Université Lille 1

Ecole Doctorale Sciences Pour l'Ingénieur Université Lille Nord-de-France

Centre de Recherche en Informatique, Signal et Automatique de Lille
(CRIStAL, UMR CNRS 9189)

Centre de Recherche INRIA Lille-Nord Europe

Thèse

pour obtenir le grade de

Docteur de l'Université de Lille 1

Discipline : Informatique

Numéro d'ordre : 42381 | Année : 2017

Optimisation combinatoire et extraction de connaissances sur données hétérogènes et temporelles : Application à l'identification de parcours patients

par

Maxence VANDROMME

Date de soutenance : 30 Mai 2017

Jury

Directeurs :	Clarisse DHAENENS	Professeur des Universités, Université Lille 1
	Laetitia JOURDAN	Professeur des Universités, Université Lille 1
Rapporteurs :	Pascale KUNTZ-COSPEREC	Professeur des Universités, Ecole Polytechnique Nantes
	Stéphane BONNEVAY	Maître de Conférences, HDR, Université Lyon 1
Examinateurs :	Vincent AUGUSTO	Maître de Conférences, Mines Saint-Etienne
	Patrick DE CAUSMAECKER	Professeur des Universités, KU Leuven
	Serge PETITON	Professeur des Universités, Université Lille 1
Invités :	Julie JACQUES	Ingénieur, Société Alicante

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Abstract Hospital data exhibit numerous specificities that make the traditional data mining tools hard to apply. In this thesis, we mainly focus on the heterogeneity associated with hospital data and on their temporal aspect. This work is done within the frame of the ANR ClinMine research project and a CIFRE partnership with the Alicante company. In this thesis, we propose two new knowledge discovery methods suited for hospital data, each able to perform a variety of tasks : classification, prediction, discovering patients profiles, etc.

In the first part, we introduce MOSC (Multi-Objective Sequence Classification), an algorithm for supervised classification on heterogeneous, numeric and temporal data. In addition to binary and symbolic terms, this method uses numeric terms and sequences of temporal events to form sets of classification rules. MOSC is the first classification algorithm able to handle these types of data simultaneously. Its application to real hospital data shows the correctness and relevance of the classification rules learned.

In the second part, we introduce HBC (Heterogeneous BiClustering), a biclustering (two-way unsupervised classification) algorithm for heterogeneous data, a problem that has never been studied so far. This algorithm is extended to support temporal data of various types : temporal events and unevenly-sampled time series. HBC is used for a case study on a set of hospital data, whose goal is to identify groups of patients sharing a similar profile. The results make sense from a medical viewpoint ; they indicate that relevant, and sometimes new knowledge is extracted from the data. These results also lead to the definition of further, more precise case studies. The integration of HBC within a software is also engaged, with the implementation of a parallel version and a visualization tool for biclustering results.

Keywords Datamining, combinatorial optimization, classification, biclustering, heterogeneous data, temporal data, machine learning.

Résumé Les données médicales hospitalières présentent de nombreuses spécificités qui rendent difficilement applicables les méthodes de fouille de données traditionnelles. Dans cette thèse, nous nous intéressons particulièrement à l'hétérogénéité de ces données ainsi qu'à leur aspect temporel. Dans le cadre du projet ANR ClinMine et d'une convention CIFRE avec la société Alicante, nous proposons deux nouvelles méthodes d'extraction de connaissances adaptées à ces types de données, permettant chacune de répondre à des problématiques différentes : classification, prédiction, découverte de profils patients, etc.

Dans la première partie, nous développons l'algorithme MOSC (Multi-Objective Sequence Classification) pour la classification supervisée sur données hétérogènes, numériques et temporelles. Cette méthode accepte, en plus des termes binaires ou symboliques, des termes numériques et des séquences d'événements temporels pour former des ensembles de règles de classification. MOSC est le premier algorithme de classification supportant simultanément ces types de données. L'application à des données hospitalières réelles permet de montrer la cohérence médicale des informations utilisées pour former les règles de classification.

Dans la seconde partie, nous proposons une méthode de classification double non-supervisée (ou *biclustering*) pour données hétérogènes, un problème qui n'a à notre connaissance jamais été exploré. Cette méthode, HBC (Heterogeneous BiClustering), est étendue pour supporter les données temporelles de différents types : événements temporels et séries temporelles irrégulières. HBC est utilisée pour un cas d'étude sur un ensemble de données hospitalières, dont l'objectif est d'identifier des groupes de patients ayant des profils similaires. Les résultats obtenus sont cohérents d'un point de vue médical ; ils permettent d'extraire des informations vraisemblables et parfois nouvelles, et amènent à la définition de cas d'étude plus précis. L'intégration dans

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

une solution logicielle est également engagée, avec une version parallèle de HBC et un outil de visualisation des résultats.

Mots-clés Fouille de données, optimisation combinatoire, classification, biclustering, données hétérogènes, données temporelles, apprentissage automatique.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Remerciements

Je tiens tout d'abord à remercier mes directrices de thèse, Clarisse et Laetitia, pour toute l'aide qu'elles m'ont apportée durant ces trois années de thèse, pour leur disponibilité exemplaire et pour les nombreuses discussions enrichissantes que nous avons pu avoir. Leurs conseils m'ont permis de mieux comprendre et aborder le monde de la recherche scientifique, et je leur en suis grandement reconnaissant.

Je remercie également mes collègues d'Alicante, en particulier Julie et Julien qui m'ont suivi tout au long de la thèse. Leur aide m'a permis de mieux saisir l'aspect industriel, les problématiques métier et le contexte général dans lequel s'inscrivaient mes travaux. Je remercie spécialement Julie pour son assistance répétée et très utile pour la relecture d'articles. Mes remerciements à David, Charles, Cédric, Samuel, Fabienne, Eve, Muriel, Valentin, pour contribuer à une atmosphère de travail agréable et productive. Alicante m'a donné l'opportunité de travailler sur des problèmes concrets dans un domaine passionnant, tout en me laissant une grande liberté sur la manière de les aborder.

J'adresse mes remerciements sincères aux membres du jury de thèse : Pascale Kuntz-Cosperec et Stéphane Bonnevay, rapporteurs, Vincent Augusto et Patrick de Causmaecker, examinateurs. Leurs retours et nombreuses questions pertinentes m'ont été très utiles. Je remercie spécialement Serge Petiton, pour avoir accepté de présider le jury et pour m'avoir apporté son aide au cours de ces dernières années.

Je remercie mes collègues d'INRIA, en particulier les amis et camarades de baby-foot : Benjamin, Alexandre, Ray, Lucien, Aymeric, Bayrem, Léo et Martin ; les membres de l'équipe Dolphin avec qui j'ai eu l'occasion d'échanger : Fanny, Marie-Eléonore, Bilel et Arnaud ; et tous ceux qui sont passés dans les équipes Dolphin et Inocs durant ces années.

Enfin, un grand merci aux amis et membres de ma famille qui ont fait le déplacement pour la soutenance, et dont le soutien compte beaucoup. A mes parents pour m'avoir permis de poursuivre mes études jusqu'à ce niveau. A mon frère pour tout ce que nous avons partagé.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Table des matières

Table des figures **11**

Liste des tableaux **14**

Introduction **17**

1 Contexte **21**
1.1 Données hospitalières . . . 21
1.1.1 Différentes sources pour les données . . . 21
1.1.2 Données PMSI . . . 22
1.1.3 Circuit des données hospitalières . . . 24
1.1.4 Exploitation des données . . . 25
1.1.5 Spécificités des données hospitalières . . . 27
1.2 Projet ANR ClinMine . . . 28
1.2.1 Présentation générale du projet . . . 29
1.2.2 Cas d'étude 1 - optimisation des trajets patients . . . 30
1.2.3 Cas d'étude 2 - inclusion dans les essais cliniques . . . 33
1.2.4 Contrainte sur l'interprétabilité des résultats . . . 34
1.3 Etat de l'art - outils de fouille de données . . . 35
1.4 Contexte scientifique . . . 35
1.4.1 Fouille de données . . . 35
1.4.2 Optimisation . . . 37
1.5 Conclusions . . . 40

I Classification supervisée pour données numériques et temporelles **41**

2 MOCA-I **43**
2.1 Modélisation . . . 43
2.1.1 Généralités sur la classification . . . 43
2.1.2 Classification à base de règles . . . 44
2.1.3 Aspect multi-objectif du problème . . . 44
2.1.4 Représentation d'une solution . . . 46
2.2 Voisinage et algorithme d'optimisation . . . 47
2.2.1 Voisinage . . . 47
2.2.2 Algorithme d'optimisation . . . 48
2.3 Choix d'une solution unique . . . 48
2.3.1 Choix dans l'archive . . . 50

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

2.3.2 Construction d'une nouvelle solution 50
2.3.3 Observations 52
2.4 Conclusion 53

3 MOCA-IQ - Classification sur données numériques 55
3.1 Problème posé par les attributs numériques 55
3.2 Première approche : discrétisation des données numériques 57
3.2.1 Généralités sur la discrétisation 57
3.2.2 Différenciation des méthodes de discrétisation 58
3.2.3 Processus de comparaison des méthodes de discrétisation 59
3.2.4 Expérimentations 60
3.3 Seconde approche : gestion des attributs numériques dans la recherche locale . . 67
3.3.1 Règles de classification avec attributs numériques 68
3.3.2 Génération des solutions initiales 68
3.3.3 Voisinage des termes numériques 69
3.3.4 Expérimentations 74
3.3.5 Discussion des résultats et limites des systèmes proposés 74
3.4 Comparaison et combinaison des approches 76
3.4.1 Comparaison des deux approches 76
3.4.2 Combinaison des approches : motivations 77
3.4.3 Reformulation proposée 78
3.4.4 Expérimentations 78
3.5 Comparaison à la littérature 80
3.5.1 Méthodes de classification 80
3.5.2 Résultats et discussion 80
3.6 Sélection d'opérateurs de voisinage 82
3.6.1 Motivations 82
3.6.2 Méthode proposée 83
3.6.3 Expérimentations 85
3.7 Amélioration de la gestion de l'archive 86
3.7.1 Motivations 86
3.7.2 Méthode proposée 87
3.7.3 Expérimentations 88
3.8 Conclusion 88

4 MOSC - Classification sur données temporelles 91
4.1 Motivations et description du problème 91
4.1.1 Motivations 91
4.1.2 Trois aspects au problème : séquence, récurrence et durée 92
4.2 Etat de l'art : classification sur données temporelles 93
4.3 Modélisation et intégration dans MOCA-I 96
4.3.1 Représentation des attributs *séquence* 96
4.3.2 Voisinage 96
4.3.3 Evaluation des solutions - *matching* de séquences 102
4.3.4 Extension du modèle - séquences multi-niveaux 106
4.3.5 Paramétrage de la méthode 107
4.3.6 Durée entre événements 108
4.4 Validation et cas d'étude 110
4.4.1 Données 110

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

4.4.2 *Sliding Windows* - discrétisation temporelle . . . 110
4.4.3 Comparaison avec d'autres algorithmes . . . 112
4.4.4 Interprétation et discussion des résultats . . . 115
4.4.5 Convergence de l'algorithme . . . 115
4.4.6 Cas d'étude - analyse qualitative des solutions . . . 117
4.5 Conclusion . . . 118

II Biclustering **119**

5 Biclustering sur données hétérogènes **121**
5.1 Définition du problème . . . 121
5.1.1 Recherche de profils patient . . . 121
5.1.2 Généralités sur le biclustering . . . 122
5.1.3 Etat de l'art . . . 125
5.1.4 Biclustering pour données hospitalières . . . 128
5.2 Première approche : résolution par méta-heuristique . . . 129
5.2.1 Modélisation . . . 129
5.2.2 Méthode proposée . . . 130
5.2.3 Expérimentations . . . 133
5.3 Seconde approche : résolution par heuristique constructive . . . 138
5.3.1 Méthode proposée . . . 138
5.3.2 Expérimentations . . . 143
5.3.3 Discussion . . . 145
5.4 Conclusion . . . 145

6 Biclustering sur données temporelles **147**
6.1 Gestion des événements temporels . . . 147
6.1.1 Motivations et modélisation . . . 147
6.1.2 Mesure de qualité . . . 149
6.1.3 Adaptations de l'algorithme . . . 151
6.1.4 Expérimentations . . . 151
6.1.5 Conclusion . . . 157
6.2 Gestion des séries temporelles irrégulières . . . 157
6.2.1 Motivations et travaux existants . . . 158
6.2.2 Aperçu de la méthode proposée . . . 161
6.2.3 Mesure de similarité . . . 161
6.2.4 Clustering de séries temporelles . . . 163
6.2.5 Intégration dans HBC . . . 166
6.2.6 Expérimentations . . . 166
6.2.7 Discussion . . . 172
6.3 Conclusion . . . 172

7 Implémentation - parallélisation et visualisation **173**
7.1 Parallélisation . . . 173
7.1.1 Motivations . . . 173
7.1.2 Première approche : MPI . . . 174
7.1.3 Seconde approche : OpenMP . . . 177
7.1.4 Expérimentations . . . 177
7.1.5 Conclusion sur la parallélisation . . . 179

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

7.2 Visualisation 180
7.2.1 Motivations et difficultés 180
7.2.2 Deux modes de visualisation : vue globale et vues détaillées 182
7.2.3 Intégration dans une application 186
7.2.4 Perspectives 189

8 Cas d'étude - identification de parcours patients 191
8.1 Présentation du problème 191
8.2 Données utilisées 192
8.3 Extension aux séries symboliques 193
8.3.1 Motivations 193
8.3.2 Méthode choisie 194
8.3.3 Expérimentations 194
8.4 Expérimentations et résultats 197
8.4.1 Préparation des données 197
8.4.2 Résultats 198
8.4.3 Analyse et discussion 200
8.5 Conclusions et perspectives 202

Conclusion 205

A Comparaison des méthodes de clustering pour séries temporelles irrégulières dans HBC 209

Bibliographie 215

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Table des figures

1.1 Parcours d'un patient dans trois unités médicales successives, génération des RUM et du RSS . . . 23
1.2 Circuit des données hospitalières . . . 24
1.3 Deux approches pour l'analyse de données : validation d'hypothèse et approche plus exploratoire . . . 27
1.4 Organisation du projet ClinMine - tâches et partenaires . . . 30
1.5 Processus proposé pour le cas d'étude Recherche de séquences d'Unités Médicales . . . 32
1.6 Critères d'inclusion et d'exclusion pour un essai clinique sur la maladie d'Alzheimer 33
1.7 Processus de découverte de connaissances dans les bases de données (KDD) . . . 36
1.8 Représentation d'un ensemble de solutions pour un problème d'optimisation bi-objectif 39

2.1 Matrice de confusion . . . 45
2.2 Sélection des solutions de l'archive de meilleure confiance et de sensibilité supérieure à 0.2, 0.4, 0.6 et 0.8 ; représentées en rouge . . . 51

3.1 Taxonomie des différents types d'attributs non-temporels . . . 56
3.2 Deux approches pour la gestion des attributs quantitatifs : discrétisation ou modification de l'algorithme . . . 57
3.3 Séparation des données en ensembles d'apprentissage et de test, génération des modèles (règles) avec l'ensemble d'apprentissage, validation sur les deux ensembles séparément . . . 62
3.4 Performance des opérateurs de voisinage : nombre de solutions conservées dans l'archive, nombre total de solutions générées, pourcentage de solutions conservées . . . 83
3.5 Surface représentant l'hypervolume de cinq solutions par rapport à une solution de référence r . . . 84

4.1 Exemple de règle de classification pour MOSC : deux termes avec attributs non-temporels et un terme avec attribut *séquence* . . . 97
4.2 Distribution du temps total d'occurence pour la séquence d'une règle (en jours) . . . 109
4.3 Processus de transformation des données pour comparaison des méthodes . . . 113
4.4 Evolution de la f-mesure de la meilleure solution de l'archive au fil des itérations . . . 116

5.1 Réorganisation des lignes et des colonnes pour identification d'un bicluster . . . 123
5.2 Différents types de biclusters définis par leur type d'homogénéité dans les données . 124
5.3 Différents types de biclusters en fonction de leur recouvrement entre eux et de la couverture de la matrice . . . 125
5.4 Biclustering par méta-heuristique - vue d'ensemble . . . 131
5.5 Vue d'ensemble de l'algorithme de biclustering par heuristique proposé, HBC . . . 138

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

5.6 Exemple de formation d'un bicluster par HBC : ajout de la colonne *num1* (et suppression de la ligne $I6$), puis ajout de la colonne $symb1$ (sans suppression de ligne). A ce point, la construction du bicluster s'arrête car il est impossible d'ajouter une colonne sans supprimer une ligne, ce qui réduirait la taille globale du bicluster . . . 140

6.1 Extraction d'un bicluster pour des données temporelles. Le bicluster extrait ici comporte trois lignes I1, I2 et I5 pour lesquelles l'ordre des événements evt2, evt3 et evt5 est identique. Cet ordre est représenté dans le tableau en bas à droite. 148
6.2 Ordre des événements composant le bicluster dont nous cherchons à calculer la qualité. Les positions des événements sont utilisées comme valeurs pour le calcul de qualité. 150
6.3 Exemple de bicluster trouvé par OPSM dans BicAT, en haut à gauche sur l'image, sur un jeu de données $n0s1$ (pas de bruit, taille 50x10). On remarque que deux autres lignes (*inst49* et *inst52*) et colonnes, en vert, pourraient être ajoutées au bicluster mais ne le sont pas 153
6.4 Données réelles - qualité et taille des biclusters trouvés sur le jeu MRB 156
6.5 Mécanisme de traitement des séries temporelles dans HBC : calcul de la matrice de similarité puis clustering sur cette matrice pour obtenir un ensemble fini de valeurs possibles, et utilisation de la mesure de similarité pour calculer la qualité des biclusters 162
6.6 Deux exemples de clusters de séries temporelles ; à gauche, un attribut de type *évolution*, à droite un attribut de type *valeurs* 171

7.1 Architecture de calcul parallèle à mémoire distribuée 175
7.2 HBC sur architecture à mémoire distribuée - première approche 176
7.3 HBC sur architecture à mémoire distribuée - seconde approche 176
7.4 Architecture de calcul parallèle à mémoire partagée 177
7.5 HBC sur architecture à mémoire partagée 178
7.6 HBC parallélisé avec OpenMP - temps d'exécution en fonction du nombre de processus utilisés, pour les quatre jeux de données détaillés dans la Table 7.1. Le temps d'exécution, en ordonnée, utilise une échelle logarithmique 179
7.7 Représentation visuelle d'un bicluster par l'outil BicAT, en haut à gauche de l'écran 181
7.8 Représentation visuelle de deux biclusters par l'outil ISA, dans une structure en damier 181
7.9 Représentation souhaitée pour la vue générale : chaque bicluster est représenté par une couleur afin de visualiser la couverture de la matrice ; les valeurs sont ignorées . 183
7.10 Vue générale : représentation de l'ensemble des biclusters et des recouvrements entre eux. Chaque couleur représente un bicluster. Les quatre parties (1, 2, 3, 4) sont détaillées dans la Section 7.2.2.1 184
7.11 Vue détaillée pour un bicluster, situé en haut à gauche et délimité par les lignes noires. Le bicluster contient 44 instances et 15 attributs, dont 14 numériques et 1 symbolique ; les couleurs représentent les valeurs de la matrice de données 185
7.12 Vue détaillée pour un bicluster contenant 1 attribut binaire et 3 attributs temporels. Ceux-ci sont groupés ensemble et dans leur ordre d'apparition afin de faciliter la visualisation 186
7.13 Vue détaillée pour deux biclusters avec recouvrement, délimités par des lignes rouges (incluses dans la vue) 187
7.14 Fonctionnement de l'outil de visualisation VisuHBC : fichiers d'entrée et de sortie. L'outil génère k vues détaillées, correspondant aux k biclusters découverts par HBC 187
7.15 Interface de l'application VisuHBC, affichant la vue détaillée pour un bicluster . . . 188
7.16 Fonctionnement souhaité pour l'application VisuHBC : intégration de l'outil de génération de vues et de l'interface de visualisation 189

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

8.1 Cluster de séries symboliques. Chaque série représente les changements d'état pour un patient, pour le type de diabète, au fil du temps. Les patients représentés ici passent d'un diabète type 1 (3 sur l'axe des ordonnées) à un état non renseigné (-1 sur l'axe des ordonnées) 196
8.2 Traitement du diabète (I) 197
8.3 Traitement du diabète (II) 197
8.4 Préparation et analyse des données pour le cas d'étude : agrégation à partir de différentes sources, séparation par GHM et biclustering 199

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Liste des tableaux

3.1 Caractéristiques des méthodes de discrétisation retenues 60
3.2 Caractéristiques des jeux de données utilisés 61
3.3 F-mesures obtenues sur l'ensemble d'apprentissage 65
3.4 F-mesures obtenues sur l'ensemble de test 65
3.5 MCC obtenus sur l'ensemble d'apprentissage 66
3.6 MCC obtenus sur l'ensemble de test 66
3.7 Rang moyen sur l'ensemble des attributs numériques pour le nombre d'intervalles générés 67
3.8 Rangs moyens sur les trois mesures 67
3.9 Comparaison des approches pour MOCA-IQ, f-mesure, ensemble d'apprentissage . . 74
3.10 Comparaison des approches pour MOCA-IQ, f-mesure, ensemble de test 74
3.11 Comparaison des approches pour MOCA-IQ, MCC, ensemble d'apprentissage 74
3.12 Comparaison des approches pour MOCA-IQ, MCC, ensemble de test 75
3.13 F-mesures obtenues sur l'ensemble d'apprentissage 77
3.14 F-mesures obtenues sur l'ensemble de test 77
3.15 Traduction d'un terme utilisant un attribut discrétisé en un terme équivalent utilisant un attribut numérique 78
3.16 F-mesures obtenues sur l'ensemble d'apprentissage 79
3.17 F-mesures obtenues sur l'ensemble de test 79
3.18 Comparaison de MOCA-IQ avec d'autres algorithmes de classification, f-mesure, ensemble d'apprentissage 81
3.19 Comparaison de MOCA-IQ avec d'autres algorithmes de classification, f-mesure, ensemble de test 81
3.20 Rangs moyens mesurés sur les ensembles de test 81
3.21 F-mesures obtenues sur l'ensemble d'apprentissage 86
3.22 F-mesures obtenues sur l'ensemble de test 86
3.23 Comparaison des méthodes d'archivage des solutions, f-mesure, ensemble d'apprentissage 88
3.24 Comparaison des méthodes d'archivage des solutions, f-mesure, ensemble de test . . 88

4.1 Relations d'ordre possibles entre les différents événements et épisodes de la séquence S 101
4.2 Informations jeux de données réels complets 111
4.3 Informations jeux de données réels réduits 113
4.4 F-mesure sur l'ensemble de test, données réduites 114
4.5 Rangs moyens sur l'ensemble de test, données réduites. Les rangs montrent la position moyenne de chaque algorithme (1 étant la meilleure), basée sur la *f-mesure* 114
4.6 F-mesure sur l'ensemble de test, données complètes 114
4.7 Rangs moyens sur l'ensemble de test, données complètes 115

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

5.1 Identification des différents types de méthodes de biclustering, avec quelques exemples pour chaque type . . . 127
5.2 Paramètres utilisés pour l'insertion des biclusters dans les données synthétiques : paramètre, valeurs et remarque éventuelle . . . 134
5.3 Comparaison de LSBC et CC - qualité et taille des biclusters générés. $nXsY$ correspond à un jeu de données avec le niveau de bruit X et la taille Y. La meilleure valeur est indiquée en gras . . . 135
5.4 Comparaison de LSBC et CC - recouvrement des biclusters injectés. $nXsY$ correspond à un jeu de données avec le niveau de bruit X et la taille Y . . . 136
5.5 Qualité et taille moyennes des biclusters . . . 144
5.6 Mesures de redécouverte des biclusters injectés U et E . . . 144

6.1 Données temporelles synthétiques - qualité q et taille des biclusters pour HBC et OPSM 152
6.2 Données temporelles synthétiques - mesures de recouvrement des biclusters (U et E) pour HBC et OPSM . . . 152
6.3 Performances de HBC - données temporelles pures, larges - qualité, taille et recouvrement des biclusters injectés . . . 154
6.4 Performances de HBC - données hétérogènes (dont temporelles), larges - qualité, taille et recouvrement des biclusters injectés . . . 155
6.5 Données réelles - caractéristiques . . . 155
6.6 Données réelles - résultats . . . 156
6.7 Qualité moyenne des biclusters formés . . . 168
6.8 Taille moyenne des biclusters formés . . . 168
6.9 Mesure de recouvrement U - part du bicluster injecté retrouvée . . . 169
6.10 Mesure de recouvrement E - part de données externes au bicluster injecté . . . 169
6.11 Recouvrement des biclusters injectés ; comparaison avec DTW . . . 170

7.1 Informations jeux de données - tests parallélisation . . . 178

A.1 Stabilité - mesure de dissimilarité pour les variantes de *k-medoids* . . . 210
A.2 Intra_max - distance maximale entre deux points . . . 211
A.3 Intra_avg - distance moyenne entre deux points . . . 211
A.4 Intra_centre - distance moyenne au centroïde . . . 211
A.5 Inter_single - distance entre les points les plus proches . . . 212
A.6 Inter_complete - distance entre les points les plus éloignés . . . 212
A.7 Inter_centre - distance entre les centroïdes . . . 212
A.8 Composite - index Davies-Bouldin . . . 213

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Introduction

Cette thèse s'intéresse aux problématiques d'extraction de connaissances à partir des données médicales hospitalières. Celles-ci présentent plusieurs spécificités qui rendent inapplicables la plupart des méthodes d'analyse des données. En particulier, ces données sont fortement hétérogènes, c'est-à-dire qu'elles contiennent des informations de différents types (quantitatives et qualitatives), et temporelles, c'est-à-dire que les informations (par exemple un acte médical ou un diagnostic) sont souvent marquées dans le temps. L'objectif de cette thèse est de conçevoir et de développer des outils de fouille de données permettant de gérer les données hospitalières. Ces outils doivent être le plus génériques possibles, afin d'être à même de répondre aux différents cas d'étude et interrogations pouvant se poser sur ces données. Les travaux de cette thèse se concentrent sur deux grandes classes de méthodes : classification supervisée et classification double non-supervisée (ou *biclustering*). Ces deux classes sont complémentaires et permettent de répondre à une grande variété de problèmes différents. Dans ces travaux, la conception des méthodes emprunte en priorité aux techniques d'optimisation, et notamment d'optimisation combinatoire, sans toutefois s'y limiter. Cette thèse s'inscrit dans le cadre d'un projet de recherche plus large incluant plusieurs partenaires hospitaliers. Par conséquent, les méthodes développées sont évaluées en permanence sur des ensembles de données réelles mises à disposition par ces partenaires, qui permettent de valider leur efficacité et d'apporter en même temps des réponses à des problématiques médicales.

Cette thèse fait l'objet d'un dispositif CIFRE (Conventions Industrielles de Formation par la REcherche) et est co-encadrée par la société Alicante et l'équipe DOLPHIN du laboratoire CRIStAL, UMR CNRS, Université Lille 1. Alicante est une PME spécialisée dans les progiciels pour les hôpitaux, qui intervient notamment dans les domaines de l'identito-vigilance, du décisionnel hospitalier, de la bibliométrie ou de la recherche clinique. La société a déjà participé à des projets de recherche, tels que AKENATON (ANR TECSAN 2007-2011) et FAROS (RNTL 2005-2008). L'équipe DOLPHIN (Discrete multiobjective Optimization for Large-scale Problems with Hybrid dIstributed techNiques) dispose d'une expertise dans les domaines de l'optimisation et de la fouille de données. Ces deux structures ont déjà participé conjointement à l'encadrement de la thèse de Julie Jacques, de 2010 à 2013, dont les travaux servent de base à une partie de la présente thèse [89]. Celle-ci s'inscrit également dans le cadre du projet ClinMine (ANR TECSAN 2014-2017), dont l'objectif est "l'optimisation de la prise en charge des patients à l'hôpital".

Compte tenu de la complexité relative du contexte afférent aux travaux de cette thèse, le Chapitre 1 est dédié à la présentation de ce contexte. Dans le présent chapitre, nous donnons cependant les motivations et objectifs de ce travail, et détaillons l'organisation générale du document.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Motivations et objectifs

Les établissements hospitaliers génèrent en permanence une quantité importante d'informations de différentes natures, liées tant à leur activité de soins qu'à leur fonctionnement interne. L'informatisation de l'intégralité de ces données a été initiée il y a environ 20 ans au niveau national, et se poursuit encore à l'heure actuelle avec des programmes tels que Hôpital 2007, Hôpital 2012 et Hôpital Numérique. Cette informatisation s'accompagne d'une uniformisation d'une partie importante des données. Ces deux phénomènes rendent possible l'exploitation des données par des outils informatiques. Par ailleurs, la notion de *parcours patient* est de plus en plus utilisée pour structurer l'activité des établissements de soins. La loi HPST (Hôpital, Patient, Santé, Territoire) votée en France en 2009 impose la création progressive de parcours coordonnés de soins au sein d'un territoire de santé. Ces parcours visent à agréger le suivi médical d'un patient dans différents types d'établissements médicaux : hospitaliers, sanitaires et sociaux, médecine de ville. Une meilleure connaissance de ces parcours permettrait de se focaliser sur ceux associés à certaines pathologies (telles que les AVC), par exemple pour prédire l'évolution future des patients (réhospitalisations, complications, etc.) en fonction de leur profil actuel.

Au-delà de l'organisation des bases de données et de l'utilisation de statistiques descriptives pour mieux connaître les données, il devient possible d'appliquer des outils d'analyse de données plus avancés. Les méthodes statistiques telles que les modèles de régression ou l'analyse des corrélations sont très populaires pour l'exploitation des données médicales. Dans cette thèse, nous nous intéressons aux méthodes d'apprentissage automatique, qui permettent d'extraire d'autres types de modèles à partir des données. L'analyse des données présente un intérêt notable pour les hôpitaux, qui peuvent ainsi mieux connaître leur clientèle, améliorer le parcours de soins des patients et potentiellement identifier de nouvelles hypothèses médicales à valider. Les questions que se posent des praticiens, responsables de l'information médicale ou de la recherche clinique, sont nombreuses et variées. Pour cette raison, des méthodes d'analyse de données génériques sont plus adaptées que des méthodes *ad hoc* pour chaque cas d'étude.

Dans ces travaux nous cherchons à développer des méthodes d'analyse de données permettant de répondre à deux grands types de problématiques. Le premier concerne la prédiction ou la classification supervisée, c'est-à-dire déterminer, à partir d'un ensemble d'informations connues pour un patient (par exemple), la valeur d'une information inconnue pour ce même patient. Un exemple d'application pour la prédiction est : compte tenu du dossier patient à l'heure actuelle, quelle est la probabilité que ce patient fasse une rechute avant cinq ans ? Le second type de problématiques concerne la recherche de groupes homogènes parmi les patients, c'est-à-dire de patients partageant un profil similaire. Il s'agit d'une tâche plus exploratoire, dite non-supervisée, dont les résultats doivent être évalués directement par des experts médicaux afin d'en valider la cohérence. Pour ces deux types de tâches, nous proposons deux méthodes différentes de fouille de données qui guident le plan de ce document, décrit dans la suite de ce chapitre.

Plan et contributions

Cette thèse est organisée selon le plan suivant.

Dans le **Chapitre 2**, nous présentons plus en détail le contexte afférent aux travaux de la thèse. Ce chapitre contient une description des données hospitalières avec leurs spécificités, leur cycle de vie et les différentes problématiques liées à leur exploitation pour analyse. Le projet de recherche plus général qui sert de cadre à cette thèse est également présenté dans ses objectifs et cas d'étude concrets. Une revue des outils existants pour la fouille de données permet de mieux situer les contributions apportées. Enfin, l'introduction des domaines de la fouille de données et

© 2017 Tous droits réservés. lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

de l'optimisation permet de définir formellement le cadre scientifique du projet.

Les contributions apportées par cette thèse sont organisées en deux parties correspondant aux deux problématiques majeures identifiées. La première partie, composée des Chapitres 3, 4 et 5, s'intéresse à la classification supervisée pour données hospitalières.

Dans le **Chapitre 3**, nous exposons le fonctionnement de l'algorithme MOCA-I (Multi-Objective Classification Algorithm for Imbalanced data) pour la classification supervisée adaptée aux particularités des données hospitalières. Une attention particulière est apportée à la modélisation sous forme d'un problème d'optimisation multi-objectif et la représentation de solutions sous forme de règles de classification. La conception de MOCA-I provient de la thèse de Julie Jacques et est donc antérieure à la thèse présentée ici, mais cet algorithme sert de base aux deux chapitres suivants et mérite donc une description détaillée. Nous étendons cependant ici la discussion menée dans les travaux sur MOCA-I, à propos du choix d'une solution unique parmi l'archive finale des solutions possibles.

Dans le **Chapitre 4**, nous présentons l'algorithme MOCA-IQ, une extension de MOCA-I que nous proposons afin de gérer en plus les données quantitatives. Deux approches différentes, par discrétisation en pré-traitement et par gestion dans la recherche locale, sont proposées et comparées. L'approche la plus performante est retenue et validée par comparaison avec d'autres algorithmes de classification de la littérature. Deux améliorations de l'algorithme, indépendantes du type de données considéré, sont également proposées et évaluées. La seconde, qui définit un nouveau mécanisme pour l'archivage des solutions, permet d'améliorer la qualité des résultats et est donc intégrée dans l'algorithme.

Dans le **Chapitre 5**, nous proposons l'algorithme MOSC (Multi-Objective Sequence Classification) pour la classification sur données temporelles. Le modèle choisi définit un nouveau type de données permettant d'encapsuler les données temporelles au sein de séquences. De nouveaux opérateurs de voisinage adaptés à ce type de données, ainsi qu'un algorithme de comparaison de séquences, sont proposés. Des expérimentations permettent de valider l'efficacité de l'approche par rapport à d'autres algorithmes de classification. MOSC est appliqué à plusieurs cas d'études sur données hospitalières réelles ; les résultats montrent l'intérêt de considérer l'aspect temporel des données.

La seconde partie, composée des Chapitres 6, 7, 8 et 9, s'intéresse au problème de la classification double non-supervisée, ou *biclustering*.

Dans le **Chapitre 6**, nous nous intéressons au biclustering sur données hétérogènes. Nous justifions l'intérêt de cette méthode pour la recherche de profils patient dans les données hospitalières. Une revue de l'état de l'art expose l'absence de méthodes existantes pour cette tâche. Nous proposons une première modélisation du biclustering, sous forme d'un problème d'optimisation bi-objectif, et une méta-heuristique de résolution. Nous montrons les limitations de cette approche et en proposons une seconde, mieux adaptée au problème et aux spécificités des données, basée sur une heuristique constructive et une simplification des données en pré-traitement. L'algorithme proposé, HBC (Heterogeneous BiClustering), est comparé favorablement à d'autres méthodes de biclustering sur des ensembles de données homogènes.

Dans le **Chapitre 7**, nous étendons le champ d'application de HBC aux données temporelles de deux types : événements temporels et séries temporelles irrégulières. Pour le premier type, nous proposons une modélisation et une adaptation de l'algorithme, et montrons expérimentalement

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

l'efficacité de l'approche par rapport à un algorithme de référence. Pour le second type, nous présentons d'abord les particularités des séries temporelles considérées (longueur et irrégularité). Nous définissons ensuite un mode de simplification de ces données pour permettre une utilisation dans HBC. Cette simplification implique la conception d'une nouvelle mesure de distance entre deux séries temporelles et le choix d'une méthode de clustering adaptée. Des expérimentations montrent la validité de la méthode proposée, ainsi que l'efficacité de la mesure de distance proposée par rapport à une mesure existante.

Dans le **Chapitre 8**, nous nous intéressons aux problématiques de l'intégration de HBC au sein d'applications fonctionnelles. Ces problématiques sont importantes pour une thèse menée dans un cadre industriel et inscrite dans un projet de recherche multi-partenaires. Dans un premier temps, nous explorons différents modèles pour la parallélisation de HBC, permettant un déploiement plus efficace sur des systèmes de calcul réels et une meilleure scalabilité. Dans un second temps, nous présentons les bases d'un outil de visualisation pour le biclustering sur données hétérogènes, et son intégration dans une application.

Dans le **Chapitre 9**, nous définissons un cas d'étude permettant de valider HBC. Ce cas d'étude concerne une problématique médicale réelle en lien avec les objectifs du projet de recherche ClinMine, et implique l'utilisation des différents aspects de HBC (en particulier données hétérogènes et temporelles). Pour pouvoir traiter l'intégralité de ces données, nous proposons une extension de HBC à un nouveau type de données, les séries symboliques irrégulières, assez proches des séries temporelles irrégulières. L'objectif de ce cas d'étude est de comparer les parcours de soins des patients ayant un profil similaire. Les résultats de cette étude sont cohérents d'un point de vue médical, reflètent des phénomènes connus pour la prise en charge des patients, et font apparaître de nouvelles connaissances à travers des cas particuliers intéressants. L'analyse de ces résultats permet la définition de nouveaux cas d'étude plus précis.

Dans le **Chapitre 10**, nous résumons les contributions principales de cette thèse et présentons différentes perspectives d'amélioration et nouveaux axes de recherche qui découlent des travaux réalisés.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Chapitre 1

Contexte

Les travaux réalisés au cours de cette thèse concernent la conception et la réalisation d'outils de fouille de données permettant de traiter les données issues des systèmes d'information des hôpitaux. Dans ce chapitre, nous présentons tout d'abord une description des données hospitalières, avec leurs spécificités et les difficultés que celles-ci engendrent. Nous présentons ensuite le projet de recherche ClinMine, soutenu par l'Agence Nationale de la Recherche (ANR), qui sert de cadre aux travaux de cette thèse. Nous dressons ensuite un état des lieux des outils existants pour la fouille de données afin de positionner notre travail dans le contexte actuel. Enfin, nous exposons le contexte scientifique de ce travail, en présentant les bases des domaines de recherche concernés : la fouille de données et l'optimisation.

1.1 Données hospitalières

Dans cette première section, nous présentons tout d'abord la richesse et l'hétérogénéité des données hospitalières. Nous détaillons ensuite les données dites PMSI, particulièrement intéressantes car communes à tous les hôpitaux. Nous présentons ensuite le circuit des données au sein des systèmes d'information hospitaliers, puis les nouveaux défis pour l'analyse de ces données. Enfin, nous exposons les spécificités de ces données, qui justifient la conception et le développement de nouvelles méthodes d'analyse.

1.1.1 Différentes sources pour les données

Les centres hospitaliers français disposent d'un nombre croissant de données numériques. Cet état de fait est le résultat, entre autres, de programmes de numérisation des données médicales. Au niveau national, le Programme de Médicalisation des Systèmes d'Information (PMSI) vise à ce que chaque hôpital dispose d'une partie de données traduisant son activité (notamment les actes et diagnostics), sous un format commun. En plus de ces données uniformisées, les centres hospitaliers disposent souvent d'un ensemble de systèmes et de logiciels visant à collecter les données provenant d'autres sources, telles que :

— les résultats d'analyses biologiques (prélèvements sanguins, biopsies, etc.) réalisées par les laboratoires d'analyse liés à l'hôpital.

— les radiographies et autres résultats d'imagerie médicale.

— les prescriptions médicamenteuses, et notamment les chimiothérapies.

— les courriers et notes de transmission rédigées lors d'un séjour.

— les dossiers médicaux des patients et les informations personnelles.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

— les outils plus administratifs, tels que la gestion des lits, des rendez-vous et consultations, du personnel (emplois du temps, répartition des moyens humains) ; la gestion logistique et financière.

Les hôpitaux génèrent et utilisent donc au quotidien une quantité importante de données très variées. L'organisation exacte du système d'information varie d'un hôpital à l'autre, mais inclut le plus souvent une variété de logiciels spécialisés, chacun dédié à une activité particulière. Dans ces conditions, unifier l'ensemble de ces données au sein d'une base unique est un défi de taille pour les établissements. Pour cette raison, nous nous concentrons dans cette présentation en priorité sur les données PMSI, qui sont communes à tous les hôpitaux.

1.1.2 Données PMSI

Le Programme de Médicalisation des Systèmes d'Information a été instauré en 1994, avec une extension aux établissements privés en 1997. Le socle principal des données dites PMSI concerne les séjours hospitaliers MCO (Médecine-Chirurgie-Obstétrique). Celles-ci représentent une majorité des activités hospitalières, mais laissent par exemple de côté les activités de psychiatrie, qui disposent d'un système dédié, et les consultations réalisées à l'hôpital. Ces données sont par exemple utilisées dans le cadre de la Tarification A l'Activité (T2A) : les hôpitaux sont subventionnés en fonction des actes médicaux réalisés et des diagnostics effectués, ce qui nécessite ce système commun de classification des activités. Nous présentons ici les différents éléments qui font partie du système PMSI.

Une **Unité Médicale** (UM) est un service hospitalier spécialisé dans un type de pathologies, correspondant de près aux différentes spécialités médicales. La définition des différentes UM est spécifique à chaque hôpital. "Médecine Physique et Rééducation", "Pathologies Digestives" et "ORL Pédiatrique" sont des exemples d'unités médicales. Au cours d'un séjour à l'hôpital, un patient passe par au moins une unité médicale. Le *parcours patient* peut alors être défini comme la séquence d'unités médicales par lesquelles passe un patient. Par exemple, un parcours peut être formé de la séquence d'UM suivante :

$$\text{Urgences} - \text{Neurochirurgie} - \text{Réanimation}$$

Chaque unité médicale fournit, à la sortie du patient, un **Résumé d'Unité Médicale** (RUM), qui contient les informations suivantes :

— date de naissance, sexe et code postal du patient,

— dates d'entrée et de sortie de l'UM,

— diagnostics principal et associés,

— actes réalisés,

— autres informations médicales (notamment score IGS2 de gravité [111]).

A la sortie du patient de l'hôpital, l'ensemble des RUM sont utilisés pour générer le **Résumé de Sortie Standardisé** (RSS), qui contient obligatoirement un *diagnostic principal*, et éventuellement un *diagnostic relié* et/ou des *diagnostics associés*.

Dans les RUM (et RSS), les *diagnostics* sont codés d'après la **CIM-10** (Classification Internationale des Maladies et recours aux services de santé n.10) [2]. Cette classification comporte vingt-deux chapitres, chaque chapitre regroupant un ensemble cohérent de pathologies, chacune décrite par un code à trois caractères (une lettre et deux chiffres). Par exemple, le chapitre IV regroupe les codes allant de E00 à E90, correspondant aux "Maladies endocriniennes, nutritionnelles et métaboliques". On y retrouve, entre autres, la Thyroïdite (E06), le Diabète sucré insulino-dépendant (E10) ou la Carence en vitamine D (E55). De plus chaque code peut se voir

© 2017 Tous droits réservés. lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

adjoindre un caractère supplémentaire afin de préciser la pathologie décrite. Par exemple, E10.2 correspond au diabète avec complications rénales, E10.3 au diabète avec complications oculaires, etc. Dans sa version courante, la CIM-10 contient au total 14 400 codes différents.

Les RUM renseignent également les actes effectués lors du passage d'un patient dans l'UM. Les actes sont codés d'après la **CCAM** (Classification Commune des Actes Médicaux) [1]. Il s'agit d'une classification hiérarchique comprenant 7623 codes (dans sa version de 2005), où chaque code est constitué de quatre lettres et de trois chiffres :

— la première lettre désigne l'appareil anatomique concerné.

— la deuxième lettre précise l'organe ou la fonction métabolique.

— la troisième lettre désigne l'acte effectué.

— la quatrième lettre précise la voie d'accès ou la technique utilisée.

— les chiffres permettent de distinguer les actes pour lesquels les quatre lettres sont identiques. Par exemple, le code BACA002 désigne une "Suture d'une plaie cutanée d'une paupière, sans atteinte du bord libre", et BACA008 une "Suture de plaie du sourcil".

La CCAM renseigne, pour chaque acte, le tarif remboursé par la Sécurité Sociale à l'hôpital ayant réalisé cet acte. Cette classification forme donc la base de la Tarification A l'Activité (T2A) déjà mentionnée plus haut.

La Figure 1.1 illustre le concept de *parcours patient* et présente les différents documents générés lors d'un séjour (RUM et RSS).

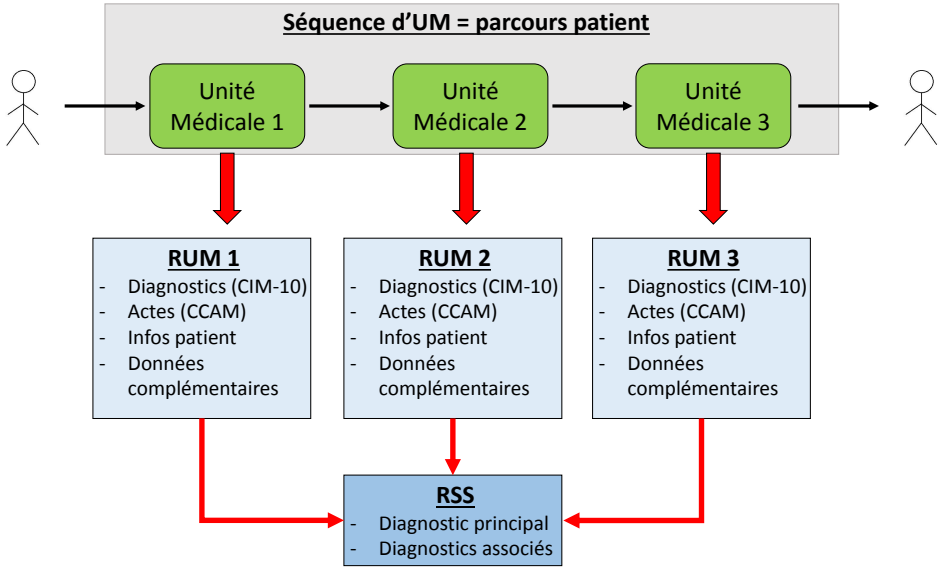


FIGURE 1.1 – Parcours d'un patient dans trois unités médicales successives, génération des RUM et du RSS

Le mode de saisie des codes PMSI du RUM varie d'un établissement à l'autre. Ces codes sont le plus souvent saisis par les praticiens de l'unité médicale concernée, ou par les membres du Département d'Information Médicale (DIM). Ce système n'est pas parfait et peut être sujet à des erreurs, volontaires ou non. Il est tout d'abord possible que la personne qui saisit les codes se trompe et renseigne, par exemple, un acte différent de celui qui a été réalisé. Il peut également

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

arriver que l'erreur soit intentionnelle. Ce genre de situations est en partie causé par l'une des finalités du PMSI, qui est la tarification à l'activité en fonction des codes enregistrés. Dans ce contexte, il est tentant d'omettre les codes n'offrant aucun intérêt pour le remboursement des actes afin de gagner du temps, ou au contraire de modifier le groupage en sur-codant afin d'optimiser le remboursement. Les données PMSI ne sont donc pas totalement fiables, mais constituent une base conséquente de données générées par les hôpitaux, car obligatoirement renseignés pour chaque patient. Dans la section suivante, nous examinons le parcours des données (PMSI et autres) au sein des systèmes d'information hospitaliers.

1.1.3 Circuit des données hospitalières

L'uniformisation, l'agrégation et le stockage des données numériques représentent un défi de taille pour les hôpitaux, qui doivent souvent repenser tout ou partie de leur système d'information. L'ensemble du parcours des données générées par l'activité hospitalière est appelée *circuit des données hospitalières*. Ce circuit des données est un processus complexe et toujours en cours d'élaboration ou de perfectionnement au sein de chaque centre. La Figure 1.2 montre un exemple de circuit complet, de la récupération des données à leur utilisation finale, d'après la vision du Département d'Information Médicale (DIM) du GHICL, l'un des partenaires du projet ClinMine (voir section suivante) :

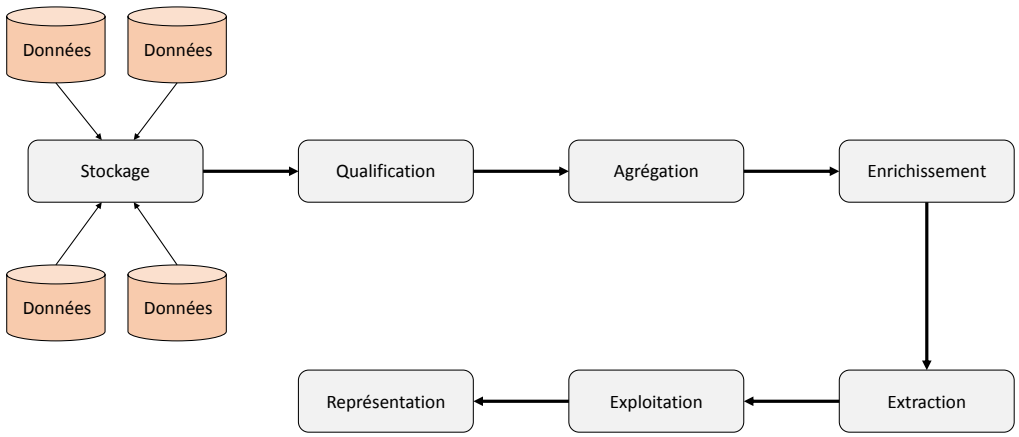


FIGURE 1.2 – Circuit des données hospitalières

Cette figure décrit les différentes étapes suivantes :

— **stockage** : la première étape consiste assez naturellement à mettre en place un système de bases de données permettant de stocker les données générées par l'activité de l'hôpital.

— **qualification** : le stockage brut des données réalisé dans la première étape peut amener à enregistrer des données erronées ou mal formatées. La *qualification* vise à garantir la qualité des données, par exemple en assurant l'uniformité des formats et unités utilisés. Une autre préoccupation majeure à ce sujet concerne *l'identitovigilance*, c'est-à-dire l'exactitude des

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

informations disponibles pour l'identité des patients : suppression ou regroupement de doublons, correction de données aberrantes, etc.

— **agrégation** : comme nous l'avons vu dans la Section 1.1.1, les données hospitalières proviennent de nombreuses sources. L'étape d'*agrégation* permet de lier entre elles ces données. Il peut s'agir, par exemple, de croiser les résultats d'analyses biologiques avec la base d'identités patients, afin de lier un patient aux résultats des analyses qu'il a subies.

— **enrichissement** : une fois la qualité et l'homogénéité des données garanties, il est possible de procéder à une phase d'*enrichissement*. Celle-ci vise à extraire des données disponibles des informations qui ne sont pas accessibles directement. Il peut s'agir, par exemple, d'appliquer des techniques de traitement du langage naturel aux courriers rédigés lors des séjours afin d'en extraire des informations complémentaires.

— **extraction** : stocker les données, et assurer leur qualité et leur accessibilité, est un travail motivé entre autres par la perspective de pouvoir utiliser ces données à des fins d'analyse. Il est donc nécessaire de pouvoir *extraire* les données des bases qui les hébergent, par exemple via des requêtes SQL.

— **exploitation** : l'étape d'*exploitation* concerne donc les différentes analyses qu'il est possible de mener sur les données disponibles. Celles-ci englobent les applications de fouille de données qui forment le cœur de cette thèse. Cette étape est exposée plus en détails dans la Section 1.1.4.

— **représentation** : l'exploitation des données produit un ensemble de modèles, qu'il est important de pouvoir *représenter* de manière à faciliter leur compréhension, en particulier par les utilisateurs finaux (personnel médical).

1.1.4 Exploitation des données

Les trois premières étapes de ce circuit, du stockage à l'agrégation, sont principalement du ressort des services informatiques internes aux centres hospitaliers. Les étapes suivantes, à partir de l'enrichissement et de l'extraction, sont celles qui nous intéressent davantage dans le cadre de cette étude. L'accessibilité à une quantité toujours croissante de données provoque de nouveaux besoins en analyse. Cela signifie également qu'il est nécessaire de revoir les outils d'analyse des données employés jusqu'à présent, lesquels ne sont pas toujours aptes à traiter cet afflux récent et massif de données. En effet, cette évolution dans les données disponibles se traduit par deux aspects majeurs :

— le **volume** des données est sans doute l'aspect le plus notable et le plus évident. Un hôpital peut accueillir plusieurs dizaines de milliers de patients par an, et la quantité d'information disponible pour chaque patient est conséquente.

— l'**hétérogénéité** des données est un aspect souvent négligé, mais potentiellement plus complexe et problématique que la simple augmentation du volume. Les données nouvellement agrégées provenant de sources différentes, elles sont par conséquent de différents types : informations personnelles, antécédents médicaux, prescriptions médicamenteuses, évolution des états d'une maladie, parcours au sein des unités médicales, etc. Ces informations diverses sont naturelles pour les praticiens médicaux, et correspondent aux différentes facettes complémentaires d'un même profil patient. Leur diversité représente néanmoins une nouveauté à laquelle les méthodes d'analyse doivent s'adapter.

Le fonctionnement traditionnel des études menées sur les données médicales implique la collecte, pour un ensemble de patients, de leurs données pour un ensemble d'informations. L'analyse de ces données est menée par des outils statistiques et permet la validation, ou non, d'une hypothèse de départ formulée au début de l'étude. Cette hypothèse guide la sélection des données.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Par exemple, si l'hypothèse suppose que "les patients ayant les symptômes S_1 et S_2 développent la pathologie P", les données sélectionnées doivent inclure *a minima* les informations concernant S_1, S_2 et P pour l'ensemble des patients étudiés. Une fois ces données rassemblées, des techniques statistiques telles que la régression ou l'analyse factorielle sont employées afin de vérifier l'hypothèse formulée. Ces techniques donnent de très bons résultats sur des données bien préparées et de taille assez réduite, avec typiquement quelques centaines de patients et quelques dizaines d'attributs. Elles montrent cependant leurs limites lorsque les données sont plus volumineuses ou mal formatées. Les deux aspects de l'évolution mentionnée ci-dessus (volume et hétérogénéité) permettent d'élargir cette approche traditionnelle, et rendent possibles d'autres types de questionnement sur les données :

— l'augmentation du nombre de patients autorise des recherches plus exploratoires parmi les données. Plutôt que de chercher à valider une hypothèse, il devient possible de rechercher, dans l'ensemble des données, des corrélations entre certaines informations ou des groupes de patients présentant des attributs communs. Ces corrélations peuvent permettre de formuler de nouvelles hypothèses.

— de manière corollaire, il est possible d'utiliser un même jeu de données pour différents cas d'étude, si celui-ci contient l'ensemble des patients et des informations disponibles. Ce processus de réutilisation des données (ou *data reuse*) permet de simplifier la phase de préparation et de sélection des données, qui n'a plus besoin d'être faite à chaque nouveau cas d'étude.

— l'augmentation du nombre de patients implique aussi le plus souvent une fiabilité accrue des modèles explicatifs ou prédictifs construits à partir de l'analyse des données.

— l'intégration de données provenant de sources plus diverses permet également d'améliorer la qualité des modèles. En considérant l'ensemble des informations disponibles pour chaque patient, on s'autorise à examiner toutes les raisons possibles pour expliquer un événement médical, par exemple. Ainsi, il est possible de trouver ou confirmer des liens entre des informations en apparence sans rapport entre elles, telles que des comorbidités ou facteurs aggravants.

Cette évolution conjointe des données et des techniques disponibles permet donc l'émergence d'une nouvelle approche, basée sur la *fouille de données*. La Figure 1.3 représente ces deux approches de manière plus concise. Il est important de souligner que celles-ci sont complémentaires, leurs buts respectifs étant différents.

L'approche par fouille de données est plus flexible et permet de s'adapter à des problématiques plus variées. En effet, il arrive qu'il soit difficile d'établir une hypothèse *a priori*, surtout si les données sont massives et/ou moins bien connues. L'objectif le plus important pour les hôpitaux est d'optimiser la prise en charge de chaque patient. Cet objectif assez générique se traduit en pratique par différentes catégories de problèmes auxquelles des outils d'analyse de données doivent pouvoir répondre :

— profilage de patients : des groupes de patients aux profils similaires sont créés ; la classification d'un nouveau patient dans un des groupes existants peut permettre de mieux orienter sa prise en charge.

— prédiction : en fonction des informations disponibles et de cas précédents, un outil de prédiction permet d'estimer le risque qu'a un patient de développer une complication dans le futur, par exemple.

— complétion de données : comme nous le verrons par la suite, les données hospitalières sont souvent très *creuses*, c'est-à-dire que peu d'informations sont disponibles pour chaque patient (pour différentes raisons). Un outil de classification peut permettre d'inférer les

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

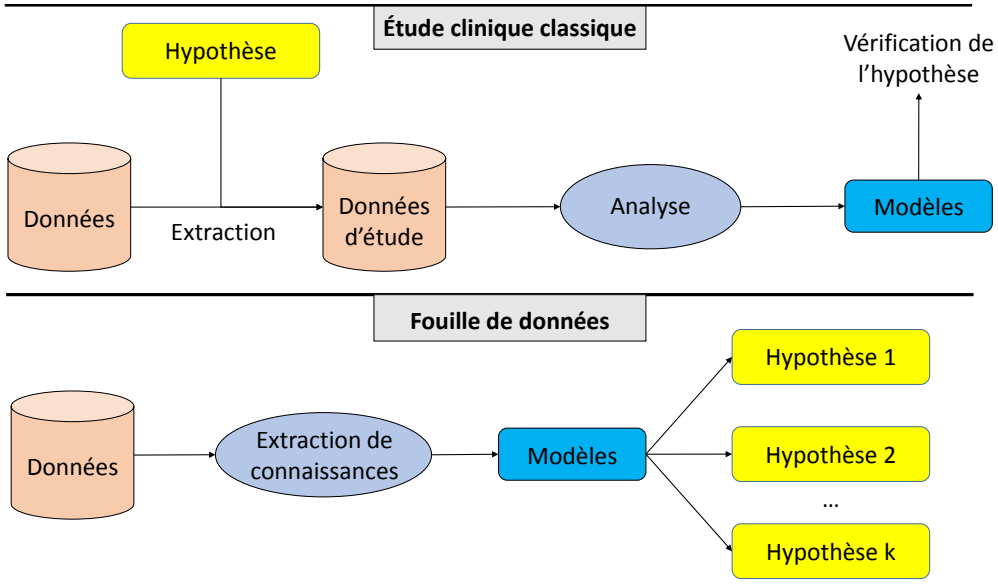


FIGURE 1.3 – Deux approches pour l'analyse de données : validation d'hypothèse et approche plus exploratoire

données manquantes si celles-ci sont importantes, par exemple pour l'inclusion dans un essai clinique.

— explication : certains outils de classification/prédiction génèrent des modèles facilement interprétables par un humain, tels que les arbres de décision ou règles de classification. Dans ce cas, ces modèles peuvent permettre d'expliquer un événement en identifiant les éléments qui y mènent le plus souvent.

1.1.5 Spécificités des données hospitalières

Les données hospitalières présentent plusieurs spécificités qu'il est important de détailler ici. En effet, celles-ci constituent autant de problèmes qu'il est nécessaire de considérer afin de proposer des outils d'analyse de données adaptés à ce contexte. Par conséquent, ces spécificités constituent la principale motivation pour les méthodes proposées dans ce travail.

— volume des données. Cette caractéristique n'est pas uniquement propre aux données médicales, mais elle n'en pose pas moins certains problèmes. Comme mentionné précédemment, un hôpital peut rapidement se retrouver avec plusieurs dizaines de milliers de patients dans ses archives. De plus, un patient peut être défini par un nombre très élevé d'attributs. Par exemple, le système PMSI associe à chaque acte et diagnostic un code unique. Au total, il existe plus de dix mille codes PMSI, donc autant d'attributs pouvant servir à caractériser un patient.

— données *creuses*. Corollaire de l'aspect précédent : un patient peut être caractérisé par plusieurs milliers d'informations différentes, mais son parcours hospitalier n'implique d'un nombre réduit d'informations. La plupart des séjours sont décrits par quelques dizaines de codes tout au plus, parmi les milliers de codes PMSI possibles. La matrice de données, où une ligne représente un patient et une colonne un attribut (code ou autre), est par

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

conséquent extrêmement creuse, avec un immense majorité de données manquantes. Ainsi, il n'est pas rare de retrouver moins de 1% de données effectivement renseignées dans un jeu de données. De plus, il serait abusif de considérer que l'absence d'une donnée implique que l'événement est absent. Par exemple, l'absence d'information pour l'attribut *hypertension* signifie uniquement que l'attribut (ici, la tension) n'a pas été mesuré. Par conséquent, il serait inexact de considérer par défaut toutes les données manquantes comme *négatives*.

— données déséquilibrées. Cet aspect est surtout notable pour les problèmes de classification. On parle de déséquilibre lorsque la proportion de cas positifs et négatifs pour l'attribut à prédire est notablement différente de 50%. Or, la plupart des événements médicalement intéressants à prédire ont une occurrence très faible. Une maladie ou affection donnée se produit presque toujours chez moins de 10% des patients, et le plus souvent moins d'1%. Par exemple, une maladie fréquente telle que le diabète a un taux d'occurrence de 4.7% de la population française en 2013.

— données incertaines. Dans un contexte médical, l'absence d'une information est ambigue et peut avoir deux significations différentes. Dans la plupart des cas, si aucune information n'est présente pour une maladie chez un patient, cela signifie que cette maladie est absente. Cependant, cela peut également signifier que la maladie est présente mais qu'elle n'a pas encore été détectée. De même pour les prescriptions ou actes médicaux, leur absence peut correspondre au cas où ils ont été réalisés, mais l'information n'a pas été enregistrée dans le système. Cette incertitude sur le codage induit entre autres une absence de réelle négation dans les données. Cela pose un problème considérable pour de nombreuses méthodes de classification/prédiction, qui apprenent souvent à partir d'ensembles de cas positifs et négatifs avérés.

— temporalité. La plupart des données hospitalières ont une importante composante temporelle : les actes, diagnostics et prescriptions sont effectués à un moment précis (date et heure), les affections peuvent évoluer au cours du temps (type, niveau de sévérité, etc.), des mesures biologiques sont souvent effectuées plusieurs fois au cours d'un séjour, entre autres. De plus, comme nous l'avons vu dans la Section 1.1.2, le parcours patient est constitué d'une séquence d'unités médicales, et est donc intrinsèquement temporel. Cet aspect temporel apporte un complément d'information aux données, et son exploitation optimale représente l'un des défis les plus notables associés aux données hospitalières.

— hétérogénéité. Les données provenant de différentes sources, il est naturel qu'elles prennent différents types : numériques, binaires, symboliques, etc. Exploiter correctement plusieurs types de données est un problème de taille et une limite de nombreux outils de fouille de données actuellement.

Dans la section suivante, nous détaillons le projet de recherche qui sert de cadre aux travaux réalisés dans cette thèse, ainsi que différents cas d'étude concrets sur les données médicales dont nous venons d'exposer les particularités.

1.2 Projet ANR ClinMine

Le cadre général de cette thèse est directement lié au projet ANR ClinMine. Cette section présente donc le projet dans son ensemble, les différents partenaires impliqués et les cas d'étude qu'il englobe.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

1.2.1 Présentation générale du projet

ClinMine est le projet ANR-13-TECS-0009 du programme TecSan (Technologies de la Santé) de l'année 2013, financé par l'Agence Nationale de la Recherche. L'objectif de ce projet est d'optimiser la prise en charge des patients à l'hôpital. Une meilleure prise en charge d'un patient est souvent synonyme d'une prise en charge plus adaptée à ce patient. Au-delà de l'amélioration pour le patient, une meilleure prise en charge présente de nombreux bénéfices pour l'hôpital : traitement plus rapide et moins coûteux, occupation des lits réduite, prévention de complications, etc. Ce projet est porté par un ensemble de partenaires issus de différents domaines :

— l'équipe de recherche DOLPHIN, du laboratoire CRIStAL - Université Lille 1. Cette équipe est spécialisée en modélisation et en résolution de problèmes d'optimisation combinatoire, avec un intérêt particulier pour une approche par optimisation des problèmes de fouille de données.

— l'équipe de recherche MODAL, du centre INRIA Lille-Nord Europe. Celle-ci travaille sur la conception de modèles génératifs probabilistes pour l'analyse de données et l'apprentissage statistique de données potentiellement hétérogènes.

— l'équipe d'accueil EA2694 "Santé Publique : épidémiologie et qualité des soins", de l'Université Lille 2 et rattachée à la Faculté de Médecine. Il s'agit d'une équipe pluridisciplinaire dans les domaines des biostatistiques, de l'informatique médicale, de l'économie de la santé et de l'évaluation ergonomique.

— l'équipe d'accueil EA1046 "Maladie d'Alzheimer et pathologies vasculaires". Cette équipe regroupe des cliniciens neurologues et pharmacologues, ainsi que des spécialistes en imagerie médicale. Leur objectif est de mener des recherches sur les accidents vasculaires cérébraux et les maladies neurodégénératives, et sur leurs interactions dans la survenue de troubles cognitifs et de démences.

— la société Alicante, spécialisée dans la conception, la réalisation, l'intégration et la maintenance évolutive de systèmes d'information hospitaliers, avec un intérêt particulier pour les systèmes experts et l'analyse de données.

— le GHICL - Groupement des Hôpitaux de l'Institut Catholique de Lille. Le GHICL est un institut privé à but non-lucratif, qui rassemble plusieurs hôpitaux dans la région lilloise et qui assume une triple vocation de soins, d'enseignement et de recherche.

Le projet regroupe donc des partenaires de natures différentes et de compétences complémentaires. La Figure 1.4 représente l'organisation de la coopération entre partenaires autour des grands thèmes du projet.

L'EA2694 et le GHICL sont principalement en charge de la collecte et l'agrégation des données. Les équipes DOLPHIN et MODAL (respectivement de CRIStAL et de INRIA) se concentrent sur la conception et le développement de méthodes adaptées pour l'analyse de ces données. Les cas d'étude permettant de valider ces méthodes sont mis en place par le GHICL et l'EA 1046. Alicante intervient de manière transversale dans l'ensemble des étapes, et en particulier pour l'intégration des méthodes développées au sein d'outils fonctionnels.

Les travaux réalisés au cours de cette thèse s'inscrivent en premier lieu dans la partie *Datamining* du schéma présenté ci-dessus. L'objectif est de concevoir, d'implémenter et de valider des méthodes de fouilles de données adaptées au contexte hospitalier. Inscrire ces travaux dans le cadre plus large du projet ClinMine permet, entre autres, de disposer de données réelles et de cas d'étude concrets. Ceux-ci aident à la compréhension des enjeux et difficultés du problème. Ainsi, les travaux de cette thèse se situent principalement dans la partie *Exploitation des données* présentée sur la Figure 1.2, mais touchent occasionellement à d'autres étapes du circuit des

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

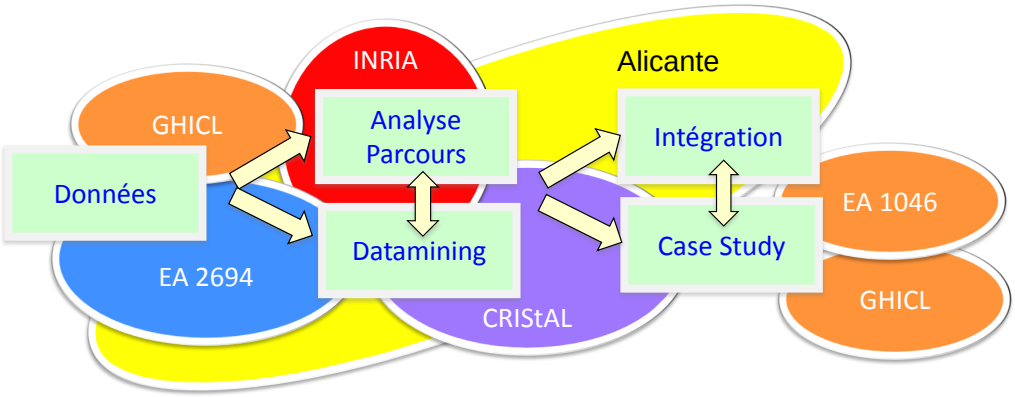


FIGURE 1.4 – Organisation du projet ClinMine - tâches et partenaires

données, comme nous le verrons par la suite. Afin d'illustrer les objectifs du projet de manière plus concrète, nous présentons ici deux des cas d'étude qui en font partie.

1.2.2 Cas d'étude 1 - optimisation des trajets patients

L'un des cas d'étude concerne l'**optimisation des trajets patients**. Par *trajet patient*, on entend, dans le cadre de ce projet, la séquence des différents services (appelés Unités Médicales, ou UM) par lesquelles passe chaque patient lors d'un séjour à l'hôpital. Optimiser ces trajets signifie chercher à orienter chaque patient vers le trajet le plus adapté à son profil. Ce problème est générique et englobe une grande variété de problèmes plus précis. Nous nous intéressons ici à trois de ces problèmes : la prise en charge en ambulatoire, la recherche de séquences d'Unités Médicales et la prédiction de codes PMSI. A noter que seul le premier problème fait partie des objectifs initiaux de ClinMine. Cependant, nous présentons également les deux autres car ils sont apparus au cours du travail et qu'ils présentent un intérêt notable pour plusieurs partenaires du projet (notamment Alicante et le GHICL).

1.2.2.1 Prise en charge en ambulatoire

Dans ce cas d'étude, nous nous concentrons sur les interventions chirurgicales. L'hospitalisation d'un patient pour une intervention peut se faire sur une journée, ou impliquer une ou plusieurs nuits passées à l'hôpital (avant ou après l'intervention). Dans le premier cas, on parle d'une *hospitalisation de jour* (HdJ) ou *ambulatoire*, dans le second cas d'une *hospitalisation complète* (HC). Une prise en charge en HdJ présente plusieurs avantages, tant pour le patient que pour l'hôpital :

— une diminution du temps passé en milieu hospitalier permet de réduire les risques d'infections nosocomiales, c'est-à-dire contractées dans un établissement de soins. La chirurgie ambulatoire tend également à promouvoir des méthodes d'opération moins invasives, qui réduisent également ces risques.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

— des enquêtes montrent que les patients pris en charge en HdJ présentent un taux de satisfaction plus important. L'HdJ est également considérée par les personnels soignants comme préférable pour certaines populations plus fragiles telles que les enfants et les personnes âgées.

— un hôpital dispose d'une quantité limitée de chambres et de lits. Par conséquent, il est naturel de chercher à minimiser l'occupation des lits afin de pouvoir accueillir davantage de patients sur une même période de temps. Cette recherche d'efficacité ne saurait néanmoins compromettre la qualité des soins apportés, qui demeure la priorité.

— un point davantage ouvert à la discussion concerne les économies réalisées lors d'une HdJ par rapport à une HC. Il est attendu qu'une HdJ soit moins coûteuse car mobilisant moins de ressources, mais cet aspect est difficile à mesurer et n'a donc pas été établi formellement à ce jour.

Compte tenu de ces avantages, il existe une tendance générale des hôpitaux en France à chercher à accroître le taux de prise en charge en HdJ. Cette tendance est soutenue par les organismes gouvernementaux, qui établissent des objectifs chiffrés pour ce taux de prise en charge. Ainsi, en 2015, la Direction Générale de l'Offre de Soins (DGOS) et le Secrétariat Général des Ministères Chargés des Affaires Sociales ont publié une directive fixant à 66.2% l'objectif de prise en charge en HdJ à l'horizon 2020. Toutes les opérations ne peuvent être réalisées en HdJ, mais les progrès des techniques chirurgicales et de récupération post-opératoire étendent progressivement son champ d'action à des opérations plus lourdes (pose de prothèse complète, par exemple). Le taux de prise en charge en HdJ au niveau national était de 32.3% en 2007 et de 37.7% en 2010, mais demeure inférieur à celui d'autres pays européens. Au GHICL, le taux d'HdJ en 2010 était de 37.4%, et l'objectif est d'atteindre 44%. Afin d'augmenter le taux d'HdJ, l'une des pistes explorées consiste à rechercher les séjours qui auraient dû se dérouler en HdJ, mais pour lesquels le patient est resté une nuit ou plus à l'hôpital. En particulier, il est intéressant de chercher les causes possibles de ces situations. Plusieurs hypothèses existent pour expliquer cela, et il est attendu qu'une approche plus factuelle d'analyse de données permette de confirmer ou non ces hypothèses, ainsi que de présenter éventuellement d'autres explications possibles.

1.2.2.2 Recherche de séquences d'Unités Médicales

Un autre problème dans l'optimisation des trajets patients concerne la recherche de séquences d'UM communes à des ensembles de patients. Plus précisément, nous nous intéressons à des groupes de patients ayant un profil similaire, c'est-à-dire présentant des similarités pour un sous-ensemble de leurs informations (par exemple, un ensemble de codes PMSI communs). Nous cherchons à déterminer dans quelle mesure les patients ayant un profil similaire suivent un parcours de soins (trajet patient) identique. Cela permet de répondre à deux problématiques :

— tout d'abord, il est important pour un hôpital que les parcours de soins soient similaires pour des patients ayant des profils semblables. En effet, un hôpital se doit de garantir l'égalité dans l'accès aux soins, afin que chaque patient puisse bénéficier du meilleur traitement possible.

— de manière plus concrète, rechercher les dissimilarités dans les parcours de soins pour des patients similaires permet à un hôpital d'identifier de possibles erreurs de fonctionnement interne. Celles-ci peuvent se situer au niveau de la prise en charge (un patient orienté par erreur dans un service non-adapté) ou de la saisie (l'enregistrement d'une UM s'est faite de manière erronnée).

Pour ce cas d'étude, la collaboration et le dialogue avec les praticiens hospitaliers est indispensable afin de déterminer si les groupes de patients similaires sont cohérents, et d'expliquer les

© 2017 Tous droits réservés. lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

dissimilarités retrouvées au niveau des parcours de soins. La Figure 1.5 résume les différentes étapes souhaitées du processus : recherche de groupes de patients, comparaison des parcours au sein de ces groupes, présentation et explication des dissimilarités observées.

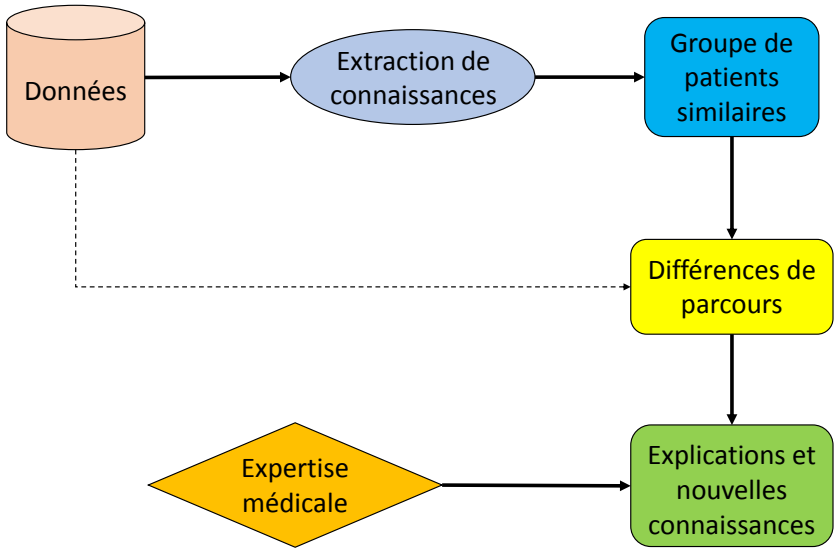


FIGURE 1.5 – Processus proposé pour le cas d'étude Recherche de séquences d'Unités Médicales

1.2.2.3 Prédiction de codes PMSI

Nous cherchons ici à automatiser en partie le travail de codage d'un séjour, en proposant une liste de codes pouvant décrire le séjour à partir des autres informations disponibles. Comme mentionné dans la description du PMSI, les codes pour chaque séjour doivent être saisis par les praticiens, et cette opération est coûteuse en temps et facilement sujette à des inexactitudes. Ainsi, proposer des codes probables pour un séjour permettrait d'accélérer le codage et de faciliter son uniformisation. Pour cela, nous nous intéressons particulièrement aux *courriers de sortie*. Ceux-ci sont rédigés à la fin de l'hospitalisation d'un patient et permettent sa sortie. Ils sont dictés ou rédigés par le praticien, qui y résume le motif d'hospitalisation, le processus de soins et les recommandations pour un éventuel suivi (prescriptions notamment). Ces courriers contiennent donc du texte libre, non-structuré, qui est par conséquence difficile à exploiter de manière automatique mais recèle souvent des informations précieuses sur les séjours. Afin de valoriser ces courriers, nous appliquons une phase d'*enrichissement* (voir Figure 1.2). Nous appliquons un outil de traitement automatique du langage naturel, qui permet d'extraire les différents *termes* présents dans un courrier et de compter leur nombre d'occurrences.

Afin de faciliter l'étude au vu de l'accessibilité des données, nous nous plaçons dans une situation *a posteriori*, c'est-à-dire lorsque l'intégralité des données pour un séjour sont connues (informations patient, codes PMSI et termes extraits des courriers). De cette manière, nous proposons de valider tout d'abord l'approche en générant, à partir d'un corpus de données établi, un ensemble de règles proposant des codes PMSI lorsque certains termes sont présents dans le courrier. Cela permet d'évaluer la pertinence de cette approche de deux manières complémentaires :

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

— étant donné que nous sommes dans le cas *a posteriori* et que toutes les données sont disponibles, nous pouvons comparer les codes PMSI prédits par les règles (donc à partir des termes des courriers), avec ceux renseignés effectivement dans les données.

— nous cherchons à générer des *règles d'association*, de la forme "SI {termes courrier} ALORS {codes PMSI}". L'objectif est que ces règles soient compréhensibles par un humain afin de favoriser leur validation, si possible par des experts médicaux.

1.2.3 Cas d'étude 2 - inclusion dans les essais cliniques

Un second cas d'étude inclus dans ClinMine concerne **l'optimisation de l'inclusion dans les essais cliniques**. Un essai clinique est une procédure consistant à étudier un ensemble de patients volontaires, afin d'acquérir de nouvelles connaissances ou de valider une nouveau traitement, médicament ou méthode de diagnostic. Cette procédure est très encadrée d'un point de vue légal, afin de garantir la sécurité des participants et la validité des résultats. L'essai clinique succède à une phase d'essais non-cliniques, et inclut plusieurs étapes : essais sur personnes saines, validation par rapport à la méthode standard, détermination des doses optimales, etc. L'inclusion d'un patient dans un essai clinique se fait sur la base du volontariat, mais doit respecter rigoureusement deux ensembles de contraintes :

— les **critères d'inclusion** sont des conditions que le patient doit remplir afin de pouvoir être inclus dans l'essai. Par exemple, si l'essai concerne un traitement pour le diabète, un critère d'inclusion évident est que les patients doivent souffrir de diabète. Chaque patient doit remplir tous les critères d'inclusion afin de pouvoir participer à l'étude.

— les **critères d'exclusion** sont des conditions qui disqualifient automatiquement un patient pour la participation à l'étude. Ces critères sont le plus souvent des cas amenant des risques pour le patient, ou risquant de fausser les conclusions de l'étude. Par exemple, une allergie à l'un des composants d'un traitement forme un critère d'exclusion évident. Un patient ne doit remplir aucun des critères d'exclusion pour pouvoir participer à l'étude.

La Figure 1.6 présente un exemple de critères d'inclusion et d'exclusion pour un essai clinique portant sur un traitement pour la maladie d'Alzheimer.

Critères d'inclusion	Critères d'exclusion
Âge compris entre 50 et 85 ans	Présence d'une autre pathologie importante ou systémique (antécédents d'AVC, épilepsie, cancer, etc.)
Diagnostic clinique de la maladie d'Alzheimer	Présence de pacemaker ou de prothèse métallique
Stade débutant de la maladie, ou troubles cognitifs légers	Variation des doses de traitements médicamenteux durant ou peu avant (<3 mois) l'étude
Être accompagné d'un proche prêt à s'impliquer dans l'étude	

FIGURE 1.6 – Critères d'inclusion et d'exclusion pour un essai clinique sur la maladie d'Alzheimer

Améliorer l'inclusion de patients dans les essais cliniques est important pour les hôpitaux, et ce pour plusieurs raisons :

— Tout d'abord, lles essais cliniques amènent le plus souvent à des publications médicales qui bénéficient aux établissements d'accueil où ces essais sont menés (en termes de subventions

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

et de notoriété). Comme nous l'avons vu, les hôpitaux ont souvent une vocation à la fois de soins et de recherche, et les essais cliniques forment un lien fort entre ces deux aspects.

— De plus, s'il s'avère impossible de recruter suffisamment de patients pour un essai, celui-ci doit être abandonné. Or, l'abandon d'un essai clinique faute de recrutements est coûteux (en temps et en argent) pour un hôpital.

— Enfin, pour les patients volontaires, participer à un essai clinique implique une prise en charge plus approfondie et souvent de meilleure qualité., ce qui se traduit par une plus grande satisfaction des patients.

Augmenter l'inclusion dans les essais cliniques est donc une priorité pour les services de recherche clinique, et bénéficie à la fois à l'hôpital et aux patients concernés. Le problème est que les informations sur les critères d'inclusion et d'exclusion ne sont pas toujours renseignées pour tous les patients. Comme nous l'avons signalé plus haut, toutes les informations concernant un patient ne sont pas toujours disponibles dans le système informatique d'un hôpital. Une information manquante peut signifier l'absence du fait que celle-ci concerne (le cas le plus fréquent), mais également un état non-diagnostiqué, ou connu mais non-renseigné dans le système. Cela implique parfois un manque d'information pour un critère d'inclusion ou d'exclusion, qui empêche de considérer des patients qu'il serait possible d'inclure dans un essai. Afin de lever cette incertitude, il est possible d'appliquer des méthodes de classification supervisée. Dans ce cas, la méthode consiste à déterminer la présence ou non du critère manquant à partir des autres informations disponibles pour le patient. De cette manière, il est possible de proposer un ensemble de patients susceptibles de remplir les conditions nécessaires à l'inclusion dans un essai. Pour ceux-ci, on ne dispose pas de toutes les informations sur les critères (inclusion et exclusion), mais un outil de classification indique qu'ils ont de bonnes chances de remplir ces critères, compte tenu de leur profil. Ces patients potentiels sont ensuite soumis pour examen au responsable du recrutement de l'essai, qui doit déterminer si ceux-ci remplissent les conditions nécessaires. Cela permet de filtrer effectivement un large groupe de patients pour ne proposer que ceux qui ont le plus fort potentiel d'inclusion. L'outil Opcyclin, développé par Alicante et installé notamment au GHICL, permet de mener ce processus de proposition de patients potentiels. L'un des objectifs des travaux de cette thèse est d'améliorer le moteur de classification au cœur d'Opcyclin afin que celui-ci gère des données plus variées.

1.2.4 Contrainte sur l'interprétabilité des résultats

Une contrainte additionnelle, posée par la société Alicante, est que les méthodes de fouille de données doivent produire des résultats facilement interprétables par un humain. Cela permet tout d'abord une utilisation plus rapide des résultats, dans le contexte d'un outil déployé dans un hôpital. Cela permet également une validation plus aisée par les experts médicaux des modèles construits par les outils d'analyse de données. Un retour direct sur les résultats produits permet de corriger ou d'adapter les méthodes en conséquence. Disposer de résultats dans un langage proche de celui utilisé par les experts médicaux permet de comparer aisément ces résultats avec leurs connaissances actuelles. Par exemple, un classifieur basé sur les SVM (Support Vector Machines) [38] est de type *black-box*, c'est-à-dire qu'un tel système apprend en interne un ensemble de paramètres, qu'il est difficile de traduire de manière intelligible. A l'inverse, un arbre de décision [137] utilise directement les différents attributs présents dans les données, et imite le processus de décision d'un humain. Le second serait donc mieux adapté que le premier pour nos besoins. Il serait également préférable que les méthodes développées ne nécessitent pas de configuration trop complexe ni de nombreux paramètres à fixer pour chaque nouveau cas d'étude. Pour cette raison, nous privilégions les méthodes robustes, capables de donner rapidement de bons résultats sur un large ensemble de jeux de données différents. Enfin, il est également souhaitable de mener une

© 2017 Tous droits réservés. lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

réflexion sur la représentation des résultats, qui est la dernière étape de la Figure 1.2. Disposer d'une représentation visuelle des résultats permet de concevoir des outils d'analyse de données plus proches des utilisateurs finaux. Dans la section suivante, nous dressons un état des lieux des outils de fouille de données existants afin de mieux nous positionner dans ce contexte.

1.3 Etat de l'art - outils de fouille de données

Compte tenu de l'importance croissante du domaine, il existe de nombreux outils et logiciels de fouille de données, en libre accès ou propriétaires. Parmi les plus connus, nous pouvons citer KNIME [18], qui inclut un ensemble de méthodes de préparation et d'analyse de données, et de présentation des résultats. Ce logiciel est largement utilisé, notamment pour des applications biologiques ; il permet de gérer des volumes importants de données et autorise l'utilisateur à implémenter ses propres composants. Cette modularité est partagée par la plupart des plateformes de fouille de données reconnues, notamment Weka [72], Orange [50], MLPACK [41], Scikit-learn [131], KEEL [6] ou le langage R et ses nombreux packages de fouille de données. Ces plateformes proposent une variété de méthodes et laissent l'utilisateur choisir les composants du cycle de traitement des données, en offrant la possibilité d'en définir de nouveaux. D'autres plateformes sont plus spécialisées, se restreignant à un type de méthodes ou se concentrant sur un type de données. Par exemple, GATE [40] est destiné au traitement du langage naturel, OpenNN [118] se concentre sur les réseaux de neurones et Torch [35] sur les méthodes de *deep learning*.

Il convient de rappeler ici que l'objectif des travaux de cette thèse n'est pas de concevoir une plateforme de fouille de données complète et générique comme celles citées ici. Nous nous intéressons donc aux logiciels existants dans le but de voir s'il est possible d'en utiliser et/ou d'en étendre un aux problèmes qui nous occupent. A cet égard, la principale difficulté provient des données étudiées. Comme mentionné précédemment, les données médicales contiennent des types de données rarement pris en compte dans les logiciels existants (données temporelles notamment). Leur hétérogénéité (plusieurs types différents) et le caractère creux de la matrice de données amènent d'autres difficultés dans la représentation et le stockage. Les plate-formes existantes proposent bien des moyens de définir des algorithmes adaptés à ces données, mais il est difficile de modifier le mode de représentation des données, au risque de dérégler l'ensemble du logiciel (notamment la visualisation des résultats). Par conséquent, nous estimons qu'il est préférable dans ces conditions de nous concentrer sur la conception et le développement de méthodes *ad hoc* pour les données hospitalières. Par ailleurs, la société Alicante dispose d'une certaine expérience en matière de fouille de données et d'outils développés en interne. Notamment, le logiciel Opcyclin, mentionné précédemment, qui utilise le moteur de classification MOCA-I [90].

1.4 Contexte scientifique

Dans cette section, nous présentons un aperçu des diffférents domaines de recherche en lien avec la problématique de cette thèse. Cette présentation vise à permettre de mieux comprendre les objectifs des travaux présentés, et à former une introduction facilitant la compréhension des méthodes développées.

1.4.1 Fouille de données

La fouille de données, ou *data mining*, est une étape du processus plus global de *découverte de connaissances dans les bases de données* (ou *KDD* pour *Knowledge Discovery in Databases*). La Figure 1.7 montre les différentes étapes de ce processus de KDD :

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

— une étape de **sélection** est tout d'abord menée afin de former l'ensemble des données à étudier. Cette phase est d'autant plus importante que le système de bases de données est complexe. Dans le cas des données hospitalières par exemple, nous sélectionnons les données concernant les activités médicales et laissons de côté les données administratives.

— les données subissent ensuite un **prétraitement** afin d'en corriger les imperfections et de former un ensemble de données cohérent et exploitable par la suite. Cette étape peut être séparée en deux phases : une pour le prétraitement des données (complétion des données manquantes, correction des valeurs aberrantes, discrétisation, etc.) et une pour la transformation en un fichier exploitable.

— une fois cette préparation des données effectuée, la **fouille de données** peut être appliquée. Lors de cette étape, un ensemble de modèles sont générés à partir des données préparées.

— les modèles créés peuvent ensuite être **interprétés** et **évalués**. L'objectif ici est de traduire les modèles obtenus dans le langage propre au métier, et de mesurer leur pertinence. La traduction est plus ou moins complexe en fonction de l'interprétabilité des résultats. L'évaluation peut être menée selon des critères internes (qualité intrinsèque du modèle) et/ou externes (validation par un expert métier).

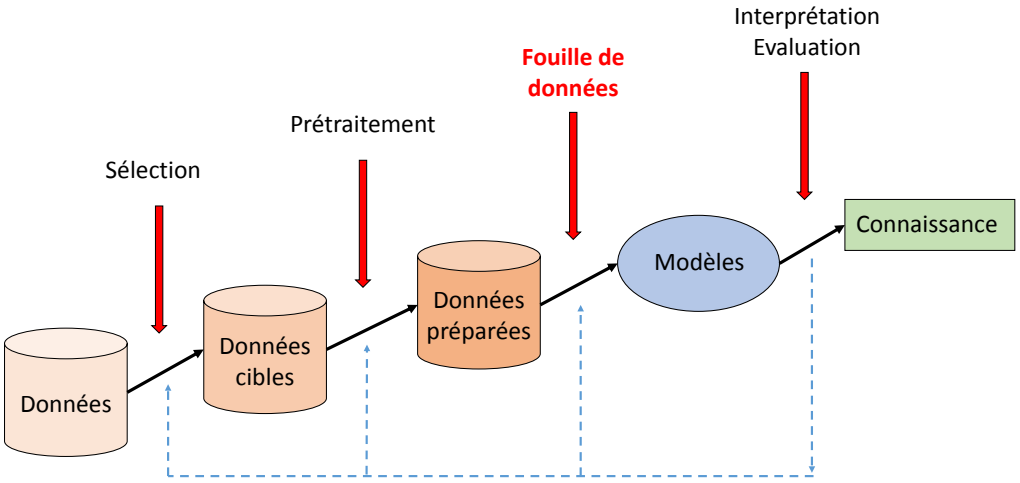


FIGURE 1.7 – Processus de découverte de connaissances dans les bases de données (KDD)

Le processus de KDD est également souvent représenté selon les termes du modèle CRISP-DM (CRoss-Industry Standard Process for Data Mining), qui inclut six phases principales :

— connaissance du métier

— connaissance des données

— préparation des données

— modélisation des données

— évaluation des modèles

— déploiement

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

CRISP-DM présente de nombreuses similitudes avec le processus précédent (voir Figure 1.7), et inclut également les phases de préparation des données, de génération des modèles et d'évaluation des résultats. Le point le plus intéressant est que CRISP-DM insiste sur la connaissance du métier, qui est indispensable à toutes les étapes du processus.

Nous nous intéressons ici à la phase qui consiste à générer un ensemble de modèles à partir des données. Deux types de modèles existent : descriptifs et prédictifs.

1.4.1.1 Modèles descriptifs

Les modèles descriptifs visent à organiser, simplifier et aider à comprendre l'information présente dans un ensemble de données. Comme leur nom l'indique, ils cherchent à décrire les données sous une forme plus compréhensible, en détectant par exemple des groupes homogènes ou des tendances générales. On retrouve dans cette catégorie de nombreuses méthodes statistiques, et notamment d'analyse factorielle. Celles-ci visent à identifier des variables cachées qui décrivent un sous-ensemble d'instances. On retrouve par exemple parmi celles-ci l'analyse en composantes principales et l'analyse des correspondances multiples.

Cette catégorie de modèles regroupe aussi les méthodes issues du domaine de l'apprentissage automatique (ou *machine learning*). Une des familles les plus connues concerne la classification non-supervisée. Cette classification consiste à répartir les instantes en plusieurs classes, de sorte que les instances présentes dans une même classe soient similaires entre elles et différentes de celles des autres classes. Par *non supervisée*, on entend que les classes ne sont pas connues à l'avance et sont découvertes au cours du processus de classification. Ces méthodes permettent de dégager des groupes homogènes de données à partir d'un ensemble qui ne l'est pas au départ, et peuvent donc servir comme première étape avant l'application d'autres méthodes de fouille de données. On retrouve donc dans cette catégorie les méthodes de classification non-supervisée (ou *clustering*), mais également de recherche de règles d'association et de *biclustering* (qui est essentiellement une classification double).

1.4.1.2 Modèles prédictifs

Les modèles prédictifs visent à expliquer ou prévoir la valeur d'une classe à prédire à partir des autres variables existantes. Dans le contexte médical, un exemple consiste à prédire la présence ou non d'une maladie à partir d'un ensemble d'informations disponibles au sujet d'un patient. Ces méthodes impliquent souvent une phase d'apprentissage, où les modèles prédictifs sont inférés à partir d'un ensemble d'instances pour lesquelles la valeur de la classe à prédire est connue. Une fois ces modèles appris, il est alors possible de les appliquer afin de prédire la valeur de la classe pour de nouvelles instances dont la classe est inconnue. Si la valeur à prédire est qualitative, on parle de *discrimination* ; si la valeur est quantitative, on parle de *régression*.

On retrouve parmi ce type de modèles un grand nombre de méthodes d'apprentissage automatique, telles que les arbres de décision, les règles de classification, les réseaux de neurones ou le raisonnement par cas. Il existe également des méthodes statistiques, telles que les modèles de régression, les modèles linéaires ou additifs généralisés, les réseaux bayésiens.

1.4.2 Optimisation

Les méthodes proposées dans cette thèse considèrent la fouille de données comme un problème d'optimisation, en accord avec les axes de recherche explorés par l'équipe DOLPHIN. Nous présentons ici une introduction au domaine de l'optimisation, notamment dans ses aspects combinatoire et multi-objectif.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

1.4.2.1 Optimisation et combinatoire

Formellement, un problème d'optimisation en mathématiques s'exprime de la manière suivante. Etant donnée une fonction $f : A \to \mathbb{R}$ définie sur un ensemble A à valeurs dans l'ensemble $\mathbb{R}$ des nombres réels, trouver un élément x_{min} de A tel que $f(x_{min}) \leq f(x) \forall x \in A$. On dit que l'on cherche à minimiser la fonction f. La définition est quasiment identique si on cherche à maximiser une fonction ; par convention, la minimisation est employée dans les définitions. La fonction f est souvent appelée *objectif* ou *fonction objectif*. L'ensemble A est appelé *ensemble admissible* (ou *réalisable*), les points $x \in A$ *points admissibles* du problème et le point x_{min} *solution optimale* (ou *globale*) du problème d'optimisation.

Pour un problème d'optimisation standard, A est donc un sous-ensemble de l'espace euclidien $\mathbb{R}^n$. On parle d'**optimisation combinatoire** lorsque A est un sous-ensemble de l'espace $\mathbb{N}^n$. Le terme d'optimisation combinatoire peut se comprendre si on considère que, dans ce cas, la solution du problème est une configuration pour l'ensemble des n attributs, où chaque attribut peut prendre un ensemble fini de valeurs possibles. Cette configuration peut être vue comme une *combinaison* de paramétrages pour chacun des attributs, comme une combinaison de chiffres forme un code secret par exemple. De manière plus formelle, étant donnés :

— un ensemble discret N,
— une fonction objectif $f : 2^N \to \mathbb{R}$,
— et un ensemble R de sous-ensembles de N dont les éléments sont appelés *solutions réalisables*,

un problème d'optimisation combinatoire consiste à trouver la solution réalisable S qui maximise la fonction objectif f, c'est-à-dire :

$$max_{S \subseteq N}\{f(S) : S \in R\}. \tag{1.1}$$

A la différence d'un problème d'optimisation général, l'ensemble des solutions possibles d'un problème d'optimisation combinatoire est discret et fini. Par conséquent, il est possible de trouver à coup sûr la solution optimale en essayant (c'est-à-dire en évaluant la fonction objectif) chaque solution. Cependant, cela n'est pas réalisable en pratique pour la plupart des problèmes intéressants ou réels. En effet, le nombre de solutions possibles augmente de manière exponentielle avec la taille du problème. Un exemple classique pour illustrer ce comportement est le problème dit du voyageur de commerce (ou TSP pour *Traveling Salesman Problem*), qui consiste à trouver le plus court chemin permettant de visiter un ensemble de n villes. Le nombre de solutions possibles pour ce problème est de $(n-1)!$. Pour une instance comprenant 25 villes, donc de taille apparamment restreinte, le nombre de solutions vaut donc $25! \approx 6.2 \times 10^{23}$. Le temps nécessaire pour évaluer l'ensemble des solutions atteint rapidement des proportions démesurées (plusieurs millions d'années) et est donc irréalisable en pratique.

1.4.2.2 Optimisation multi-objectif

De nombreux problèmes d'optimisation n'ont pas une seule fonction objectif, mais plusieurs qui doivent être optimisées simultanément. Dans le cas du TSP par exemple, un problème multi-objectif peut consister à la fois à minimiser la distance totale parcourue (comme dans la formulation originale) et à minimiser également le temps total du trajet. Formellement, un problème d'optimisation multi-objectif peut être défini comme

$$min(f_1(x), f_2(x), \ldots, f_k(x)) \text{ tel que } x \in X \tag{1.2}$$

où $k \geq 2$ est le nombre d'objectifs, l'ensemble des f_i telles que $i \in [1, k]$ sont les fonctions objectif, et X est l'ensemble des solutions réalisables. Dans la plupart des problèmes d'optimisation

© 2017 Tous droits réservés. lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

non-triviaux, il n'existe pas de solution optimale sur tous les objectifs, c'est-à-dire de solution qui minimise l'ensemble des k fonctions objectif. Pour cette raison, les méthodes d'optimisation pour les problèmes multi-objectifs visent à trouver un ensemble de *solutions de compromis*, qui minimisent un sous-ensemble $m < k$ de fonctions objectif. Afin d'expliquer plus formellement cette notion, nous introduisons la notion de *dominance*, ou *dominance Pareto*. Une solution x_1 domine une solution x_2 sur les fonctions objectifs à minimiser f_1, ..., f_k si :

$$\forall i \in 1..k, f_i(x_1) \leq f_i(x_2) \tag{1.3}$$

et

$$\exists j : f_j(x_1) < f_j(x_2). \tag{1.4}$$

En d'autres termes, x_1 domine x_2, noté $x_1 \succ x_2$, si x_1 est au moins aussi bonne que x_2 sur l'ensemble des k objectifs, et strictement meilleure sur au moins un des objectifs. Nous définissons un ensemble de *solutions de compromis* $x_1, x_2, \ldots, x_c$ comme un ensemble de solutions entre lesquelles il n'existe aucune relation de dominance, ou $\forall i, j \in [1, c] : x_i \not\succ x_j$, et tel que pour toute solution x_d extérieure à cet ensemble, il existe au moins une solution x_n de l'ensemble telle que $x_n \succ x_d$. Cet ensemble est appelé *solutions non-dominées*, ou *front Pareto*. La Figure 1.8 présente une illustration de ces définitions pour un problème d'optimisation bi-objectif ($k = 2$).

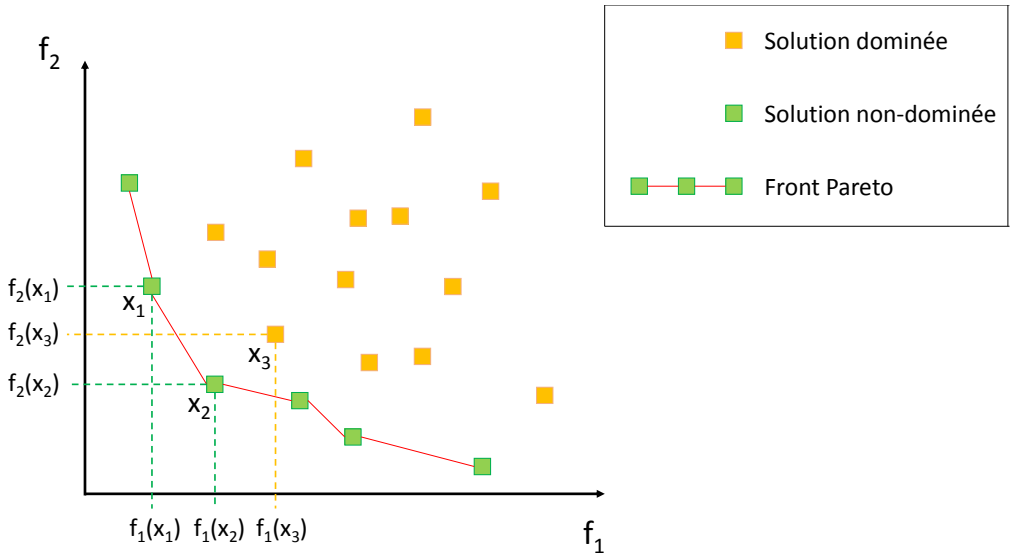


FIGURE 1.8 – Représentation d'un ensemble de solutions pour un problème d'optimisation bi-objectif

Sur cette figure, on observe que les deux critères de définition du front Pareto sont respectés. Par exemple, les solutions x_1 et x_2 (qui font partie du front Pareto) ne sont pas dominées entre elles car $f_1(x_1) < f_1(x_2)$ et $f_2(x_1) > f_2(x_2)$. De plus, x_2 domine x_3, qui ne fait pas partie du front. Ces deux conditions demeurent vraies pour l'ensemble des autres solutions représentées sur cette figure.

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

1.5 Conclusions

Dans ce chapitre, nous avons présenté les données hospitalières, et notamment leurs spécificités : volume, hétérogénité, données manquantes ou incertaines, aspect temporel, déséquilibre. Nous avons placé les travaux de cette thèse dans le cadre du projet ANR ClinMine, qui vise à concevoir de nouveaux outils de fouille de données adaptés à ces spécificités. L'intérêt de ces travaux est motivé par plusieurs cas d'étude ayant un réel intérêt pour les établissements hospitaliers : optimisation du parcours de soins, inclusion dans les essais cliniques. Nous avons vu que les outils existants ne permettent pas de traiter ces données tout en répondant aux contraintes d'interprétabilité imposées pour les résultats. Pour cette raison, nous choisissons de développer de nouveaux outils de fouille de données inspirés de l'expérience de la société Alicante dans le domaine, de l'expertise de l'équipe DOLPHIN pour la résolution par méthodes d'optimisation et de l'apport de l'ensemble des partenaires du projet, notamment pour l'aspect médical.

L'objectif des travaux de cette thèse est donc de concevoir un ensemble d'outils permettant de traiter les données hospitalières dans toute leur diversité, et capables de répondre aux différentes problématiques pouvant se poser sur ces données, et en priorité telles que définies dans les cas d'étude. Afin de parvenir à cet objectif, nous proposons deux outils distincts qui sont autant de contributions majeures de ce travail :

— dans un premier temps, nous nous intéressons au problème de la classification supervisée, qui englobe différentes applications : classification, prédiction, explication. Pour cela, nous proposons plusieurs extensions et améliorations à un système de classification existant, MOCA-I. Cet outil a été initialement conçu pour composer avec certains particularités des données médicales, notamment taille, déséquilibre et hétérogénéité basique. Nous apportons plusieurs améliorations à cette approche afin de gérer efficacement les données temporelles et numériques. La nouvelle approche, appelée MOSC, est validée sur plusieurs cas d'étude réels.

— dans un second temps, nous proposons une méthode pour l'exploration non-supervisée des données hospitalières. Celle-ci vise, par exemple, à trouver parmi ces données des groupes de patients présentant des caractéristiques similaires. Cela répond d'une manière différente à la problématique plus générale de classification de parcours patient. L'avantage de cette technique est sa nature exploratoire, qui permet potentiellement de découvrir des groupes intéressants et inattendus. Nous développons pour cela une méthode de biclustering capable de supporter les données hétérogènes et temporelles. Cette méthode fait également l'objet d'une validation sur un cas d'étude complet incluant des données hospitalières provenant de plusieurs sources.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Première partie

Classification supervisée pour données numériques et temporelles

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Chapitre 2

MOCA-I

Nous présentons dans ce chapitre les grandes lignes de l'algorithme MOCA-I (Multi-Objective Classification Algorithm for Imbalanced Data). Celui-ci a été développé dans des travaux antérieurs à cette thèse, afin de répondre au problème de la classification sur données massives et déséquilibrées [89, 90]. Il est donc nécessaire d'exposer le fonctionnement de cet algorithme, afin de comprendre les contributions que nous y avons apporté, qui seront détaillées par la suite. Dans une première section, nous présentons la modélisation adoptée par MOCA-I pour le problème de la classification sur données volumineuses et déséquilibrées. Dans une deuxième section, nous détaillons l'algorithme d'optimisation, basé sur une recherche locale multi-objectif. Enfin, nous discutons des mécanismes de choix d'une solution unique parmi l'ensemble des solutions de compromis contenues dans l'archive en fin d'exécution de MOCA-I.

2.1 Modélisation

Dans cette section, nous détaillons le problème de la classification afin de compléter l'aperçu qui en a été donné dans la partie Contexte (voir Section 1.4.1.1). Nous présentons ensuite le modèle employé par MOCA-I pour les règles de classification dans un contexte multi-objectif.

2.1.1 Généralités sur la classification

Le problème de classification en apprentissage automatique consiste à déterminer à quel ensemble appartient une observation, étant donné un ensemble d'observations utilisées pour construire le modèle. Par exemple, une application possible de cette tâche consiste à déterminer si un patient est atteint par une maladie compte tenu d'un ensemble d'informations dont on dispose sur lui (âge, poids, symptômes, mesures, etc.). Dans cet exemple, on cherche donc à prédire la *classe* "maladie", de type binaire (c'est-à-dire que sa valeur est "oui" ou "non") à partir des *attributs* correspondant aux informations disponibles pour ce patient. Ces attributs sont de différents types ; on distingue particulièrement les attributs quantitatifs des attributs qualitatifs. Les premiers (par exemple l'âge) prennent comme valeur un nombre, tandis que les seconds possèdent une liste de valeurs possibles restreinte et fixée. Les attributs binaires sont un type d'attributs qualitatifs, où cette liste est composée de deux valeurs possibles.

Une méthode de classification construit un modèle permettant d'attribuer à des observations, dont la classe est inconnue, la classe qui leur convient le mieux. Ce modèle est construit à partir d'un ensemble d'observations dont la classe est connue : il s'agit donc d'un problème d'apprentissage supervisé. Le processus se déroule en deux phases : apprentissage et évaluation. Dans la phase d'apprentissage, l'algorithme construit le modèle de classification à partir de cas connus.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Dans la phase d'évaluation, le modèle est appliqué à des cas dont la classe à prédire est inconnue. De nombreux types de méthodes de classification ont été développés ; ces méthodes peuvent être très différentes, mais leur point commun est d'utiliser le *vecteur d'attributs* d'une observation pour en déduire sa classe. Parmi ces méthodes, on distingue entre autres les classifieurs linéaires, les réseaux de neurones ou les arbres de décision. Celles-ci représentent le modèle sous diverses formes : arbres, règles, fonction, ensemble de paramètres. Dans la suite de ce travail, nous nous intéressons aux classifieurs sous forme de règles [122].

2.1.2 Classification à base de règles

Une règle de classification est de la forme *SI condition ALORS prédiction*. La partie gauche de la règle (*condition*) est composée d'un ensemble de *termes*, où chaque terme correspond à un test réalisé sur les attributs de l'observation dont on cherche à prédire la classe. Par exemple, la condition *âge* > 60 ET *hypertension* $=$ *oui* est constituée de deux termes ; si ces deux termes sont vérifiés dans une observation, alors la règle est activée et donne la prédiction contenue en partie droite de cette règle.

Plus formellement, on définit un *terme* comme étant formé de trois éléments :

— un **attribut** sur lequel le test est effectué

— un **opérateur**, par exemple $<$, $>$ ou $=$

— une **valeur**, quantitative ou qualitative

Cette représentation sous forme de règles présente au moins deux avantages majeurs pour le problème qui nous occupe. Tout d'abord, les règles de classification sont directement compréhensibles par l'utilisateur, étant donné que celles-ci sont exprimées en utilisant directement les attributs. Cela permet également de valider plus rapidement les résultats obtenus, un spécialiste métier pouvant rapidement déterminer si les règles générées ont un sens ou non. A l'inverse, les résultats produits par des méthodes telles que les classifieurs linéaires ou les réseaux de neurones, qui ajustent un ensemble de paramètres, sont moins facilement compréhensibles.

Un autre atout de cette représentation est son adéquation au problème posé par le déséquilibre présent dans les données à classifier. En effet, nous avons vu dans la Section 1.1.5 que les cas que nous cherchons à prédire sont souvent très rares dans les données (moins de 10%). Afin de répondre à ce problème, on se place dans un contexte de classification partielle, dans lequel on ne cherche à prédire que partiellement une classe. Cela signifie que les prédictions ne porteront que sur une partie des valeurs possibles. Par exemple, si la prédiction est un attribut binaire *Alzheimer*, alors un modèle de classification partielle ne génèrera que des règles de la forme *SI condition ALORS Alzheimer* $=$ *oui*. De cette manière, l'algorithme de classification se concentre sur les cas "intéressants" (positifs), et ce même si ceux-ci sont faiblement représentés dans les données. Un autre avantage de l'approche partielle est son incapacité à générer des règles contradictoires. Deux règles sont contradictoires sur une observation si elles prédisent chacune une classe différente pour cette observation. Il est impossible que cette situation se produise si toutes les règles prédisent la même valeur.

2.1.3 Aspect multi-objectif du problème

Afin de choisir le modèle de classification le plus performant, nous avons donc besoin de définir une mesure de la qualité d'une règle de classification. Ce problème est loin d'être trivial, et de nombreuses mesures ont été développées dans ce but ; un résumé de celles-ci est présenté dans une étude précédente [89]. Traditionnellement, les algorithmes de classification basés sur des règles

© 2017 Tous droits réservés. lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

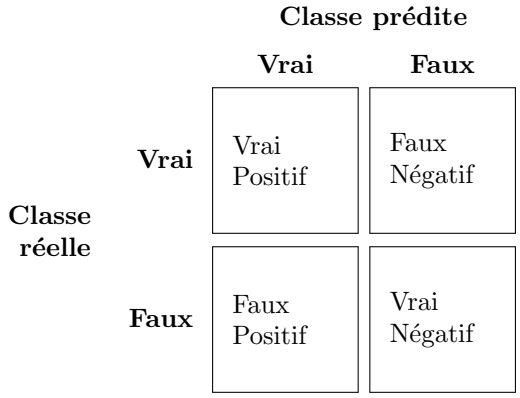


FIGURE 2.1 – Matrice de confusion

évaluent les solutions générées selon deux critères : exactitude et support. L'exactitude correspond à la proportion d'observations correctement classifiées par une règle, tandis que le support mesure la proportion des observations qui vérifient cette règle. Parmi les méthodes fonctionnant ainsi, on peut citer notamment l'algorithme *Apriori* [5]. Cependant, ces deux mesures, exactitude et support, présentent certains problèmes lorsque les données sont fortement déséquilibrées. En effet, si un événement intéressant ne se produit que dans 1 cas sur 1000, alors une règle prédisant que cet événement ne se produit jamais aura une exactitude de 99.9%. Une telle règle manque cependant tous les cas positifs, qui sont les plus intéressants. L'association de ces deux mesures ne paraît donc pas appropriée à notre problème.

Compte tenu de ces observations, il a été déterminé que les deux mesures les plus intéressantes dans ce cas sont la confiance et la sensibilité [89]. Celles-ci sont basées sur la matrice de confusion associée à la règle, et sont exprimées selon les formules présentées ci-dessous. On rappelle auparavant qu'une matrice de confusion, présentée dans la Table 2.1, regroupe les valeurs suivantes :

— Vrais positifs (ou TP : True Positives) : le nombre d'observations vérifiant à la fois la condition et la prédiction. Ce sont les cas positifs correctement prédits comme tels par la règle.

— Faux positifs (ou FP : False Positives) : le nombre d'observations vérifiant la condition mais pas la prédiction. Ce sont les cas qui ont été incorrectement classifiés comme positifs.

— Faux négatifs (ou FN : False Negatives) : le nombre d'observations vérifiant la prédiction mais pas la condition. Ce sont les cas positifs qui n'ont pas été identifiés par la règle.

— Vrais Négatifs (ou TN : True Negatives) : le nombre d'observations ne vérifiant ni la condition ni la prédiction. Ce sont les cas négatifs correctement prédits comme tels par la règle.

$$\textit{confiance} = \frac{TP}{TP + FP} \tag{2.1}$$

$$\textit{sensibilité} = \frac{TP}{TP + FN} \tag{2.2}$$

Différentes mesures permettant d'évaluer la qualité d'un classifieur sont construites à partir de la confiance, de la sensibilité, ainsi que des éléments de la matrice de confusion. En particulier, la f-mesure, aussi appelée F_1 *score* ou *F-score*, est obtenue en calculant la moyenne harmonique

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

de la confiance et de la sensibilité (présentées dans la Section 2.1.3), c'est-à-dire :

$$\textit{f-mesure} = 2 \times \frac{\textit{confiance} \times \textit{sensibilité}}{\textit{confiance} + \textit{sensibilité}} \tag{2.3}$$

Une autre mesure, appelée MCC, peut être quant à elle obtenue directement à partir de la matrice de confusion (également présentée en Section 2.1.3) :

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2.4}$$

Cette seconde mesure correspond à un coefficient de corrélation entre observation et prédiction. Elle est fortement corrélée avec la f-mesure, mais peut se révéler plus sensible que cette dernière, notamment du fait qu'elle admette des valeurs entre -1 et 1, et non 0 et 1.

MOCA-I est un algorithme d'optimisation multi-objectif qui cherche à optimiser ces deux objectifs pour les règles générées. Ainsi, le but est d'obtenir des règles prédisant correctement les cas positifs comme tels (bonne confiance) et identifiant la plus grande proportion possible de ces cas (bonne sensibilité). En d'autres termes, il s'agit d'identifier correctement tous les cas positifs.

Cet aspect multi-objectif peut se justifier par le fait que les deux objectifs sont différents, souvent antagonistes, mais aussi importants l'un que l'autre. Dans une telle situation, deux approches sont possibles. La première consiste à agréger les différents objectifs en une unique fonction, par le biais d'une pondération de chacun des objectifs. La fonction objectif nouvellement obtenue serait donc de la forme $f(x) = w_1 \times \textit{confiance}(x) + w_2 \times \textit{sensibilité}(x)$, où x est une solution à évaluer, et w_1 et w_2 sont les pondérations respectives des deux objectifs. Cette approche présente l'avantage de simplifier le problème en le ramenant à un problème d'optimisation mono-objectif, pour lequel de nombreuses méthodes existent. Cependant, l'efficacité d'une telle méthode dépend fortement d'un choix judicieux pour les pondérations, lequel est bien souvent assez arbitraire. Une autre approche est de considérer les divers objectifs séparément et de chercher à générer un ensemble de solutions de compromis, parmi lesquelles il faudra choisir la plus appropriée (ce choix pouvant être laissé à l'utilisateur ou mené de manière automatique en fin d'apprentissage). Ainsi, une solution de cet ensemble pourra avoir une confiance élevée mais une faible sensibilité, tandis qu'une autre présentera les caractéristiques inverses et qu'une troisième sera plus équilibrée. Cette approche est basée sur la dominance. On cherche à conserver un ensemble de solutions non-dominées entre elles (au sens de dominance Pareto), cet ensemble contient donc les solutions de meilleur compromis.

Il convient également de mentionner ici que le passage à un contexte multi-objectif peut, dans certains cas, améliorer les résultats d'une méthode d'optimisation. Ce passage peut se faire naturellement, ou en ajoutant à l'objectif initial un objectif additionnel, dit *auxiliaire* [100]. L'étude originale sur ce sujet concernait le problème du voyageur de commerce, mais d'autres travaux ont ensuite montré que ces résultats pouvaient être étendus à d'autres problèmes [86, 165]. Les études réalisées sur MOCA-I ont d'ailleurs confirmé l'intérêt d'une telle démarche dans le cas qui nous intéresse [90].

2.1.4 Représentation d'une solution

Avant de détailler l'algorithme d'optimisation utilisé, il est nécessaire de définir précisément la nature des solutions que celui-ci va rechercher. En effet, deux modélisations différentes sont possibles pour les règles de classification. La première, appelée Michigan, considère qu'une solution consiste en une règle unique. La seconde, Pittsburgh, considère qu'une solution est un ensemble de règles. C'est cette dernière approche qui a été employée dans l'algorithme MOCA-I. L'un des avantages de cette modélisation est de pouvoir générer des solutions plus riches, qui

© 2017 Tous droits réservés. lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

incluent plusieurs "causes" possibles pour une même prédiction. De plus, l'un des inconvénients habituellement associés à cette représentation, à savoir la génération de solutions contenant des règles contradictoires, est évité par l'emploi de la classification partielle décrite précédemment. De cette manière, on évite de devoir déterminer des systèmes de vote (majorité, pondération, etc.) pour déterminer la classe d'une observation.

Etant donné que toutes les règles ont la même prédiction, on considère qu'il suffit qu'une observation vérifie une seule règle d'une solution pour que cette observation soit classifiée comme ayant la prédiction en question. En d'autres termes, on peut voir une solution comme une règle de classification unique exprimée sous forme normale disjonctive, où chaque clause serait la partie gauche d'une des règles de la solution. Nous présentons ci-dessous un exemple de règle utilisant la représentation choisie (Pittsburgh) :

$$(\hat{a}ge > 55) \text{ OU } (tension > normale \text{ ET } diabète = vrai) \Rightarrow risque = vrai \tag{2.5}$$

Cette solution est un ensemble disjonctif de deux règles, et chaque règle est une conjontion de termes. Il est à noter que, dans un contexte de classification partielle, préciser la prédiction ($risque = vrai$) est superflu, puisque toutes les règles portent sur la même prédiction. Maintenant que nous avons vu comment représenter les solutions, nous présentons dans la partie suivante l'algorithme d'optimisation de ces solutions par recherche locale.

2.2 Voisinage et algorithme d'optimisation

2.2.1 Voisinage

Le voisinage d'une solution correspond à l'ensemble des solutions "proches" de celle-ci, c'est-à-dire les solutions qu'il est possible d'obtenir en modifiant "légèrement" la solution initiale. Il est donc nécessaire de définir les *opérateurs de voisinage*, les opérations permettant d'obtenir ces solutions voisines. L'algorithme MOCA-I utilise les opérateurs suivants :

- **add_rule** : ajoute une règle à la solution (ensemble de règles). Cette nouvelle règle comporte un unique terme formé à partir d'un attribut choisi au hasard.
- **remove_rule** : supprime une règle de la solution, s'il reste au moins une règle dans celle-ci après suppression.
- **add_term** : ajoute un terme (formé au hasard) à une des règles de la solution.
- **remove_term** : supprime un terme d'une règle, s'il reste au moins un terme dans la règle après suppression.
- **replace_term** : remplace un terme d'une règle par un autre terme utilisant le même attribut, mais une valeur et/ou un opérateur différent.

Pour illustrer l'effet de ces différents opérateurs, nous les appliquons à la solution suivante :

$$(\hat{a}ge > 55) \text{ OU } (genre = M \text{ ET } antécédents = vrai) \tag{2.6}$$

$add_rule \Rightarrow (\hat{a}ge > 55) \text{ OU } (genre = M \text{ ET } antécédents = vrai) \textbf{ OU } (\boldsymbol{tension > normale})$

$remove_rule \Rightarrow (\hat{a}ge > 55)$

$add_term \Rightarrow (\hat{a}ge > 55 \textbf{ ET } \boldsymbol{antibiotiques = vrai}) \text{ OU } (genre = M \text{ ET } antécédents = vrai)$

$remove_term \Rightarrow (\hat{a}ge > 55) \text{ OU } (antécédents = vrai)$

$replace_term \Rightarrow (\hat{a}ge > 55) \text{ OU } (genre \textbf{ = F} \text{ ET } antécédents = vrai)$

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

A noter que ces cinq solutions voisines ne forment pas le voisinage exhaustif de la solution initiale. Certains de ces opérateurs peuvent s'appliquer sur d'autres parties de la solution ; par exemple, *replace_term* peut également s'appliquer sur les deux autres termes existants. Ces solutions ne sont que des exemples de voisins qu'il est possible d'obtenir via les différents opérateurs utilisés.

2.2.2 Algorithme d'optimisation

L'algorithme utilisé pour l'optimisation des solutions est DMLS (Dominance-based Multi-objective Local Search) [116]. Celui-ci est une adaptation du mécanisme classique de recherche locale, par exemple le *Hill Climbing*, pour supporter plusieurs objectifs. DMLS a été montré aussi efficace que le meilleur algorithme d'optimisation multi-objectif connu dans la littérature, NSGA-II [47]. DMLS requiert moins de configuration de paramètres que NSGA-II, ce qui est un atout majeur dans le contexte qui nous intéresse. L'algorithme génère un ensemble de solutions de compromis réparties sur le *Front Pareto*. Le fonctionnement de DMLS est décrit dans l'Algorithme 1. L'algorithme maintient une archive contenant l'ensemble actuel de solutions de compromis. Cette archive est initialisée avec un ensemble de 100 solutions non-dominées ; le nombre a été choisi après expérimentations, présentées dans le travail original sur MOCA-I [89]. Idéalement, les solutions initiales doivent être formées de manière aléatoire, c'est-à-dire à partir d'attributs pris au hasard. Cependant, pour des jeux de données tels que ceux qui nous intéressent, avec plusieurs milliers d'attributs, cela n'est pas une bonne idée. En effet, une règle ainsi formée au hasard a très peu de chances d'être vraie ne serait-ce que pour une observation dans le jeu de données. De telles règles sont inutiles, et il ne paraît pas raisonnable d'appeler ce processus de formation de règles aléatoires jusqu'à ce qu'on obtienne par chance suffisamment de règles non-nulles. Par conséquent, l'algorithme utilise à la place une méthode de génération de solutions initiales à partir d'observations présentes dans les données. Cette méthode est décrite dans l'Algorithme 2. Les ensembles de règles (solutions) générés comportent au moins deux règles, et chacune de ces règles est valide pour au moins une observation dans les données. Une fois ces solutions générées, l'ensemble de solutions non-dominées est conservé et forme l'archive initiale. Ces solutions sont marquées comme "non-visitées" ; l'algorithme d'optimisation s'arrête lorsqu'il n'y a plus aucune solution "non-visitée" dans l'archive. Tant qu'il en reste au moins une, DMLS en choisit une (ou davantage) au hasard. Pour chacune des solutions choisies, l'ensemble des solutions voisines est généré et évalué. Chaque voisin non-dominé est ajouté à l'archive. Lorsque tout le voisinage d'une solution est exploré, celle-ci est marqué comme "visitée". A la fin de l'exploration, les solutions dominées sont supprimées de l'archive. L'algorithme se répète en utilisant la nouvelle archive, jusqu'à ce que la condition d'arrêt soit satisfaite. Différentes stratégies sont possibles quand au choix des solutions dans l'archive et à l'exploration du voisinage. La configuration la plus efficace est de choisir une solution unique dans l'archive, et de générer son voisinage de manière exhaustive [89].

2.3 Choix d'une solution unique

L'algorithme MOCA-I renvoie, en fin d'optimisation, une archive de solutions non-dominées pour les deux objectifs consistant à maximiser la confiance et la sensibilité des solutions (ensembles de règles). Dès lors, le problème de la sélection d'une solution unique se pose ; on souhaite obtenir un classifieur unique en fin de processus afin de pouvoir l'appliquer aux données à traiter. Cependant, déterminer la "meilleure" solution non-dominée implique naturellement de savoir ce qui rend une solution meilleure qu'une autre, et donc de disposer, en termes plus formels, d'une relation d'ordre total entre deux solutions. Dans cette partie, nous proposons diverses réponses possibles à cette question. Dans un premier temps, nous cherchons à déterminer des

© 2017 Tous droits réservés. lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Algorithme 1 : Fonctionnement de l'algorithme DMLS - recherche locale multi-objectif

```
archive ← solutionsInitiales(100);
while archive.getNbNonVisitees() > 0 do
    solutionVisitee ← selection(archive);
    resultats ← [];
    voisins ← solutionVisitee.genereVoisins();
    for V_n ∈ voisins do
        resultats.add(V_n);
    end
    solutionVisitee.visitee ⇐ true;
    for r ∈ results do
        for r_arch ∈ archive do
            if r > r_arch then
                archive.supprime(r_arch);
            end
            if r_arch > r then
                r.dominee ← true;
            end
        end
        if !r.dominee then
            archive.add(r);
        end
    end
end
```

Algorithme 2 : MOCA-I - génération des solutions initiales à partir d'observations positives

```
sélectionner une observation aléatoire Obs qui vérifie la prédiction;
créer une règle vide r;
for attributA ∈ Obs do
    r.ajouterTerme(A, " = ", Obs.valeur(A));
end
taille ← random(2, 9);
while r.taille() < taille do
    r.supprimeTermeRandom();
end
for i ∈ max(r.size()/3, 1) do
    r.ajouteTerme(getTermeAleatoire());
end
for i ∈ max(r.size()/3, 1) do
    r.supprimeTermeRandom();
end
```

critères supplémentaires (au-delà de la confiance et sensibilité) permettant de juger de la qualité

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

respective des solutions et de choisir la meilleure dans l'archive. Dans un second temps, nous explorons plusieurs méthodes de recombinaison des ensembles de règles visant à construire une nouvelle solution "optimale". Nous concluons cette section par une discussion de l'efficacité de ces mécanismes de sélection. Cette section vient compléter l'étude sur ce sujet réalisée dans le cadre des travaux menés sur l'algorithme MOCA-I original [89].

2.3.1 Choix dans l'archive

Une première méthode assez basique consiste à choisir la solution de l'archive ayant la meilleure valeur pour une mesure unique. Comme discuté précédemment, cela revient à transformer temporairement un problème multi-objectif en mono-objectif. La mesure utilisée peut être l'un des objectifs du problème d'optimisation, telle que la confiance. En effet, il ne paraît pas absurde de vouloir un ensemble de règles de confiance maximale pour un problème où le nombre de faux positifs (c'est-à-dire de cas négatifs ayant été prédits comme positifs) doit être minimal. Néanmoins, les solutions présentant une confiance très élevée concernent en général très peu d'individus, et les règles obtenues peuvent être par conséquent difficiles à généraliser car trop spécifiques, et décrivant trop littéralement un petit nombre d'individus. Ces solutions sont, de plus, souvent les plus sensibles au sur-apprentissage ; leurs performances peuvent se dégrader fortement lors de leur évaluation sur de nouvelles données.

Il est également possible d'utiliser des mesures uniques plus complexes. Par défaut, MOCA-I sélectionne la solution de l'archive présentant la meilleure *f-mesure* (sur l'ensemble d'apprentissage). Bien que toujours simple, cette méthode a été montrée plus performante que plusieurs autres dans l'étude précédente [89]. En fonction des besoins de l'utilisateur, d'autres mesures peuvent être employées : MCC, surprise, conviction, support, etc.

Une autre stratégie, davantage axée sur la réduction du sur-apprentissage, est de réserver une partie de l'ensemble d'apprentissage (souvent appelée *ensemble de validation*) à l'évaluation des solutions de l'archive. Dans cette méthode, chaque solution est évaluée sur l'ensemble de validation à la fin du processus d'optimisation [142]. La solution de meilleure qualité (par exemple, *f-mesure*) sur cet ensemble est choisie comme solution unique.

Une approche différente consiste à sélectionner un ensemble de solutions de l'archive, en laissant le choix à l'utilisateur de celle qui convient le mieux à ses besoins. Le but ici est de sélectionner un échantillon le plus représentatif possible de l'ensemble des solutions trouvées, mais également assez réduit pour être appréhendable par un utilisateur humain. Une méthode efficace est de rechercher la solution ayant la meilleure valeur pour l'un des deux objectifs du problème, avec une valeur minimale pour le second objectif. En faisant varier cette valeur minimale, on obtient un échantillon réparti sur l'ensemble du front Pareto. On peut, par exemple, rechercher les solutions de meilleure confiance en fixant une sensibilité minimale à 0.2, 0.4, 0.6 et 0.8. La Figure 2.2 illustre ce mécanisme : les solutions en rouge sont celles retenues et forment un échantillon représentatif de l'archive des solutions.

Une telle approche n'est cependant applicable que si les résultats de l'algorithme sont présentés directement à un expert métier, ce qui n'est pas toujours le cas. En effet, les résultats peuvent être directement utilisés par d'autres systèmes automatiques, auquel cas cette méthode n'est pas adaptée.

2.3.2 Construction d'une nouvelle solution

Etant donné qu'on dispose d'une archive complète de solutions (*i.e.* d'ensembles de règles), on cherche maintenant à exploiter davantage qu'une unique solution pour déterminer le classifieur optimal. Nous cherchons donc ici à combiner différentes solutions de l'archive afin d'obtenir une

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

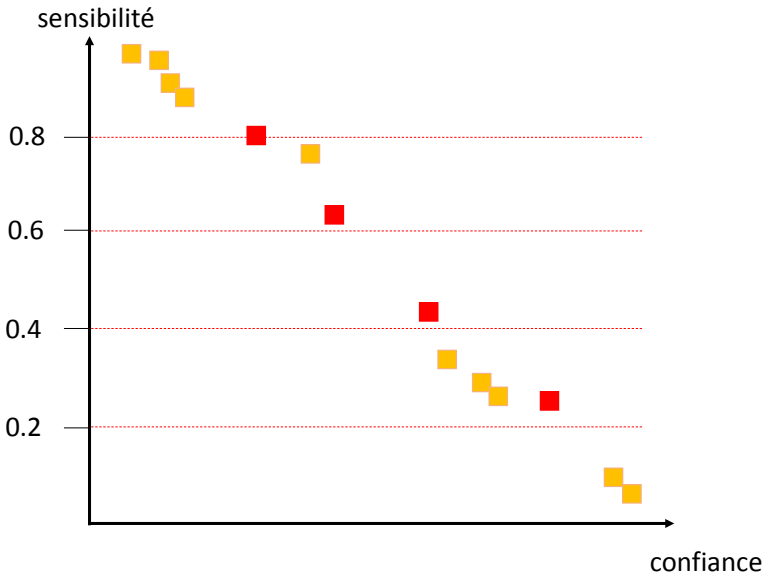


FIGURE 2.2 – Sélection des solutions de l'archive de meilleure confiance et de sensibilité supérieure à 0.2, 0.4, 0.6 et 0.8 ; représentées en rouge

nouvelle solution, que nous espérons meilleure que celles présentes dans l'archive. L'ensemble des méthodes développées à cet effet sont décrites en détail dans l'étude précédente [89]. Nous en présentons ici une vue d'ensemble avant d'exposer une approche légèrement différente sur la construction de ces solutions. Quelques observations découlant d'expérimentations qualitatives réalisées à ce sujet viennent illustrer ces méthodes en fin de partie.

Le principe commun aux méthodes de construction développées est le suivi de ces trois étapes principales, qui rejoignent dans l'idée le processus utilisé dans l'algorithme CBA (*Classification Based on Association rules*) [119] :

— Tri des solutions de l'archive, selon un critère à définir.

— Création d'une nouvelle solution, initialement vide. Dans cette solution sont insérées un sous-ensemble des solutions de l'archive (ordonnée grâce à la phase précédente). Il est possible d'insérer l'ensemble de l'archive sans restriction, ou de n'ajouter que celles satisfaisant certains seuils de performances sur l'une ou l'autre mesure (e.g. $confiance_{min} = 0.4$).

— "Coupe" de la nouvelle solution. Celle-ci est en général très grande après l'insertion de plusieurs solutions de l'archive. A cette étape, on cherche à définir le point à partir duquel on estime que le restant de la nouvelle solution n'est plus intéressant. Cela est dû au fait que les règles de la nouvelle solution sont naturellement ordonnées selon le même critère que celui utilisé dans la première étape. Par conséquent, les règles situés au début de la solution sont les plus intéressantes ; il convient juste de déterminer à quel moment s'arrête ce "début".

La première étape se fait le plus souvent en triant les solutions par confiance décroissante. En effet, l'un des comportements souhaités de la combinaison de solutions est de rassembler plusieurs classifieurs de forte confiance mais de faible sensibilité afin d'augmenter cette dernière. Il est

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

également possible de considérer le ratio (*sensitivité*/(1−*sensibilité*)), qui correspond à la mesure utilisée pour obtenir la courbe ROC d'une solution et qui représente l'évolution du compromis entre les deux objectifs de l'optimisation. La courbe ROC est une mesure de performance d'un classifieur binaire, originalement utilisée en médecine [74].

La deuxième étape est sans doute la plus simple. Les seuils sont inclus afin de réduire la taille de la nouvelle solution. Cependant, ceux-ci sont facultatifs, étant donné que les solutions moins intéressantes sont écartées au cours de la phase suivante.

La dernière étape, visant à déterminer le point de coupe, peut se faire de différentes manières :

— Parcourir les règles de la solution (en partant du haut). A chaque nouvelle règle, calculer la mesure de qualité pour la solution constituée de l'ensemble des règles parcourues jusqu'à présent. Couper lorsqu'une nouvelle règle dégrade la qualité du classifieur. En d'autres termes, cela revient à ajouter des règles tant que cela améliore la qualité. Comme mesure de qualité, on utilise le plus souvent la *f-mesure*, même si d'autres sont possibles.

— Utiliser comme mesure de qualité la proximité au point idéal de la courbe ROC, et couper lorsque le fait d'ajouter une nouvelle solution éloigne de ce point. A noter que cette méthode de coupe est fortement liée à un tri des solutions en fonction du ratio (*sensitivité*/(1−*sensibilité*)), pour des raisons assez évidentes.

— Utiliser directement le ratio (*sensitivité*/(1−*sensibilité*)). Lorsque ce ratio devient inférieur à 1, cela signifie que l'ajout d'une règle apporte davantage de Faux Positifs que de Vrais Positifs, ce qui marque le point de coupe recommandé.

Afin d'affiner le processus, nous ajoutons la possibilité d'utiliser ces méthodes, en considérant indépendamment les règles plutôt que les solutions de l'archive. Ainsi, la qualité de chaque règle est calculée plutôt que celles des solutions ; cela revient à construire autant de classifieurs que de règles différentes dans les solutions de l'archive et d'utiliser les méthodes de construction sur ce nouvel ensemble de classifieurs plutôt que sur l'archive originale. Le but est de permettre la sauvegarde d'éventuelles règles "intéressantes" (de forte confiance), qui feraient partie d'une solution moins intéressante dans son ensemble, par exemple car composée d'autres règles moins précises (faible confiance, forte sensibilité). De telles règles pourraient être laissées de côté en considérant les colutions entières, alors qu'elles présentent un intérêt potentiel à la construction d'un classifieur de qualité. On pourrait arguer que procéder ainsi revient à invalider en partie l'approche fondamentale de l'algorithme MOCA-I, qui manipule et optimise des ensembles de règles, dans la mesure où on sépare totalement les règles du contexte plus général du classifieur dont elles font partie. Ce n'est bien entendu pas l'objectif de ce développement ; nous souhaitons principalement étudier les différences entre une approche aux niveaux "solutions" et "règles", dans le cadre de la construction d'une solution à partir de l'archive.

2.3.3 Observations

L'étude réalisée dans la thèse de J. Jacques sur les méthodes de sélection dans l'archive et de construction de nouvelles solutions soulignait le manque d'intérêt de ce type de méthodes plus complexes par rapport à des options plus simples, telles que prendre la solution de meilleure *f-mesure* [89]. En effet, les solutions obtenues par recombinaison de différents éléments de l'archive s'avèrent similaires aux solutions de l'archive, tant du point de vue de la structure que de la qualité (confiance, sensibilité, f-mesure). Certaines variantes du processus de combinaison aboutissent à des solutions comportant un nombre important de règles (plusieurs dizaines), mais même celles-ci ne présentent pas d'intérêt réel. En effet, bien que structurellement différentes, elles se situent au même niveau que d'autres solutions de l'archive, à divers endroits sur le front

© 2017 Tous droits réservés. lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Pareto. Compte tenu de cela, il peut être tentant de considérer l'utilité de ces méthodes dans les cas où la solution unique renvoyée doit satisfaire certaines contraintes (par exemple, une forte sensitivité). Cependant, il est difficile de prévoir quelle méthode permettra de générer quel "type" de solution, tant le résultat de ces processus de recombinaison dépend du problème en général, de la composition de l'archive et de paramètres associés (seuils d'insertion, etc.). Pour ces raisons, dans le cas où la méthode basique de sélection de la solution de meilleure *f-mesure* ne convient pas, nous recommandons d'employer l'une des alternatives simples présentées plus haut, consistant à fixer un seuil pour la confiance et/ou la sensibilité, en fonction des besoins.

Cependant, les tests menés ont permis de faire une constatation intéressante et potentiellement utile pour l'outil d'extraction d'information. Lorsque l'algorithme s'exécute jusqu'à convergence ou avec une limite de temps assez large (au moins une heure), les méthodes de recombinaison de solutions ne semblent présenter aucun intérêt. Cependant, lorsqu'on laisse moins de temps à l'algorithme, les solutions contenues dans l'archive finales sont logiquement de moins bonne qualité. Or, dans ce cas-là, on observe expérimentalement que le processus de recombinaison permet d'obtenir une solution de qualité notablement meilleure que les solutions de l'archive. L'explication à ce phénomène est à la fois simple et instinctive : les solutions n'ont pas le temps de converger vers des ensembles de règles "optimaux" ; certaines règles de bonne qualité sont formées, mais isolées au sein de solutions n'ayant pas encore convergé dans leur ensemble. Dans cette situation, combiner les règles de meilleure qualité individuelle permet d'obtenir une solution de meilleure qualité globale. A noter que cette solution est très proche de celles auxquelles aboutit l'algorithme à l'approche de la convergence ; le processus de recombinaison permet simplement d'en obtenir au moins une, ce bien plus rapidement qu'en laissant converger l'algorithme. Ainsi, allouer une demi-heure pour l'optimisation, puis recombiner les solutions en extrayant les meilleures règles, permet d'obtenir une solution de qualité équivalente à celles obtenues en quatre heures d'optimisation. Une telle méthode paraît quelque peu complexe à mettre en place, mais nous pensons néanmoins que cette observation, couplée aux techniques de combinaison de solutions développées, peut s'avérer utile dans le cas hypothétique où attendre la (quasi-)convergence de l'algorithme se révèlerait trop long. La construction d'une nouvelle solution recombinée à partir de l'archive est également assez coûteuse en termes de temps (plusieurs minutes, dans le cas de données réelles), et ne saurait par conséquent être employée comme opérateur standard de voisinage (comme dans un algorithme génétique), comme marqueur de convergence du processus d'optimisation ou pour générer de nouvelles solutions de départ après un *restart* de l'algorithme. Ce coût est dû à l'évaluation individuelle de chaque règle composant chaque solution de l'archive.

Il est cependant difficile de mesurer à partir de quel moment de l'exécution de MOCA-I il devient plus intéressant de recombiner les solutions plutôt que de continuer l'optimisation par recherche locale. Pour cette raison, nous choisissons de conserver le mécanisme original de sélection, où la règle de meilleure f-mesure sur l'ensemble d'apprentissage est retenue dans l'archive finale. Le processus de recombinaison demeure une piste intéressante si des contraintes étaient placées sur le temps d'exécution, pour des cas d'étude où il serait nécessaire d'obtenir des solutions rapidement.

2.4 Conclusion

Dans ce chapitre, nous avons présenté le problème de la classification ainsi que l'algorithme MOCA-I, qui génère un ensemble de règles de classification partielle dans un contexte multi-objectif. Nous avons également discuté des mécanismes de choix de la solution parmi l'archive produite en fin d'exécution de l'algorithme. Une limitation de MOCA-I est qu'il ne supporte que les attributs qualitatifs. Ceux-ci peuvent être de différents types (binaire, symbolique ordonné ou non-ordonné) ; par conséquent, MOCA-I gère dans une certaine mesure l'hétérogénéité dans les

© 2017 Tous droits réservés. lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

données. Cependant, les types d'attributs supportés ne sont que qualitatifs, ce qui ne correspond pas à la diversité réelle des données. Dans le chapitre suivant, nous développons une première extension à MOCA-I afin de prendre en compte les données quantitatives.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Chapitre 3

MOCA-IQ - Classification sur données numériques

Dans le chapitre précédent, les grandes lignes de l'algorithme MOCA-I pour la génération de règles de classification ont été présentées. Dans ce chapitre, une première extension est détaillée, appelée MOCA-IQ, visant à supporter les données quantitatives en plus des autres types déjà supportés. Les données quantitatives forment une part importante des données hospitalières, provenant de diverses sources (résultats d'analyse, constantes vitales, informations personnelles telles que l'âge, etc.).

Ce chapitre est associé aux publications suivantes :

— M. Vandromme, J. Jacques, J. Taillard, L. Jourdan, C. Dhaenens : *Impact de la discrétisation des données numériques sur l'efficacité d'un algorithme de classification par métaheuristique*, **Conférence ROADEF**, 2015.

— M. Vandromme, J. Jacques, J. Taillard, L. Jourdan, C. Dhaenens : *Handling numerical data to evolve classification rules using a Multi-Objective Local Search*, **Metaheuristics International Conference**, MIC 2015.

3.1 Problème posé par les attributs numériques

Nous définissons tout d'abord les différents types que peuvent prendre les attributs dans les données, résumés dans la Figure 3.1. Nous ne nous concentrons dans ce chapitre que sur les attributs non-temporels ; les attributs temporels seront traités dans le chapitre suivant. Ces attributs non-temporels sont répartis en deux grandes classes :

— un attribut **qualitatif** inclut un ensemble fini de valeurs possibles. Par exemple, l'attribut *groupe sanguin* peut prendre les valeurs $\{A, B, AB, O\}$.

— un attribut **quantitatif** inclut un intervalle de valeurs possibles. Par exemple, l'attribut *âge* peut prendre n'importe quelle valeur dans l'intervalle $[0, 130]$. Dans sa forme la moins restrictive, un attribut quantitatif peut prendre n'importe quelle valeur sur l'ensemble des réels $\mathbb{R}$.

Les attributs qualitatifs peuvent être séparés plus précisément en trois sous-types :

— un attribut **symbolique ordonné** (ou ordinal) inclut une liste ordonnée de valeurs possibles. Par exemple, l'attribut *sévérité* peut prendre les valeurs {*basse, moyenne, haute*}. Le point important ici est que les valeurs sont comparables entre elles ; par exemple, *haute* est supérieure à *moyenne*.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

— un attribut **symbolique non-ordonné** (ou catégoriel) correspond au cas plus général, où les valeurs possibles sont incomparables entre elles. L'attribut *groupe sanguin* présenté plus haut illustre ce type de données.

— un attribut **binaire** est un cas particulier d'attribut symbolique non-ordonné, avec uniquement deux valeurs possibles 0 et 1. Ce type d'attribut peut être géré de la même façon que le précédent, mais nous choisissons de séparer les deux cas dans l'ensemble de ce travail. En effet, les données binaires ont été particulièrement étudiées dans les problèmes de fouille de données, et disposent de certaines méthodes permettant de les exploiter au mieux, comme nous le verrons plus tard.

Les attributs quantitatifs peuvent également être séparés en deux sous-types distincts :

— un attribut **entier** est défini sur un intervalle $I_e \in \mathbb{N}$.

— un attribut **réel** est défini sur un intervalle $I_r \in \mathbb{R}$.

A noter que, si l'intervalle I_e est fini, un attribut entier peut être considéré comme un attribut symbolique ordonné. Cependant, nous séparons les deux afin de considérer le cas général des attributs entiers, avec un nombre infini de valeurs possibles quand $I_e = \mathbb{N}$.

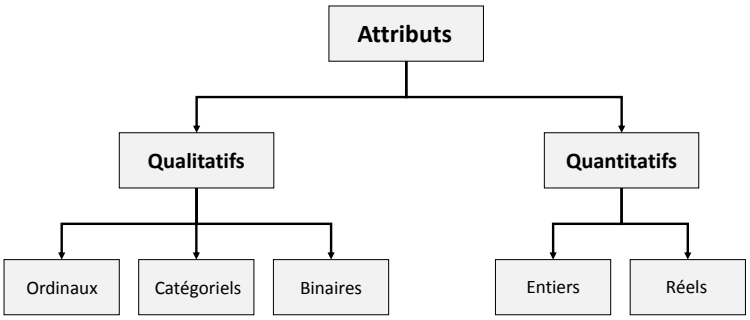


FIGURE 3.1 – Taxonomie des différents types d'attributs non-temporels

Une des limitations de l'algorithme MOCA-I original est son incapacité à gérer directement les attributs quantitatifs (ou numériques), tels que l'âge. Cette limitation est récurrente dans le domaine des méta-heuristiques, étant donné que ces méthodes cherchent le plus souvent à répondre à des problèmes d'optimisation combinatoire. Plus précisément, ces méthodes visent à explorer rapidement l'espace des solutions possibles afin de trouver une (ou plusieurs) bonne(s) solution(s). Cet espace des solutions est certes trop large pour pouvoir être exploré par des méthodes exhaustives, mais il est cependant fini : il existe un nombre fini de solutions possibles au problème. C'est le cas lorsque chaque attribut du problème ne peut prendre lui-même qu'un nombre fini de valeurs.

Cette dernière hypothèse ne tient plus dès lors qu'au moins un des attributs peut prendre une infinité de valeurs différentes ; l'espace des solutions possibles devient alors infini. Dans le cas des attributs prenant des valeurs entières telles que l'âge, il serait envisageable de définir une liste bornée des valeurs possibles ; par exemple, tous les nombres entiers dans l'intervalle [0, 130]. Cependant, cette approche n'est possible que si cet intervalle n'est pas trop grand, au risque d'accroître la taille de l'espace de recherche de manière déraisonnable. De plus, il est impossible de procéder ainsi pour les attributs prenant des valeurs réelles. Par exemple, dans le contexte

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

qui nous intéresse ici, un résultat de mesure biologique tel que le taux de glycémie peut prendre une infinité de valeurs possibles.

Ce problème peut être traité de deux manières différentes, présentées sur la Figure 3.2. La première est de transformer les données numériques en amont par discrétisation, afin de restreindre les valeurs possibles à un ensemble fini. On se ramène ainsi à un problème d'optimisation combinatoire sur un espace de recherche fini, auquel on peut appliquer MOCA-I dans sa version originale. La seconde approche consiste à modifier MOCA-I de sorte que celui-ci soit capable d'explorer les espaces de recherche infinis obtenus en considérant les attributs numériques sans transformation. De nombreuses méthodes d'*optimisation continue* ont été développées pour ce type de problème [63]. Nous nous inspirons de ces méthodes pour développer des mécanismes de voisinage adaptés aux données numériques dans le cadre de cette seconde approche.

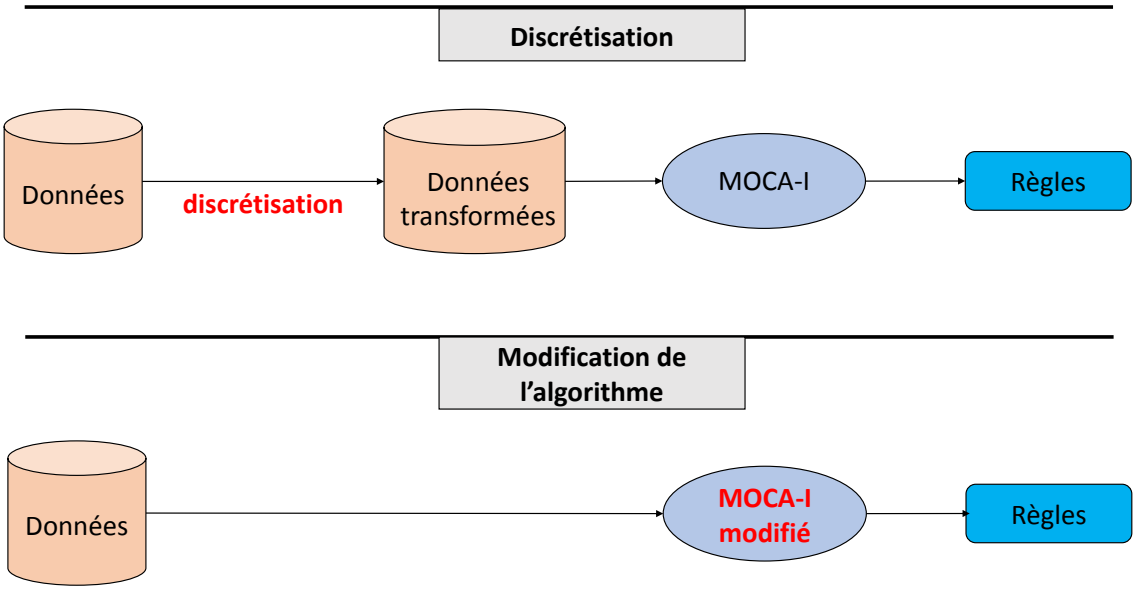


FIGURE 3.2 – Deux approches pour la gestion des attributs quantitatifs : discrétisation ou modification de l'algorithme

Dans ce chapitre, nous nous posons la question de l'efficacité de ces deux approches. Ainsi, chaque approche est explorée, puis les deux sont comparées afin de déterminer la plus adaptée.

3.2 Première approche : discrétisation des données numériques

Dans cette approche, nous cherchons à nous ramener à des cas gérés par MOCA-I, en transformant les attributs quantitatifs en attributs qualitatifs par discrétisation. Nous étudions donc différentes méthodes de discrétisation afin de choisir la plus efficace.

3.2.1 Généralités sur la discrétisation

La discrétisation consiste à transformer un espace continu C en un espace discret D, en définissant une partition complète et sans chevauchement de l'espace C. En d'autres termes, cela

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

revient à transformer $C = [d_0, d_m]$ en un ensemble D de k intervalles discrets, avec :

$$D = \{[d_0, d_1], (d_1, d_2], \ldots, (d_{m-2}, d_{m-1}], (d_{m-1}, d_m]\}, \text{ avec } d_i < d_{i+1} \text{ pour } i = 0, 1, \ldots, m-1. \tag{3.1}$$

Une méthode de discrétisation produit donc un ensemble de points de séparation (ou cut points) $P = \{d_1, d_2, d_{m-2}, d_{m-1}\}$ définissant la partition de l'espace continu C. Définir une "bonne" partition d'un espace continu est un problème complexe, notamment du fait que la qualité d'une partition n'admet pas de mesure universellement reconnue. Afin de répondre à ce problème, de nombreuses méthodes de discrétisation, ainsi que plusieurs mesures de la qualité d'une partition, ont été proposées. Certaines de ces méthodes sont présentées dans la sous-partie suivante ; le lecteur pourra se référer à [66] pour une vision plus complète de la discrétisation en général et une taxonomie détaillée des différentes méthodes existantes.

Dans la version originale de MOCA-I, les données numériques sont discrétisées par *equal-width binning*. Cette méthode simple consiste à diviser l'espace continu en un nombre fixé k d'intervalles de même largeur. Par exemple, le *k-binning* d'un espace continu $C = [0, 100]$ avec $k = 5$ produira l'ensemble de *cut points* $P = \{20, 40, 60, 80\}$, lequel définit l'espace discret $D = \{[0, 20], (20, 40], (40, 60], (60, 80], (80, 100]\}$.

Discrétiser les données en amont présente certains avantages. En plus de la simplicité, cette solution permet de réduire la taille de l'espace de recherche, ce qui peut être bénéfique à l'algorithme d'optimisation. Une étude a montré que, dans certains cas, discrétiser les données pouvait améliorer les performances d'une méta-heuristique, même si celle-ci est capable de gérer les données numériques [29].

3.2.2 Différenciation des méthodes de discrétisation

L'étude mentionnée précédemment référence plus de 80 méthodes de discrétisation différentes [66]. Ce nombre rend pratiquement impossible toute comparaison exhaustive, étant donné que ces techniques ne disposent que rarement d'une implémentation standard et prête à l'emploi. Il est néanmoins possible de différencier les algorithmes de discrétisation en fonction d'un certain nombre de propriétés. L'étude de référence [66] propose une taxonomie des techniques de discrétisation à partir des propriétés discriminantes suivantes :

- Statique ou dynamique : une méthode statique est appliquée lors d'une phase de pré-traitement des données, tandis qu'une méthode dynamique se déroule en même temps que l'algorithme d'apprentissage. Une très grande majorité des méthodes existantes sont statiques, ce qui explique que nous ayons introduit le concept de discrétisation en lui-même comme un pré-traitement des données. Dans la comparaison des méthodes qui fait l'objet de la sous-partie suivante, les méthodes dynamiques sont, par nécessité, détachées de l'algorithme d'apprentissage ; une variante statique de ces méthodes est par conséquent employée pour les besoins de l'étude.
- Univariable ou multivariable : les méthodes univariables considèrent les attributs numériques à discrétiser un par un. A l'inverse, les méthodes multivariables traitent l'ensemble des attributs numériques comme un seul problème et cherchent à tirer parti des corrélations pouvant exister entre différents attributs. Les secondes sont par conséquent plus riches, mais aussi plus complexes. La plupart des techniques actuelles appartiennent à la première catégorie.
- Supervisé ou non-supervisé : une discrétisation supervisée tient compte des labels attribués aux instances pour construire les intervalles. Par exemple, dans le cas d'un problème de classification, une méthode supervisée peut utiliser la classe de chaque instance. A l'inverse, les méthodes non-supervisées, telles que le binning présenté plus haut, n'utilisent pas cette

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

information. Pour cette raison, ces dernières sont souvent moins efficaces, mais présentent l'avantage d'être plus simples à mettre en place.

— Séparation ou fusion d'intervalles : l'ensemble d'intervalles produit par la discrétisation peut être créé de deux manières différentes. Une première option est de partir d'un unique intervalle et de le séparer progressivement en plusieurs ; une seconde option est de définir au départ autant d'intervalles qu'il y a de valeurs différentes, puis de fusionner ces intervalles de départ entre eux.

— Globale ou locale : une méthode globale peut utiliser l'intégralité des données disponibles pour l'attribut à discrétiser, tandis qu'une méthode locale n'en exploite qu'une partie. La plupart des approches sont globales, les méthodes locales procèdent le plus souvent par séparation récursive des intervalles.

— Directe ou incrémentale : les techniques incrémentales définissent un unique nouvel intervalle à chaque itération, tandis que les techniques directes en créent plusieurs (voire même tous les intervalles) au cours d'une itération. Par conséquent, les premières utilisent à chaque étape les résultats produits jusque là, c'est-à-dire les intervalles définis à cet instant.

— Mesure d'évaluation : une méthode de discrétisation produit, durant son exécution, un certain nombre de solutions possibles, et sélectionne de manière incrémentale la solution (c'est-à-dire un ensemble d'intervalles) la mieux adaptée. Chaque méthode doit donc inclure une mesure permettant de comparer les solutions entre elles. De nombreuses mesures de comparaison ont été proposées dans la littérature. Ces mesures peuvent être classées en fonction de leur type : information (mesure d'entropie, le plus souvent), statistique (corrélations entre attributs, etc.), ensembles approximatifs, *wrapper* (utilisation de l'erreur d'un classifieur appliqué à la solution à évaluer), ou absence de mesure dans le cas de certaines méthodes non-supervisées.

D'autres critères existent pour différencier les méthodes de discrétisation, mais ceux-ci sont moins critiques, ou fortement liés à ceux présentés ci-dessus. Cependant, il convient de mentionner que l'aspect non-paramétrique d'une méthode revêt une importance particulière dans le cadre applicatif qui nous occupe. En effet, il nous apparaît toujours plus intéressant de réduire le plus possible le nombre de paramètres requis par l'algorithme d'optimisation dans son ensemble.

3.2.3 Processus de comparaison des méthodes de discrétisation

Diverses mesures ont été proposées pour évaluer l'efficacité des différentes méthodes de discrétisation. Cette évaluation peut se faire suivant deux axes différents. Le premier consiste à mesurer et comparer l'efficacité intrinsèque de chaque méthode suivant plusieurs critères : nombre d'intervalles produits, taux d'incohérence, temps d'exécution requis, etc. Le second consiste à comparer l'efficacité d'un même algorithme d'apprentissage utilisant un même jeu de données, discrétisé suivant différentes méthodes. De cette manière, on mesure indirectement la qualité d'une discrétisation en observant son effet sur la qualité de l'apprentissage.

Plusieurs études ont été menées afin de déterminer de manière empirique les méthodes les plus efficaces [55, 66]. Ces études explorent les deux axes de comparaison : à la fois mesure directe et indirecte de la qualité des différentes méthodes de discrétisation. La seconde partie est traitée le plus souvent en utilisant plusieurs classifieurs standard, reconnus pour leur efficacité, tels que C4.5, k-Nearest-Neighbors ou Naive Bayes. Comme expliqué précédemment, il est pratiquement impossible de comparer exhaustivement l'intégralité des méthodes existantes. Pour cette raison, nous avons choisi un sous-ensemble, contenant des méthodes le plus représentatives possible de la diversité (voir critères de différenciation présentés plus haut).

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

La mesure indirecte de l'efficacité d'une discrétisation est celle qui nous intéresse le plus dans le cadre de ce travail. En effet, qu'une discrétisation soit meilleure qu'une autre ne nous intéresse que dans la mesure où cela permet à l'algorithme d'optimisation d'obtenir de meilleurs résultats. Par conséquent, les mesures de qualité intrinsèque des différentes méthodes sont de moindre importance dans ce contexte. L'objet des expérimentations qui suivent est de déterminer quelle méthode de discrétisation permet à MOCA-I d'obtenir les meilleurs résultats. En ce sens, ce travail diffère des études empiriques mentionnées plus haut, étant donné que nous nous concentrons sur un algorithme d'apprentissage en particulier, sans chercher à déterminer quelle discrétisation est plus performante en général sur un ensemble d'algorithmes standard. Sur le principe, on peut rapprocher cela d'une phase préparatoire de tuning des paramètres de MOCA-I, en considérant chaque méthode de discrétisation comme une valeur possible d'un paramètre à fixer.

Un ensemble représentatif des différentes méthodes est donc choisi à des fins de comparaison. La Table 3.1 présente les propriétés des méthodes retenues :

Méthode	Statique	Multivar.	Supervisée	Séparation	Globale	Directe	Mesure
10-bin	Oui	Non	Non	Oui	Oui	Oui	Bin.
1R	Oui	Non	Oui	Oui	Oui	Oui	Bin.
CAIM	Oui	Non	Oui	Oui	Oui	Non	Stat.
ChiMerge	Oui	Non	Oui	Non	Oui	Non	Stat.
Chi2	Oui	Non	Oui	Non	Oui	Non	Stat.
ExtChi2	Oui	Non	Oui	Non	Oui	Non	Stat.
Fayyad	Oui	Non	Oui	Oui	Non	Non	Info.
FUSINTER	Oui	Non	Oui	Non	Oui	Non	Info.
ID3	Non	Non	Oui	Oui	Non	Non	Info.
Zeta	Oui	Non	Oui	Oui	Oui	Oui	Stat.

TABLE 3.1 – Caractéristiques des méthodes de discrétisation retenues

Ces méthodes ont été choisies parmi celles implémentées dans la plate-forme KEEL [6]. A noter que certaines des méthodes initialement retenues ne fonctionnaient pas correctement sur les jeux de données choisis ; pour cette raison, nous n'avons pas pu tester de méthode de discrétisation multi-variable ou utilisant des mesures de qualité basées sur les ensembles approximatifs ou les *wrappers*. Nous estimons cependant que la sélection retenue présente une bonne diversité tout en intégrant les méthodes les plus largement utilisées. Les méthodes $ChiMerge$, $Chi2$ et $ExtChi2$ présentent les mêmes caractéristiques dans le tableau, mais elles n'en possèdent pas moins certaines différences conceptuelles qui les rendent intéressantes à comparer. Il semblait également instructif d'observer si les modifications apportées à une technique de base améliorent ou non les résultats ($ExtChi2$ est basée sur $Chi2$, elle-même basée sur $ChiMerge$). Maintenant que les méthodes sont choisies, nous pouvons les comparer entre elles.

3.2.4 Expérimentations

Dans cette section, nous définissons le protocole expérimental utilisé, puis nous présentons les résultats et en déduisons la méthode de discrétisation la plus efficace.

3.2.4.1 Protocole expérimental

Jeux de données Afin de mesurer l'efficacité des méthodes implémentées, nous avons besoin d'un ensemble de jeux de données à classifier, contenant des attributs quantitatifs. Nous déci-

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

dons d'utiliser également quelques jeux "mixtes", c'est-à-dire contenant à la fois par des attributs qualitatifs et quantitatifs, afin de mieux correspondre aux données réelles. Ces jeux de données proviennent de la base de données de l'UCI [115], qui rassemble des jeux standard largement utilisés dans la littérature. Ce choix a été principalement motivé par l'indisponibilité de données réelles adaptées aux besoins de l'étude (c'est-à-dire, contenant un nombre significatif d'attributs numériques) lors de la période de développement de ces méthodes. Mener les expérimentations sur des données de la littérature présente l'avantage de faciliter la comparaison avec d'autres méthodes de classification, et assure une plus grande généricité. Cependant, cela présente l'inconvénient de s'éloigner quelque peu du cadre du problème en validant les algorithmes développés sur des données détachées des cas d'étude proposés par le projet.

Les jeux de données sont choisis afin de garantir une certaine diversité sur les différentes caractéristiques que ces données peuvent présenter : nombre d'instances, nombre d'attributs, répartition qualitatifs/quantitatifs, pourcentage d'instances positives. Cette dernière propriété est d'une grande importance, étant donné que MOCA-I vise à classifier des données qui sont, par nature, fortement déséquilibrées, donc contenant très peu d'instances positives. Il reste néanmoins intéressant d'évaluer les performances de l'algorithme sur des données présentant une répartition plus équilibrée, aussi différentes valeurs de déséquilibre (ou *asymétrie*) sont choisies. A noter que certains des jeux de données présentés sont des versions binarisées de jeux existants, obtenus en réduisant un problème de classification multi-classes à un problème de classification binaire. Afin d'y parvenir, l'une des classes est choisie comme positive, et toutes les autres sont considérées comme négatives. Cette méthode est par ailleurs utile pour obtenir des jeux de données déséquilibrés. La Table 3.2 présente les caractéristiques des différents jeux retenus : nombre d'instances, nombre d'attributs, type des attributs, pourcentage d'instances positives. Dans la colonne "Type attributs", la notation "R/I/C" correspond au nombre d'attributs de type réel/entier/catégoriel.

Nom	Nb. inst.	Nb. attrs.	Type attrs. (R/I/C)	% positifs
adult	45 000	14	0/6/8	25%
arrythmia	452	279	206/0/73	46%
page-blocks	5 472	10	4/6/0	10%
penbased	11 000	16	0/16/0	10%
ring	7 400	20	20/0/0	50%
segment	2 308	19	19/0/0	17%
spambase	4 597	57	55/2/0	39%
vowel	988	13	10/3/0	10%

TABLE 3.2 – Caractéristiques des jeux de données utilisés

Implémentation des méthodes de discrétisation Nous cherchons à comparer plusieurs méthodes de discrétisation comme phase de pré-traitement des données en amont de MOCA-I. Ce pré-traitement peut être séparé de l'algorithme d'optimisation. Nous choisissons donc d'utiliser les implémentations des différentes méthodes de discrétisation disponibles dans l'outil KEEL [6]. Cet outil regroupe un grand nombre de méthodes de pré-traitement (discrétisation, sélection d'attributs), d'algorithmes d'optimisation et de tests statistiques, que nous utiliserons également afin de valider les résultats obtenus.

Protocole expérimental Afin d'évaluer la qualité d'un classifieur, il convient de séparer les données utilisées en deux parties : un ensemble d'apprentissage et un ensemble de test. Le premier

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

est utilisé par l'algorithme pour construire le modèle de classification à partir de données correctement classifiées. Le second n'est utilisé qu'en fin du processus d'optimisation, pour déterminer les performances du modèle généré, c'est-à-dire sa capacité à classifier correctement des nouvelles instances (qui n'ont pas été utilisées pour l'apprentissage). La Figure 3.3 résume cette séparation des données et l'utilisation des deux ensembles distincts. L'ensemble de test permet de mesurer les capacités de généralisation du modèle de classification produit. Dans les expérimentations qui ont été menées pour cette étude, chaque jeu de données est séparé de manière égale entre apprentissage et test : la moitié des instances est allouée à chaque ensemble.

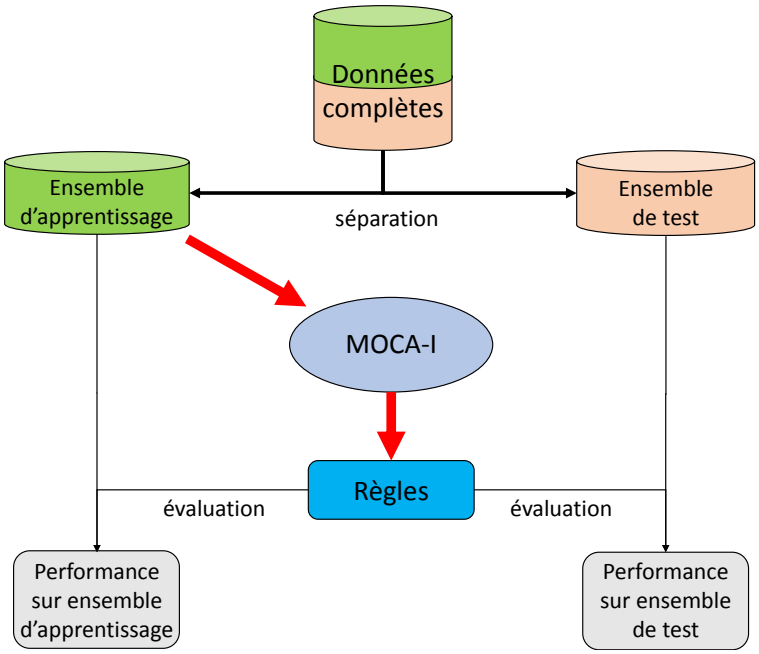


FIGURE 3.3 – Séparation des données en ensembles d'apprentissage et de test, génération des modèles (règles) avec l'ensemble d'apprentissage, validation sur les deux ensembles séparément

Etant donné le caractère stochastique de l'algorithme d'optimisation, il est nécessaire de réaliser plusieurs exécutions de celui-ci sur un même jeu de données, et de considérer la moyenne des performances mesurées. Par conséquent, les résultats présentés correspondent à la moyenne des performances obtenue par chaque méthode sur 30 exécutions. Les écarts-types sont également inclus, principalement afin de détecter d'éventuelles anomalies ou valeurs aberrantes. La graine (ou *seed*) utilisée pour initialiser le générateur de nombres aléatoires change à chaque exécution ; les mêmes 30 graines sont utilisées pour les 30 exécutions de chaque méthode. Tous les tests sont réalisés sur un ordinateur Dell T1700 quad-core à 3.30GHz, disposant de 4 GB de RAM, sous Ubuntu 14 avec gcc 4.8.2.

Tests statistiques Relever les performances respectives de chaque méthode sur un ensemble de jeux de données nous permet de voir (le plus souvent de manière approximative) si telle méthode est plus efficace que telle(s) autre(s) ou non. Cependant, afin de valider plus formellement ces impressions, il est utile de procéder à des tests statistiques permettant de déterminer si les écarts observés sont significatifs. Un test statistique permet de savoir si les différences de per-

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

formance sont probablement dues au hasard. Plus formellement, ces tests permettent d'évaluer une hypothèse nulle H_0, qui correspond souvent au prédicat "les différences observées entre les résultats sont dues au hasard". Un test statistique calcule la probabilité p (aussi appelée *p-value*) de faire une erreur si on conclut que l'hypothèse H_0 est vraie, c'est-à-dire que les méthodes comparées ne présentent pas de différence significative. On considère le plus souvent qu'il est possible de rejeter l'hypothèse nulle, c'est-à-dire de conclure que les résultats sont significativement différents, si la probabilité p est suffisamment faible, le plus souvent $p < 0.05$ ou $p < 0.10$. Les tests statistiques peuvent être paramétriques ou non-paramétriques ; pour les problèmes d'apprentissage automatique, les tests non-paramétriques sont recommandés [49], étant donné que ceux-ci, à la différence des premiers, ne requièrent pas que les données respectent certains critères, tels qu'une distribution normale, qui pourraient être difficiles à garantir dans ce contexte.

Il existe plusieurs tests statistiques différents, chacun étant plus ou moins approprié en fonction des comparaisons à effectuer. Certains tests ne sont applicables que pour comparer deux méthodes, ou ne peuvent comparer les méthodes que sur une unique instance. Une présentation exhaustive des différents tests existant ne rentre pas dans le cadre de cette étude ; le lecteur intéressé pourra se référer à [89] pour un aperçu détaillé des tests statistiques dans le domaine qui nous occupe ici. Etant donné que plusieurs variantes des algorithmes développés seront comparés à chaque fois, et que les exécutions seront réalisées sur plusieurs instances (jeux de données), nous avons besoin de tests permettant une comparaison statistique de plusieurs algorithmes sur plusieurs instances. Les tests les plus adaptés dans ce cas sont ceux de Friedman [64] et d'Iman-Davenport [84] (le second étant une amélioration du premier). Pour la comparaison des méthodes, nous utilisons donc le test de Friedman, dont l'implémentation est également fournie dans le framework KEEL. Ce test fonctionne en affectant à chaque méthode un rang en fonction de ses performances par rapport aux autres méthodes. Chaque méthode possède ainsi un rang par jeu de données ; le test cherche alors à déterminer si les rangs moyens de chaque méthode sont significativement différents. Dans ce test, l'hypothèse nulle est que les rangs moyens sont statistiquement identiques.

KEEL inclut également des tests *post-hoc*, qui servent à déterminer quelle méthode se démarque des autres dans le cas où des différences significatives sont relevées par le premier test. Dans ce cas, nous utilisons les tests de Nemenyi [127] et de Shaffer [150].

Mesures de qualité Nous comparons les méthodes selon trois mesures : f-mesure (à maximiser) [161], MCC (pour *Matthew's Correlation Coefficient*, à maximiser) [121] et nombre d'intervalles générés par la discrétisation (à minimiser). Les deux premières sont des mesures standard de la qualité d'un classifieur binaire. Le nombre d'intervalles générés présente aussi un intérêt, étant donné qu'une des motivations de l'emploi d'une discrétisation concerne la réduction de l'espace de recherche, et que cette réduction est d'autant plus importante que le nombre d'intervalles générés est faible. Ainsi, il est attendu que MOCA-I fonctionne plus rapidement lorsque le nombre d'intervalles est réduit ; garder trace de ce nombre permet donc de valider ou non cette hypothèse. Cependant, mesurer le nombre d'intervalles en additionnant simplement le nombre d'intervalles générés pour chaque attribut ne paraît pas judicieux, pas plus que de considérer le nombre moyen d'intervalles pour l'ensemble des attributs d'un jeu de données. Pour répondre à ce problème, nous choisissons de relever, pour chaque attribut de chaque jeu de données, le rang obtenu par chacune des méthodes de discrétisation. Ce rang est déterminé en ordonnant les méthodes par nombre d'intervalles croissant pour l'attribut considéré. Le rang global pour un jeu de données entier est ensuite déterminé en calculant la moyenne des rangs obtenus sur chaque attribut.

Comme expliqué dans la partie précédente, 30 exécutions sont réalisées pour chaque jeu de données et chaque méthode. Les Tables 3.3 et 3.4 présentent les valeurs moyennes et écarts-

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

type obtenus pour la f-mesure sur les ensembles d'apprentissage et de test respectivement, les Tables 3.5 et 3.6 pour le MCC (apprentissage et test également). La Table 3.7 présente le rang moyen des méthodes pour le nombre d'intervalles créés lors de la discrétisation, donc avant séparation en apprentissage/test. La Table 3.8 présente les rangs pour chacune des trois mesures considérées sur l'ensemble de test. Dans les tables, les valeurs en gras indiquent la méthode la plus performante pour le jeu de données en question. Nous rappelons que l'objectif est de **maximiser la f-mesure et le MCC** et de **minimiser le nombre d'intervalles**. La qualité des méthodes de discrétisation est donc mesurée à travers l'efficacité de MOCA-I, pour lequel elles agissent en pré-traitement des données.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

	binning	1R	CAIM	ChiMerge	Chi2	ExtendedChi2	Fayyad	FUSINTER	ID3	Zeta
adult	0.658 ± 0.006	0.61 ± 0.006	0.66 ± 0.005	0.66 ± 0.006	0.676 ± 0.007	0.668 ± 0.006	**0.678 ± 0.005**	0.677 ± 0.005	0.635 ± 0.016	0.656 ± 0.004
arrythmia	0.985 ± 0.007	0.967 ± 0.019	0.977 ± 0.008	0.971 ± 0.008	0.94 ± 0.01	0.717 ± 0.02	0.96 ± 0.008	**0.99 ± 0.007**	0.938 ± 0.014	0.957 ± 0.016
page-blocks	0.768 ± 0.017	0.871 ± 0.008	0.753 ± 0.015	0.769 ± 0.017	0.874 ± 0.013	0.785 ± 0.015	0.891 ± 0.009	**0.893 ± 0.009**	0.867 ± 0.011	0.779 ± 0.013
penbased	**0.999 ± 0.001**	0.19 ± 0.004	0.992 ± 0.002	0.982 ± 0.004	0.875 ± 0.007	0.982 ± 0.004	0.996 ± 0.002	0.996 ± 0.001	0.994 ± 0.004	0.927 ± 0.008
ring	0.878 ± 0.009	0.904 ± 0.004	0.881 ± 0.004	0.896 ± 0.002	**0.918 ± 0.003**	0.67 ± 0.005	0.888 ± 0.007	0.883 ± 0.003	0.718 ± 0.012	0.867 ± 0.003
segment	0.998 ± 0.002	**1.0 ± 0.001**	0.985 ± 0.005	0.967 ± 0.008	0.922 ± 0.011	0.975 ± 0.005	**1.0 ± 0.0**	**1.0** ± 0.0	**1.0** ± 0.0	0.714 ± 0.026
spambase	0.786 ± 0.009	0.894 ± 0.007	0.906 ± 0.005	0.909 ± 0.006	0.895 ± 0.004	0.891 ± 0.008	**0.914 ± 0.004**	0.913 ± 0.005	0.889 ± 0.008	0.898 ± 0.005
vowel	**1.0 ± 0.0**	1.0 ± 0.0	0.912 ± 0.027	0.907 ± 0.021	0.862 ± 0.024	0.739 ± 0.036	0.986 ± 0.011	1.0 ± 0.0	1.0 ± 0.0	0.866 ± 0.032

TABLE 3.3 – F-mesures obtenues sur l'ensemble d'apprentissage

	binning	1R	CAIM	ChiMerge	Chi2	ExtendedChi2	Fayyad	FUSINTER	ID3	Zeta
adult	0.655 ± 0.005	0.609 ± 0.007	0.656 ± 0.005	0.656 ± 0.005	0.673 ± 0.005	0.664 ± 0.006	**0.674 ± 0.006**	**0.674 ± 0.005**	0.634 ± 0.016	0.653 ± 0.005
arrythmia	0.645 ± 0.031	0.649 ± 0.045	0.732 ± 0.037	0.739 ± 0.046	**0.773 ± 0.027**	0.598 ± 0.036	0.758 ± 0.04	0.709 ± 0.038	0.706 ± 0.049	0.679 ± 0.049
page-blocks	0.722 ± 0.021	0.808 ± 0.022	0.714 ± 0.019	0.739 ± 0.022	0.836 ± 0.011	0.745 ± 0.014	0.837 ± 0.018	**0.846 ± 0.02**	0.802 ± 0.016	0.759 ± 0.017
penbased	**0.985 ± 0.006**	0.184 ± 0.014	**0.985 ± 0.004**	0.976 ± 0.005	0.868 ± 0.009	0.971 ± 0.006	0.982 ± 0.005	0.982 ± 0.005	0.974 ± 0.006	0.924 ± 0.008
ring	0.86 ± 0.013	0.885 ± 0.005	0.855 ± 0.005	0.877 ± 0.005	**0.901 ± 0.005**	0.672 ± 0.005	0.866 ± 0.009	0.863 ± 0.004	0.707 ± 0.01	0.843 ± 0.007
segment	0.973 ± 0.01	0.955 ± 0.012	0.972 ± 0.006	0.949 ± 0.011	0.922 ± 0.012	0.97 ± 0.007	**0.98 ± 0.009**	0.968 ± 0.011	0.97 ± 0.013	0.718 ± 0.025
spambase	0.764 ± 0.01	0.866 ± 0.009	0.887 ± 0.009	**0.891 ± 0.006**	0.881 ± 0.009	0.875 ± 0.008	**0.891 ± 0.007**	0.89 ± 0.008	0.87 ± 0.01	0.882 ± 0.007
vowel	0.882 ± 0.037	0.892 ± 0.044	0.822 ± 0.043	0.857 ± 0.031	0.831 ± 0.031	0.747 ± 0.047	**0.927 ± 0.031**	0.898 ± 0.044	0.893 ± 0.04	0.814 ± 0.042

TABLE 3.4 – F-mesures obtenues sur l'ensemble de test

Thèse de Maxence Vandromme, Lille 1, 2017

	binning	1R	CAIM	ChiMerge	Chi2	ExtendedChi2	Fayyad	FUSINTER	ID3	Zeta
adult	0.542 ± 0.008	0.491 ± 0.015	0.553 ± 0.007	0.549 ± 0.008	0.568 ± 0.008	0.556 ± 0.009	**0.57 ± 0.007**	**0.57 ± 0.006**	0.512 ± 0.022	0.555 ± 0.007
arrythmia	0.972 ± 0.014	0.939 ± 0.035	0.957 ± 0.015	0.946 ± 0.016	0.88 ± 0.02	0.461 ± 0.046	0.926 ± 0.016	**0.981 ± 0.013**	0.886 ± 0.028	0.92 ± 0.031
page-blocks	0.747 ± 0.018	0.857 ± 0.009	0.727 ± 0.016	0.754 ± 0.017	0.861 ± 0.014	0.769 ± 0.017	0.88 ± 0.01	**0.882 ± 0.01**	0.853 ± 0.012	0.755 ± 0.015
penbased	**0.999 ± 0.001**	-0.915 ± 0.317	0.991 ± 0.002	0.98 ± 0.005	0.86 ± 0.007	0.98 ± 0.004	0.996 ± 0.002	0.996 ± 0.002	0.993 ± 0.004	0.921 ± 0.008
ring	0.748 ± 0.023	0.804 ± 0.008	0.756 ± 0.009	0.784 ± 0.004	**0.833 ± 0.007**	-1.0 ± 0.0	0.768 ± 0.014	0.758 ± 0.005	0.332 ± 0.045	0.725 ± 0.006
segment	0.998 ± 0.002	**1.0 ± 0.001**	0.983 ± 0.005	0.962 ± 0.009	0.91 ± 0.013	0.971 ± 0.006	**1.0 ± 0.0**	**1.0 ± 0.0**	**1.0 ± 0.0**	0.712 ± 0.024
spambase	0.647 ± 0.014	0.826 ± 0.01	0.848 ± 0.009	0.849 ± 0.008	0.828 ± 0.007	0.82 ± 0.012	**0.859 ± 0.006**	0.857 ± 0.008	0.816 ± 0.014	0.834 ± 0.008
vowel	**1.0 ± 0.0**	**1.0 ± 0.0**	0.906 ± 0.028	0.9 ± 0.022	0.85 ± 0.025	0.722 ± 0.037	0.984 ± 0.012	**1.0 ± 0.0**	**1.0 ± 0.0**	0.86 ± 0.031

TABLE 3.5 – MCC obtenus sur l'ensemble d'apprentissage

	binning	1R	CAIM	ChiMerge	Chi2	ExtendedChi2	Fayyad	FUSINTER	ID3	Zeta
adult	0.539 ± 0.006	0.49 ± 0.015	0.548 ± 0.008	0.544 ± 0.006	0.564 ± 0.007	0.551 ± 0.009	**0.566 ± 0.008**	**0.566 ± 0.007**	0.511 ± 0.022	0.552 ± 0.008
arrythmia	0.36 ± 0.056	0.37 ± 0.081	0.515 ± 0.063	0.524 ± 0.082	**0.585 ± 0.052**	0.245 ± 0.044	0.565 ± 0.067	0.476 ± 0.063	0.465 ± 0.083	0.409 ± 0.092
page-blocks	0.697 ± 0.022	0.789 ± 0.024	0.686 ± 0.019	0.721 ± 0.023	0.82 ± 0.012	0.727 ± 0.015	0.821 ± 0.019	**0.831 ± 0.021**	0.781 ± 0.017	0.734 ± 0.018
penbased	**0.984 ± 0.006**	-0.919 ± 0.303	0.983 ± 0.004	0.973 ± 0.006	0.853 ± 0.01	0.968 ± 0.007	0.98 ± 0.005	0.98 ± 0.005	0.971 ± 0.007	0.918 ± 0.008
ring	0.707 ± 0.031	0.765 ± 0.009	0.699 ± 0.009	0.741 ± 0.011	**0.795 ± 0.011**	-1.0 ± 0.0	0.718 ± 0.019	0.711 ± 0.007	0.293 ± 0.04	0.673 ± 0.013
segment	0.968 ± 0.012	0.947 ± 0.014	0.967 ± 0.008	0.94 ± 0.012	0.908 ± 0.014	0.965 ± 0.008	**0.977 ± 0.01**	0.963 ± 0.013	0.965 ± 0.015	0.714 ± 0.022
spambase	0.611 ± 0.015	0.78 ± 0.014	0.816 ± 0.014	0.819 ± 0.011	0.805 ± 0.014	0.792 ± 0.011	**0.821 ± 0.013**	0.819 ± 0.011	0.786 ± 0.016	0.807 ± 0.01
vowel	0.871 ± 0.041	0.883 ± 0.046	0.808 ± 0.044	0.846 ± 0.034	0.816 ± 0.034	0.732 ± 0.04	**0.921 ± 0.034**	0.889 ± 0.047	0.885 ± 0.042	0.805 ± 0.042

TABLE 3.6 – MCC obtenus sur l'ensemble de test

Thèse de Maxence Vandromme, Lille 1, 2017

	UnifWid	1R	Fayyad	ChiM	Chi2	ExtChi2	CAIM	Zeta	ID3	FUS.
adult	6.333	3.333	5.167	**1.667**	7.167	8.5	2.667	3.667	10.0	6.5
arrythmia	6.896	4.14	**2.427**	4.28	3.022	3.624	5.993	6.993	9.527	8.1
page-blocks	8.2	8.7	6.3	**1.2**	4.7	3.1	2.5	3.5	10.0	6.8
penbased	9.0	**1.313**	6.875	2.688	2.5	4.688	4.563	5.563	10.0	7.813
ring	6.714	8.667	5.857	2.095	4.048	**1.238**	3.667	4.667	9.952	8.095
segment	8.632	5.789	5.632	2.947	**1.737**	3.053	4.474	5.474	9.789	7.474
sonar	8.183	8.083	**1.533**	3.867	2.383	2.45	5.1	6.1	10.0	7.3
spambase	7.526	6.667	3.579	2.316	**1.579**	8.93	3.667	4.667	9.947	6.123
vowel	8.462	7.154	3.769	**2.538**	2.769	3.0	4.923	5.923	9.769	6.692

TABLE 3.7 – Rang moyen sur l'ensemble des attributs numériques pour le nombre d'intervalles générés

Afin d'avoir une vue plus complète des performances de chaque méthode, nous examinons également les rangs moyens de chaque méthode sur l'ensemble des jeux de données, tels qu'utilisés par les tests statistiques.

Algorithme	Rang f-mesure	Rang MCC	Rang nb. intervalles
UniformWidth	6.9	6.9	8.1
1R	7	6.85	6.4
Fayyad	**2.8**	**2.95**	4
ChiMerge	4.6	4.55	**2.5**
Chi2	4.7	4.5	2.9
ExtendedChi2	6.8	7.6	4.2
CAIM	5.55	4.85	4
Zeta	6.9	6.7	5.5
ID3	6.2	6.5	10
FUSINTER	3.55	3.6	7.4

TABLE 3.8 – Rangs moyens sur les trois mesures

Les tests statistiques menés sur ces résultats n'ont relevé que quelques différences significatives sur le nombre d'intervalles (entre ChiMerge/ID3 et Chi2/ID3). Cependant, cet objectif est secondaire par rapport aux deux autres, aussi les résultats de ces tests n'ont-ils qu'une importance réduite. On observe en revanche que la méthode de discrétisation Fayyad [58] obtient un meilleur rang moyen que les autres sur les objectifs de f-mesure et de MCC, tout en affichant des performances correctes pour ce qui est du nombre moyen d'intervalles générés. Ces résultats, couplés au fait que cette méthode soit reconnue comme standard et implémentée dans des outils tels que la plate-forme de *data mining* Weka [72], nous incitent à la retenir comme plus performante que les autres. Par conséquent, nous choisissons d'implémenter cette méthode dans MOCA-I, afin de permettre une discrétisation automatique comme phase de pré-processing. Cela nous permet de conclure cette première approche en proposant une version améliorée de MOCA-I, où les attributs numériques sont transformés par Fayyad en pré-traitement.

3.3 Seconde approche : gestion des attributs numériques dans la recherche locale

Employer une discrétisation est une manière simple et intéressante de répondre au problème lié aux données numériques. Cependant, cela implique par définition une perte dans l'information

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

exploitable par l'algorithme, étant donné que le processus de discrétisation résume et agrège les données originales. Il est donc possible que cette perte de qualité sur les données induise une efficacité moindre de l'algorithme d'optimisation. Pour cette raison, nous proposons dans cette partie un modèle de représentation des attributs numériques dans les règles de classification, ainsi que différentes méthodes pour la recherche locale sur ces attributs d'un nouveau type. Nous détaillons la représentation des attributs numériques dans les règles de classification, une méthode adaptée pour la génération des solutions initiales, un voisinage pour les termes numériques, et une comparaison de plusieurs variantes. Nous appelons le nouvel algorithme MOCA-IQ (MOCA-I pour données Quantitatives).

3.3.1 Règles de classification avec attributs numériques

Dans MOCA-I, comme décrit précédemment, une règle est constituée d'un ou plusieurs terme(s), chaque terme étant décrit par un attribut, un opérateur et une valeur. Dans le cas des attributs numériques discrétisés, la valeur est choisie parmi une liste ordonnée, par exemple, pour un attribut A1, les valeurs possibles sont {[0, 20], (20, 40], (40, 60], (60, 80], (80, 100]}. Trois opérateurs peuvent être utilisés pour ces attributs : { $<$, $>$, $=$ }, avec les significations suivantes :

- $A1 < (20, 40] \longrightarrow A1 \leq 20$
- $A1 > (20, 40] \longrightarrow A1 > 40$
- $A1 = (20, 40] \longrightarrow 20 < A1 \leq 40$

Dans MOCA-IQ, nous reprenons la même structure et les mêmes opérateurs pour définir les termes portant sur des attributs numériques. Cependant, une distinction est faite entre attributs entiers et attributs réels. Dans le cas d'un attribut entier tel que l'âge, l'opérateur d'égalité permettant de créer des termes tels que $\hat{a}ge = 52$ a un sens, et il est donc conservé. Cependant, dans le cas d'un attribut numérique prenant des valeurs réelles, cet opérateur ne permet plus de créer des termes ayant un sens véritable. Par exemple, le terme $taux = 0.85697$ ne représente pas une information concrète, mais plutôt un cas particulier difficilement généralisable. Pour cette raison, l'opérateur d'égalité n'est pas employé pour les attributs réels. Afin de traduire, dans la nouvelle représentation, les termes tels que $A1 = (20, 40]$, une règle peut dorénavant contenir deux termes de même attribut (numérique), à condition que les opérateurs de ces deux termes soient opposés. Ainsi, le terme $A1 = (20, 40]$ sera traduit par les deux termes $A1 > 20$ et $A1 < 41$. A noter que ces deux termes sont créés et gérés de telle sorte qu'il soit impossible de générer des intervalles invalides, tels que $A1 > 10$ et $A1 < 5$.

3.3.2 Génération des solutions initiales

Comme tous les algorithmes d'optimisation à population, MOCA-I inclut une méthode pour générer une population initiale de solutions, qui sert de point de départ à la recherche locale. Chacune de ces solutions est constituée de deux règles, et chacune de ces règles est créée suivant la méthode déjà décrite par l'Algorithme 2 dans la Section 2.2.2. Cette méthode vise à générer des règles initiales basées sur des observations positives, c'est-à-dire contenant la prédiction, auxquelles sont ajoutés des termes aléatoires afin de garantir une certaine diversité. Le point qui nous intéresse en particulier ici est l'étape de traduction de l'observation en termes de la règle. Dans le cas d'attributs qualitatifs, cela ne pose aucun problème de traduire "directement" en terme. Par exemple, si une observation contient l'information $genre = \text{M}$, alors l'algorithme ajoutera le terme $genre = \text{M}$ à la règle R. Le problème provient, ici aussi, des attributs quantitatifs. En effet, comme mentionné dans la sous-partie précédente, utiliser l'opérateur d'égalité pour les termes

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

numériques signifie qu'on risque de générer des termes tels que $x = 0.1216$, qui ne correspondent probablement qu'à une unique observation et sont difficilement généralisables sous cette forme.

Par conséquent, une autre méthode est adoptée pour la génération des solutions initiales sur les attributs numériques. Au lieu de l'égalité, les opérateurs d'infériorité et de supériorité seront employés, et choisis au hasard pour chaque nouveau terme. Cependant, on conserve la possibilité d'utiliser l'opérateur d'égalité pour les termes numériques entiers, étant donné que cet opérateur peut avoir un sens dans ce cas. La méthode ainsi modifiée est décrite dans l'Algorithme 3.

Algorithme 3 : Génération des solutions initiales dans MOCA-IQ

```
sélectionner une observation aléatoire Obs qui vérifie la prédiction;
créer une règle vide R;
for attributA ∈ Obs do
    if A est un attribut numérique then
        if A est entier then
            op ← getRandomOperator(<, >, =);
        else
            op ← getRandomOperator(<, >);
        end
        R.ajouterTerme(A, op, Obs.valeur(A));
    else
        R.ajouterTerme(A, " = ", Obs.valeur(A));
    end
end
réduire la taille de R;
ajouter des termes aléatoires;
supprimer des termes aléatoires;
```

Le point central de la modélisation concerne cependant la définition du voisinage des termes d'une règle. Ici, il s'agit donc de définir une méthode pour générer les voisins d'un terme portant sur un attribut numérique, tel que $A2 > 12.5$. Dans la suite de cette partie, plusieurs méthodes sont proposées pour répondre à cette question.

3.3.3 Voisinage des termes numériques

Nous nous intéressons ici exclusivement à l'opérateur *replace_term* (voir Section 2.2.1) ; les quatre autres demeurent inchangés car indépendants du type de données.

3.3.3.1 Généralités

Par termes voisins, on entend les termes par lesquels un terme original sera remplacé dans une règle de classification. Lorsque l'ensemble des valeurs possibles pour l'attribut de ce terme est fini, le voisinage est obtenu en remplaçant, de manière exhaustive, par toutes les autres valeurs possibles. Dans le cas d'un ensemble infini, il est donc nécessaire de définir dans quelle mesure la valeur du terme original sera perturbée. Nous proposons de définir le voisinage d'un terme $A < v$ d'attribut numérique A et de valeur v comme l'union des termes voisins suivants :

- $A < (v + d)$
- $A < (v - d)$
- $A < v$ et $A > w$

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

où d est la variation apportée à v, appelée *step*, et w est une valeur telle que $w < v$, permettant de définir un intervalle clos de valeurs. L'opérateur d'infériorité (dans $A < v$) est choisi ici à des fins d'exemple, mais la méthode est identique pour les autres opérateurs. A noter que l'opérateur d'un terme reste par conséquent fixe et non sujet aux perturbations. Ce choix a été réalisé en considérant qu'une telle perturbation modifierait trop largement le terme pour que le nouveau puisse être réellement considéré comme un voisin de l'original.

Le troisième élément de la liste permet de traduire la notion d'intervalles de valeurs présente dans la version originale de l'algorithme. Ajouter un terme de même attribut mais d'opérateur opposé permet de clore un intervalle.

Les deux premiers éléments de la liste sont ceux qui forment le cœur du problème, à savoir déterminer dans quelle mesure il convient de faire varier la valeur d'un terme. La suite de cette partie sera consacrée à l'étude de différentes méthodes permettant de fixer la valeur de d, qui sera également appelée *step*.

3.3.3.2 Variante basique : statique

La définition de l'amplitude de la perturbation à apporter pose un problème identique à celui rencontré par les algorithmes génétiques appliqués à des problèmes d'optimisation continue. En effet, ces algorithmes doivent établir à la fois des opérateurs de croisement, ou *cross-over*, et des opérateurs de mutation. Ce sont des derniers qui rappellent assez naturellement le problème qui nous occupe dans cette partie, étant donné qu'il s'agit là aussi de faire varier une valeur numérique. Cependant, les opérateurs de mutation proposés dans la littérature sont le plus souvent assez simples, l'aspect innovant de ces travaux se concentrant en général sur la définition d'opérateurs de cross-over performants. Ainsi, les perturbations de la valeur sont obtenues en ajoutant ou en soustrayant une valeur unitaire fixe ou une simple valeur aléatoire.

Par conséquent, une première méthode basique pour notre problème est de fixer le step à une valeur constante. Néanmoins, il convient de considérer que différents attributs numériques peuvent avoir des plages de valeurs très différentes. Un attribut peut admettre des valeurs entre 0 et 100, tandis qu'un autre est défini sur l'espace entre 0.001 et 0.1. Afin de tenir compte de ces disparités, une valeur de step différente est utilisée pour chaque attribut, dépendant de la taille de sa plage de valeurs. Cette taille est définie comme suit, pour un attribut A :

$$plage(A) = valmax(A) - valmin(A) \tag{3.2}$$

Cette approche est similaire à celle utilisée dans [36], qui définit également une valeur de mutation, appelée *creep rate*, proportionnelle à l'amplitude de la plage de valeurs de l'attribut. Le step associé à chaque attribut A est donc égal à une fraction de plage(A) ou, plus formellement :

$$step(A) = plage(A) * (p/100) \tag{3.3}$$

où p est un paramètre de l'algorithme à fixer. Des tests empiriques ont été menés afin de déterminer quelle valeur de p apporte les meilleures performances. Aucune différence significative n'a été trouvée, mais de faibles valeurs ($p < 5$) semblent mieux convenir pour cette méthode. Dans l'implémentation, une valeur de $p = 2$ a été retenue.

3.3.3.3 Variante aléatoire avec décroissance progressive

Cette première option pour fixer la valeur du step présente l'avantage d'être simple, mais n'est pas tout à fait satisfaisante d'un point de vue conceptuel. En premier lieu, l'aspect statique de la méthode, qui utilise une valeur fixe pour la variation, signifie que les capacités d'optimisation de l'algorithme sur les attributs numériques sont limitées. La granularité choisie a priori ne

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

correspond peut-être pas à celle qui serait la plus appropriée pour chaque attribut. De plus, cette méthode n'est pas adaptative, elle reste constante durant tout le processus d'optimisation et ne tire aucune information de l'état du processus au moment présent.

Une seconde approche vise à répondre à ces limitations tout en restant le plus simple possible. Dans celle-ci, la valeur du step est fixée à une valeur aléatoire chaque fois que le voisinage d'un terme numérique doit être généré. Cette valeur aléatoire suit une distribution normale centrée sur la valeur actuelle du terme, et utilise la variance associée à l'attribut concerné. Nous implémentons un système où la variance est calculée au moment du chargement des données en mémoire, en utilisant les différentes valeurs relevées dans les données pour cet attribut. De cette manière, nous espérons tirer parti de la répartition des données : pour un attribut dont les valeurs sont assez groupées, la variation sera plus faible que pour un attribut dont les valeurs sont plus dispersées. L'aspect aléatoire du procédé permet, quant à lui, une plus grande diversité dans les valeurs par rapport à une variation statique. Les méthodes d'optimisation aléatoire présentent également une propriété de convergence théorique vers l'optimum global [12], laquelle est intéressante en dépit du fait que les temps de convergence sont souvent prohibitifs dans les problèmes réels [146].

Afin de répondre à la seconde limitation soulevée plus haut, un mécanisme de décroissance progressive de la variance est ajouté au processus. Ce processus est fortement inspiré de l'heuristique du *Recuit Simulé*, ou *Simulated Annealing* (SA) [98], pour l'optimisation continue. Après chaque génération d'une valeur aléatoire, la variance (liée à un attribut) qui a été utilisée est diminuée légèrement. Plus précisément, la variance σ associée à un attribut A est mise à jour, après chaque génération, suivant la formule :

$$\sigma(A) = \sigma(A) * r \tag{3.4}$$

avec r un paramètre à fixer. De la même manière que pour la première approche, des tests empiriques ont été menés afin de déterminer la valeur de r la plus adéquate, et une valeur de $r = 0.99$ a été retenue.

Pour justifier cette approche, on estime qu'à mesure des évaluations, les termes ont tendance à converger vers les valeurs optimales, et qu'il convient par conséquent de réduire progressivement l'amplitude des variations. Ainsi, les variations seront en moyenne plus larges en début d'optimisation, et plus réduites en fin de processus. L'aspect aléatoire garantit toujours la possibilité de générer des variations importantes même avec une faible variance, mais de moins en moins fréquemment. Le SA utilise également une décroissance progressive visant à accompagner la convergence du processus d'optimisation. Il inclut souvent un mécanisme permettant d'augmenter fortement le taux décroissant, en général lorsque celui-ci atteint un certain seuil. Cela permet d'éviter au système de se figer à un endroit non-optimal de l'espace de recherche. Dans la méthode proposée, un mécanisme simple de restart est ajouté : à chaque fois que l'algorithme d'optimisation converge et redémarre (c'est-à-dire, lorsque toutes les solutions de l'archive ont été explorées), les variances sont réinitialisées.

3.3.3.4 Variante avec adaptation automatique

La méthode précédente permet d'apporter de la diversité dans l'amplitude des perturbations de par son aspect aléatoire, et de suivre la convergence supposée des solutions générées par le processus d'optimisation, via la décroissance progressive des variances. Cependant, cette méthode présente l'inconvénient de n'être pas adaptable et de suivre un schéma prédéfini, sans tenir compte de l'état actuel de la recherche locale.

L'adaptation automatique des paramètres d'un algorithme d'optimisation est un sujet complexe qui a été largement étudié, et plusieurs méthodes ont été proposées à cette fin, formant le domaine appelé Reactive Search Optimization [15]. Une des méthodes les plus reconnues est

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

l'algorithme de Solis-Wets [153]. L'idée centrale de celui-ci consiste à enregistrer les performances du processus d'optimisation, en termes de *succès* et *échecs*, et d'adapter les paramètres de la recherche en conséquence. Plus précisément, si plusieurs succès consécutifs sont notés, alors le step de la recherche locale est augmenté, afin d'exploiter plus rapidement une direction prometteuse. A l'inverse, si plusieurs échecs successifs se produisent, alors la taille du step est réduite. Cette idée est similaire à la *règle du 1/5* développée dans le cadre des Stratégies d'Évolution [76]. Suivant cette règle, le taux de succès sert à ajuster l'amplitude de la perturbation : un fort taux de succès l'augmente, tandis qu'un faible taux la diminue. Cette méthode a été par exemple utilisée avec succès dans le cadre d'un algorithme mémétique hybride pour l'optimisation continue [123]. Afin de pouvoir utiliser cet outil dans notre problème, il est nécessaire de répondre à deux questions. Tout d'abord, qu'est-ce qu'un succès dans le contexte applicatif qui nous occupe, à savoir une métaheuristique de recherche locale multi-objectif avec une archive de solutions ? Ensuite, à quel niveau convient-il d'imputer ces succès (ou échecs) : concernent-ils la solution entière, la règle, le terme, ou l'attribut associé ? Pour ce qui est de la première question, deux réponses semblent possibles :

— Un succès a lieu lorsqu'une solution générée par recherche locale est meilleure que la solution d'origine.

— Un succès a lieu lorsqu'une solution générée par recherche locale est insérée dans l'archive des solutions.

Déclarer qu'une solution est meilleure qu'une autre est difficile dans le cadre d'un problème multi-objectif. Pour cette raison, la notion de Pareto-dominance est utilisée pour comparer des solutions entre elles. Les deux options possibles pour définir un succès paraissent également valides. Nous choisissons la seconde : un succès est enregistré lorsqu'au moins une solution est insérée dans l'archive (parmi celles générées par recherche locale), un échec lorsque ce n'est pas le cas. Ce choix est principalement motivé par le fait que, à l'inverse de la première, cette option ne requiert pas d'évaluation supplémentaire de la qualité des solutions à comparer. A noter que la première option est plus restrictive que la seconde : un succès au sens de la première option est toujours un succès au sens de la seconde, l'inverse n'est pas toujours vrai.

Il reste maintenant à répondre à la seconde question : à quel niveau enregistrer les succès et échecs ? Si l'algorithme de recherche locale crée une nouvelle solution en explorant le voisinage d'une solution existante, et que cette nouvelle solution est ajoutée à l'archive, cela signifie que la mutation apportée a été bénéfique pour améliorer la qualité de la solution. Ainsi, un des termes numériques d'une des règles de la solution a été modifié (on ne considère ici que l'opérateur *replace_term*), et cette modification a apporté un gain de qualité. Dès lors, faut-il considérer que la mutation a eu un effet positif sur le terme, sur la règle ou sur la solution complète ? Cette question admet davantage de réponses possibles que la précédente, mais il est possible de grouper ces réponses en deux catégories :

— Gestion locale : enregistrer les succès/échecs pour chaque solution ou chaque sous-partie de solution (règle ou terme).

— Gestion globale : enregistrer les succès/échecs pour chaque attribut.

La première option paraît intéressante pour exploiter les particularités locales de l'espace de recherche et pour ajuster l'algorithme d'optimisation plus précisément. Néanmoins, cette option pose plusieurs problèmes qui rendent sa mise en œuvre difficile ou qui limitent son intérêt. Tout d'abord, il paraît impossible de faire une synthèse de l'ensemble des comportements locaux afin d'en extraire quelque information sur le problème ou l'espace de recherche en général. Ensuite, l'information accumulée peut être perdue à tout moment : les règles et termes peuvent être supprimés, et les solutions marquées comme explorées ne seront plus utilisées par la suite. Enfin, il est nécessaire de procéder à un certain nombre d'explorations successives du voisinage pour

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

modifier la variance associée. En multipliant le nombre de compteurs, on augmente la dispersion de l'information, avec le risque de n'extraire au final rien d'utile pour ajuster les paramètres de l'algorithme. L'idée même de requérir plusieurs modifications successives est cohérente avec les techniques de recherche locale, mais s'adapte mal à un algorithme à population tel que MOCA-I.

Compte tenu de ces problèmes, nous retenons plutôt la seconde option, qui consiste à enregistrer les succès et échecs pour chaque attribut. De cette manière, l'information est préservée même en cas de suppression d'éléments, et reste exploitable car synthétisée à un niveau plus élevé. Cette option n'en présente pas moins certaines limites, qui sont discutées plus tard dans cette partie. Le processus dans son ensemble, inspiré de l'algorithme de Solis-Wets, peut donc être résumé par le pseudo-code présenté dans l'Algorithme 4.

Algorithme 4 : Mécanisme d'adaptation automatique : mise à jour de la variance en fonction des succès ou échecs consécutifs, effectuée après chaque génération d'une nouvelle solution contenant un terme avec un attribut numérique

```
Data : un nouveau terme T' d'attribut numérique A
if la nouvelle solution contenant T' est insérée dans l'archive then
    SuccèsConsécutifs(A)++;
    EchecsConsécutifs(A) = 0;
    if SuccèsConsécutifs(A) == 5 then
        Variance(A) = Variance(A) × 2;
        SuccèsConsécutifs(A) = 0;
    end
else
    EchecsConsécutifs(A)++;
    SuccèsConsécutifs(A) = 0;
    if EchecsConsécutifs(A) == 3 then
        Variance(A) = Variance(A) ÷ 2;
        EchecsConsécutifs(A) = 0;
    end
end
```

Le choix qui a été fait de se placer à un niveau global pour l'enregistrement des succès/échecs n'est pas forcément instinctif, dans la mesure où on peut craindre que l'adaptation du step pour chaque attribut dans son ensemble ne tienne pas compte de la position du terme à modifier dans l'espace de recherche. L'aspect conceptuel de ce problème est discuté à la fin de cette partie à la lumière des résultats obtenus, mais nous pouvons dès à présent essayer de l'atténuer.

Pour cela, nous proposons une variante de cette méthode, dans laquelle la plage de valeurs de chaque attribut numérique est séparée en plusieurs intervalles, et à chaque intervalle est attribuée une variance distincte, qui peut donc varier indépendemment. Ainsi, lorsqu'un voisin est généré pour un terme d'attribut numérique, le processus aléatoire utilise, pour déterminer le step, la variance associée à l'intervalle dans lequel la valeur du terme exploré se situe. La logique derrière ce mécanisme est qu'on s'attend à ce que la recherche locale suivant un attribut numérique se comporte différemment dans des parties différentes de l'espace de recherche. Cette méthode permettrait, par exemple, d'avoir des variances plus importantes dans certaines régions de l'espace, afin de diriger plus rapidement les solutions vers les régions plus "intéressantes", dans lesquelles les variances sont plus faibles afin d'affiner la recherche.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

3.3.4 Expérimentations

Nous cherchons à présent à déterminer laquelle de ces trois variantes de MOCA-IQ est la plus performante : variation fixe (MIQ1), décroissance progressive (MIQ2) ou adaptation automatique (MIQ3). Pour cela, nous suivons le protocole détaillé précédemment : 8 jeux de données, moyenne des mesures sur 30 exécutions indépendantes de chaque algorithme. Les résultats sont présentés dans les Tables 3.9 à 3.12.

	MIQ1	MIQ2	MIQ3
adult	0.641 ± 0.011	0.636 ± 0.012	**0.642 ± 0.012**
arrythmia	0.751 ± 0.016	**0.759 ± 0.021**	0.755 ± 0.02
page-blocks	0.759 ± 0.014	**0.783 ± 0.034**	0.767 ± 0.015
penbased	**0.909 ± 0.012**	0.906 ± 0.016	0.906 ± 0.021
ring	0.843 ± 0.006	0.833 ± 0.007	**0.844 ± 0.004**
segment	0.99 ± 0.003	**0.991 ± 0.003**	0.987 ± 0.004
spambase	0.798 ± 0.012	**0.807 ± 0.01**	0.803 ± 0.009
vowel	0.886 ± 0.016	**0.912 ± 0.015**	0.875 ± 0.017

TABLE 3.9 – Comparaison des approches pour MOCA-IQ, f-mesure, ensemble d'apprentissage

	MIQ1	MIQ2	MIQ3
adult	0.637 ± 0.011	0.632 ± 0.01	**0.639 ± 0.011**
arrythmia	**0.614 ± 0.034**	0.609 ± 0.039	0.604 ± 0.049
page-blocks	0.727 ± 0.027	**0.745 ± 0.042**	0.729 ± 0.029
penbased	**0.903 ± 0.014**	0.896 ± 0.018	0.899 ± 0.02
ring	0.831 ± 0.008	0.822 ± 0.008	**0.832 ± 0.008**
segment	**0.97 ± 0.012**	0.967 ± 0.015	0.968 ± 0.01
spambase	0.791 ± 0.013	**0.796 ± 0.013**	0.79 ± 0.016
vowel	0.774 ± 0.063	**0.787 ± 0.048**	0.759 ± 0.053

TABLE 3.10 – Comparaison des approches pour MOCA-IQ, f-mesure, ensemble de test

	MIQ1	MIQ2	MIQ3
adult	**0.527 ± 0.016**	0.518 ± 0.018	0.526 ± 0.018
arrythmia	0.509 ± 0.033	**0.534 ± 0.052**	0.524 ± 0.039
page-blocks	0.733 ± 0.015	**0.76 ± 0.037**	0.743 ± 0.017
penbased	**0.9 ± 0.013**	0.896 ± 0.017	0.896 ± 0.022
ring	0.668 ± 0.013	0.645 ± 0.016	**0.671 ± 0.009**
segment	0.988 ± 0.004	**0.99 ± 0.004**	0.985 ± 0.005
spambase	0.672 ± 0.023	**0.686 ± 0.018**	0.682 ± 0.022
vowel	0.878 ± 0.017	**0.905 ± 0.016**	0.867 ± 0.019

TABLE 3.11 – Comparaison des approches pour MOCA-IQ, MCC, ensemble d'apprentissage

3.3.5 Discussion des résultats et limites des systèmes proposés

L'observation la plus importante que nous pouvons faire à partir de ces résultats est qu'aucune des trois méthodes proposées ne se démarque particulièrement des autres. La deuxième, basée

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

	MIQ1	MIQ2	MIQ3
adult	**0.523 ± 0.015**	0.514 ± 0.016	0.523 ± 0.017
arrythmia	0.235 ± 0.055	**0.244 ± 0.079**	0.229 ± 0.075
page-blocks	0.699 ± 0.028	**0.72 ± 0.044**	0.704 ± 0.028
penbased	**0.894 ± 0.015**	0.886 ± 0.019	0.889 ± 0.022
ring	0.638 ± 0.018	0.618 ± 0.018	**0.642 ± 0.015**
segment	**0.965 ± 0.013**	0.962 ± 0.017	0.963 ± 0.011
spambase	0.661 ± 0.025	**0.668 ± 0.023**	0.664 ± 0.026
vowel	0.763 ± 0.064	**0.773 ± 0.051**	0.745 ± 0.057

TABLE 3.12 – Comparaison des approches pour MOCA-IQ, MCC, ensemble de test

sur un mécanisme de décroissance progressive du step, se révèle être plus performante que les autres sur une majorité des jeux de données, mais les écarts sont trop faibles et trop irréguliers pour en conclure que cette méthode est réellement mieux adaptée. Les tests statistiques menés sur ces résultats n'ont d'ailleurs relevé aucune différence significative entre ces trois approches pour l'ensemble des jeux de données. Les écarts-type présentés dans ces tableaux sont également assez importants (par rapport aux moyennes) pour renforcer l'idée que ces méthodes donnent des performances très similaires.

Une interprétation possible de cette constatation est que trouver la meilleure valeur pour le step n'est pas un problème crucial pour l'algorithme d'optimisation avec gestion des attributs quantitatifs. En effet, il est plausible que, quelle que soit la méthode employée pour fixer cette valeur, les solutions obtenues par l'algorithme convergent dans l'ensemble vers les mêmes valeurs. Ainsi, les faibles différences entre les valeurs des termes numériques des solutions n'auraient au final qu'une influence mineure sur la qualité de ces solutions. Une telle différence ne peut sans doute provoquer, tout au plus, qu'une modification dans la classification d'un nombre très réduit d'instances du jeu de données considéré ; ceci expliquerait les faibles écarts de performance observés. En d'autres termes, la seule caractéristique indispensable pour une méthode permettant de fixer le step serait d'avoir une granularité "correcte" : assez faible pour approximer convenablement les solutions idéales, mais assez importante pour ne pas se retrouver gelée au cours de la recherche locale. Il convient également de rappeler que le step n'est utilisé que pour l'opérateur *replace_term*, et n'influe donc que sur une partie du voisinage généré.

Un choix non-trivial a été fait lors de l'implémentation des seconde et troisième méthodes : celui d'élaborer le processus de détermination du *step* à un niveau "global", c'est-à-dire commun à toutes les solutions. Par exemple, le processus de décroissance est séparé pour chaque attribut, mais ces attributs sont utilisés dans plusieurs solutions en simultané. Comme nous l'avons vu dans la présentation de la troisième méthode (avec adaptation automatique), une seconde option aurait été de situer ce processus à un niveau plus local, en attribuant, par exemple, un tel mécanisme à chaque solution, ce mécanisme pouvant ainsi évoluer indépendamment des autres en se concentrant sur la solution à laquelle il est associé. Revenons à présent sur ce choix en considérant les résultats présentés ci-dessus. Les observations et postulats qui seront présentés par la suite concernent en premier lieu la troisième méthode (adaptation automatique), mais également la deuxième (décroissance).

L'observation des résultats indique assez clairement que l'intégration de mécanismes plus sophistiqués que l'emploi d'une simple valeur fixe pour le step n'apporte pas le gain de performance initialement escompté. Il est néanmoins très difficile de déterminer avec certitude les raisons de cet état de fait ; nous pouvons tout au plus proposer une explication possible tirée de l'analyse des mécanismes ajoutés. L'aspect le plus important à cet égard est le caractère global d'un tel sys-

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

tème, qui cherche à fixer une valeur unique pour le processus d'optimisation dans son ensemble. Ce principe paraît naturel dans le cadre d'une optimisation par recherche locale, dans lequel les solutions explorées consécutivement seront souvent très similaires entre elles. On peut considérer, dans ce cas, que le mécanisme servant à fixer le step fournira des valeurs proches pour des solutions situées dans une même région de l'espace de recherche. De plus, si ce mécanisme apprend de l'évolution des performances de l'algorithme d'optimisation, cette évolution suivra l'exploration de l'espace de recherche, laquelle se fait en considérant, ici aussi, des solutions proches.

C'est le cas pour les algorithmes utilisant une solution unique (tels que le Simulated Annealing), mais pas pour les algorithmes à population tels que MOCA-I. Dans ce dernier, une solution de l'archive est sélectionnée au hasard à chaque itération, le voisinage de cette solution est exploré et les nouvelles "bonnes" solutions sont ajoutées à l'archive. Par conséquent, des itérations successives de l'algorithme explorent le plus souvent des régions de l'espace de recherche très éloignées les unes des autres, étant donné que les solutions explorées peuvent être elles-mêmes très différentes. De cette manière, une valeur de step adéquate à une itération i ne le sera sans doute pas à l'itération $i + 1$, ce qui affecte sérieusement l'idée même d'apprentissage sur cette valeur. La motivation qui a poussé à l'implémentation de ces mécanismes en dépit de leurs limitations inhérentes est que nous espérions ainsi moduler l'intensité du step en fonction de la convergence progressive de l'ensemble de l'archive (pour la méthode avec décroissance progressive) ou tirer parti des spécificités de chaque attribut (pour la méthode avec adaptation automatique).

A cet égard, il serait intéressant de relever les performances de MOCA-I dans sa variante "exhaustive", dans laquelle l'ensemble des solutions de l'archive sont explorées à chaque itération, couplée avec les mécanismes de détermination du step présentés dans cette partie. En effet, il paraît raisonnable de s'attendre à ce que ces mécanismes soient plus performants dans ce cas, étant donné que l'évolution du step pourrait ainsi suivre la convergence de l'archive entière vers un ensemble optimal de solutions non-dominées. Une étude de ce type n'aurait cependant probablement aucune application concrète, cette variante de MOCA-I ayant été prouvée moins performante de manière générale que celle où une unique solution est choisie à chaque étape [89].

Une autre explication envisageable concerne plus spécifiquement la méthode avec adaptation automatique. Ce mécanisme cherche à extraire de l'information des résultats obtenus en modifiant un terme numérique d'une règle en utilisant une certaine valeur pour le step. Le problème est que ces termes ne sont pas considérés dans le contexte de la règle à laquelle ils appartiennent, ni même de la solution à laquelle cette règle appartient. Autrement dit, un terme dépend en général fortement des autres composantes d'une solution dans sa contribution à la qualité de cette solution, aussi l'information extraite de la modification d'un terme unique ne représenterait-elle aucune connaissance réelle ni exploitable directement.

Dans cette section, nous avons proposé une méthode de gestion des attributs numériques au sein de la recherche locale pour l'algorithme MOCA-IQ. Dans la section suivante, nous comparons cette approche à celle développée précédemment, qui discrétise les attributs numériques en prétraitement. Nous proposons également une méthode combinant ces deux approches et évaluons son efficacité.

3.4 Comparaison et combinaison des approches

3.4.1 Comparaison des deux approches

Nous cherchons à présent à comparer les deux approches proposées pour la gestion des données numériques : 1) discrétisation a priori, et 2) gestion au sein de la recherche locale. Pour cela, nous regroupons dans les Tables 3.13 et 3.14 les résultats obtenus par les deux approches. Pour la première approche, nous considérons l'algorithme MOCA-I avec deux variantes pour les deux

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

méthodes de discrétisation : binning (utilisé dans MOCA-I original), et Fayyad (qui a été montré plus performant que les autres). Pour la seconde approche, nous conservons les trois variantes de MOCA-IQ (MIQ1, MIQ2 et MIQ3), car celles-ci présentent des résultats très proches. Nous ne montrons que les résultats sur la f-mesure ; les résultats sur le MCC aboutissent aux mêmes conclusions.

	Approche 1		Approche 2		
	binning	Fayyad	MIQ1	MIQ2	MIQ3
adult	0.658 ± 0.006	**0.678 ± 0.005**	0.641 ± 0.011	0.636 ± 0.012	0.642 ± 0.012
arrythmia	**0.985 ± 0.007**	0.96 ± 0.008	0.751 ± 0.016	0.759 ± 0.021	0.755 ± 0.02
page-blocks	0.768 ± 0.017	**0.891 ± 0.009**	0.759 ± 0.014	0.783 ± 0.034	0.767 ± 0.015
penbased	**0.999 ± 0.001**	0.996 ± 0.002	0.909 ± 0.012	0.906 ± 0.016	0.906 ± 0.021
ring	0.878 ± 0.009	**0.888 ± 0.007**	0.843 ± 0.006	0.833 ± 0.007	0.844 ± 0.004
segment	0.998 ± 0.002	**1.0 ± 0.0**	0.99 ± 0.003	0.991 ± 0.003	0.987 ± 0.004
spambase	0.786 ± 0.009	**0.914 ± 0.004**	0.798 ± 0.012	0.807 ± 0.01	0.803 ± 0.009
vowel	**1.0 ± 0.0**	0.986 ± 0.011	0.886 ± 0.016	0.912 ± 0.015	0.875 ± 0.017

TABLE 3.13 – F-mesures obtenues sur l'ensemble d'apprentissage

	Approche 1		Approche 2		
	binning	Fayyad	MIQ1	MIQ2	MIQ3
adult	0.655 ± 0.005	**0.674 ± 0.006**	0.637 ± 0.011	0.632 ± 0.01	0.639 ± 0.011
arrythmia	0.645 ± 0.031	**0.758 ± 0.04**	0.614 ± 0.034	0.609 ± 0.039	0.604 ± 0.049
page-blocks	0.722 ± 0.021	**0.837 ± 0.01**	0.727 ± 0.027	0.745 ± 0.042	0.729 ± 0.029
penbased	**0.985 ± 0.006**	0.982 ± 0.005	0.903 ± 0.014	0.896 ± 0.018	0.899 ± 0.02
ring	0.86 ± 0.013	**0.866 ± 0.009**	0.831 ± 0.008	0.822 ± 0.008	0.832 ± 0.008
segment	0.973 ± 0.01	**0.98 ± 0.009**	0.97 ± 0.012	0.967 ± 0.015	0.968 ± 0.01
spambase	0.764 ± 0.01	**0.891 ± 0.007**	0.791 ± 0.013	0.796 ± 0.013	0.79 ± 0.016
vowel	0.882 ± 0.037	**0.927 ± 0.031**	0.774 ± 0.063	0.787 ± 0.048	0.759 ± 0.053

TABLE 3.14 – F-mesures obtenues sur l'ensemble de test

Les résultats montrent clairement que la première approche est plus performante que la seconde. Au sein de la première approche, Fayyad est plus efficace, en particulier sur les ensembles de test, ce qui indique une plus grande robustesse. Il est donc plus efficace d'appliquer une discrétisation adéquate sur les données numériques plutôt que d'adapter l'algorithme d'optimisation pour les traiter directement. Une seconde observation est que les variantes de la seconde approche donnent souvent des résultats inférieurs à ceux de l'algorithme MOCA-I original (avec discrétisation par binning simple). Par conséquent, on peut supposer que la discrétisation en général est la méthode la plus efficace pour traiter ces données. Cette observation a déjà été faite pour d'autres tâches de fouille de données [29], et n'est donc pas anormale. Dans la suite de cette section, nous cherchons néanmoins à voir s'il est possible de tirer parti des deux approches en même temps.

3.4.2 Combinaison des approches : motivations

Nous avons vu précédemment qu'appliquer l'algorithme sur des données discrétisées permettait de restreindre l'espace de recherche, et par conséquent d'arriver plus rapidement à de bonnes solutions. Cependant, cela se fait au prix d'une perte d'information, laquelle risque de limiter la qualité des solutions finales en empêchant une optimisation avec une granularité plus faible que

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

celle définie par la discrétisation. A l'inverse, travailler directement sur les données numériques permet potentiellement une optimisation plus fine, mais rend l'exploration complète de l'espace de recherche plus délicate.

Au vu de ces constatations, une option serait donc de coupler les deux approches : rechercher un premier ensemble de bonnes solutions en utilisant les données discrétisées, puis optimiser plus finement ces solutions via une recherche locale continue sur les données numériques originales. On définit dans ce cas une méthode hybride en mode *relais* [92]. De cette manière, on suppose pouvoir tirer le meilleur parti des deux approches en combinant leurs avantages respectifs : une couverture rapide de l'espace de recherche et une optimisation précise des solutions.

Cette approche possède une certaine ressemblance avec le principe développé dans l'algorithme de Reformulation Local Search [23]. Il s'agit en effet de changer la formulation du problème en cours d'optimisation ; dans notre cas, en passant de données discrètes à des données numériques. Les deux parties (discrète et continue) de l'algorithme ont été décrites en détail dans les parties précédentes, aussi reste-t-il à expliciter la transition entre les deux, c'est-à-dire la reformulation en elle-même.

3.4.3 Reformulation proposée

Le problème à résoudre ici est le suivant : comment transformer un ensemble de règles sur données discrétisées en son équivalent sur données numériques "originales" ?

Répondre à cette question paraît assez simple ; il s'agit juste de remplacer les attributs discrétisés en leurs versions numériques respectives, et de faire correspondre, aux intervalles utilisés comme valeurs dans la version discrète, les valeurs bornant ces intervalles. Cependant, il convient de tenir compte de certaines particularités lors de la traduction des intervalles. Les différents cas pouvant se présenter sont résumés dans la Table 3.15, avec leur traduction en termes numériques.

Terme discrétisé	Terme numérique
$a < [x_i, x_{i+1}]$	$a < x_i$
$a > [x_i, x_{i+1}]$	$a > x_{i+1}$
$a = [x_i, x_{i+1}]$	$a > x_i$ AND $a < x_{i+1}$

TABLE 3.15 – Traduction d'un terme utilisant un attribut discrétisé en un terme équivalent utilisant un attribut numérique

Une variante de cette méthode a aussi été envisagée : la reformulation se ferait dans les deux sens (discret vers continu, et vice-versa), et ce de manière cyclique. De cette manière, la phase de recherche sur l'espace discret agirait comme une sorte de *shaking* en apportant des perturbations plus importantes aux solutions trouvées par recherche sur l'espace continu. En parallèle, les valeurs des termes numériques trouvées par cette dernière seraient utilisées pour modifier la discrétisation. L'idée derrière une telle méthode est proche de celle développée dans [23], qui étend le principe du *Variable Neighborhood Search* [77]. Cette variante n'a cependant pas été implémentée, mais elle demeure une piste de recherche potentiellement intéressante.

3.4.4 Expérimentations

Le principe de cette méthode étant de coupler recherches locales discrète et continue, il convient donc de déterminer quelle variante de recherche locale continue sera employée, parmi celles proposées dans la Section 3.3. Plusieurs variantes sont testées, basées sur MIQ1 et MIQ3 pour la recherche locale continue. MIQ2, avec décroissance progressive, n'est pas incluse dans

© 2017 Tous droits réservés. lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

ces tests car nous estimons qu'elle perdrait de son intérêt dans le cas de solutions ayant déjà convergé vers des régions intéressantes de l'espace de recherche.

La première phase de ces différentes méthodes *relais* est identique pour chacune ; il s'agit d'une recherche locale sur données discrétisées par Fayyad [58]. Pour cette raison, l'archive des solutions obtenue par cette première partie est enregistrée (pour chacune des 30 itérations), et sert ainsi de point de départ commun aux variantes de recherche locale continue. De cette manière, nous nous concentrons sur l'apport de cette seconde phase, en mettant les méthodes sur un pied d'égalité au départ. La première méthode (Fayyad30) présentée dans les Tables 3.16 et 3.17 correspond à la solution trouvée par MOCA-I sur données discrétisées par Fayyad, ce qui constitue la phase 1 du processus. Les deuxième et troisième colonnes (MIQ1_2p et MIQ1_5p) correspondent à deux variantes de MOCA-IQ avec l'approche basique et un *step* fixe respectivement égal à 2% et 5% de la plage de valeurs. La quatrième colonne (MIQ3) correspond à la méthode avec adaptation continue, sans séparation de l'espace des attributs (voir Section 3.3.3).

	Fayyad30	MIQ1_2p	MIQ1_5p	MIQ3
adult	**0.676 ± 0.005**	0.665 ± 0.007	0.67 ± 0.006	0.669 ± 0.006
arrythmia	0.958 ± 0.008	0.957 ± 0.013	0.955 ± 0.014	**0.96 ± 0.011**
page-blocks	0.889 ± 0.011	0.886 ± 0.012	0.886 ± 0.012	**0.894 ± 0.012**
penbased	**0.995 ± 0.002**	0.994 ± 0.002	**0.995 ± 0.002**	0.994 ± 0.002
ring	0.886 ± 0.007	0.895 ± 0.008	0.893 ± 0.008	**0.898 ± 0.008**
segment	**1.0 ± 0.0**	**1.0 ± 0.0**	**1.0 ± 0.0**	**1.0 ± 0.0**
spambase	**0.912 ± 0.004**	**0.912 ± 0.004**	**0.912 ± 0.004**	**0.912 ± 0.004**
vowel	0.985 ± 0.011	**0.992 ± 0.007**	**0.992 ± 0.007**	0.99 ± 0.008

TABLE 3.16 – F-mesures obtenues sur l'ensemble d'apprentissage

	Fayyad30	MIQ1_2p	MIQ1_5p	MIQ3
adult	**0.672 ± 0.005**	0.66 ± 0.007	0.665 ± 0.006	0.666 ± 0.006
arrythmia	0.759 ± 0.03	0.762 ± 0.037	0.755 ± 0.031	**0.767 ± 0.031**
page-blocks	0.839 ± 0.018	0.835 ± 0.017	0.838 ± 0.016	**0.844 ± 0.022**
penbased	**0.983 ± 0.005**	0.98 ± 0.005	0.982 ± 0.005	**0.983 ± 0.005**
ring	0.864 ± 0.009	0.871 ± 0.009	0.872 ± 0.008	**0.873 ± 0.008**
segment	**0.979 ± 0.01**	**0.979 ± 0.01**	**0.979 ± 0.01**	**0.979 ± 0.01**
spambase	**0.891 ± 0.007**	0.89 ± 0.006	0.89 ± 0.007	0.89 ± 0.006
vowel	**0.922 ± 0.032**	0.912 ± 0.032	0.913 ± 0.041	0.918 ± 0.033

TABLE 3.17 – F-mesures obtenues sur l'ensemble de test

D'après ces résultats, on observe que le système de *relais* discret-continu n'apporte au mieux que des améliorations marginales de la qualité des solutions. Les mesures étant très proches les unes des autres, il est impossible de déceler des différences majeures entre les méthodes ; la troisième variante, qui reprend la technique d'adaptation automatique, semble cependant être légèrement plus performante que les autres dans la majorité des cas. Néanmoins, l'observation majeure reste le fait qu'ajouter une seconde phase d'optimisation en changeant la formulation du problème n'amène pas de hausse sensible des performances de l'algorithme. Cela peut être dû, soit à des lacunes dans le potentiel exploratoire des méthodes de recherche locale continue, soit au fait que les solutions trouvées par recherche locale discrète (lors de la première phase)

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

sont déjà très proches des solutions “optimales”, et qu'il est par conséquent difficile de les améliorer de manière significative. Déterminer laquelle de ces options est la plus vraisemblable est compliqué ; nous espérons cependant que la comparaison de MOCA-I avec d'autres méthodes de classification performantes de la littérature permettra de répondre à ce problème. En effet, si ces autres méthodes ne trouvent pas de meilleure solution, il est probable que les modèles de classification générés par MOCA-I soient très proches des solutions “optimales”. Cette comparaison sera effectuée à la fin de cette partie.

Les légères dégradations occasionnelles de la qualité entre les phases 1 et 2 peuvent paraître étranges, étant donné que la méthode employée ne fait que traduire les solutions d'une phase à l'autre, et qu'il est impossible dans MOCA-I de supprimer la meilleure solution de l'archive. Ces variations sont en réalité dues au changement de l'ensemble d'apprentissage : au début de la seconde phase, les données (avec attributs quantitatifs) sont de nouveau chargées en mémoire, mais la partition apprentissage/test est différente de celle opérée pour la première phase, car aléatoire.

3.5 Comparaison à la littérature

Les sections précédentes de ce chapitre se sont attachées à présenter différentes méthodes et variantes pour gérer les attributs quantitatifs, que ce soit par pré-traitement ou en modifiant l'algorithme de recherche locale de MOCA-I. Ces variantes ont été comparées entre elles afin de déterminer empiriquement les plus performantes. Dans cette partie, nous nous intéressons à présent à la comparaison de ces méthodes les plus efficaces à un ensemble d'algorithmes de classification issus de la littérature, étape nécessaire pour valider l'apport des mécanismes proposés.

3.5.1 Méthodes de classification

Un nombre conséquent d'algorithmes ont été proposés afin de répondre au problème de classification en général. De la même manière que dans [89], nous ne nous intéressons ici qu'aux algorithmes ayant été conçus, ou adaptés, pour pouvoir gérer des données asymétriques (ou déséquilibrées). Une première comparaison sur 22 algorithmes a été réalisée par [59], et les expérimentations menées dans [89] ont permis d'identifier trois méthodes robustes et plus performantes en moyenne que les autres : BioHEL [13], C4.5-CS [158] et Ripper [34]. Comme pour la comparaison des méthodes de discrétisation décrite précédemment, nous utilisons les implémentations de ces algorithmes fournies dans le framework KEEL [6] avec les paramètres par défaut (qui sont ceux suggérés par les auteurs respectifs de ces algorithmes). Quand celles-ci existent, nous choisissons les versions *cost-sensitive* de ces méthodes, mieux adaptées aux problèmes de classification sur données déséquilibrées. Dans ces expérimentations, nous laissons les algorithmes s'exécuter jusqu'à convergence de la méthode.

Nous comparons ces méthodes à la variante la plus efficace de MOCA-IQ. Celle-ci correspond à une discrétisation des données par Fayyad, avec une amélioration de la méthode d'archivage telle que décrite dans la Section 3.7. Les résultats sont présentés dans les Tables 3.18 et 3.19.

3.5.2 Résultats et discussion

Une observation intéressante ici est que MOCA-IQ donne des résultats moins bons que les autres méthodes sur l'ensemble d'apprentissage, mais meilleurs sur l'ensemble de test, et ce de manière constante sur tous les jeux de données. Ceci montre que le modèle de classification généré s'adapte mieux aux nouvelles données et ne traduit pas trop directement les données utilisées par l'apprentissage. En d'autres termes, l'algorithme semble moins sujet au *sur-apprentissage*. Cette

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

	MOCA-IQ	Ripper	C45CS	BioHEL
adult	0.695 ± 0.004	0.63 ± 0.014	0.705 ± 0.015	**0.735 ± 0.008**
arrythmia	0.868 ± 0.018	0.899 ± 0.036	0.951 ± 0.014	**1.0 ± 0.001**
page-blocks	0.907 ± 0.009	**0.945 ± 0.01**	0.91 ± 0.021	0.884 ± 0.014
penbased	0.996 ± 0.002	**0.998 ± 0.001**	0.987 ± 0.008	0.996 ± 0.005
ring	0.943 ± 0.021	0.944 ± 0.009	**0.985 ± 0.002**	0.951 ± 0.017
segment	**1.0 ± 0.0**	0.997 ± 0.004	0.985 ± 0.007	0.993 ± 0.005
spambase	0.942 ± 0.006	0.956 ± 0.006	0.958 ± 0.008	**0.975 ± 0.004**
vowel	0.984 ± 0.01	0.969 ± 0.043	0.914 ± 0.081	**0.993 ± 0.008**

TABLE 3.18 – Comparaison de MOCA-IQ avec d'autres algorithmes de classification, f-mesure, ensemble d'apprentissage

	MOCA-IQ	Ripper	C45CS	BioHEL
adult	**0.689 ± 0.006**	0.621 ± 0.013	0.649 ± 0.012	0.683 ± 0.005
arrythmia	0.661 ± 0.034	**0.704 ± 0.034**	0.652 ± 0.056	0.589 ± 0.048
page-blocks	**0.841 ± 0.018**	0.832 ± 0.019	0.815 ± 0.019	0.782 ± 0.019
penbased	**0.984 ± 0.004**	0.974 ± 0.005	0.965 ± 0.008	0.971 ± 0.007
ring	**0.924 ± 0.024**	0.85 ± 0.008	0.895 ± 0.007	0.919 ± 0.009
segment	**0.979 ± 0.01**	0.96 ± 0.013	0.959 ± 0.01	0.961 ± 0.011
spambase	**0.903 ± 0.006**	0.889 ± 0.006	0.896 ± 0.007	0.899 ± 0.006
vowel	**0.928 ± 0.033**	0.871 ± 0.047	0.853 ± 0.051	0.819 ± 0.11

TABLE 3.19 – Comparaison de MOCA-IQ avec d'autres algorithmes de classification, f-mesure, ensemble de test

observation rejoint celles faites dans [89], où il a été montré que les modèles générés par MOCA-I sont en général plus compacts, c'est-à-dire qu'ils comportent moins de règles, ou des règles moins longues, que ceux produits par d'autres algorithmes. Cette compacité traduit une capacité à isoler les attributs les plus discriminants pour la classification. Au-delà d'un pouvoir prédictif accru sur de nouvelles données, cette compacité implique que les modèles de classification produits sont plus facilement compréhensibles pour l'utilisateur, ce qui peut s'avérer intéressant dans le contexte applicatif qui nous intéresse. La Table 3.20 ci-dessous présente les rangs moyens tels qu'utilisés par les tests statistiques.

Algorithme	Rang f-mesure	Rang MCC
MOCA-IQ	**1.125**	**1.125**
Ripper	2.875	2.75
C45CS	3.25	3.25
BioHEL	2.75	2.875

TABLE 3.20 – Rangs moyens mesurés sur les ensembles de test

Les tests statistiques menés relèvent une dominance significative de MOCA-IQ sur les trois autres algorithmes de classification, avec une *p-value* < 0.05 pour chacune. Dans cette section, nous avons donc validé que MOCA-IQ est une méthode efficace pour la classification sur données numériques. Dans la suite de ce chapitre, nous explorons deux pistes d'amélioration de l'algorithme de classification, indépendemment du type de données utilisées.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

3.6 Sélection d'opérateurs de voisinage

La première piste d'amélioration concerne la sélection dynamique des opérateurs de voisinage. Nous exposons d'abord les motivations justifiant l'intégration d'un tel mécanisme, puis nous détaillons le fonctionnement de la méthode proposée. Enfin, nous présentons des expérimentations visant à mesurer l'apport de ce mécanisme.

3.6.1 Motivations

Dans l'implémentation actuelle, MOCA-IQ explore de manière exhaustive l'ensemble du voisinage de la solution choisie à chaque itération. Ce processus est très coûteux en termes de temps d'exécution, principalement en raison de l'évaluation de chaque nouvelle solution générée, ce qui implique de parcourir l'ensemble des données chargées en mémoire à chaque fois. Etant donné qu'aucune méthode satisfaisante n'a pu être trouvée pour réduire le temps nécessaire à l'évaluation, et dans le but d'accélérer l'algorithme d'optimisation dans son ensemble, nous cherchons dans cette partie à nous éloigner de l'exhaustivité de l'exploration du voisinage. Le but ici est de réduire le nombre de nouvelles solutions générées, tout en conservant les propriétés exploratoires de l'algorithme. En effet, un défaut de MOCA-I est qu'il tend à converger plus lentement que la plupart des autres algorithmes de classification.

Comme mentionné dans la description générale de MOCA-I, les voisins d'une solution (c'est-à-dire d'un ensemble de règles) sont obtenus en appliquant cinq opérateurs de voisinage : *add_rule*, *remove_rule*, *add_term*, *remove_term*, *replace_term*. Jusqu'à présent, nous avons surtout étudié le dernier, étant donné que celui-ci est dépendant du type de données considérées, et change donc avec l'ajout de données numériques et temporelles.

Nous avons souhaité analyser la pertinence de l'utilisation de ces opérateurs en analysant leur apport et leur coût d'utilisation. Pour cela, nous avons relevé, pour chaque opérateur, le nombre total de solutions qu'il avait servi à engendrer, ainsi que le nombre de solutions parmi celles-ci qui avaient été insérées dans l'archive. Cela nous renseigne à la fois sur le poids de l'opérateur sur le temps d'exécution total (via le nombre de solutions générées) et sur l'efficacité de chaque opérateur (via le taux de succès). Les données utilisées ici sont celles qui ont servi pour comparaison dans le Chapitre 3. Ces résultats sont présentés dans la Figure 3.4. Pour chaque opérateur, la première ligne contient le nombre de solutions générées par cet opérateur qui ont été conservées dans l'archive. La seconde ligne contient le nombre total de solutions générées par l'opérateur. A partir de ces deux informations, l'efficacité de l'opérateur est calculée comme le pourcentage de solutions ayant été conservées dans l'archive. Ces trois lignes sont répétées pour chaque opérateur. Dans cette table, chaque colonne correspond à un jeu de données différent. En raison de contraintes d'implémentation, nous n'avons pas pu mesurer ces valeurs pour l'opérateur *remove_rule* ; d'autres expérimentations ont cependant permis de vérifier que que celui-ci montre un comportement similaire à celui des autres opérateurs.

La première observation à partir de ces résultats est que les opérateurs ont des performances très différentes entre eux. Notamment, *remove_term* produit moins de voisins que les autres opérateurs, mais montre une grande efficacité en générant une majorité de solutions qui améliorent l'archive (jusqu'à 90% pour le jeu de données *segment*). La seconde observation est que la performance d'un opérateur varie en fonction des données. Par exemple, l'efficacité de *add_rule* varie entre moins de 1% et 24%. Par conséquent, dans la suite de cette section, nous étudions la possibilité d'un mécanisme d'adaptation automatique pour utiliser en priorité l'opérateur le plus efficace compte tenu de la situation.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

	adult	arcene	arrythmia	page-blocks	penbased	ring	segment	spambase	vowel
Add-rule (success)	42085	14873	248516	87922	107405	55091	220081	58271	1909574
Add-rule (evals)	248566	6289900	15565869	1608806	443946	2993139	10290715	7741582	13027310
Add-rule (%success)	**16,93**	**0,24**	**1,60**	**5,47**	**24,19**	**1,84**	**2,14**	**0,75**	**14,66**
Add-term (success)	79058	11181	321656	388931	224015	495771	959294	316663	2456954
Add-term (evals)	1110216	24738295	46649194	5454434	2836450	4791713	14537007	11951809	25414076
Add-term (%success)	**7,12**	**0,05**	**0,69**	**7,13**	**7,90**	**10,35**	**6,60**	**2,65**	**9,67**
Remove-term (success)	12492	25	141555	6010	15056	24185	112132	92663	60471
Remove-term (evals)	16794	50	278001	7525	31808	31415	123585	119740	98381
Remove-term (%success)	**74,38**	**50,00**	**50,92**	**79,87**	**47,33**	**76,99**	**90,73**	**77,39**	**61,47**
Replace-term (success)	40493	3728	239488	776946	146348	986555	2045150	479380	3447146
Replace-term (evals)	186851	16604	600035	2794381	788119	1990568	4997513	904022	9537006
Replace-term (%success)	**21,67**	**22,45**	**39,91**	**27,80**	**18,57**	**49,56**	**40,92**	**53,03**	**36,14**

FIGURE 3.4 – Performance des opérateurs de voisinage : nombre de solutions conservées dans l'archive, nombre total de solutions générées, pourcentage de solutions conservées

3.6.2 Méthode proposée

Nous proposons dans cette partie une méthode permettant de n'employer, pour une itération donnée, qu'un seul des opérateurs lors de la génération du voisinage d'une solution. Par exemple, si *add_term* est choisi, alors seules les solutions dans lesquelles un terme est ajouté à une règle de la solution seront générées et évaluées. L'objectif est bien entendu de choisir, à chaque itération, le "bon" opérateur, c'est-à-dire celui dont on espère qu'il permettra d'obtenir de "bonnes" nouvelles solutions.

Ce problème, ainsi formulé, est proche du dilemme "exploration-exploitation" [33]. Celui-ci, très présent en apprentissage automatique, consiste, pour un processus décisionnel, à la fois à acquérir de l'information sur le système et à prendre des décisions basées sur les connaissances qu'il a acquises sur ce système. On peut voir la première tâche, appelée *exploration*, comme une phase dans laquelle l'agent essaie de nouvelles options ou des options potentiellement moins performantes, afin de compléter ses connaissances. D'un autre côté, en phase *d'exploitation*, l'objectif est de maximiser le gain en choisissant la solution jugée comme étant la meilleure à cet instant. Dans le cas qui nous intéresse, l'exploration consisterait à appliquer des opérateurs moins performants, et l'exploitation à l'opérateur considéré comme le meilleur à ce moment.

De nombreuses stratégies ont été développées afin de répondre au dilemme "exploration-exploitation", mais celui-ci demeure un problème ouvert. Nous ne détaillons pas ici l'ensemble des méthodes proposées sur le sujet ; le lecteur intéressé peut se référer à [60] pour un état de l'art récent des techniques existantes et une explication plus complète du problème et de ses applications. Dans tous les cas, une stratégie de décision comporte deux composantes distinctes :

— un mécanisme de récompense (aussi appelé *reward* ou *credit assignment*), qui sert à fixer et à mettre à jour une mesure d'intérêt, ou de qualité, de chaque opérateur
— un mécanisme de sélection, qui choisit l'opérateur en fonction de sa qualité

Le premier élément à définir concerne donc l'attribution de *crédit* à un opérateur en fonction de sa performance lorsque celui-ci est choisi. Diverses méthodes existent pour cela, la plupart faisant usage de la qualité des nouvelles solutions générées. Le fait que l'algorithme de recherche locale génère plusieurs nouvelles solutions à chaque étape n'est habituellement pas un problème ; certaines méthodes utilisent la somme ou la moyenne des qualités de chaque solution, d'autres comparent directement les populations avant et après modification. Le problème est cependant plus complexe dans un contexte multi-objectif, étant donné qu'il devient alors difficile de mesurer directement la qualité d'une solution. Une méthode répandue dans ce cas consiste à considérer la somme pondérée de plusieurs mesures (en incluant le plus souvent une mesure de diversité).

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Cependant, nous avons vu dans la Section 2.1.3 que ce genre d'approche présentait certaines limitations. Afin de répondre à ce problème, nous proposons de nous inspirer des travaux qui ont été menés dans le domaine des *indicator-based algorithms* [178]. L'idée générale de ceux-ci est de ramener un problème multi-objectif à un problème mono-objectif, en appliquant l'optimisation sur un indicateur unique, calculé à partir des ensembles de solutions générés. L'un des avantages affichés de cette méthode est qu'elle ne requiert pas l'ajout de mesures de diversité. En contrepartie, son efficacité dépend grandement du choix de l'indicateur utilisé. En effet, on peut penser qu'il aurait été possible d'appliquer directement cette idée dans MOCA-I en utilisant, par exemple, la f-mesure [161] comme objectif unique du processus d'optimisation. Le problème est que la f-mesure, comme les autres mesures telles que la confiance ou la sensibilité, s'applique toujours à une unique solution. Afin de pouvoir l'appliquer sur un ensemble de solutions, il est donc nécessaire de calculer, par exemple, la f-mesure moyenne de l'archive (avant et après insertion), ou la f-mesure de la meilleure solution. De telles méthodes peuvent être suffisantes dans certains cas, mais elles impliquent une perte d'information liée à l'inadéquation des mesures employées à capturer le gain réel apporté par un ensemble de nouvelles solutions. Par conséquent, l'emploi d'indicateurs spécifiques paraît plus adapté.

Plusieurs indicateurs ont été proposés à cet effet [75, 178], le plus reconnu et le plus utilisé étant l'*hypervolume* [19, 88]. Celui-ci représente, dans l'espace des objectifs, le volume couvert par un ensemble de solutions par rapport à un point de référence, souvent noté r ou z^*. La Figure 3.5 représente, de manière plus visuelle, l'hypervolume d'un ensemble de cinq solutions d'un problème bi-objectif, où les deux objectifs sont à maximiser, par rapport au point de référence r.

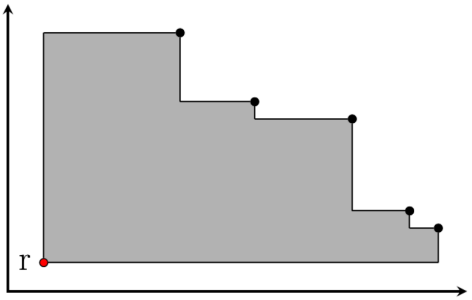


FIGURE 3.5 – Surface représentant l'hypervolume de cinq solutions par rapport à une solution de référence r

La question qui se pose lors de l'utilisation de l'hypervolume est celle du choix du point de référence. Dans notre cas, deux options sont envisageables :

— la première consiste à prendre comme point de référence la solution "originale", celle dont le voisinage a été exploré pour former les nouvelles solutions.

— la seconde consiste à prendre un point de référence correspondant à une "mauvaise" solution théorique, par exemple le point de coordonnées $(0, 0)$ dans l'espace des solutions. Dans ce cas, l'hypervolume de l'ensemble de l'archive est calculé avant et après insertion des nouvelles solutions, et c'est la différence entre les deux qui permet de mesurer l'apport des nouvelles solutions.

Ces deux options représentent des objectifs légèrement différents. Dans la première, on cherche à mesurer l'efficacité d'un opérateur de voisinage pour améliorer une solution. Dans la seconde, on cherche à mesurer son efficacité pour améliorer la qualité générale de l'archive. Les deux présentent chacune un intérêt ; dans cette étude, la première option a été retenue, et ce pour

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

deux raisons. D'abord, celle-ci est moins coûteuse, étant donné qu'un seul calcul d'hypervolume est requis, et que ce calcul concerne en général moins de solutions (par rapport à une archive complète), ce qui le rend moins complexe. Il est important que le calcul d'hypervolume soit rapide et ne ralentisse pas la recherche locale, sans quoi l'intérêt du mécanisme de sélection d'opérateur serait remis en cause. Une seconde raison qui motive ce choix est que cette option permet de mesurer l'efficacité d'un opérateur plus directement, de mesurer son apport à la recherche locale sans tenir compte de l'état général de l'archive, lequel pourrait biaiser les résultats.

L'implémentation choisie pour le calcul de l'hypervolume est inspirée de celle décrite dans [62]. La complexité annoncée de cet algorithme est $O(m^3n^2)$, avec m le nombre de points (*i.e.* solutions) et n le nombre d'objectifs (deux, dans notre cas). Un article ultérieur [169] a contesté cette valeur et annoncé une complexité de $O(m^n)$. Cet article, ainsi que d'autres, ont proposé des méthodes plus efficaces pour le calcul de l'hypervolume. Cependant, les gains d'efficacité apportés par ces améliorations sont surtout notables lorsque les problèmes étudiés impliquent un nombre plus élevé d'objectifs. Pour les problèmes à deux ou trois objectifs, le gain est négligeable par rapport à des méthodes "simples" telles que celle utilisée ici.

Le second élément requis pour former une stratégie est un mécanisme de décision, c'est-à-dire un algorithme qui choisit l'opérateur à utiliser en fonction de la qualité de chaque opérateur. Pour ce problème également, de nombreuses méthodes ont été proposées dans la littérature sur le sujet. Parmi celles-ci, nous avons retenu deux méthodes simples : *ProbabilityMatching* et *AdaptivePursuit* [157]. La première consiste en un choix aléatoire de l'opérateur à utiliser, où la probabilité de choisir chaque opérateur dépend de la qualité de celui-ci. La seconde est une variation de la première, et cherche à répercuter plus rapidement les changements dans les résultats sur les probabilités de sélection des différents opérateurs. Des tests préliminaires ont révélé que la méthode *ProbabilityMatching* était légèrement plus performante que la méthode *AdaptivePursuit* dans notre cas, aussi est-ce cette méthode que nous considérons pour les expérimentations présentées ci-dessous.

3.6.3 Expérimentations

Nous comparons MOCA-IQ dans sa version la plus efficace (discrétisation par Fayyad) à une variante utilisant le mécanisme de *Probability Matching* (appelée MOCA-IQ-ProbMatch), avec calcul d'hypervolume par rapport à la solution utilisée pour la génération du voisinage. Cette méthode choisit, à chaque génération de voisinage, un unique opérateur à utiliser. Les expérimentations sont menées sur les jeux de données de la littérature déjà employés dans les sections précédentes. Pour chaque algorithme, 30 exécutions indépendantes sont réalisées, avec une limite de temps fixée à une heure. Les valeurs moyennes et écarts-type pour la solution de meilleure f-mesure sont présentés dans les Tables 3.21 et 3.22, respectivement pour les ensembles d'apprentissage et de test. Les valeurs en gras correspondent celles de l'algorithme le plus performant pour le jeu de données concerné.

On observe que les résultats obtenus par MOCA-IQ ne varient presque pas, que le mécanisme de sélection d'opérateur soit utilisé ou non. Une explication possible à cela est que le gain obtenu en termes de vitesse de convergence via ce mécanisme compense tout juste le surcoût de complexité engendré. Cette explication recoupe les observations réalisées sur la convergence : l'algorithme avec sélection d'opérateur effectue moins d'itérations au cours d'un temps t que l'algorithme sans ce mécanisme, on peut donc supposer que chacune de ces itérations en plus performante en moyenne. En conclusion, la sélection d'opérateur de voisinage ne justifie pas son coût par des performances réellement accrues.

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

	MOCA-IQ	MOCA-IQ-ProbMatch
adult	**0.678 ± 0.005**	0.677 ± 0.005
arrythmia	**0.96 ± 0.008**	0.957 ± 0.009
page-blocks	0.891 ± 0.009	**0.894 ± 0.01**
penbased	**0.996 ± 0.002**	0.995 ± 0.001
ring	**0.888 ± 0.007**	0.885 ± 0.004
segment	**1.0 ± 0.0**	**1.0 ± 0.0**
spambase	0.914 ± 0.004	**0.915 ± 0.005**
vowel	**0.986 ± 0.011**	0.985 ± 0.013

TABLE 3.21 – F-mesures obtenues sur l'ensemble d'apprentissage

	MOCA-IQ	MOCA-IQ-ProbMatch
adult	**0.674 ± 0.006**	0.674 ± 0.005
arrythmia	**0.758 ± 0.04**	0.757 ± 0.06
page-blocks	**0.837 ± 0.018**	0.835 ± 0.015
penbased	**0.982 ± 0.005**	0.981 ± 0.005
ring	**0.866 ± 0.009**	0.863 ± 0.006
segment	**0.98 ± 0.009**	0.978 ± 0.01
spambase	0.891 ± 0.007	**0.892 ± 0.006**
vowel	**0.927 ± 0.031**	**0.927 ± 0.03**

TABLE 3.22 – F-mesures obtenues sur l'ensemble de test

3.7 Amélioration de la gestion de l'archive

Dans cette section, nous explorons la seconde piste d'amélioration envisagée, qui concerne la gestion de l'archivage des nouvelles solutions. Nous rappelons le fonctionnement du mécanisme d'archivage actuel, puis proposons une nouvelle méthode. Des expérimentations permettent de valider l'apport de ce nouveau mécanisme sur la qualité des solutions trouvées par MOCA-IQ.

3.7.1 Motivations

Nous avons vu précédemment que les nouvelles solutions pouvaient être insérées dans l'archive, à condition qu'elles ne soient pas dominées par les solutions courantes. Le déroulement précis de cette phase d'insertion se déroule selon le processus décrit dans l'Algorithme 5.

Algorithme 5 : Méthode standard pour l'insertion d'une nouvelle solution dans l'archive

```
Data : une nouvelle solution S
if S n'est dominée par aucune solution de l'archive then
    supprimer les solutions de l'archive dominées par S;
    if taille(archive) < MAX then
        insérer S dans l'archive;
    end
end
```

Pour des raisons évidentes d'espace mémoire, il n'est pas raisonnable d'envisager une archive de taille illimitée. Par conséquent, il est possible qu'une nouvelle solution non-dominée ne soit

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

pas incluse dans l'archive, si celle-ci a déjà atteint sa capacité maximale et que la nouvelle solution ne domine aucune de celles de l'archive. Ainsi, il existe un risque que certaines solutions potentiellement intéressantes ne soient pas retenues. Ce problème est récurrent dans les méta-heuristiques à population et a été le sujet de nombreuses études. Diverses méthodes ont été proposées, la plupart visant à maintenir une bonne diversité sur l'ensemble de l'archive, c'est-à-dire faire en sorte que les solutions de l'archive soient bien réparties sur le front Pareto [114].

3.7.2 Méthode proposée

La diversité d'une archive est un aspect difficile à mesurer, et aucune méthode de gestion de l'archive ne semble faire consensus dans la littérature. De plus, la diversité n'est pas primordiale dans le contexte qui nous occupe ; la qualité des solutions, quant à elle, est d'un intérêt majeur. Pour cette raison, nous proposons le système décrit dans l'Algorithme 6 pour l'insertion des nouvelles solutions dans l'archive.

Algorithme 6 : Nouvelle méthode pour l'insertion d'une nouvelle solution dans l'archive

```
Data : une nouvelle solution S
supprimer les solutions de l'archive dominées par S;
if S n'est dominée par aucune solution de l'archive then
    if taille(archive) < MAX then
        insérer S dans l'archive;
    else
        choisir une solution aléatoire S_A issue de l'archive;
        if f-mesure(S) > f-mesure(S_A) then
            supprimer S_A de l'archive;
            insérer S dans l'archive;
        end
    end
end
```

De cette manière, la qualité générale de l'archive des solutions doit logiquement augmenter. Il convient néanmoins de garder à l'esprit qu'en procédant ainsi, le problème est temporairement transformé en un problème mono-objectif cherchant à maximiser la f-mesure. Cependant, un tel procédé est déjà employé dans l'algorithme : dans l'archive finale, la "meilleure" solution, celle qui est produite en sortie, est la solution de meilleure f-mesure. On peut donc considérer la f-mesure comme une mesure utile pour départager plusieurs solutions non-dominées entre elles, même si ce choix reste contestable.

Le fait de ne choisir qu'une unique solution au hasard parmi l'archive est bien entendu un choix arbitraire assez simpliste. Afin de juger de l'efficacité de ce choix, une variante de cette méthode est implémentée, dans laquelle trois solutions sont choisies au lieu d'une seule. La nouvelle solution vient remplacer la première des trois dont la *f-mesure* est inférieure. On espère ainsi que de tels remplacements par une solution de meilleure *f-mesure* se produiront plus fréquemment, ce qui se traduirait par une amélioration globale de la qualité de l'archive. Des tests préliminaires ont été menés, mais cette variante ne s'est pas révélée plus intéressante que la méthode de base. Plus précisément, le gain apporté par les insertions supplémentaires ne compense pas le surcoût en termes d'évaluation de solutions, et le léger "ralentissement" ainsi provoqué suffit à rendre cette variante moins performante.Pour cette raison, les expérimentations présentées dans la suite de cette section ne se concentrent que sur la méthode avec sélection et remplacement d'une seule solution.

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

3.7.3 Expérimentations

Pour ces expérimentations, nous utilisons les mêmes jeux de données que dans la section précédente. 30 exécutions indépendantes sont réalisées pour chaque réalisées, avec une limite de temps d'une heure pour chaque. Nous considérons la moyenne et l'écart-type sur ces 30 exécutions, pour la solution de meilleure f-mesure sur l'ensemble d'apprentissage à la fin du processus d'optimisation. Les Tables 3.23 et 3.24 présentent une comparaison des résultats obtenus avec MOCA-IQ, et MOCA-IQ avec gestion de l'archive améliorée, respectivement pour les ensembles d'apprentissage et de test.

	MOCA-IQ	MOCA-IQ_arch
adult	0.678 ± 0.005	**0.695 ± 0.004**
arrythmia	**0.96 ± 0.008**	0.868 ± 0.018
page-blocks	0.891 ± 0.009	**0.907 ± 0.009**
penbased	0.996 ± 0.002	0.996 ± 0.002
ring	0.888 ± 0.007	**0.943 ± 0.021**
segment	**1.0 ± 0.0**	**1.0 ± 0.0**
spambase	0.914 ± 0.004	**0.942 ± 0.006**
vowel	**0.986 ± 0.011**	0.984 ± 0.01

TABLE 3.23 – Comparaison des méthodes d'archivage des solutions, f-mesure, ensemble d'apprentissage

	MOCA-IQ	MOCA-IQ_arch
adult	0.674 ± 0.006	**0.689 ± 0.006**
arrythmia	**0.758 ± 0.04**	0.661 ± 0.034
page-blocks	0.837 ± 0.018	**0.841 ± 0.018**
penbased	0.982 ± 0.005	**0.984 ± 0.004**
ring	0.866 ± 0.009	**0.924 ± 0.024**
segment	**0.98 ± 0.009**	0.979 ± 0.01
spambase	0.891 ± 0.007	**0.903 ± 0.006**
vowel	0.927 ± 0.031	**0.928 ± 0.033**

TABLE 3.24 – Comparaison des méthodes d'archivage des solutions, f-mesure, ensemble de test

En-dehors d'une dégradation difficile à expliquer sur le jeu de données *arrythmia*, nous pouvons observer que ce mécanisme simple de gestion de l'archive améliore les performances de l'algorithme d'optimisation assez régulièrement sur l'ensemble des jeux de données. Une amélioration notable de la qualité a lieu sur le jeu *adult*. Ce point est intéressant car ces données contiennent un mélange d'attributs qualitatifs et quantitatifs ainsi qu'une volumétrie plus importante que la moyenne, ce qui les rapprochent des données réelles auxquelles nous nous intéressons.

3.8 Conclusion

Dans ce chapitre, nous avons défini le problème de l'adaptation des règles de classification aux données numériques. Nous avons présenté deux approches possibles : discrétisation des données numériques en pré-traitement, et modification de l'algorithme de recherche locale pour prendre en compte ce nouveau type de données. Pour chacune de ces deux approches, nous avons proposé et comparé plusieurs variantes. Nous avons ensuite comparé les variantes les plus performantes

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

de chaque approche. Les résultats montrent que l'approche par discrétisation est plus performante que la seconde. Nous avons déterminé la méthode de discrétisation donnant les meilleurs résultats et l'avons implémentée au sein du nouvel algorithme, appelé MOCA-IQ. Des expérimentations ont montré que celui-ci produit des classifieurs de meilleure qualité que trois autres algorithmes de classification de la littérature : BioHEL, C45-CS et Ripper. Nous avons également exploré deux pistes d'amélioration de l'algorithme, indépendantes du type de données utilisées. Nous avons d'abord proposé un mécanisme visant à sélectionner dynamiquement les opérateurs de voisinage à appliquer au cours du processus d'optimisation, afin d'accélérer la convergence de l'algorithme. L'introduction de ce mécanisme n'apporte pas l'amélioration escomptée, mais cette idée de sélection dynamique demeure une piste de recherche intéressante. La seconde amélioration proposée concerne la gestion de l'archive. Nous avons développé un mécanisme alternatif pour l'insertion des nouvelles solutions dans l'archive. Des expérimentations ont montré que celui-ci amène MOCA-IQ à trouver des solutions de meilleure qualité par rapport à la méthode d'archivage originale. Pour cette raison, nous conservons ce mécanisme d'archivage dans la suite de ces travaux. Dans le chapitre suivant, nous explorons le problème de la gestion des données temporelles et proposons MOSC, une extension de MOCA-IQ supportant ce type de données.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Chapitre 4

MOSC - Classification sur données temporelles

Dans le chapitre précédent, nous nous sommes intéressés aux données numériques et avons proposé une méthode permettant de les prendre en compte, formant ainsi l'algorithme MOCA-IQ. Dans ce chapitre, nous nous concentrons sur une autre particularité des données hospitalières, à savoir le caractère temporel de ces données. En premier lieu, nous présentons le problème lié à ces données et motivons leur utilisation au sein d'un algorithme de classification. Nous dressons ensuite un état de l'art des méthodes existantes pour gérer ce type de données. Nous proposons ensuite une modélisation pour des règles de classification comprenant des séquences temporelles, ainsi qu'un algorithme d'optimisation adapté pour ces règles. Enfin, nous présentons les résultats d'expérimentations menées sur des données hospitalières réelles, permettant de valider l'efficacité de la méthode proposée. Cette méthode est appelée MOSC (Multi-Objective Sequence Classification).

Ce chapitre est associé aux publications suivantes :

— M. Vandromme, J. Jacques, J. Taillard, A. Hansske, L. Jourdan, C. Dhaenens : *Extraction and optimization of classification rules for temporal sequences : application to hospital data*, **Knowledge-Based Systems**, Volume 122, pp 148-158, 2017.

4.1 Motivations et description du problème

4.1.1 Motivations

Dans le Section 1.1, nous avons vu que les données hospitalières sont fortement hétérogènes. Nous avons également vu qu'une partie significative des données disponibles pour les hôpitaux concernent les données PMSI (actes et diagnostics). Or, ces données sont intrinsèquement temporelles : à chaque code correspond une date et une heure d'occurrence. Ces informations ne sont pas exploitées dans l'algorithme de fouille de données présenté dans les parties précédentes. En effet, celui-ci omet entièrement toute notion de temporalité dans les données : les événements sont représentés chacun par un attribut binaire, qui prend la valeur *true* pour un individu si l'événement s'est produit au moins une fois pour cet individu. Ce faisant, trois aspects potentiellement intéressants des données sont laissés de côté :

— l'ordre dans lequel se produisent les événements,

— le fait qu'un événement arrive à plusieurs reprises,

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

— le temps écoulé entre deux événements.

Dans un premier temps, il convient de s'interroger sur l'utilité réelle d'intégrer tout ou partie de ces aspects temporels dans un outil de classification. Qu'apporterait concrètement un système pouvant utiliser des séquences temporelles pour construire un ensemble de règles de classification ?

Un premier avantage évident est qu'une telle modélisation permettrait de représenter plus fidèlement la réalité, en évitant de perdre l'information sur les données véhiculée par les *timestamps* (date et heure) associés à chaque événement. Un second apport est qu'un classifieur portant sur des données temporelles sera naturellement plus intéressant, et ce pour deux raisons majeures. Ces raisons sont fortement liées à l'aspect métier du problème traité, mais sont transposables à d'autres problèmes du même type.

Tout d'abord, une règle contenant un aspect temporel est en général plus utile et plus parlante à un expert médical qu'une règle qui en serait dépourvue. Cela permet une interprétation plus rapide et plus aisée des solutions trouvées par l'outil d'optimisation et, par conséquent, une validation plus simple des résultats. Par exemple, une règle du type "si A puis B se produisent à moins de 3 jours d'intervalle, alors P est vrai" est plus riche et plus facilement interprétable qu'une règle du type "si A et B se produisent, alors P est vrai".

Ensuite, cela rend possible l'utilisation de l'outil de classification dans des tâches plus diverses. On peut penser assez naturellement à un système d'alerte, qui signale quand la séquence d'événements associée à un patient contient le début d'une règle de classification. Par exemple, s'il existe une règle de classification telle que "si A puis B puis C se produisent, alors P est vrai" et que les événements A puis B viennent de se produire chez un patient, alors un tel système pourrait signaler le cas. Cette utilisation est proche des problèmes d'*early classification* surtout utilisée pour les séries temporelles, où l'objectif est de classifier une série de valeurs le plus rapidement possible (sans toutefois perdre en exactitude dans les prédictions) [170].

4.1.2 Trois aspects au problème : séquence, récurrence et durée

Nous avons vu plus haut qu'une modélisation complète du problème devait considérer les trois parties qu'il comporte : l'ordre des événements, la possibilité d'avoir un même événement plusieurs fois, et la durée entre les événements. Parmi celles-ci, il nous semble nécessaire de séparer les deux premières de la troisième. En effet, si l'aspect séquentiel ainsi que la récurrence d'un événement paraissent être des éléments indispensables à la nouvelle modélisation souhaitée, la durée qui sépare les événements n'est qu'une précision supplémentaire, une contrainte qui vient s'ajouter sur les séquences. Cette contrainte revêt une importance moindre que la structure séquentielle des séquences.

Par conséquent, nous choisissons de ne pas intégrer directement cet aspect dans la représentation (qui sera détaillée par la suite). Au-delà de la justification conceptuelle de la différence d'importance, cette décision est justifiée par plusieurs raisons :

— inclure les contraintes sur la durée dans la représentation des séquences (et des règles qui en découlent) ne pose pas de difficulté majeure. Cependant, cela impliquerait, quelle que soit la modélisation choisie, de traiter ces contraintes comme des attributs utilisés par le processus d'optimisation dans sa recherche de classifieurs. En pratique, cela se traduirait par un ajout d'attributs, et donc une complexité accrue, à la fois de la modélisation et du mécanisme de recherche locale, par le biais d'un espace de recherche plus important et d'une comparaison (*matching*) des séquences plus coûteuse.

— une conséquence directe de considérer les durées inter-événements comme des paramètres est que cela autorise l'algorithme à générer des règles contenant les mêmes événements dans le même ordre, dont seules les contraintes sur les durées diffèrent. Un risque à cela est la multiplication de règles ne différant que par ces contraintes, à la fois dans l'archive en

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

général et au sein d'une même solution. Il est naturel pour l'algorithme d'explorer en permanence les voisins d'une solution existante, mais un niveau de détail trop grand peut nuire à l'efficacité globale de l'optimisation, en remplissant l'archive de solutions structurellement très proches, au risque de dégrader la diversité.

Il convient de noter qu'aucun de ces problèmes n'est rédhibitoire, et qu'il demeure tout à fait possible d'ajouter ces contraintes sur les durées directement dans le processus d'optimisation. Cependant, la raison pour laquelle nous avons choisi de ne pas le faire, raison transversale aux différentes difficultés évoquées plus haut, est que le gain escompté est trop faible par rapport aux divers problèmes engendrés. Par conséquent, il nous paraît plus approprié de reléguer cette gestion des durées à une méthode de post-traitement. De cette manière, l'outil d'optimisation travaillerait dans un premier temps sur l'aspect purement séquentiel du problème, et les séquences intéressantes ainsi trouvées seraient, en fin de traitement, précisées et complétées par des contraintes temporelles si besoin. Les mécanismes proposés à cet effet sont décrits dans la Section 4.3.6.

A noter que l'aspect séquentiel inclut également la nécessité de pouvoir représenter des événements simultanés. Ce point est en grande partie motivé par le problème traité ainsi que la nature des données : un *parcours patient* est composé d'un ensemble de *séjours*, chaque séjour regroupant un ensemble d'événements considérés comme se produisant au même moment ou, plus exactement, dans la même unité de temps. Cette notion de simultanéité n'est pas propre uniquement aux données médicales ; d'autres applications, telles que les études marketing réalisées sur les biens de consommation achetés dans un même panier [5], se servent également d'une représentation de ce type. La modélisation des données temporelles doit donc pouvoir traduire des événements ordonnés les uns par rapport aux autres dans tout ce que cela implique, y compris la simultanéité dans l'occurrence de plusieurs événements.

4.2 Etat de l'art : classification sur données temporelles

Vue d'ensemble La fouille de données temporelles dans son ensemble est un domaine qui a attiré de nombreuses contributions. En effet, de nombreuses applications concrètes utilisent des données de ce type ; il est donc indispensable de disposer d'outils permettant de les traiter. Ce domaine de recherche englobe plusieurs problèmes bien distincts. Nous nous intéressons ici à la *classification de séquences temporelles*, qui correspond à la problématique exposée au début de cette section. Ce problème particulier a reçu comparativement assez peu d'attention, comme souligné dans une revue du domaine [11]. Bien que cette étude ait été publiée en 2001, ses conclusions restent valides à ce jour. Le problème de classification de séquences temporelles a néanmoins été exploré, notamment dans le contexte de la sécurité informatique et de la détection d'intrusion dans les systèmes. Pour ce type d'applications, les données sont des séquences d'actions, et la dimension temporelle est primordiale. Les premières études sur le sujet cherchaient à prédire les brèches de sécurité dans les systèmes à partir de ces séquences d'évènements antérieurs [156, 105]. La plupart des travaux sur ce sujet emploient des algorithmes basés sur la recherche de règles d'association. D'autres utilisent des méthodes alternatives telles que les arbres de décision [39] ou les réseaux de neurones. L'une des limitations principales des méthodes actuelles est leur problème d'adaptation aux données déséquilibrées, qui rendent difficile la définition d'un niveau de support minimum pour les règles. Afin de répondre à cela, une étude notable propose une méthode pour la prédiction d'événements rares [163]. Cependant, l'aspect temporel de cette méthode reste limité. Celle-ci vise à former des règles de prédiction à partir d'ensembles d'événements se produisant dans une fenêtre de temps avant le fait à prédire, mais sans relation temporelle au sein de ces événements.

© 2017 Tous droits réservés. lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Méthodes statistiques Différentes méthodes statistiques ont été appliquées au problème de classification de séquences temporelles. En particulier, les Modèles de Markov Cachés (Hidden Markov Models, ou HMM) [172] et les Champs Aléatoires Conditionnels (Conditional Random Fields, ou CRF) [106] se sont rélévés efficaces, avec des applications en reconnaissance vocale [149] et détection de mouvement [173] notamment. Les CRF sont mieux adaptés et davantage utilisés pour le *marquage de séquences*, qui associe à chaque élément d'une séquence un label, mais sont également utilisés pour le problème qui nous intéresse ici. Le problème de ces méthodes est qu'elles ne supportent souvent qu'un nombre assez réduit d'attributs, quelques centaines tout au plus. Cela rend la sélection d'attributs nécessaire pour les données plus volumineuses [102]. Un second problème est que les HMM et CRF peuvent gérer la séquentialité dans les données, mais pas la simultanéité, qui est un point central des données sur lesquelles nous travaillons.

Classification de séquences Une étude intéressante sur ce sujet propose de générer des attributs supplémentaires pour représenter les évènements à différents moments dans une séquence [112]. Une phase de sélection d'attributs permet de ne retenir que les attributs les plus utiles, qui sont utilisés pour la classification en elle-même. Les auteurs utilisent les classifieurs Bayésien et de Winnow. Cette approche améliore significativement l'efficacité de la classification, mais n'est applicable que pour un nombre d'attributs initiaux assez restreint. En effet, la création d'attributs additionnels fait croître ce nombre de manière exponentielle ; quelques dizaines d'attributs produisent plusieurs dizaines ou centaines de milliers d'attributs supplémentaires. Ainsi, cette approche n'est pas applicable pour les cas qui nous intéressent, avec des milliers d'attributs initiaux. Un autre travail sur le sujet utilise des Cartes Cognitives Floues (Fuzzy Cognitive Maps, ou FCM) [65]. Plus précisément, le système proposé cherche à décrire et prédire la présence de maladies à partir de séquences d'événements médicaux (interventions, évolutions de l'état de santé, etc.). Cependant, les FCM nécessitent un intervalle de temps constant entre deux éléments consécutifs d'une séquence, ce qui n'est pas le cas dans les données qui nous intéressent. Cette limitation apparaît également dans une autre étude, qui utilise des réseaux de neurones récurrents pour classifier des séquences [70]. Dans cette dernière, les auteurs prédisent le prochain événement d'une séquence, et se servent de cela pour classifier cette séquence. Un autre problème est que cette méthode suppose que les séquences sont périodiques et peuvent être étendues, ce qui n'est que rarement vrai dans notre cas. Les réseaux de neurones récurrents ont également été utilisés dans une étude plus récente, toujours afin de prédire le prochain événement (avec davantage de détails, tels que le temps estimé avant l'événement) [32]. Ce système est néanmoins limité par le nombre d'attributs supportés ; les auteurs recommandent de se limiter à une centaine. Une dernière étude tangente à notre problème se concentre sur l'induction d'Automates Finis Déterministes (Deterministic Finite Automata, ou DFA), où le but est de construire des automates décrivant des langages naturels [108]. Ici, on considère des séquences de mots et de lettres comme données. Cependant, comme précédemment [32], cette méthode n'autorise pas de simultanéité dans les données. De plus, l'induction de DFA est un problème très spécifique et difficile à résoudre même pour des instances de taille assez réduite (quelques centaines d'attributs). Les expérimentations menées dans l'étude mentionnée [108] soulignent également que les données manquantes compliquent considérablement la tâche, ce qui rend ce type d'algorithmes mal adaptés à notre problème.

Règles d'association temporelles Un sous-domaine de recherche tangent à celui nous intéresse est la recherche de règles d'association temporelles. Ici, le but n'est pas de classifier chaque instance, mais de découvrir dans les données des règles intéressantes formées de séquences d'événements. Cette question a été largement explorée depuis le travail fondateur sur l'algorithme Apriori [5], et inclut de nombreuses applications dans le marketing, la santé, la sécurité des

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

réseaux [107] ou la sécurité financière [117]. L'algorithme Apriori original portait déjà sur des données séquentielles, et des développement ultérieurs ont étendu son champ d'application aux séries temporelles [69, 138]. Une autre étude notable utilise SPEA2 [179], un algorithme d'optimisation multi-objectif reconnu, pour construire un ensemble de règles d'association fiables [145]. Une étude très influente dans le domaine propose d'utiliser les règles d'association trouvées comme base pour construire des règles de classification [119]. Cette étude propose l'algorithme CBA (Classsification Based on Associations), qui ne supporte pas les données temporelles dans sa version originale, mais peut être aisément étendu afin de les prendre en compte. Le problème de cette approche est qu'elle n'est pas adaptée aux données déséquilibrées. Comme mentionné brièvement plus haut, un problème récurrent dans la recherche de règles d'association est de déterminer le niveau de support minimal pour les règles. Si l'une des classes est présente moins souvent dans les données, ce seuil minimal doit être assez bas pour capturer des règles décrivant la classe rare. Cependant, un seuil plus bas mène à la création de davantage de règles. Lorsqu'une classe est très rare (1% de présence, par exemple), le processus de recherche crée un nombre considérable de règles sans réel intérêt, ce qui altère la performance du classifieur. Une étude plus récente sur le sujet propose de modifier Apriori afin de supporter les données déséquilibrées [144]. Néanmoins, les données utilisées dans celle-ci sont assez réduites (230 instances et 11 attributs), aussi la scalabilité de cette méthode à des données plus massives reste-t-elle à vérifier.

Séries temporelles L'analyse de séries temporelles est un domaine très largement étudié, et qui a certaines similarités avec le problème qui nous occupe. En effet, séries et séquences temporelles sont composées d'une suite ordonnée de valeurs, avec chaque valeur associée à un point dans le temps. De nombreuses études proposent des méthodes pour extraire des règles de classification pour séries temporelles [39, 69, 79, 145, 166]. Cependant, séries et séquences temporelles présentent des différences qui rendent la transposition de telles méthodes à notre problème impossible. La principale est que les valeurs d'une série temporelle sont numériques, et celles d'une séquence sont symboliques. Dans ce contexte, la plupart des outils statistiques utilisés habituellement pour l'analyse de séries temporelles (transformée de Fourier, par exemple) ne sont pas applicables. Des difficultés supplémentaires apparaissent lorsqu'on considère que l'intervalle de temps entre événements d'une séquence n'est pas constant, et que ces séquences sont parfois très courtes. Une approche commune pour l'analyse de séries temporelles emploie la quantification pour transformer les séries en séquences de symboles. Bien qu'intéressante, cette approche vise à n'obtenir qu'un nombre très restreint de symboles possibles (moins de dix, le plus souvent), ce qui s'accorde mal avec les milliers d'attributs que nous considérons ici.

Classification de données hétérogènes Dans cette revue de l'état de l'art, nous nous sommes concentrés sur la classification de séquences temporelles. Cependant, le véritable problème est plus large, car nous souhaitons classifier des données hétérogènes, incluant entre autres des données temporelles. Ce problème particulier ne semble pas avoir été étudié jusqu'à présent. La classification sur données hétérogènes a été étudiée, et plusieurs outils logiciels reconnus sont disponibles, tels que Weka [72] ou KEEL [6]. La classification de séquences temporelles a également été étudiée, comme montré dans cette section. Néanmoins, il ne semble exister aucune méthode permettant de résoudre les deux problèmes simultanément. Ceci constitue la contribution principale de ce chapitre. Nous proposons d'étendre l'algorithme MOCA-IQ développé dans le chapitre précédent afin qu'il supporte les séquences d'événements temporels comme parties de règles de classification.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

4.3 Modélisation et intégration dans MOCA-I

Dans cette section, nous définissons un nouveau type de données, appelé *séquence* permettant de représenter les attributs temporels. Nous proposons ensuite un ensemble d'opérateurs de voisinage adaptés à ce nouveau type, ainsi que des restrictions de l'espace de recherche réduisant notablement la complexité associée à la génération du voisinage. Nous définissons ensuite une méthode d'évaluation de la similarité entre deux *séquences*, nécessaire pour la mesure de qualité des solutions. Nous présentons une extension du modèle à des séquences comprenant plusieurs niveaux de temporalité. Nous discutons ensuite le paramétrage de la méthode. Enfin, nous proposons un mécanisme permettant de préciser les règles de classification générées en intégrant des informations sur les durées entre les événements, qui ont été écartées de l'algorithme de classification (voir Section 4.1.2).

4.3.1 Représentation des attributs *séquence*

L'algorithme de classification MOCA-IQ supporte déjà plusieurs types d'attributs : symboliques (listes ordonnées ou non) et numériques (entiers ou réels). Afin de répondre aux besoins explicités ci-dessus concernant la modélisation de la temporalité dans les données, nous introduisons un nouveau type d'attribut, nommé *séquence*. Cet attribut vise à exprimer l'ensemble des événements associés à un patient dans leur aspect temporel. Cela signifie qu'un unique attribut concentre, selon cette représentation, une vaste majorité des informations disponibles pour un patient. Plus précisément, l'attribut *séquence* regroupe toutes les informations auxquelles sont associées un *timestamp*, donc principalement les codes PMSI. Les attributs non-temporels, qualitatifs ou quantitatifs, restent en-dehors de l'attribut *séquence*. D'après ce modèle, il est théoriquement possible d'avoir plusieurs attributs de type *séquence* pour représenter différentes parties d'un "parcours patient". Des tests ont été menés à ce sujet en définissant d'un côté une séquence d'actes et diagnostics, et de l'autre une séquence de prescriptions médicamenteuses (antibiotiques). L'objectif ici est d'identifier les séjours de patients porteurs de bactéries multi-résistantes aux antibiotiques ; ces résistances pouvant être provoquées par les antibiotiques. Multiplier les séquences présente l'inconvénient de ne plus pouvoir capturer les corrélations entre différents éléments issus de ces séquences (par exemple, "l'acte A suivi du médicament M"). Cependant, une telle modélisation des données s'avère nécessaire si les granularités temporelles diffèrent. Dans cet exemple, les actes et diagnostics portent sur un unique séjour, tandis que les antibiotiques dépassent ce cadre. En effet, on considère que les antibiotiques prescrits jusqu'à trois mois avant le séjour peuvent favoriser l'émergence de bactéries résistantes. Pour cette raison, il est nécessaire de les inclure dans les données, avec une temporalité propre. Cet exemple illustre la nécessité pour l'algorithme de pouvoir traiter plusieurs séquences différentes, ce qui est bien le cas avec la représentation proposée.

Les besoins définis pour la modélisation impliquant à la fois séquentialité et simultanéité, nous définissons la valeur d'un attribut de type *séquence* comme une séquence ordonnée d'*épisodes*, chaque *épisode* étant constitué d'un ou plusieurs événements simultanés. Les durées inter-épisodes sont également enregistrées dans la structure *séquence*, mais leur utilisation est réservée au post-traitement uniquement, comme nous le verrons plus tard dans ce chapitre. La Figure 4.1 présente un exemple de règle de classification R incluant une *séquence* parmi ces termes, afin d'illustrer les notions introduites.

4.3.2 Voisinage

De la même manière que pour les attributs numériques traités dans le Chapitre 3, il convient à présent de définir le voisinage des attributs de type *séquence*, étant donné que la valeur de

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

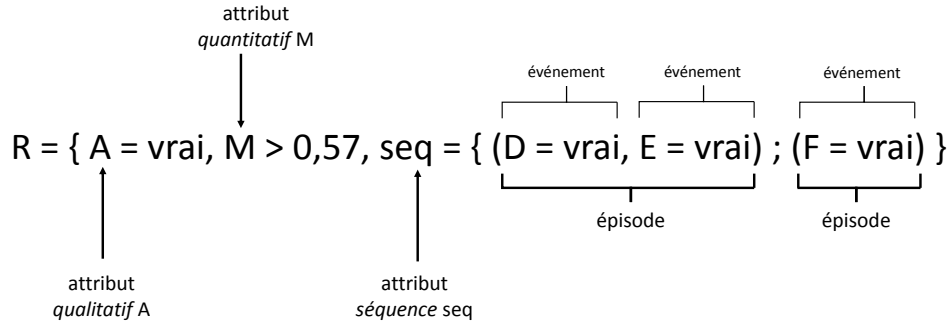


FIGURE 4.1 – Exemple de règle de classification pour MOSC : deux termes avec attributs non-temporels et un terme avec attribut *séquence*

ces attributs est une structure de données plus élaborée qu'un simple entier ou symbole d'une liste. Un terme de type *séquence* demeure malgré tout un terme d'une règle, ce qui signifie que l'opérateur *remove_term* s'applique toujours. Néanmoins, les opérateurs qui nous intéressent particulièrement ici sont *replace_term* et, dans une moindre mesure *add_term*. Les opérateurs *add_rule* et *remove_rule* sont quant à eux inchangés, car indépendants des types de données considérés.

4.3.2.1 Opérateurs de voisinage

Nous proposons de définir l'opérateur de voisinage *replace_term* pour les attributs *séquence* comme l'union des opérateurs suivants. Tous ces opérateurs sont appliqués de manière exhaustive :

- *add_episode* : ajouter un épisode, c'est-à-dire un ensemble d'événements simultanés, à la séquence actuelle. Le nouvel épisode peut se situer à n'importe quel endroit dans la séquence. Initialement, un nouvel épisode contient un unique événement.
- *remove_episode* : supprimer un épisode parmi ceux existants dans la séquence. Cet opérateur n'est pas applicable s'il ne reste qu'un seul épisode dans la séquence. La suppression du terme entier peut se faire via l'opérateur *remove_term* uniquement ; cette restriction a été ajoutée afin de mieux cloisonner le champ d'action des différents opérateurs.
- *add_event* : ajouter un événement à un épisode existant dans la séquence.
- *remove_event* : supprimer un événement d'un épisode. Cet opérateur n'est pas applicable s'il ne reste qu'un seul événement dans l'épisode.
- *replace_event* : changer la valeur d'un événement. Cet opérateur fonctionne de la même manière que pour les attributs hors-séquence. Cela signifie qu'un événement n'est pas forcément un attribut binaire (qui marquerait la présence ou l'absence d'un symptôme, par exemple), mais peut également être une mesure numérique, avec son voisinage propre (cf. Chapitre 3). La seule restriction imposée ici est qu'il est interdit d'avoir, au sein d'une séquence, un événement qui soit lui-même de type *séquence*, et ce pour des raisons évidentes de cohérence de la représentation.

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

L'opérateur *add_term* demeure applicable aux attributs de type *séquence*. Ajouter un nouveau terme d'attribut de type *séquence* se fait en initialisant la nouvelle séquence comme un unique épisode contenant un unique événement (de la même façon que pour *add_episode*).

4.3.2.2 Restrictions de l'espace de recherche

On observe ici que le voisinage d'un terme de type *séquence* est clairement plus complexe, et bien plus large, que le voisinage d'un terme symbolique ou numérique. En effet, bien qu'il n'existe aucune limite fixe au nombre de valeurs que peut prendre un attribut symbolique, ce nombre excède rarement quelques dizaines. Le voisinage d'un terme numérique, selon les méthodes décrites dans le Chapitre 3, ne comprend que trois termes voisins. A l'inverse, la taille du voisinage exhaustif d'un terme *séquence* dépend directement du nombre d'attributs présents dans les données, multiplié par le nombre d'épisodes dans la séquence en question. Etant donné le nombre souvent important des attributs dans les problèmes applicatifs considérés (plus de 10 000), le voisinage atteint rapidement des proportions démesurées, ce qui ralentit considérablement l'exploration des solutions et le processus d'optimisation en général. Par exemple, si les données contiennent 10 000 attributs, une règle de classification non-temporelle (MOCA-IQ) génère 10 000 voisins en essayant d'ajouter chaque attribut à la règle. Pour une règle temporelle, avec une séquence de trois épisodes, les opérateurs définis pour MOSC essaient d'ajouter ces 10 000 attributs dans chaque épisode, entre chaque paire d'épisodes consécutifs et avant et après la séquence, pour un total de 70 000 voisins. Il est par conséquent nécessaire de développer des méthodes permettant de réduire la taille de ce voisinage.

Une première amélioration assez basique par rapport à la méthode exhaustive consiste à ne considérer, pour la génération du voisinage, que les attributs (ou événements) se produisant au moins une fois parmi les individus positifs pour la prédiction considérée. En effet, il est assez trivial de constater qu'un attribut qui ne fait jamais partie des événements pour aucun des individus positifs, ne pourra pas être employé pour construire des règles de classification prédisant ces individus. Bien qu'évidente, cette simple optimisation permet de diviser par trois la taille du voisinage dans les cas d'étude, en raison de la faible proportion d'individus positifs. Ce système était déjà présent dans l'algorithme MOCA-I original, nous n'avons pour l'instant procédé qu'à une adaptation de celui-ci aux attributs de type *séquence*.

Les autres restrictions possibles du voisinage d'une séquence découlent en quelque sorte de cette amélioration primitive. En effet, l'objectif commun sous-jacent est d'éviter à l'algorithme de générer des solutions qui ne correspondent à aucun exemple positif, étant attendu que ces solutions seront naturellement écartées par le processus de sélection. Bien entendu, simplement vérifier au moment de la création d'une nouvelle solution si celle-ci correspond à au moins un exemple positif, via un parcours de l'ensemble des données, ne présente pas d'intérêt en termes de gain de performance. Il convient de trouver des mécanismes permettant d'écarter rapidement les solutions sans intérêt, et ce avant même de les évaluer, en ne générant que les solutions potentiellement intéressantes. La motivation derrière cette recherche est que le processus d'évaluation des solutions est la partie la plus coûteuse de l'algorithme, en particulier lors de l'emploi d'attributs de type *séquence*, et ce malgré les optimisations apportées à ce processus telles que détaillées dans la Section 4.3.3.

La première amélioration proposée est un exemple basique de ce concept ; en enregistrant la liste des attributs présents dans les exemples positifs, l'algorithme peut utiliser cette liste réduite pour générer les nouvelles solutions. Les mécanismes de restriction du voisinage doivent donc satisfaire à plusieurs conditions :

— ils ne doivent pas empêcher l'algorithme de recherche locale d'explorer des solutions potentiellement intéressantes, c'est-à-dire correspondant à au moins un exemple positif.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

— leur implémentation doit accélérer l'exploration du voisinage d'une solution. En d'autres termes, le surcoût engendré en termes de temps d'exécution ne doit pas être trop important, de sorte qu'employer ces mécanismes reste rentable.

Une optimisation importante et moins triviale peut être obtenue en considérant la structure générale d'une séquence. Une séquence est composée d'une suite ordonnée d'épisodes, chaque épisode étant un ensemble (non-ordonné) d'événements simultanés. Par conséquent, pour deux événements distincts E_1 et E_2 se produisant dans la même séquence, trois relations différentes peuvent exister entre ceux-ci :

— E_1 précède E_2, si E_1 se situe dans un épisode antérieur à celui de E_2.

— E_1 et E_2 sont simultanés, s'ils se situent dans le même épisode.

— E_1 succède à E_2, si E_1 se situe dans un épisode postérieur à celui de E_2.

Une observation intéressante ici est que pour un couple d'événements $\{E_1, E_2\}$, les trois relations d'ordre ne sont pas toujours présentes dans les données. Par exemple, il est possible que ces deux événements se produisent toujours en même temps (*i.e.* dans le même épisode) dans les séquences présentes dans les données. Dans ce cas, il est inutile de générer une nouvelle solution contenant, par exemple, E_1 avant E_2, étant donné qu'une telle situation ne correspond à aucun exemple positif. Afin de pouvoir utiliser cette information, au moment du chargement des données en mémoire, l'algorithme enregistre également, pour chaque événement E quels événements peuvent se produire avant, en même temps ou après E. Ainsi, pour un événement E_1, on pourrait par exemple obtenir les trois listes suivantes :

— $\mathit{Avant}(E_1) = \{E_5, E_6\}$

— $\mathit{Simult}(E_1) = \{E_2, E_3, E_4, E_5\}$

— $\mathit{Après}(E_1) = \{\}$

Selon cette méthode, un événement est introduit dans la liste correspondante dès lors qu'un exemple positif est trouvé dans les données qui contient la relation d'ordre ainsi décrite. Il serait également possible de n'introduire un évènement que lorsque la relation d'ordre est présente dans plusieurs exemples distincts, afin d'éviter de générer des solutions ne correspondant qu'à un seul cas, et par conséquent sans réel intérêt car difficiles à généraliser. Cependant, fixer le seuil (nombre d'occurrences) au-delà duquel une relation est considérée comme présente est un problème complexe, en partie dépendant des données étudiées. Pour cette raison, nous utilisons dans l'implémentation la méthode basique, dans laquelle une relation est enregistrée dès qu'elle est repérée dans un exemple positif.

Cette méthode permet de procéder à plusieurs restrictions sur les solutions générées par les différents opérateurs de voisinage propres aux séquences décrits précédemment. Ces restrictions sont expliquées de manière assez générique pour le moment ; un exemple complet de génération de voisinage restreint est présenté par la suite afin de faciliter la compréhension.

Pour *add_episode*, la restriction se fait au niveau des événements pouvant être inclus dans le nouvel épisode. Ceux-ci doivent pouvoir se situer à la fois après les événements présents dans les épisodes antérieurs (au nouvel épisode), et avant les événements présents dans les épisodes postérieurs.

Pour *add_event*, la restriction se fait également sur la liste des événements pouvant être insérés. Ceux-ci doivent pouvoir se situer à la fois après les événements présents dans les épisodes antérieurs à celui-ci, avant les événements présents dans les épisodes postérieurs, et en même temps que les événements déjà présents dans l'épisode modifié.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Les opérateurs *remove_episode*, *remove_event* et *replace_event* ne sont pas concernés par les restrictions. Ceux-ci ne contribuent de toute façon que marginalement à la taille du voisinage d'une séquence, aussi présentent-ils moins d'intérêt compte tenu de l'objectif présent.

Le système proposé répond aux deux exigences définies précédemment : il n'écarte que des solutions ne correspondant à aucun exemple positif dans les données, et le surcoût impliqué se limite à la construction des listes d'événements au moment du chargement. D'autres améliorations demeurent néanmoins possibles ; on pourrait par exemple utiliser le nombre exact d'épisodes séparant deux épisodes au lieu d'employer les notions plus génériques d'antériorité et de postériorité. Ce perfectionnement n'a pas été implémenté, principalement en raison du nombre souvent assez réduit d'épisodes présents en moyenne pour chaque séquence dans les données manipulées.

4.3.2.3 Exemple - voisinage d'une séquence

Afin d'illustrer le fonctionnement des différents opérateurs de voisinage introduits dans cette section ainsi que l'apport des restrictions imposées à la génération des voisins, nous présentons ici un exemple complet d'exploration du voisinage d'une séquence. Dans cet exemple (artificiel), nous considérons les attributs $\{A, B, C, D, E, F, G, H, J, K, L, M\}$ et la séquence $S = \{(A, B, C); (F)\}$. Les données sont représentées par six individus $I_1, \ldots, I_6$ possédant les séquences d'événements suivantes :

— Instance $I_1 = \{(A, E, F, G); (A, B, F)\}$

— Instance $I_2 = \{(C, D); (F, G); (L)\}$

— Instance $I_3 = \{(A, B, D, E); (J); (K, L)\}$

— Instance $I_4 = \{(M); (C); (H)\}$

— Instance $I_5 = \{(A); (C); (L)\}$

— Instance $I_6 = \{(A, B, C); (F, G)\}$

Examinons tout d'abord les opérateurs de voisinage les plus simples, qui ne sont pas affectés par les restrictions : *remove_event* et *remove_episode*. Appliqués à la séquence S, ceux-ci produisent respectivement les voisins suivants :

— *remove_event* $(S) : \{(B, C); (F)\}, \{(A, C); (F)\}, \{(A, B); (F)\}, \{(A, B, C)\}$

— *remove_episode* $(S) : \{(A, B, C)\}, \{(F)\}$

Nous présentons dans la Table 4.1 les relations d'ordre possibles entre événements, requises par les opérateurs *add_episode* et *add_event*. Pour une meilleure lisibilité, seules les lignes concernant les événements présents dans la séquence S étudiée sont montrées. A partir de ces informations de base, nous déterminons ensuite les événements pouvant entrer en relation avec l'épisode complet (A, B, C) par intersection des listes de chacun de ses événements constituants (A, B et C). Ce dernier choix peut amener certains problèmes, qui sont discutés dans la suite de cette partie.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

événement/épisode	Avant	Simult	Après
A	A, E, F, G	B, C, D, E, F, G	A, B, C, F, G, J, K, L
B	A, E, F, G	A, C, D, E, F	F, G, J, K, L
C	A, M	A, B, D	F, G, H, L
F	A, C, D, E, F, G	A, B, E, G	A, B, F, L
(A, B, C)	A	D	F, G, L

TABLE 4.1 – Relations d'ordre possibles entre les différents événements et épisodes de la séquence S

L'opérateur *add_event* peut être appliqué à deux endroits dans S, au niveau de l'épisode (A, B, C) ou de l'épisode (F). Pour ces deux emplacements, les événements possibles à ajouter sont définis ainsi :

— dans le premier cas, on cherche les événements pouvant se produire en même temps que (A, B, C) mais avant F. Ici, on a donc $evts = Simult((A, B, C)) \cap Avant(F) = \{D\} \cap \{A, C, D, E, F, G\} = \{D\}$.

— de la même manière, dans le second cas, on cherche les événements pouvant se produire en même temps que F mais après (A, B, C). On a donc $evts = Simult(F) \cap Après((A, B, C)) = \{A, B, E, G\} \cap \{F, G, L\} = \{G\}$.

Les voisins possibles de la séquence S générés par cet opérateur sont donc $\{(A, B, C, D); (F)\}$ et $\{(A, B, C); (F, G)\}$.

L'opérateur *add_episode* peut quant à lui être appliqué à trois endroits dans S : avant (A, B, C), entre les deux épisodes ou après (F). Pour ces trois emplacements, les événements possibles à ajouter dans le nouvel épisode créé sont définis ainsi :

— si l'épisode se situe en début de séquence, on cherche les événements pouvant se produire avant (A, B, C) et F, c'est-à-dire :
$evts = Avant((A, B, C)) \cap Avant(F) = \{A\} \cap \{A, C, D, E, F, G\} = \{A\}$.

— si l'épisode se situe entre (A, B, C) et (F), on cherche les événements :
$evts = Après((A, B, C)) \cap Avant(F) = \{F, G, L\} \cap \{A, C, D, E, F, G\} = \{F, G\}$.

— enfin, si l'épisode se situe en fin de séquence, on cherche alors les événements :
$evts = Après((A, B, C)) \cap Après(F) = \{F, G, L\} \cap \{A, B, F, L\} = \{F, L\}$.

Les voisins possibles de la séquence S générés par cet opérateur sont donc :

— $\{(A); (A, B, C); (F)\}$, $\{(A, B, C); (F); (F)\}$,

— $\{(A, B, C); (G); (F)\}$,

— $\{(A, B, C); (F); (F)\}$,

— $\{(A, B, C); (F); (L)\}$.

A noter que dans cet exemple, la méthode aboutit à la création de deux voisins identiques (la séquence $\{(A, B, C); (F); (F)\}$). Au total, ces deux opérateurs permettent de générer 7 nouvelles

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

solutions potentielles. En comparaison, une génération exhaustive sans restriction génère 60 solutions (5 emplacements possibles, 12 événements différents).

On remarque assez aisément, néanmoins, que même après restriction, tous les voisins nouvellement créés ne correspondent pas à des séquences présentes dans les données. En réalité, seule $\{(A, B, C); (F, G)\}$ fait partie de la séquence d'au moins un individu (en l'occurrence, I_6). L'algorithme d'optimisation continue à procéder à des évaluations de solutions n'ayant aucun support dans les données ; les restrictions appliquées au voisinage ne sont donc pas optimales. Par conséquent, il convient de se demander s'il est possible de faire mieux à ce niveau. Plus précisément, nous posons à présent la question inverse : de quelle manière peut-on écarter tous les voisins n'ayant aucun support dans les données avant même de les évaluer ?

Afin de rendre cette interrogation plus factuelle, nous nous intéressons à la séquence $\{(A, B, C, D); (F)\}$, générée par l'opérateur *add_event*. Ici, le problème réside dans le fait que l'événement D ne devrait pas pouvoir venir s'ajouter à l'épisode (A, B, C), étant donné que l'épisode (A, B, C, D) n'apparaît jamais dans les données. Ce problème serait donc résolu si on disposait d'une table répondant à la question "quels autres événements peuvent apparaître dans un épisode contenant déjà A, B et C ?", c'est-à-dire un équivalent de la table $Simult(evt)$, mais pour les épisodes et non plus les événements. Cette solution ne paraît cependant pas réalisable en pratique. En effet, le nombre d'épisodes possibles croît de manière exponentielle avec le nombre d'attributs, et ce dernier est souvent important (plus de 10 000). Même en ne considérant que les épisodes apparaissant dans les séquences des individus positifs, il faudrait inclure dans les tables une entrée pour chaque tel épisode, et pour chacun de ses sous-épisodes possibles, l'ensemble ainsi défini formant l'ensemble des composants (épisodes) possibles pour les règles de classification. Cela nécessiterait également d'ajouter une table, recensant les épisodes pouvant se produire dans la même séquence que celui considéré. Ce changement de niveau, d'événement vers épisode, serait donc très probablement impossible pour la plupart des cas d'étude réels. De plus, même si la solution s'avérait réalisable en pratique, il n'est pas certain que le surcoût engendré, beaucoup plus important que dans la version "événement", soit amorti par le gain en termes d'évaluations évitées.

La méthode de restriction du voisinage proposée n'est donc certes pas idéale, dans la mesure où elle ne permet pas d'écarter toutes les solutions inintéressantes, c'est-à-dire ne correspondant à aucune instance des données. Cependant, elle permet, à moindre coût, de réduire considérablement le nombre de nouvelles solutions à évaluer, ce qui constituait l'objectif initial de la démarche. Dans l'exemple artificiel présenté ci-dessus, le nombre de voisins générés passe de 60 à 7, ce qui constitue déjà une baisse notable. Sur des cas pratiques contenant plusieurs milliers d'attributs, le facteur de réduction atteint en général un ordre de grandeur de 100 ou plus, pour un surcoût minime.

4.3.3 Evaluation des solutions - *matching* de séquences

Concepts L'évaluation de la qualité des nouvelles solutions, générées lors de l'exploration du voisinage, est en général la composante la plus coûteuse (en termes de temps d'exécution) de l'algorithme d'optimisation dans son ensemble. Cela est d'autant plus vrai que les données manipulées sont de taille importante, étant donné que chaque évaluation requiert un parcours complet des données. L'ajout de séquences d'événements complique également cette phase d'évaluation. En effet, dans les versions précédentes de MOCA-I, les règles ne contenaient que des termes d'attributs symboliques ou numériques. Ces termes peuvent être évalués de manière immédiate : ils correspondent ou non à un exemple présent dans les données. Appliquer le même principe à l'évaluation des séquences reviendrait à rechercher, dans les règles générées, des séquences correspondant exactement à des séquences existantes dans les données. Procéder ainsi serait contraire

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

à l'idée même qui motive l'emploi de règles de classification, c'est-à-dire l'identification de sous-ensembles d'attributs dont la présence permet de décider de la classe d'une instance. Dans le cas des séquences, cela signifie qu'on recherche les *patterns* communs à plusieurs exemples, c'est-à-dire des sous-parties de ces exemples. Pour cette raison, lors de l'évaluation d'un terme de type *séquence* d'une règle, on cherche plutôt à déterminer si cette séquence est contenue dans la séquence d'événements décrivant chaque exemple présent dans les données. Il est donc nécessaire de définir la méthode permettant de comparer deux séquences entre elles, ou plutôt (le processus étant assymétrique) de déterminer si une séquence est contenue dans une autre.

Définition On dit qu'une séquence S_1 est présente dans une séquence S_2 si S_2 contient les mêmes événements que S_1 et que ceux-ci sont groupés en épisodes de manière identique. A noter cependant que S_2 peut contenir, en plus, d'autres épisodes et d'autres événements que ceux de S_1. Etant donné le caractère assymétrique de cette opération, nous parlons dans la suite de la recherche du *pattern* P dans une séquence S, équivalent à la recherche de S_1 dans S_2.

Exemple Afin de mieux comprendre cette définition, l'exemple ci-dessous expose la réponse souhaitée à l'évaluation de la présence du pattern $P = \{(A); (B)\}$ dans plusieurs séquences :

— $S = \{(D); (A); (C); (B); (E)\} \Longrightarrow$ vrai

— $S = \{(A, C); (B, D)\} \Longrightarrow$ vrai

— $S = \{(C); (B); (A)\} \Longrightarrow$ faux

— $S = \{(A, B); (C, D)\} \Longrightarrow$ faux

La méthode de comparaison que nous proposons s'inspire des travaux qui ont été menés dans le domaine des algorithmes de *string searching*, c'est-à-dire la détermination de la présence d'un chaîne de caractères au sein d'une autre. Le problème qui nous occupe contient en effet quelques similarités avec ce dernier, principalement sur la notion d'ordre entre les différents éléments du *pattern* recherché (caractères ou épisodes). La méthode proposée reprend certains des principes majeurs de l'algorithme de Knuth-Morris-Pratt [101], adapté à la structure particulière des séquences, notamment pour ce qui est des événements se produisant "en même temps". Cet algorithme permet d'accélérer le processus de comparaison des séquences en évitant de réexaminer des sous-séquences déjà comparées. L'objectif est de réduire autant que possible la complexité de la phase d'évaluation, en utilisant un algorithme performant couplé à diverses améliorations propres aux structures de données manipulées.

Implémentation L'Algorithme 7 décrit le processus utilisé pour déterminer si un pattern P est présent dans une séquence S. Une première technique permettant de réduire le temps d'évaluation moyen est de déterminer en premier lieu si tous les événements présents dans P sont également présents dans S, pour l'instant sans restriction sur l'ordre dans lequel ces événements sont agencés. De manière assez évidente, si P contient des événements absents de S, il est impossible que P *matche* S, et il est inutile d'examiner les séquences en détail. La raison pour laquelle ce mécanisme permet de gagner du temps sur l'évaluation est que nous enregistrons, dans la structure de données décrivant une séquence, la liste des événements présents dans cette séquence. Plus précisément, ces informations sont en réalité stockées sous la forme d'un tableau

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

de valeurs booléennes, chacune indiquant la présence ou non de l'attribut d'indice correspondant dans la séquence. Ce choix accroît légèrement l'espace mémoire requis pour enregistrer les données, mais permet de vérifier si un attribut est présent avec une complexité $O(1)$. De cette manière, vérifier la présence dans S de chaque attribut présent dans P se fait très rapidement. Ce processus permet donc d'éviter un nombre conséquent de comparaisons détaillées des deux séquences, et ce au prix d'un surcoût négligeable en temps.

Algorithme 7 : Algorithme de *matching* d'une séquence dans une autre

```
Data : deux séquences P et S
if tous les événements présents dans P sont aussi présents dans S then
    epi_P = epi_S = 0;
    while epi_S < nbEpisodes(S) do
        evt_P = evt_S = 0;
        while evt_S < nbEvenements(S, epi_S) do
            if attributeId(S, epi_S) == attributeId(P, epi_P) then
                evt_P++;
                if evt_P == nbEvenements(P, epi_P) then
                    evt_P = 0;
                    epi_P++;
                    if epi_P == nbEpisodes(P) then
                        return true;
                    end
                end
            else
                if attributeId(S, epi_S) > attributeId(P, epi_P) then
                    epi_S++;
                    evt_S = −1;
                    evt_P = 0;
                end
            end
            evt_S++;
            if (nbEvenements(P, evt_P) − evt_P) > (nbEvenements(S, evt_S) − evt_S) then
                evt_S = nbEvenements(S, epi_S);
            end
        end
        epi_S++;
        if (nbEpisodes(P) − epi_P) > (nbEpisodes(S) − epi_S) then
            return false;
        end
    end
else
    return false;
end
```

Dès lors que cette première phase a permis d'assurer que tous les événements présents dans P apparaissent dans S, il devient nécessaire de vérifier qu'ils apparaissent dans le même ordre dans les deux séquences. Les événements contenus au sein d'un même épisode sont ordonnés en permanence par *attributeId* (un entier associé à chaque attribut) croissant. Cela ne traduit

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

en aucun cas un ordre d'apparition entre eux, mais sert uniquement à accélérer l'évaluation. L'idée générale de cette méthode d'évaluation est de ne parcourir qu'une seule fois la séquence S, et d'utiliser le fait que les événements sont triés au sein de chaque épisode pour épargner des comparaisons inutiles.

Passée cette première étape, le reste du mécanisme est similaire à l'algorithme de Knuth-Morris-Pratt mentionné précédemment [101]. Au début de la phase d'évaluation, deux pointeurs i_p et i_s sont positionnés sur le premier événement du premier épisode de P et S, respectivement. Si i_s pointe sur un événement dont l'*attributeId* est supérieur à celui de l'événement pointé par i_p, alors il est impossible que ce dernier soit présent dans l'épisode courant de S, étant donné que les événements sont ordonnés au sein de l'épisode. Dans ce cas, i_s est incrémenté jusqu'à pointer sur le premier événement de l'épisode suivant de S, et i_p revient au début de l'épisode courant de P. Si les événements pointés par i_p et i_s sont identiques, les deux pointeurs sont incrémentés une fois. Si i_s pointe sur un événement dont l'*attributeId* est inférieur à celui pointé par i_p, alors seul i_s est incrémenté une fois. Lorsque i_p dépasse le dernier événement de l'épisode de P courant, i_p est incrémenté pour pointer au début de l'épisode suivant. A tout moment, s'il reste davantage d'événements à *matcher* dans l'épisode courant de P que dans l'épisode courant de S, cela signifie qu'il sera impossible de *matcher* l'intégralité de l'épisode de P dans l'épisode de S. Dans ce cas, i_s est incrémenté jusqu'au début de l'épisode suivant. De la même manière, s'il ne reste pas suffisamment d'épisodes dans S pour *matcher* le restant de la séquence P, alors la recherche s'arrête et renvoie un résultat négatif.

L'exemple ci-dessous présente une exemple de *matching* du pattern P dans la séquence S, avec :

$P = \{(A); (B, F); (K, M)\}$
$S = \{(C, R, Q); (A, P); (M, Z); (A, B, C, F, G, L); (K, L, M, N, Z)\}$

A chaque étape, les symboles en gras et soulignés sont ceux pointés par les indices i_p et i_s respectivement. Le vert indique un *matching* réussi, le rouge un échec.

$P = (\underline{\mathbf{A}}); (B, F); (K, M)$
$S = (\underline{\mathbf{C}}, R, Q); (A, P); (M, Z); (A, B, C, F, G, L); (K, L, M, N, Z)$
$\Rightarrow$ *matching* impossible car A < C $\Rightarrow$ épisode suivant de S

$P = (\underline{\mathbf{A}}); (B, F); (K, M)$
$S = (C, R, Q); (\underline{\mathbf{A}}, P); (M, Z); (A, B, C, F, G, L); (K, L, M, N, Z)$
$\Rightarrow$ *matching* terminé pour l'épisode de P $\Rightarrow$ épisode suivant de P et de S

$P = (A); (\underline{\mathbf{B}}, F); (K, M)$
$S = (C, R, Q); (A, P); (\underline{\mathbf{M}}, Z); (A, B, C, F, G, L); (K, L, M, N, Z)$
$\Rightarrow$ *matching* impossible car B < M $\Rightarrow$ épisode suivant de S

$P = (A); (\underline{\mathbf{B}}, F); (K, M)$
$S = (C, R, Q); (A, P); (M, Z); (\underline{\mathbf{A}}, B, C, F, G, L); (K, L, M, N, Z)$
$\Rightarrow$ *matching* toujours possible car B > A $\Rightarrow$ événement suivant de S

$P = (A); (\underline{\mathbf{B}}, F); (K, M)$
$S = (C, R, Q); (A, P); (M, Z); (A, \underline{\mathbf{B}}, C, F, G, L); (K, L, M, N, Z)$
$\Rightarrow$ *matching* $\Rightarrow$ événement suivant de P et S

$P = (A); (B, \underline{\mathbf{F}}); (K, M)$
$S = (C, R, Q); (A, P); (M, Z); (A, B, \underline{\mathbf{C}}, F, G, L); (K, L, M, N, Z)$
$\Rightarrow$ *matching* toujours possible car F > C $\Rightarrow$ événement suivant de S

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

$P = (A); (B, \underline{\mathbf{F}}); (K, M)$
$S = (C, R, Q); (A, P); (M, Z); (A, B, C, \underline{\mathbf{F}}, G, L); (K, L, M, N, Z)$
$\Rightarrow$ *matching* terminé pour l'épisode de $P \Rightarrow$ épisode suivant de P et de S

$P = (A); (B, F); (\underline{\mathbf{K}}, M)$
$S = (C, R, Q); (A, P); (M, Z); (A, B, C, F, G, L); (\underline{\mathbf{K}}, L, M, N, Z)$
$\Rightarrow$ *matching* $\Rightarrow$ événement suivant de P et S

$P = (A); (B, F); (K, \underline{\mathbf{M}})$
$S = (C, R, Q); (A, P); (M, Z); (A, B, C, F, G, L); (K, \underline{\mathbf{L}}, M, N, Z)$
$\Rightarrow$ *matching* toujours possible car M > L $\Rightarrow$ événement suivant de S

$P = (A); (B, F); (K, \underline{\mathbf{M}})$
$S = (C, R, Q); (A, P); (M, Z); (A, B, C, F, G, L); (K, L, \underline{\mathbf{M}}, N, Z)$
$\Rightarrow$ *matching* terminé ; P est bien présent dans S

Dans cette implémentation, nous ne considérons que le *matching* exact de séquences. Cela signifie que la séquence évaluée doit être présente en entier et dans le même ordre dans la séquence avec laquelle elle est comparée. On peut légitimement imaginer d'autres interprétations moins strictes de cette notion de *matching*, en autorisant par exemple un certain nombre d'erreurs, où une erreur pourrait être l'absence d'un événement ou une inversion de position entre deux événements, comme cela se fait pour l'alignement de séquences d'ADN en bioinformatique [125]. Une autre approche à ce problème est l'utilisation de *classifieurs flous* (*fuzzy classifiers* [87]), dans lesquels le résultat d'un *matching* n'est pas un résultat binaire (vrai/faux), mais un score de certitude, typiquement choisi entre 0 et 1. Cette approche est intéressante (voir [141] pour un exemple), mais est trop éloignée des principes au cœur de MOCA-I pour pouvoir être considérée directement.

4.3.4 Extension du modèle - séquences multi-niveaux

Le modèle proposé ci-dessus décrit une séquence comme une suite ordonnée d'épisodes, chacun comprenant un ou plusieurs événement(s), ces événements étant considérés comme simultanés (au sein d'un même épisode). Ce faisant, on définit implicitement une granularité temporelle : des événements sont considérés comme simultanés s'ils sont "suffisamment rapprochés" dans le temps. Ce modèle permet de s'adapter correctement à différents problèmes, en choisissant la granularité en fonction des besoins et des données. Pour prendre des exemples issus du contexte hospitalier, on peut considérer que les événements sont regroupés en fonction du séjour auquel ils appartiennent, ou de l'unité médicale dans laquelle l'événement a eu lieu. Pour davantage de précision et si le besoin s'en fait sentir, il est possible de regrouper les événements se produisant dans la même journée, ou dans la même heure, et ainsi de suite.

Des discussions avec les experts métier ont permis d'établir que des granularités telles que la journée (ou plus fines) ne présentent *a priori* pas d'intérêt, du moins pour les cas d'étude qui nous intéressent. Cependant, il serait souhaitable de pouvoir inclure plusieurs granularités différentes dans la représentation d'une séquence, afin d'élargir le spectre des applications possibles de la méthode. En particulier, la modélisation du parcours patient à la fois inter-séjour et intra-séjour représente un aspect important des besoins. Pour cette raison, nous faisons évoluer le modèle proposé afin d'intégrer deux niveaux de séquentialité différents. Considérons par exemple la séquence suivante telle qu'interprétée par la modélisation précédente :

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

$seq = (A, B); (C); (D, E, F); (K)$

Cette séquence est constituée de quatre épisodes, ordonnés par ordre d'occurrence, obtenus en appliquant une granularité g_1 à l'ensemble des événements A, B, C, D, E, F, K et à leurs dates/heures d'occurrence respectives. Par exemple, on considère que g_1 groupe les événements s'ils se produisent dans la même unité médicale.

Une modélisation étendue à deux niveaux envisageable pour cette séquence serait :

$seq = [(A, B); (C)]\ [(D, E, F); (K)]$

Dans celle-ci, on retrouve les quatre épisodes de la précédente version, mais ces groupes ont été eux-mêmes rassemblés dans deux ensembles (délimités par les crochets). Ces nouveaux ensembles sont créés par l'application d'une seconde granularité g_2, laquelle peut par exemple agglomérer les groupes d'événements se produisant au cours du même séjour hospitalier.

Dans une telle représentation, les processus de chargement des données, de génération des règles initiales et d'évaluation des solutions ne sont que peu modifiés, et suivent assez naturellement le modèle proposé. Le mécanisme le plus impacté par cette extension du modèle est la génération du voisinage des séquences. L'ensemble des opérateurs de voisinage employés pour la génération se doit d'être étendu pour tenir compte de la nouvelle structure. Aux opérateurs précédemment définis viennent s'ajouter les suivants :

— *add_set* : ajoute un nouvel ensemble, constitué d'un unique épisode, lui-même contenant un unique événement.

— *remove_set* : supprime un ensemble existant (et tous les épisodes et événements qu'il contient). Ne peut pas être utilisé si la séquence ne contient qu'un seul ensemble.

Ces opérateurs sont similaires dans l'idée à ceux définis dans la modélisation de base, mais adaptés à la nouvelle représentation. Les restrictions apportées au voisinage au moment de la génération ne sont pas modifiées ; prendre en compte la nouvelle structure à plusieurs niveaux pour définir des listes d'événements plus fines est envisageable, mais le gain escompté est assez faible par rapport à la surcharge en termes de complexité. Cela demeure cependant une amélioration possible à explorer, de même qu'une éventuelle extension à davantage de niveaux (ou granularités) différents.

Il est à noter que cette évolution du modèle a été apportée assez tardivement dans le processus de développement, et ne fait donc pas partie de la version de l'algorithme qui a été testée de manière extensive (voir partie suivante). Des tests ont permis de valider que les deux versions de l'algorithme retrouvaient les mêmes solutions quand appliquées sur les mêmes données, au prix d'une légère augmentation du temps d'exécution, augmentation imputable à un voisinage plus large et à une évaluation plus coûteuse. L'une des applications principales imaginées pour cette extension du modèle visait à représenter les différents séjours d'un patient comme un niveau supérieur de séquence (séjour 1 puis séjour 2, etc.). Faute de données adéquates, nous n'avons pas pu tester cette évolution du modèle sur des cas d'étude concrets.

4.3.5 Paramétrage de la méthode

A l'inverse des méthodes développées pour la gestion des données numériques, l'algorithme MOSC développé dans ce chapitre ne présente pas réellement de variantes différentes, et ne nécessite que peu de calibrage visant à déterminer les valeurs optimales pour les paramètres. En

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

effet, les seuls paramètres introduits dans cette méthode sont la limite du nombre d'épisodes dans une séquence, et la limite du nombre d'événements dans un épisode. Par défaut, ceux-ci avaient été fixés à 3, et des tests préliminaires ont confirmé qu'augmenter cette limite ne permettait pas d'obtenir des séquences plus complexes. Il est probable que ces paramètres n'aient qu'une influence réduite sur l'algorithme ; autoriser des séquences et épisodes plus longs accroît le travail de la méthode de recherche locale, mais les restrictions apportées sur le voisinage des séquences permettent d'éviter que le nombre de voisins devienne trop grand même pour des séquences plus longues.

La seule réelle variante possible de l'algorithme concerne la gestion, ou non, des événements "hors-séquence". Dans la modélisation actuelle, on considère que tous les attributs marqués comme événements possibles ne peuvent apparaître, dans une règle, qu'au sein d'un attribut de type *séquence*. Ainsi, chaque événement de la règle doit être ordonné par rapport à tous les autres. Une variante serait donc d'autoriser les règles à inclure des attributs-événements en-dehors de la séquence, donc sans notion d'ordre par rapport aux termes de la séquence. Un exemple de telle règle serait :

$R = \{A = true, age > 65, seq = \{(B = true); (C = true)\}\}$

où A, B et C sont des attributs temporels. Dans cette règle, l'événement A n'a pas de contrainte sur son ordre d'apparition par rapport à B et C ; cette règle est donc plus générique que si A devait impérativement être inclus dans la séquence. L'intérêt de cette variante est discutable ; les règles obtenues sont potentiellement plus riches de cette manière, mais cela implique une charge accrue pour la méthode de recherche locale. Les tests menés n'ont pas permis de mettre en avant l'apport de cette variante, qui peut de plus sembler quelque peu contradictoire avec l'objectif de détection de séquences d'événements dans les données. Pour ces raisons, cette fonctionnalité n'a pas été retenue dans l'implémentation de la méthode utilisée pour la validation des performances et la comparaison avec d'autres algorithmes de classification.

4.3.6 Durée entre événements

Comme expliqué dans la partie introductive de ce chapitre qui en détaillait les motivations et objectifs, la représentation souhaitée des données temporelles doit comporter trois parties : séquentialité, récurrence et durée (quantitative) entre événements. Nous avons choisi de n'intégrer que les deux premiers aspects dans la modélisation et l'implémentation de MOSC. Ce choix a été amplement discuté et argumenté ; il impliquait par ailleurs la possibilité de traiter l'aspect manquant (temporel "quantitatif") en post-traitement, c'est-à-dire après que le processus d'optimisation ait trouvé un ensemble de solutions non-dominées. De cette manière, les séquences trouvées dans les règles de ces solutions peuvent être précisées en allant rechercher des informations non-exploitées par l'algorithme d'optimisation. On rappelle en effet que les données temporelles ont un marqueur (ou *timestamp*) associé à chaque événement. Cette information n'est pour le moment utilisée que pour déterminer l'ordre respectif des événements, mais il est possible de l'exploiter davantage afin d'affiner les règles "séquentielles" construites par l'algorithme. Afin de mieux visualiser l'apport de telles techniques, nous utilisons dans la suite de cette section une règle de classification fictive $R = \{\{(A); (B,C); (D)\} \Rightarrow P\}$, où $Q = (A); (B,C); (D)$ est le *prédicat* et P est la *prédiction*.

Deux types de précisions revêtent un intérêt majeur, notamment dans le cadre applicatif qui nous occupe dans cette étude :

— une première idée consiste à déterminer le délai d'occurrence de Q, c'est-à-dire le temps total écoulé entre le premier et le dernier épisodes de la séquence Q. Dans l'exemple fictif utilisé, cela revient à calculer le temps entre l'apparition de l'épisode (A) et de l'épisode (D), chez un individu contenant le prédicat complet Q. D'un point de vue métier, il est en

© 2017 Tous droits réservés. lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

effet intéressant de pouvoir estimer ce délai ; un *pattern* se produisant en moyenne en trois jours est différent d'un *pattern* se produisant en six mois. Cette précision peut également faciliter la validation par un expert des règles trouvées.

— une seconde possibilité est de rechercher la durée entre l'occurrence du *pattern* du prédicat Q et l'occurrence de la prédiction P. Cette précision permet, par exemple, de différencier les règles prédisant des effets à court ou long terme [43].

Nous nous sommes surtout intéressés au premier point, à savoir le délai d'occurrence de Q. En effet, nous ne disposions pas pour les cas d'étude des temps d'occurence des prédictions, ce qui rend le calcul du délai entre Q et P impossible. Afin de calculer le délai d'occurence de Q, il convient tout d'abord de séparer les solutions obtenues en leurs règles constituantes respectives. Il paraît en effet difficile de rechercher un délai d'occurrence sur une solution complète (*i.e.* un ensemble de règles) ; nous nous situons bien ici au niveau *règles*. Ensuite, pour chaque règle, le système examine chaque instance des données, et repère les séquences contenant le prédicat Q. Pour chacun de ces individus, le délai d'occurence de Q est calculé et ajouté à une liste. Une fois cet examen achevé, le système calcule diverses statistiques pour chaque règle individuellement : moyenne, écart-type, valeurs extrêmes (min-max) et quartiles/déciles. Ces valeurs fournissent une information plus complète sur les règles, en plus de permettre une représentation visuelle, sous forme de graphiques, des délais d'occurrence impliqués et de leur répartition. Un exemple d'un tel graphique, obtenu à partir de valeurs fictives mais réalistes, est présenté dans la Figure 4.2 :

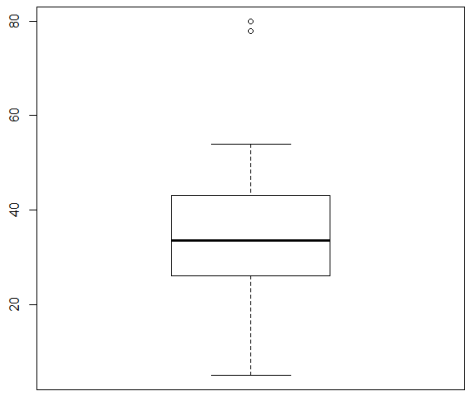


FIGURE 4.2 – Distribution du temps total d'occurence pour la séquence d'une règle (en jours)

Ici, on observe que le *pattern* d'événements correspondant à ce graphique se produit la plupart du temps sur une durée allant de 30 à 40 jours, avec quelques valeurs extrêmes aux alentours de 80 jours.

Cependant, toutes les règles trouvées parmi les solutions de l'archive ne présentent pas le même intérêt par rapport à une analyse aussi fine. Il paraît plus instructif de ne procéder à celle-ci que pour les règles de confiance élevée, dans la mesure où ces règles incluent souvent des *patterns* plus complets que ceux trouvés par les règles de confiance moindre. Ces *patterns* plus riches et plus complexes semblent mieux se prêter à l'analyse décrite dans cette section.

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Pour cette raison, seules les règles de meilleure confiance sont inspectées. Le procédé demeure cependant applicable à n'importe quelle règle contenant une séquence d'au moins deux épisodes.

La précision des règles de classification séquentielles par ajout d'informations sur la temporalité vient conclure cette section. Nous pouvons à présent évaluer l'algorithme proposé, MOSC, afin de valider son efficacité.

4.4 Validation et cas d'étude

Dans cette section, nous cherchons à évaluer l'efficacité de l'algorithme MOSC. Dans un premier temps, nous détaillons les données des cas d'étude réels utilisés pour l'évaluation. Ensuite, nous présentons une méthode de transformation des données nécessaire pour la comparaison avec des algorithmes de classification ne supportant pas les données temporelles. Nous comparons ensuite MOSC avec plusieurs autres algorithmes, puis analysons les résultats obtenus. Nous concluons cette section par une discussion de la convergence de MOSC, et la présentation de règles de classification comportant des séquences telles que trouvées sur les données des cas d'étude.

4.4.1 Données

Les différents jeux de données utilisés dans cette étude sont extraits du système d'information médico-administratif du GHICL (Groupement des Hôpitaux de l'Institut Catholique de Lille). Plus précisément, la base de ces données est formée par les codes PMSI enregistrés pour les activités de Médecine, Chirurgie et Obstétrique (MCO). Nous considérons ici cinq jeux de données, correspondant à cinq prédictions d'événements ou d'états médicaux. La Table 4.2 présente différentes statistiques pour ces cinq jeux de données : nombre d'attributs, nombre d'instances et pourcentage d'instances positives. A noter que la matrice de ces données est très creuse, avec moins de 1% de données présentes, et fortement déséquilibrées (faible pourcentage d'instances positives). Ces données constituent donc de bons exemples de cas d'étude sur lesquels nous souhaitons pouvoir appliquer la méthode de classification proposée. Une instance correspond à un séjour ; un patient peut donc être décrit par plusieurs instances s'il a effectué plusieurs séjours. Dans les séquences, des événements sont considérés comme simultanés s'ils se produisent au sein de la même unité médicale (par exemple, chirurgie ou neurologie). Les trois premiers jeux correspondent au même ensemble de données avec trois prédictions différentes ; soins palliatifs, escarres et chute. Le quatrième concerne les mêmes patients que précédemment, mais avec moins d'attributs. Cela est lié à la phase d'extraction des données à partir du système d'information hospitalier. Les attributs manquants ne sont pas cruciaux, et sont souvent redondants avec d'autres informations présentes dans les données. Le dernier jeu de données regroupe les informations disponibles pour la prédiction de présence de *bactéries résistantes à la méthicilline*. Ce cas d'étude concerne un problème d'hygiène très important dans les hôpitaux, avec l'apparition et la propagation de bactéries (telles que le staphylocoque doré) ne pouvant être combattues par les antibiotiques.

4.4.2 *Sliding Windows* - discrétisation temporelle

Compte tenu du manque de travaux sur le problème de la classification sur données hétérogènes avec temporalité, nous ne sommes pas en mesure de comparer directement la méthode proposée avec d'autres approches. Nous n'affirmons pas que la solution développée dans cette étude soit la seule possible. Par exemple, il est probable que l'algorithme Apriori [5] étendu aux données déséquilibrées [144], couplé avec une méthode telle que CBA [119] pour construire les

© 2017 Tous droits réservés. lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

	Nombre d'attributs	Nombre d'instances	% de positifs
palliatif	12 535	102 996	1.57%
escarre	12 535	102 996	3.7%
chute	12 535	102 996	1.15%
deces	11 413	102 896	3.18%
bmrNoso	11 450	97 359	3.7%

TABLE 4.2 – Informations jeux de données réels complets

classifieurs, soit une autre approche viable. Cependant, cela nécessiterait un travail conséquent qui pourrait faire l'objet d'une contribution à part entière. Cela serait clairement hors du champ de la présente étude, aussi choisissons-nous de laisser le développement de telles méthodes à d'autres.

Afin de pouvoir malgré tout comparer notre approche à d'autres méthodes de classification, nous nous intéressons ici à une méthode de transformation des données temporelles en données non-temporelles. De cette manière, il devient possible d'appliquer des algorithmes de classification classique (non-temporelle) sur ces données, ce qui fournit une base de comparaison. La méthode employée pour cette phase de pré-traitement des données est appelée *Sliding Windows*. Il s'agit d'une technique souvent utilisée pour le traitement des flux constants de données (par exemple en provenance d'appareils de mesure), qui consiste à découper une séquence en plusieurs parties et de considérer chaque partie comme une instance indépendante des données [3]. Toute notion de temporalité est donc abolie par ce processus, que ce soit entre les différentes *fenêtres* ou au sein de chaque fenêtre. L'idée sous-jacente ici est néanmoins de considérer des événements rapprochés dans le temps comme faisant partie d'une même instance, ce qui implique de leur reconnaître un lien commun. L'une des problématiques majeures liées à cette méthode est la détermination d'une taille convenable pour la fenêtre. C'est d'ailleurs sur ce point qu'il est possible d'établir un parallèle intéressant avec le format des données temporel que nous avons utilisé jusqu'à présent. En effet, il semble naturel d'utiliser le découpage "séquence/épisodes/événements" pour définir qu'une *fenêtre* correspond à un épisode, c'est-à-dire un groupe d'événements.

Afin d'illustrer ce procédé, nous prenons l'exemple d'une instance présente dans des données temporelles, traduite en instances de données non-temporelles via *sliding windows*.

$$temp = \{type = B, M = 0.57, seq = \{(D, E); (F); (G, H, L)\}\}$$

$$\Downarrow$$

$$nonTemp_1 = \{type = B, M = 0.57, D = true, E = true\}$$
$$nonTemp_2 = \{type = B, M = 0.57, F = true\}$$
$$nonTemp_3 = \{type = B, M = 0.57, G = true, H = true, L = true\}$$

Comme on le voit sur cet exemple, une nouvelle instance est créée pour chaque épisode présent dans la séquence initiale. Aux événements de l'épisode sont ajoutés les attributs dépourvus de notion de temporalité (*i.e.* hors de la séquence), qui s'appliquent pour chaque partie de la séquence. Cette méthode peut s'apparenter à une discrétisation temporelle ; la transformation simplifie les données afin de les rendre exploitables par d'autres algorithmes moins spécialisés, mais cela se fait au prix d'une perte d'information au niveau de la séquentialité.

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

4.4.3 Comparaison avec d'autres algorithmes

Nous utilisons la méthode des *Sliding Windows* afin de pouvoir appliquer des algorithmes de classification pour données non-temporelles aux données (temporelles) dont nous disposons. De cette manière, nous pouvons comparer l'approche proposée, MOSC, avec les algorithmes de classification suivants : MOCA-IQ, C45CS [158], Ripper [34], CART [22] et RandomForest [81]. Le premier est l'algorithme qui a inspiré MOSC. C45CS et Ripper sont deux algorithmes performants, qui ont déjà été utilisés pour comparaison dans le chapitre précédent. Nous utilisons toujours l'implémentation disponible sur KEEL [6] pour ceux-ci. CART et RandomForest sont deux autres algorithmes de classification reconnus, qui disposent d'une implémentation supportant les matrices de données larges et creuses dans le package Python Scikit-learn [131].

Cependant, afin que la comparaison soit équitable, les modèles de classification obtenus par tous les algorithmes doivent être évalués sur les données temporelles originales. Cela signifie que les règles de classification obtenues par les algorithmes autres que MOSC doivent être transformées en règles temporelles. Pour cela, les termes utilisant des attributs temporels dans les règles non-temporelles sont regroupés et considérés comme simultanés. Ce processus, décrit dans l'exemple ci-dessous, agit à l'inverse des *Sliding Windows* afin de retrouver la temporalité :

$$nonTemp_1 = \{type = B, M = 0.57, D = true, E = true\}$$

$$\Downarrow$$

$$temp_1 = \{type = B, M = 0.57, seq = \{(D, E)\}\}.$$

Après traduction en règles temporelles, les classifieurs peuvent être comparés aux résultats obtenus par MOSC. Le processus de transformation des données et résultats est résumé dans la Figure 4.3. Un problème important est que tous les algorithmes utilisés pour comparaison ne sont pas applicables aux données larges et creuses sont nous disposons. Pour cette raison, nous séparons le processus d'évaluation en trois étapes dans le reste de ce chapitre. Dans un premier temps, nous comparons MOSC avec les autres algorithmes sur des jeux de données réduits, qui correspondent à des problèmes de classification plus faciles et plus habituels. Dans un second temps, nous comparons MOSC sur les données complètes avec les algorithmes supportant celles-ci. Enfin, nous présentons des exemples de règles de classification temporelles.

4.4.3.1 Comparaison sur données réduites

Compte tenu de l'implémentation de ces méthodes dans KEEL, il est impossible d'appliquer C45CS et Ripper sur les jeux de données complets, à moins de disposer de centaines de GB de RAM au minimum pour charger les données, ce qui n'est pas réaliste. Comme dans la plupart des autres logiciels, l'implémentation dans KEEL n'est pas conçue pour supporter les matrices de données très larges et creuses qui nous intéressent ici. Afin de pouvoir malgré tout comparer MOSC et MOCA-IQ avec C45CS et Ripper, nous devons réduire la taille des jeux de données utilisés. Pour cela, nous réduisons d'abord le nombre d'instances, puis nous supprimons les attributs qui n'apparaissent pas dans les instances restantes. Cette réduction des données est effectuée aléatoirement pour éviter tous biais : toutes les instances positives sont conservées, puis un nombre égal d'instances négatives sont choisies au hasard et conservées dans les données réduites. Les jeux de données réduits ainsi formés sont décrits par les statistiques montrées dans la Table 4.3 : nombre d'attributs, nombre d'instances sur données temporelles, nombre d'instances sur données transformées par SW, pourcentage d'instances positives. Nous avons donc supprimé

© 2017 Tous droits réservés. lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

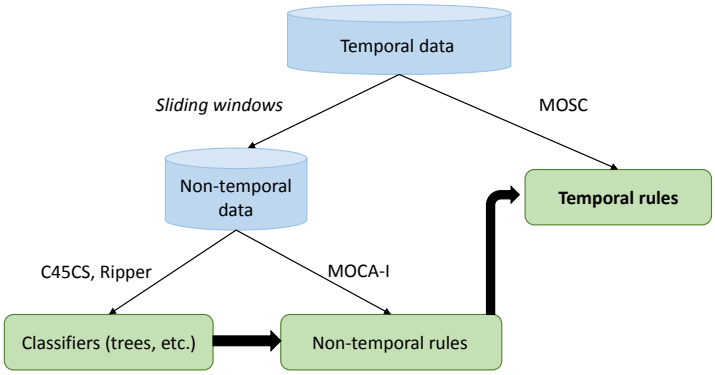


FIGURE 4.3 – Processus de transformation des données pour comparaison des méthodes

le déséquilibre des données, en conservant autant d'instances positives que négatives, par sous-échantillonnage des instances négatives. En effet, conserver le même déséquilibre que dans les données complètes signifierait ne conserver qu'une dizaine d'instances positives. Cela n'est pas faisable, car un nombre aussi faible de positifs rend le phénomène de sur-apprentissage prédominant pour n'importe quelle méthode de classification. Ces jeux de données réduits représentent donc des problèmes de classification plus simples à deux niveaux : taille des données réduite et absence de déséquilibre.

	Nombre d'attributs	Nombre d'instances (tempo)	Nombre d'instances (SW)	% de positifs
palliatif$_{red}$	1 290	1 475	2 901	50%
escarre$_{red}$	2 611	3 582	7 139	50%
chute$_{red}$	1 305	1 137	2 200	50%
deces$_{red}$	3 010	3 124	9 626	50%
bmrNoso$_{red}$	2 599	3 382	4 405	50%

TABLE 4.3 – Informations jeux de données réels réduits

Protocole La méthode de discrétisation temporelle par *Sliding Windows* est appliquée en prétraitement pour tous les algorithmes sauf MOSC. Les expérimentations sont menées selon le protocole et sur la même machine que dans le chapitre précédent. La Table 4.4 présente les valeurs moyennes de la f-mesure (et écarts-type associés) sur les 30 itérations pour chaque méthode et chaque jeu de données, pour l'ensemble de test. La Table 4.5 présente les rangs moyens de chaque méthode, également sur l'ensemble de test. La partie inférieure de cette table contient les différences statistiquement significatives détectées entre les différentes méthodes (test de Friedman), avec la $p-value$ associée. Par exemple, $MOSC > C45CS(p = 0.003)$ indique que MOSC est statistiquement meilleur que C45CS, avec une $p-value$ de 0.003. Tous les algorithmes sont exécutés jusqu'à convergence, avec une limite d'une heure pour MOCA-IQ et MOSC. Nous reportons l'analyse des résultats à la Section 4.4.4, après présentation de tous les résultats quantitatifs.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

	MOCA-IQ	MOSC	Ripper	C45CS	CART	RF
palliatif$_{red}$	0.84 ± 0.01	**0.88 ± 0.01**	0.63 ± 0.03	0.35 ± 0.23	0.75 ± 0.09	0.76 ± 0.07
escarre$_{red}$	0.83 ± 0.01	**0.84 ± 0.01**	0.61 ± 0.03	0.58 ± 0.02	0.73 ± 0.04	0.73 ±0.04
chute$_{red}$	**0.84 ± 0.01**	**0.84 ± 0.01**	0.57 ± 0.06	0.63 ± 0.01	0.72 ± 0.09	0.73 ± 0.08
deces$_{red}$	0.82 ± 0.01	**0.84 ± 0.01**	0.64 ± 0.02	0.50 ± 0.09	0.64 ± 0.08	0.64 ± 0.09
bmrNoso$_{red}$	**0.82 ± 0.01**	0.80 ± 0.01	0.42 ± 0.04	0.35 ± 0.13	0.71 ± 0.06	0.73 ± 0.05

MOSC > C45CS	($p = 0.003$)
MOSC > Ripper	($p = 0.027$)
MOCA-IQ > C45CS	($p = 0.007$)

TABLE 4.4 – F-mesure sur l'ensemble de test, données réduites

Algorithm	Ranking
MOSC	1.3
MOCA-IQ	1.7
CART	3.7
RF	3.9
Ripper	4.8
C45CS	5.6

TABLE 4.5 – Rangs moyens sur l'ensemble de test, données réduites. Les rangs montrent la position moyenne de chaque algorithme (1 étant la meilleure), basée sur la *f-mesure*

4.4.3.2 Comparaison sur données complètes

Dans cette seconde étape, nous comparons MOSC avec MOCA-IQ, CART [22] et Random-Forest [81] sur les données complètes. Comme à l'étape précédente, *Sliding Windows* est employé en phase de pré-traitement pour ces deux algorithmes ainsi que pour MOCA-IQ. Les résultats sur l'ensemble de test sont présentés dans la Table 4.6. Les rangs moyens de chaque algorithme sont présentés dans la Table 4.7. Tous les algorithmes sont exécutés jusqu'à convergence, avec une limite de temps de quatre heures pour MOCA-IQ et MOSC.

	MOCA-IQ	MOSC	CART	RF
palliative	0.38 ± 0.01	**0.58 ± 0.01**	0.37 ± 0.06	0.33 ± 0.07
ulcer	0.38 ± 0.01	**0.42 ± 0.08**	0.33 ± 0.03	0.21 ± 0.06
fall	0.21 ± 0.01	**0.25 ± 0.01**	0.15 ± 0.06	0.05 ± 0.05
death	0.29 ± 0.01	**0.41 ± 0.02**	**0.41 ± 0.02**	0.34 ± 0.04
MRB	0.31 ± 0.01	**0.34 ± 0.02**	0.32 ± 0.04	0.18 ± 0.05

MOSC > RF	($p = 0.002$)

TABLE 4.6 – F-mesure sur l'ensemble de test, données complètes

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Algorithm	Ranking
MOSC	1.1
CART	2.5
MOCA-IQ	2.6
RF	3.8

TABLE 4.7 – Rangs moyens sur l'ensemble de test, données complètes

4.4.4 Interprétation et discussion des résultats

Données réduites La comparaison sur les jeux de données réduits montre que MOSC et MOCA-IQ présentent les meilleures performances. Cette observation est confirmée par un test statistique de Friedman (mieux adapté aux comparaisons entre plus de deux méthodes), qui établit une dominance de MOSC sur C45CS et Ripper, et de MOCA-IQ sur C45CS. Ces résultats peuvent s'expliquer en regardant les classifieurs générés par C45CS et Ripper. Ceux-ci classent souvent toutes les instances comme positives (ou négatives) et ne produisent pas de règles de classification utiles d'un point de vue métier. En effet, ces règles utilisent uniquement les attributs renseignés pour toutes les instances (l'âge, par exemple), ce qui traduit une incapacité à gérer les matrices de données creuses, même pour un volume réduit. CART et RandomForest construisent des classifieurs de bonne qualité, mais restent inférieurs en termes de performance. La comparaison entre MOSC et MOCA-IQ donne un léger avantage à la première méthode, sans que cette différence soit significative. Ces deux méthodes produisent des classifieurs de très bonne qualité sur les jeux de données réduits.

Données complètes Nous comparons tout d'abord MOSC et MOCA-IQ sur ces données, car les données réduites ne permettent pas d'établir de différence. Ici, MOSC produit des résultats de meilleure qualité que MOCA-IQ sur tous les jeux de données, pour les ensembles d'apprentissage et de test. Cette différence de qualité n'est pas suffisante pour être statistiquement significative, mais peut être observée sur tous les jeux de données. Une comparaison qualitative des règles obtenues montre que les deux algorithmes utilisent souvent les mêmes attributs. MOSC obtient des résultats de meilleure qualité en classifiant correctement un nombre assez réduit d'instances supplémentaires qui ne peuvent être décrites que par une séquence d'événements. Dans certains cas cependant, MOSC crée des séquences d'un seul événement (donc sans temporalité), mais les règles sont meilleures que celles de MOCA-IQ. Nous expliquons ce comportement par une exploration plus efficace de l'espace de recherche pour MOSC, grâce aux restrictions de voisinage qui évitent de perdre du temps dans des régions non-prometteuses.

La comparaison avec CART et RandomForest montre que MOSC reste le plus efficace sur l'ensemble des jeux de données, et est statistiquement meilleur que RandomForest ($p\text{-}value = 0.002$). Il est également difficile de regarder en détail les classifieurs générés par CART et RandomForest, ceux-ci étant très complexes (plusieurs centaines de nœuds pour CART) et compliqués à interpréter, ce qui est problématique dans notre contexte. A la lumière de l'ensemble de ces résultats, nous pouvons donc affirmer que MOSC est la méthode la mieux adaptée pour traiter le problème de la classification des données temporelles qui nous intéressent.

4.4.5 Convergence de l'algorithme

Dans les expérimentations précédentes, nous avons choisi de fixer un critère d'arrêt sur le temps d'exécution à MOSC, de la même manière que pour les expérimentations qui ont été me-

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

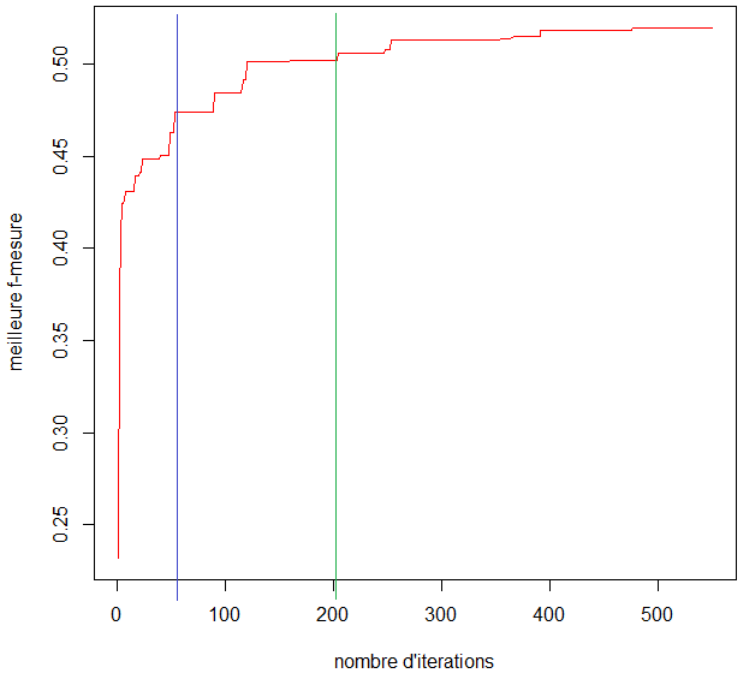


FIGURE 4.4 – Evolution de la f-mesure de la meilleure solution de l'archive au fil des itérations

nées dans le chapitre précédent. Cet arrêt forcé est nécessaire, dans la mesure où l'algorithme met en général longtemps à converger vers une archive de solutions intégralement explorées, trop longtemps pour pouvoir mener les expérimentations dans des délais raisonnables. Afin de vérifier que cet arrêt avant convergence ne biaise pas l'évaluation des performances, nous laissons l'algorithme converger en relevant à chaque itération la *f-mesure* la plus élevée parmi les solutions de l'archive. Cette solution correspond à celle retournée en sortie par MOSC dans l'implémentation actuelle. L'expérience est réalisée sur le jeu de données *deces*, qui représente un cas d'étude complet, avec une volumétrie importante, comme décrit précédemment.

Sur la Figure 4.4, la courbe rouge représente l'évolution de la f-mesure (sur l'ensemble d'apprentissage) de la meilleure solution de l'archive en fonction du temps. Les traits verticaux bleu et vert indiquent l'état de l'algorithme après une et quatre heures, respectivement. La convergence se fait en environ douze heures sur cette exécution, ce qui représente un peu moins de 600 itérations. On observe qu'après quatre heures (trait vert), la qualité de la meilleure solution est déjà très proche ($\approx 95\%$) de celle obtenue en laissant converger. Ce comportement général de l'algorithme, marqué par une forte amélioration de la qualité en début d'exécution puis une convergence lente vers l'optimum, est similaire à celui observé lors des expérimentations menées dans le chapitre précédent. Nous choisissons d'accepter cette légère dégradation de performance au profit d'un temps d'exécution plus acceptable, ce qui permet entre autres de valider les performances sur davantage de jeux de données.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

4.4.6 Cas d'étude - analyse qualitative des solutions

Nous présentons ici quelques exemples de règles de classification temporelles trouvées par MOSC sur les jeux de données complets utilisés pour évaluation et comparaison. Un premier exemple, pour la prédiction *décès* :

$$seq = \{\text{malnutrition grave} \rightarrow \text{douleurs chroniques}\} \Rightarrow \text{décès} \tag{4.1}$$

Les patients couverts par cette règles sont hospitalisés avec une condition médicale sévère. Des complications apparaissent ensuite durant le séjour, engendrant des douleurs chroniques et finalement la mort.

Une deuxième règle concerne la prédiction de bactéries (MRB) :

$$seq = \{(\text{injection d'adrénaline AND radiographie thoracique}) \rightarrow (\text{grabataire})\} \Rightarrow \text{MRB} \tag{4.2}$$

Cette règle correspond aux patients infectés par des bactéries multi-résistantes. L'injection ou la perfusion peuvent favoriser le passage de bactéries de la peau à d'autres parties du corps, notamment internes. Etre grabataire constitue également un facteur de risque connu dans la littérature pour certaines bactéries multi-résistantes [104].

Un autre exemple combine attributs temporels et non-temporels dans une même règle, pour la prédiction *décès* :

$$age > 84 \text{ AND } seq = \{\text{insuffisance respiratoire} \rightarrow \text{embolie pulmonaire}\} \Rightarrow \text{décès} \tag{4.3}$$

Cette règle décrit les patients âgés souffrant d'un problème respiratoire chronique, et victimes d'une complication soudaine entraînant la mort. Les deux règles suivantes sont chacune composées d'un unique ensemble d'événements simultanés, c'est-à-dire d'événements se produisant au sein d'une même unité médicale. Cette notion de simultanéité est une partie importante du modèle proposé pour gérer les données temporelles, comme discuté précédemment. La règle suivante concerne la prédiction *escarre* :

$$seq = \{\text{troubles cognitifs légers}, \text{rétention d'urine}, \text{douleurs chroniques}\} \Rightarrow \text{escarre} \tag{4.4}$$

A l'hôpital, les escarres apparaissent souvent chez les patients qui restent inactifs et immobilisés dans la même position pendant un long moment (le plus souvent, allongés dans un lit). Cette règle décrit un ensemble de symptômes qui se produisent souvent ensemble chez les patients ayant un état de santé général dégradé. Il convient de remarquer que cette règle exprime davantage une corrélation entre ces symptômes et l'apparition d'escarres, mais pas une causalité.

La dernière règle concerne la prédiction *chute* :

$$seq = \{\text{ischémie traumatique d'un muscle}, \text{électrocardiogramme}\} \Rightarrow \text{chute} \tag{4.5}$$

Une ischémie est une réduction de l'apport en sang aux tissus (notamment musculaires), causant une carence localisée en oxygène et en glucose. L'ischémie peut avoir différentes causes, parmi lesquelles une soudaine compression causée par une chute. Ce diagnostic est souvent accompagné d'un électrocardiogramme, afin de vérifier l'état du système cardiovasculaire.

De telles règles sont combinées au sein de solutions (ensembles de règles) par MOSC. Toutes les règles ne font pas usage de la temporalité ; une séquence peut n'être composée que d'un seul événement. Cependant, les règles temporelles ont en général une confiance élevée et un support faible. Elles décrivent le plus souvent des situations précises, et constituent un ajout précieux aux ensembles de règles. Toutes les règles présentées ici ont une confiance supérieure à 0.7 et un support inférieur à 0.2.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

4.5 Conclusion

Dans ce chapitre, nous avons présenté le problème de la classification de séquences temporelles. En utilisant MOCA-IQ comme base, nous avons développé l'algorithme MOSC, capable de générer des règles de classification mêlant attributs temporels et non-temporels. C'est, à notre connaissance, la première méthode permettant de gérer une telle hétérogénéité dans les données. Cette méthode répond aux besoins formulés précédemment et s'adapte bien aux particularités des données médicales. Nous avons également introduit plusieurs restrictions du voisinage pour les attributs *séquence*, ainsi qu'une méthode de *matching* efficace et adaptée à la structure des séquences utilisée. Nous avons comparé MOSC avec MOCA-IQ ainsi que plusieurs algorithmes de classification, sur données simplifiées et complètes. Les résultats montrent que MOSC est capable de générer des règles de bonne qualité pour tous les cas d'étude considérés, et qu'il est plus efficace que les autres algorithmes utilisés pour comparaison. L'étude qualitative montre que les règles générées sur les données réelles ont un sens et sont vraies d'un point de vue médical.

Au cours de cette étude, nous avons identifié plusieurs pistes d'amélioration possibles. Une première concerne les mécanismes employés pour réduire l'espace de recherche pour le voisinage des termes de type *séquence*. Nous avons choisi une approche simple qui permet de n'écarter aucune solution potentiellement intéressante, mais une approche plus élaborée (par modèles génératifs, par exemple) doit permettre de réduire encore davantage l'espace de recherche sans diminution notable de la qualité. Une deuxième extension concerne les séquences avec plusieurs niveaux de temporalité. Nous avons conçu et implémenté un modèle permettant de gérer des séquences à deux niveaux, mais sans le valider, faute de données adéquates. Des séquences à plus de deux niveaux de temporalité sont également envisageables, mais nécessitent d'altérer le modèle proposé. Enfin, une autre amélioration est liée à l'exploitation des durées entre les événements des séquences pour préciser les règles de classification obtenues. Ces informations ne sont pour le moment utilisées que de manière descriptive, et ne permettent pas de préciser le temps entre le prédicat et la prédiction.

Nous avons donc conçu et développé un algorithme de classification, MOSC, permettant de gérer les données hétérogènes, incluant des données numériques et temporelles, afin de pouvoir répondre à des cas d'étude sur données hospitalières réelles. Cela vient conclure les travaux de cette thèse sur le problème de la classification. Dans la partie suivante, nous nous intéressons au biclustering, une méthode de fouille de données non-supervisée permettant notamment la recherche de groupe de patients présentant un profil similaire.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Deuxième partie

Biclustering

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Chapitre 5

Biclustering sur données hétérogènes

Dans ce chapitre, nous étudions le biclustering sur données hétérogènes, omniprésentes dans les données hospitalières. Dans la première section, nous définissons le problème de la recherche de profils de patients et justifions sa modélisation en un problème de biclustering. Nous formulons ensuite ce problème sous forme d'un problème d'optimisation combinatoire multi-objectif et proposons une méta-heuristique pour le biclustering sur données hétérogènes. Nous identifions les limitations de cette première approche, puis proposons une seconde approche afin de répondre au problème de manière plus efficace. Cette seconde approche utilise une heuristique constructive et une simplification des données en pré-traitement. Dans ce chapitre, les données temporelles ne sont pas considérées ; elles font l'objet d'une étude détaillée dans le chapitre suivant.

Ce chapitre est associé aux publications suivantes :

— M. Vandromme, J. Jacques, J. Taillard, L. Jourdan, C. Dhaenens : *A local search-based multi-objective metaheuristic for biclustering on heterogeneous data*, **Metaheuristics International Conference**, MIC 2017.

— M. Vandromme, J. Jacques, J. Taillard, L. Jourdan, C. Dhaenens : *A scalable biclustering method for heterogeneous medical data*, **Second International Workshop on Machine Learning, Optimization and Big Data**, MOD 2016.

5.1 Définition du problème

Dans cette section, nous détaillons tout d'abord le problème de la recherche de profils patient dans les données hospitalières. Nous présentons ensuite une introduction générale au biclustering. Nous dressons ensuite un état de l'art des méthodes de biclustering existantes. Enfin, nous identifions les problèmes posés par l'application du biclustering aux données hospitalières et justifions le besoin d'une méthode de biclustering adaptée.

5.1.1 Recherche de profils patient

Dans la partie précédente, nous nous sommes intéressés à des méthodes de classification supervisée. Celles-ci répondent, d'une certaine manière, à la problématique générale d'identifier différents profils de patients. En effet, les règles de classification générées par MOSC permettent d'établir plusieurs profils-type de patients pour une prédiction spécifiée en amont. Chaque règle correspond à un profil ; plusieurs règles sont incluses dans chaque solution (ensemble de règles) et plusieurs solutions sont incluses dans l'ensemble de solutions générées (archive). Par l'utilisation

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

d'une mesure de qualité appropriée, il est alors possible d'identifier les profils les plus vraisemblables pour un problème donné. Cette méthode de classification supervisée a deux utilisations possibles. Tout d'abord, la simple observation des règles générées permet parfois de découvrir des relations de cause à effet jusque là insoupçonnées, ou de confirmer des intuitions par des résultats chiffrés. De cette manière, il est possible d'agir sur les causes d'un problème, qui ont été identifiées comme telles par l'outil de génération de règles. Une deuxième utilisation consiste à utiliser ces règles, qui ont été apprises à partir d'un ensemble d'instances (patients ou séjours), pour déterminer si une nouvelle instance présente des caractéristiques qui l'associent à un profil spécifique.

Cette approche est utile pour répondre à de nombreux cas d'étude, mais présente l'inconvénient de requérir une prédiction, sur laquelle porteront l'ensemble des règles de classification générées. Par exemple, on peut rechercher un ensemble de règles décrivant les patients atteints de diabète. Cela permet d'obtenir un ensemble de caractéristiques associées à la présence de diabète, et de prédire quels nouveaux patients souffrent probablement de cette pathologie même si celle-ci n'est pas renseignée dans les données. Cependant, les règles ainsi générées ne permettent de classifier les données que selon un axe précis (présence de diabète), et ignorent l'ensemble des autres profils pouvant exister dans les données. Utiliser différentes prédictions et rechercher itérativement des règles de classification pour chacune de ces prédictions permet d'obtenir davantage de diversité dans les profils générés. Cependant, étant donné le nombre conséquent de prédictions possibles (au moins autant que d'actes et de pathologies, donc plusieurs dizaines de milliers) et le temps de génération de ces règles, générer les classifieurs pour chaque prédiction n'est pas réalisable en pratique. Une solution intermédiaire consiste à choisir un ensemble de prédictions correspondant à des cas intéressants, afin de garantir une certaine diversité dans les profils. Cependant, cela requiert une expertise métier préalable, et cela introduit un biais conséquent. En effet, cela ne permet toujours pas de découvrir des corrélations surprenantes entre des attributs exclus de ce choix préalable.

Pour répondre à cette question plus générale de la découverte de profils de patients, nous avons besoin d'une méthode d'extraction de connaissance non-supervisée. Ce type de méthode fonctionne sans but *a priori*, et cherche simplement à détecter des structures dans les données. Cet aspect non-supervisé est un avantage, mais pose d'autres problèmes. En effet, il est fréquent que ces méthodes produisent un certain nombre de résultats évidents, déjà connus ou sans intérêt pratique. Par conséquent, comme nous le verrons dans les chapitres suivants, la préparation des données ainsi que le filtrage des résultats générés sont des étapes primordiales. Dans le reste de ce chapitre, nous présentons le biclustering en général et justifions son application pour l'extraction d'information à partir des données médicales hospitalières.

5.1.2 Généralités sur le biclustering

Le biclustering est une méthode permettant le clustering (classification non-supervisée) simultané des lignes et des colonnes d'une matrice de données, introduite pour la première fois dans les années 1970 par Hartigan [78]. Cette méthode produit un ensemble de biclusters, où chaque bicluster est composé d'un ensemble de lignes et d'un ensemble de colonnes, ce qui décrit un ensemble de lignes présentant un comportement similaire sur un ensemble de colonnes, et vice versa. La détection d'un bicluster au sein d'une matrice de données peut être vue comme la réorganisation des lignes et des colonnes pour former un bloc de données cohérent. La Figure 5.1 présente un exemple de réorganisation, qui aboutit à l'isolation d'un bicluster de quatre lignes et trois colonnes contenant des valeurs identiques.

Le biclustering en lui-même est une méthode très générique, qui dépend fortement de deux choix de conception : la définition du *comportement similaire* (ou *homogénéité*) des données dans

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

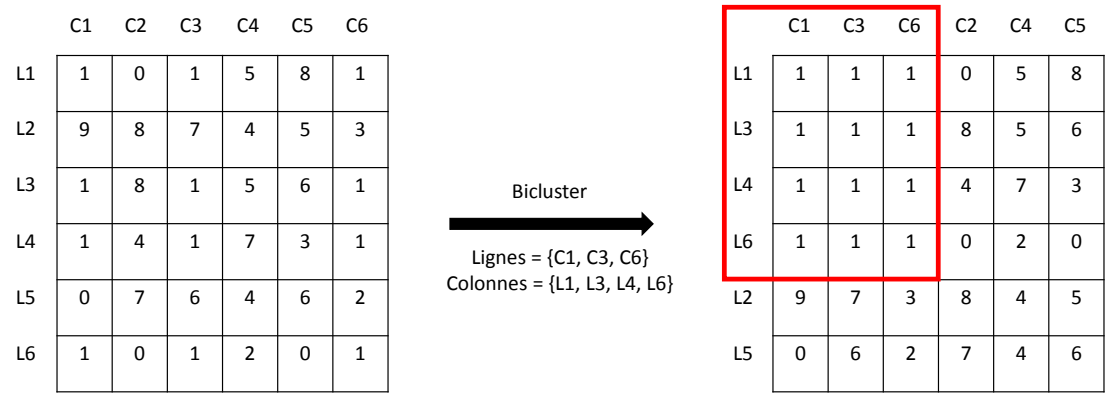


FIGURE 5.1 – Réorganisation des lignes et des colonnes pour identification d'un bicluster

un bicluster, et la disposition des biclusters sur la matrice de données.

5.1.2.1 Homogénéité d'un bicluster

Le premier choix de conception concerne la définition de l'homogénéité d'un bicluster. Un bicluster est décrit par un bloc de données homogène au sein de la matrice de données. Cette homogénéité peut néanmoins prendre différentes formes, en fonction du type des données étudiées ainsi que de l'application. Par conséquent, chaque méthode de biclustering se concentre sur un sous-ensemble de ces types, ceux qui ont le plus de sens dans le cadre de l'application. La Figure 5.2 présente les différents types de biclusters existants.

On distingue cinq types de biclusters différents :

(a) Valeurs constantes dans l'ensemble du bicluster : il s'agit du cas le plus simple et évident, qui est également illustré par la Figure 5.1. Celui-ci est un cas particulier de tous les autres types de biclusters. Par conséquent, tous les algorithmes de biclustering sont capables de détecter ce type de bicluster.

(b) Valeurs constantes sur les lignes : toutes les valeurs d'une même ligne sont identiques. Les valeurs d'une ligne à l'autre ne doivent respecter aucune relation particulière.

(c) Valeurs constantes sur les colonnes : semblable au cas précédent, pour les colonnes.

(d) Valeurs cohérentes - *shifting pattern* : d'une colonne j à une colonne $j+1$, chaque valeur est modifiée de sorte que $V_{j+1} = V_j + k_j$, où k_j est spécifique à chaque colonne et $\forall j, k_j \in \mathbb{R}$. Les valeurs d'une ligne à l'autre n'ont pas de lien entre elles, mais chaque ligne suit la même évolution. Sur l'exemple de la Figure 5.2(d), $\{k_1, k_2, k_3, k_4\} = \{-1, 3, -2, 1\}$.

(e) Valeurs cohérentes - *scaling pattern* : semblable au cas précédent, mais avec $V_{j+1} = V_j \times k_j$. L'évolution est multiplicative et non plus additive. Sur l'exemple de la Figure 5.2(e), $\{k_1, k_2, k_3, k_4\} = \{0.5, 4, 0.1, 4\}$.

5.1.2.2 Disposition des biclusters

Le second choix de conception concerne la disposition des biclusters sur la matrice de données originale. Ce choix comporte lui-même deux aspects : le recouvrement des biclusters entre eux, et

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

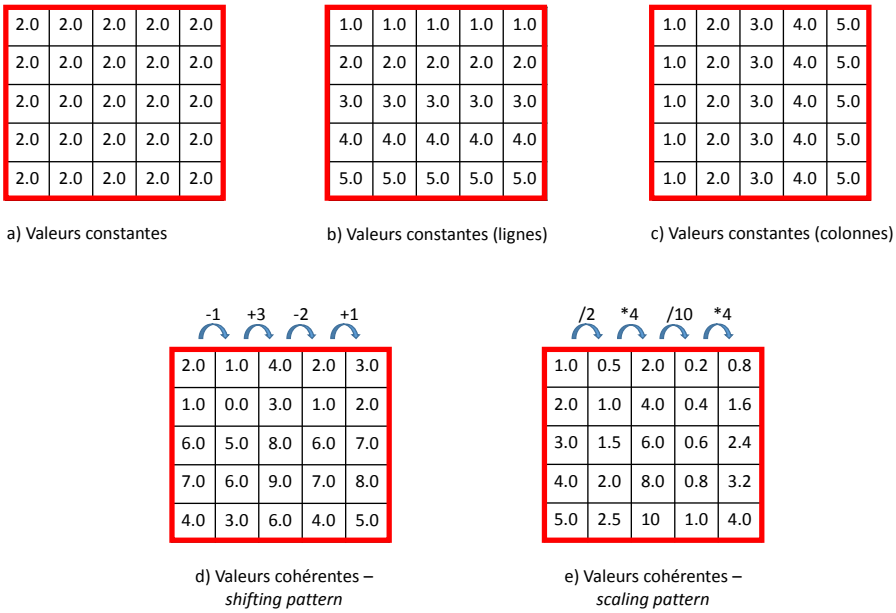


FIGURE 5.2 – Différents types de biclusters définis par leur type d'homogénéité dans les données

la couverture de la matrice de données. Pour l'aspect recouvrement, quatre options sont possibles :

- Aucun recouvrement autorisé : chaque élément de la matrice de données ne peut appartenir qu'à un seul bicluster.
- Recouvrement sur lignes uniquement : une ligne peut faire partie de plusieurs biclusters, mais une colonne ne peut faire partie que d'un seul.
- Recouvrement sur colonnes uniquement : une colonne peut faire partie de plusieurs biclusters, mais une ligne ne peut faire partie que d'un seul.
- Recouvrement complet : lignes et colonnes peuvent appartenir à plusieurs biclusters.

L'autre aspect de la disposition concerne la couverture de la matrice de données. Trois options existent :

- Couverture intégrale : chaque élément de la matrice doit appartenir à (au moins) un bicluster.
- Couverture partielle : des éléments peuvent n'appartenir à aucun bicluster.
- Couverture en damier : un cas particulier du précédent, où chaque ligne et chaque colonne doivent appartenir à au moins un bicluster. Cette structure est mentionnée car plusieurs algorithmes de biclustering génèrent des résultats de ce type.

La Figure 5.3 présente des exemples de différentes configurations pour ces deux aspects :

(a) Aucun recouvrement, couverture partielle.

(b) Aucun recouvrement, couverture en damier.

(c) Recouvrement sur lignes, couverture en damier.

(d) Recouvrement sur colonnes, couverture en damier.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

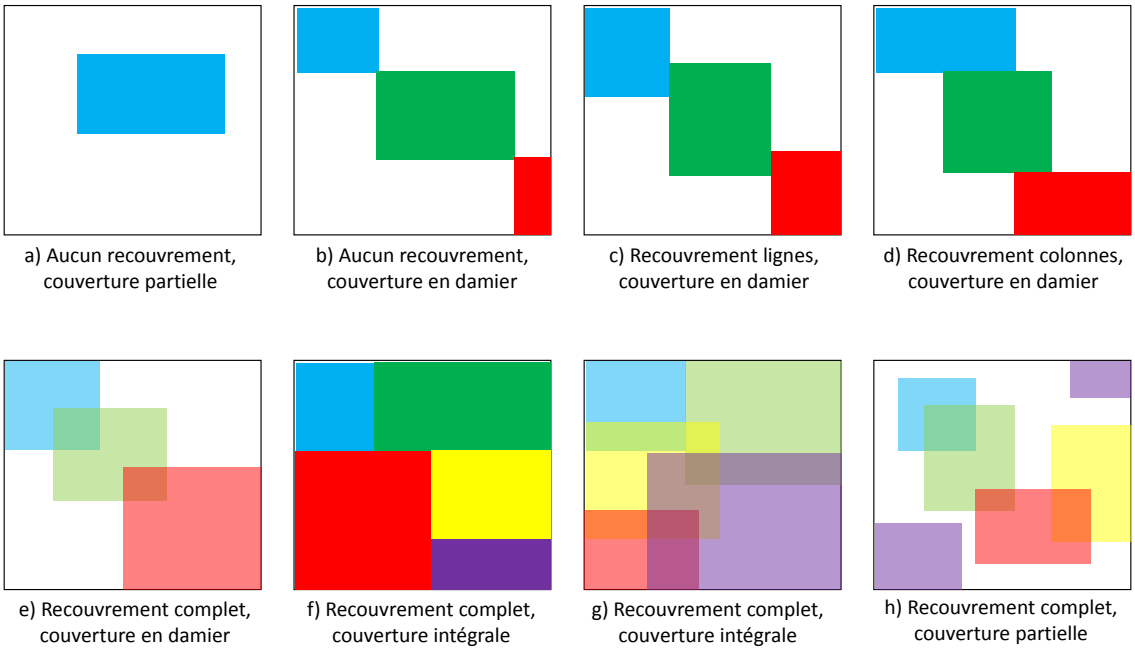


FIGURE 5.3 – Différents types de biclusters en fonction de leur recouvrement entre eux et de la couverture de la matrice

(e) Recouvrement complet (lignes et colonnes), couverture en damier.

(f) Recouvrement complet, couverture intégrale.

(g) Recouvrement complet, couverture intégrale. La différence entre f) et g) est que dans g), certains éléments de la matrice de données font partie de plusieurs biclusters. La configuration f) est un cas particulier, où il y a bien recouvrement complet même si aucun élément de la matrice ne fait partie de plusieurs biclusters.

(h) Recouvrement complet, couverture partielle. Sur cette figure, le bicluster représenté en violet est séparé en deux parties. Il arrive en effet qu'il soit impossible de réorganiser les lignes et colonnes pour n'avoir que des biclusters en une seule partie. Cela pose un problème majeur pour la représentation des résultats du biclustering dans les cas où peu de contraintes existent pour le placement des biclusters.

Les cas les plus difficiles pour les algorithmes de biclustering sont ceux pour lesquels il existe le moins de contraintes sur le placement des biclusters, donc recouvrement total entre biclusters et couverture partielle de la matrice de données (cas (h) sur la Figure 5.3). En effet, rechercher une structure en damier dans la matrice, par exemple, implique des contraintes fortes sur les résultats, qui peuvent être utilisées afin de simplifier le fonctionnement de l'algorithme de biclustering.

5.1.3 Etat de l'art

L'idée du clustering simultané des lignes et des colonnes d'une matrice de données a été formulée pour la première fois en 1972 [78], mais le premier algorithme de biclustering fonctionnel date de 2000 [31]. Ce travail fondateur s'intéresse à l'analyse des données génomiques. Dans ce problème, les lignes représentent les différents gènes, les colonnes les conditions (biologiques

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

ou environnementales), et les valeurs les niveaux d'expression des gènes en fonction des conditions. Appliqué à ce problème, le biclustering permet d'identifier des groupes de gènes ayant des profils d'expression similaires sous certaines conditions. Cette méthode permet de résoudre les limitations du clustering unidimensionnel, en considérant que la similarité des gènes dépend des conditions internes et externes. L'analyse des données d'expression génomique est donc le problème initial auquel le biclustering a été appliqué, et il demeure prédominant dans la littérature sur le sujet [133]. Une application assez proche, également dans le domaine biologique, concerne l'analyse de l'interaction entre protéines [53]. Ces applications sont toujours d'actualité, en particulier depuis l'avènement et les progrès de la technique des puces à ADN, qui permettent d'obtenir directement des matrices de données d'expression [147]. Le biclustering a également été appliqué à divers problèmes de fouille de données [26] : analyse de textes [45, 52], systèmes de recommandation [8, 175], analyse de transactions [85] ou de recherche d'information en ligne [113], recherche de patterns dans des données binaires [143].

Au-delà des applications, les algorithmes utilisés pour le biclustering sont également assez variés [82]. La Table 5.1 offre un aperçu des différentes classes de méthodes existantes et des travaux marquants sur ce sujet.

— Le travail pionnier de Cheng et Church [31] utilise une heuristique constructive, qui ajoute lignes et colonnes à un bicluster en fonction d'une mesure d'homogénéité interne. Cette étude est représentative de l'ensemble de méthodes fonctionnant par *minimisation de la variance* pour former les biclusters. D'autres méthodes de ce type sont les σ-clusters [28] et FLOC (FLexible Overlapped biClustering) [174].

— Une deuxième classe de méthodes connue consiste à considérer le biclustering comme un problème de clustering à deux dimensions (ou *Two-Way Clustering*), et de former itérativement des clusters de lignes et de colonnes [68, 155]. Une extension de cette idée utilise des cartes auto-adaptatives pour combiner les deux clusterings (sur lignes et colonnes) [25].

— Une troisième catégorie d'algorithmes considère le biclustering comme un problème de *recherche de patterns* dans les données. Certaines méthodes simplifient les données en amont pour faciliter la recherche. Par exemple, xMOTIF [126] et Bimax [135] binarisent les données et recherchent des blocs de valeurs cohérentes sur celles-ci. OPSM (Order-Preserving Sub-Matrices) recherche des biclusters ayant le même ordre de valeurs dans leurs colonnes [16]. L'algorithme de Clustering Spectral [99] effectue une décomposition en valeurs singulières des données. D'autres méthodes de cette classe sont ISA (*Iterative Signature Algorithm*) [83] et CCC (*Contiguous Column Coherent*) [120].

— Une quatrième catégorie regroupe les méthodes *probabilistiques et génératives*. Parmi celles-ci, SAMBA (*Statistical-Algorithmic Method for Bicluster Analysis*) utilise un graphe bipartite pour représenter les deux dimensions de la matrice de données, et recherche des sous-graphes fortement connectés [154]. Une autre approche utilise un échantillonage de Gibbs pour estimer les paramètres d'un modèle décrivant le *pattern* d'un bicluster [151]. PRM (*Probabilistic Relational Models*) [67] et ProBic [160] combinent modélisation probabiliste et logique relationnelle dans des modèles génératifs. Enfin, cMonkey représente les biclusters comme des processus de Markov [140]. Les modèles par *plaid* font également partie de la famille des méthodes génératives, et utilisent un modèle additif général pour expliquer les valeurs d'un bicluster [27, 110].

— Une dernière catégorie inclut les approches par métaheuristiques, qui ont également été appliquées avec succès au problème du biclustering. Différentes méthodes ont été proposées pour l'analyse de l'expression génomique : une première utilise une recherche adaptative aléatoire gloutonne [51], une autre emploie une recherche par faisceau (*Scatter Search*) basée sur les corrélations [129], une dernière un algorithme évolutionnaire [128]. D'autres

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

travaux proposent des métaheuristiques hybrides [46, 148], et une approche par systèmes immunitaires artificiels pour l'analyse textuelle [45].

Classe de méthodes	Méthode	Référence
Minimisation de la variance	Cheng et Church	[31]
	σ-clusters	[28]
	FLOC (FLexible Overlapped biClustering)	[174]
Two-Way Clustering	CTWC (Coupled Two-Way Clustering)	[68]
	ITWC (Interrelated Two-Way Clustering)	[155]
	DCC (Double Conjugated Clustering)	[25]
Recherche de patterns	xMOTIF	[126]
	Bimax	[135]
	OPSM (Order-Preserving SubMatrices)	[16]
	Clustering Spectral	[99]
	ISA (Iterative Signature Algorithm)	[83]
	CCC (Contiguous Column Coherent)	[120]
Probabilistiques et génératives	SAMBA	[154]
	Echantillonage de Gibbs	[151]
	PRM (Probabilistic Relational Models)	[67]
	ProBic (Iterative Signature Algorithm)	[160]
	cMonkey	[140]
	Plaid1	[110]
	Plaid2	[27]
Métaheuristiques	Recherche adaptative aléatoire gloutonne	[51]
	Scatter Search basée sur les corrélations	[129]
	Algorithme évolutionnaire	[128]
	Métaheuristique hybride	[46, 148]
	Systèmes immunitaires artificiels	[45]

TABLE 5.1 – Identification des différents types de méthodes de biclustering, avec quelques exemples pour chaque type

Le biclustering est donc un problème assez récent mais qui a reçu une attention soutenue, principalement à des fins d'analyse de données biologiques mais également pour des applications de plus en plus variées en fouille de données. Cependant, une limitation importante de ces différentes méthodes est qu'elles ne s'appliquent qu'à des matrices de données homogènes, c'est-à-dire de même type. Ces méthodes sont conçues pour des matrices de données numériques le plus souvent, quoique certaines binarisent les données [135] ou fonctionnent directement sur données binaires [143]. Cependant, aucune application ne considère de matrice de données où les colonnes contiennent différents types de données. Or, cette hétérogénéité est fondamentale dans les données hospitalières que nous étudions ici. Deux autres particularités de ces données sont la volumétrie et les valeurs manquantes. Ces thèmes ont longtemps été ignorés dans le contexte du biclustering, car les données génomiques comportent le plus souvent quelques centaines ou milliers de lignes et colonnes tout au plus, sans valeurs manquantes. Des études récentes commencent à s'intéresser à des cas où les matrices de données sont sensiblement plus larges, nécessitant une bonne scalabilité des méthodes [80, 94] ou une possible parallélisation [177]. Le caractère creux des matrices de données n'a pas été exploré avant une première étude en 2008 [159], qui traite de matrice binaires avec environ 20% de données manquantes. Les méthodes traditionnelles de biclustering utilisent en général des techniques d'inférence pour estimer les valeurs manquantes,

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

ce qui n'est pas applicable quand celles-ci sont majoritaires. Plus récemment, une étude intéressante s'est concentrée sur l'analyse de données larges et très creuses, dans des proportions assez similaires aux données hospitalières de notre problème : plusieurs dizaines de milliers de lignes, des milliers de colonnes et moins de 1% de données présentes [8]. Celle-ci cherche à extraire des biclusters à partir des données (numériques) de systèmes de recommandation, et donne des résultats intéressants sur ce type de données.

5.1.4 Biclustering pour données hospitalières

Le biclustering est donc une méthode de fouille de données non-supervisée, assez générique et modulable pour rechercher différentes corrélations dans les données en fonction des besoins. Il convient donc de déterminer comment adapter cette méthode aux particularités des données médicales sur lesquelles nous travaillons.

La première particularité, et sans doute la plus importante, est l'hétérogénéité dans les données. En effet, comme nous l'avons vu dans la Section 5.1.3, les méthodes de biclustering existantes sont conçues pour gérer des données homogènes. L'hétérogénéité pose des contraintes sur le type de biclusters que nous recherchons dans les données. Pour reprendre la définition du problème général, nous cherchons des groupes de patients ayant un comportement similaire sur un ensemble d'attributs. Cela correspond au cas (c) de la Figure 5.2 : les biclusters avec des valeurs constantes en colonnes. Ainsi, les patients ont bien des valeurs identiques (ou proches) pour chaque attribut. Les attributs sont incomparables ; chercher des corrélations en ligne (c'est-à-dire entre les différentes informations présentes sur une même ligne) n'a pas de sens dans ce contexte. En effet, cela reviendrait à vouloir définir un critère permettant de dire, par exemple, que l'âge, le groupe sanguin et le genre d'un individu sont identiques (cas (b) sur la Figure 5.2) ou qu'ils marquent une évolution cohérente (cas (d) et (e)). Cela étant impossible, nous sommes cantonnés à la découverte de biclusters constants en colonnes, qui répond parfaitement au problème qui nous intéresse.

Il convient ensuite de déterminer quel type de couverture de la matrice des données, et quel type de recouvrement entre biclusters nous souhaitons obtenir. Pour le premier aspect, une couverture partielle est davantage adaptée. En effet, les matrices de données traitées sont souvent larges et creuses, aussi parait-il difficile de les couvrir intégralement par un ensemble de biclusters. De plus, toutes les données ne sont pas toujours intéressantes à exploiter, et forcer la formation de biclusters peut amener des résultats inutiles ou contre-productifs. Nous ne cherchons pas à classifier chaque patient, mais davantage à découvrir des profils cohérents, même si ceux-ci ne couvrent au final qu'une petite partie des données. Pour le second aspect, nous choisissons d'autoriser un recouvrement total (sur lignes et colonnes) entre les différents biclusters. Cela signifie qu'un patient peut avoir plusieurs profils intéressants, et qu'un attribut peut être utilisé pour former plusieurs profils. Cette liberté dans la formation des biclusters paraît naturelle dans le cas des données médicales. Un patient peut souffrir de plusieurs pathologies qui justifient son inclusion dans plusieurs biclusters, et un attribut (par exemple, l'âge) peut appartenir à la description de plusieurs profils-type.

Les autres particularités concernent le volume important et l'aspect creux des matrices de données étudiées. Comme nous l'avons vu dans la section précédente, le biclustering n'a été jusqu'à présent utilisé que pour des matrices compactes (sans données manquantes) et de taille réduite, avec quelques milliers de lignes et colonnes tout au plus. Ces particularités n'impliquent cependant pas de réels choix de conception *a priori*, comme c'est le cas pour l'hétérogénéité, mais elles nécessitent de porter une attention particulière à la scalabilité de la méthode (complexité en temps et en espace) et à la gestion des valeurs manquantes.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Dans cette section, nous avons présenté la méthode du biclustering en général et explicité en quoi l'application de celle-ci aux données dont nous disposons serait intéressante. Nous avons également mis en exergue les limitations actuelles des méthodes de biclustering existantes, et les différents points qu'il nous faut résoudre. Dans le suite de ce chapitre, nous proposons deux approches pour le biclustering sur données hétérogènes, permettant de répondre aux problèmes susmentionnés.

5.2 Première approche : résolution par méta-heuristique

Dans cette section, nous proposons une première approche pour le biclustering, basée sur une méta-heuristique déjà employée pour MOCA-I et MOSC. Dans un premier temps, nous détaillons la modélisation choisie, avec notamment une mesure de qualité adaptée aux données hétérogènes. Nous présentons ensuite les différents éléments de l'algorithme proposé. Des résultats d'expérimentations visant à valider cette approche viennent conclure la section.

5.2.1 Modélisation

5.2.1.1 Modélisation sous forme d'un problème d'optimisation combinatoire multi-objectif

Nous choisissons de nous concentrer en première approche sur une résolution du problème de biclustering par méta-heuristique, car plusieurs travaux récents ont établi l'efficacité de ces méthodes pour ce problème (voir Table 5.1 dans la section précédente). Le biclustering peut être traduit assez simplement en un problème d'optimisation bi-objectif, avec les deux objectifs suivants :

— Qualité : la qualité d'un bicluster correspond au niveau d'homogénéité entre ses valeurs. Un bicluster avec des valeurs très proches entre elles est très homogène, et donc de bonne qualité. Dans le contexte du biclustering sur données hétérogènes (plusieurs types de données), nous recherchons des biclusters ayant des valeurs constantes en colonnes. Il n'existe pas de mesure standard de qualité pour les biclusters hétérogènes ; nous en proposons une dans la section suivante.

— Taille : définie comme le nombre de lignes multiplié par le nombre de colonnes du bicluster. Nous considérons qu'un grand bicluster est plus intéressant qu'un petit, cet objectif de taille est donc à maximiser. En effet, nous estimons qu'un bicluster plus grand est plus intéressant, car il couvre une plus grande partie de la matrice de données, et qu'il y a davantage de chances que les corrélations détectées soient autre chose que du bruit. Pour prendre un cas extrême, un bicluster contenant une ligne et une colonne (et donc une unique valeur) est parfaitement homogène par définition, mais ne présente néanmoins aucun intérêt.

Ces deux objectifs sont fréquemment utilisés dans les approches multi-objectif pour le biclustering. En effet, ils sont mutuellement antagonistes : un bicluster plus grand a généralement une qualité moins bonne, car la variance augmente naturellement avec le nombre de valeurs. Ces deux objectifs permettent donc de former un ensemble de solutions non-dominées intéressantes : certains biclusters petits mais très homogènes, d'autres plus grands mais de moins bonne qualité.

5.2.1.2 Mesure de qualité

Nous avons vu que la qualité d'un bicluster est l'un des deux objectifs que l'algorithme doit optimiser. Il convient donc de définir une mesure de qualité adéquate et appropriée aux biclusters sur données hétérogènes. En effet, dans ce contexte, nous ne pouvons pas utiliser telles quelles les

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

mesures classiques telles que la variance, le *mean-squared residue error*, l'indice de relevance ou les mesures basées sur les corrélations (voir [134] pour une revue récente des mesures de qualité pour le biclustering). Celles-ci tiennent en général compte des variations sur les lignes et colonnes, et ne sont donc pas applicables à notre problème. Etant donné que nous cherchons des biclusters avec valeurs constantes en colonnes, nous séparons le bicluster en ses colonnes constituantes, et calculons la qualité de chaque colonne indépendemment, avec l'une des deux mesures suivantes :

— Variance : la mesure statistique de variance classique, utilisée pour les attributs numériques et symboliques ordonnés. Plus précisément, le ratio entre la variance interne (valeurs dans le bicluster) et la variance globale (l'ensemble des valeurs pour cet attribut) est utilisé.

— Homogénéité : utilisée pour les attributs symboliques non-ordonnés et binaires. Cette mesure correspond au nombre d'occurrences de la valeur la plus fréquente dans le bicluster, divisé par le nombre de lignes dans le bicluster.

Une qualité partielle est calculée pour chaque colonne ; la qualité d'ensemble pour le bicluster est la moyenne de toutes celles-ci :

$$quality(bc) = \frac{1}{|A|} \times \left[\sum_{j \in A_{num}} \frac{\sigma^2(d_{I_{bc}j})}{\sigma^2(d_{Ij})} + \sum_{j \notin A_{num}} \left(1 - \frac{|\text{valeur la plus fréquente dans } d_{I_{bc}j}|}{|I_{bc}|}\right)\right] \tag{5.1}$$

où A est l'ensemble des attributs (ou colonnes) du bicluster bc, A_{num} l'ensemble des attributs numériques et symboliques ordonnés dans bc, σ^2 la variance d'un ensemble de valeurs, I_{bc} les lignes dans bc, et d_{ij} la valeur de la matrice de données à la ligne i et colonne j. Il est à noter que nous cherchons à minimiser cette mesure de *qualité* ; les biclusters avec une *qualité* proche de 0 sont ceux avec une variance interne minimale, donc ceux qui nous intéressent le plus. Compte tenu des spécificités des données hospitalières, il est possible que les biclusters contiennent des valeurs manquantes. Dans le calcul de qualité, les valeurs manquantes sont ignorées ; la qualité ne mesure donc que l'homogénéité des valeurs présentes dans le bicluster. Cette méthode de traitement des valeurs manquantes n'est sans doute pas optimale, mais nous reléguons à des travaux futurs la conception d'une mesure mieux adaptée, celle-ci ne s'avérant pas indispensable pour les travaux présentés dans cette thèse.

5.2.2 Méthode proposée

Ayant défini la modélisation, en particulier le type de biclusters recherchés et les critères à optimiser, nous pouvons à présent détailler la méthode de biclustering proposée.

5.2.2.1 Vue d'ensemble

Nous présentons dans la Figure 5.4 une vue d'ensemble de l'algorithme de biclustering proposé. Le fonctionnement général de celui-ci est similaire à DMLS [116], présenté au Chapitre 2 et déjà utilisé comme base à MOCA-I et MOSC. Les différents composants sont présentés plus en détail dans le reste de cette section, mais le fonctionnement général est le suivant. Un premier ensemble de biclusters est créé au cours de la phase d'initialisation, et constitue l'archive initiale des solutions. A chaque itération, une ou plusieurs solutions sont sélectionnées, et leur voisinage est généré. Parmi les nouvelles solutions ainsi créées, celles qui sont meilleures que les solutions actuelles sont conservées et ajoutées à l'archive. Quand les conditions d'arrêt sont remplies, l'algorithme renvoie comme résultat l'archive courante, qui contient un ensemble de solutions non-dominées.

© 2017 Tous droits réservés. lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

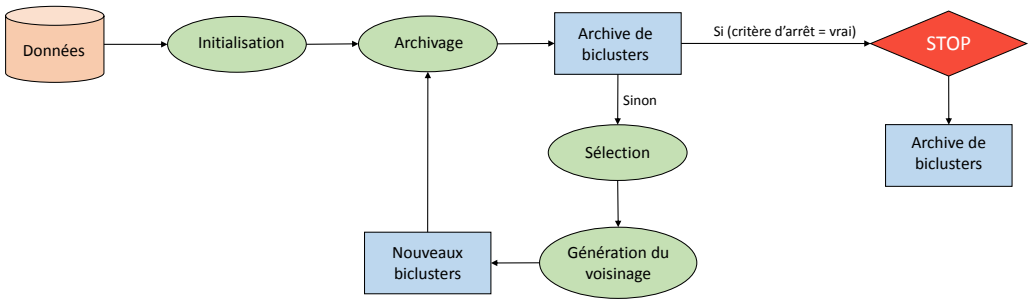


FIGURE 5.4 – Biclustering par méta-heuristique - vue d'ensemble

5.2.2.2 Initialisation

La recommandation la plus fréquente pour la génération des solutions initiales dans une méta-heuristique est de les construire de manière aléatoire, car cela permet en théorie la meilleure converture de l'espace de recherche. Ce principe, qui est également recommandé par DMLS, est appliqué dans l'algorithme proposé ici. Les biclusters initiaux contiennent donc un nombre aléatoire de lignes et de colonnes, elles-mêmes choisies aléatoirement. Cependant, nous devons tenir compte du fait que les matrices de données traitées sont parfois très creuses. En effet, si seules 1% (ou moins) des valeurs sont présentes, ce qui est le cas dans les jeux de données réels étudiés jusqu'à présent, une sélection purement aléatoire de lignes et de colonnes a de fortes chances de générer des biclusters ne contenant que des valeurs manquantes. De tels biclusters ne peuvent être considérés comme valides et ne constituent pas des points de départ valides pour l'algorithme d'optimisation. Par conséquent, nous modifions légèrement le processus afin de garantir qu'aucun des biclusters initiaux ne contient de valeur manquante. Le processus modifié est présenté dans l'Algorithme 8. Le principe général est de commencer à partir d'une ligne et de conserver toutes les colonnes sans valeurs manquantes. A chaque étape, une ligne est ajoutée et les colonnes ayant des valeurs manquantes supprimées. Cette étape est itérée jusqu'à ce que le nombre de lignes déterminé soit atteint et qu'il reste autant ou davantage de colonnes que le nombre de colonnes déterminé. Ce processus de formation est répété jusqu'à obtenir le nombre de biclusters initiaux souhaité.

5.2.2.3 Archivage

Comme mentionné précédemment, nous utilisons les deux objectifs de qualité et de taille. Ces deux mesures sont calculées à la création de chaque bicluster, et sont mises à jour dès que celui-ci est modifié. Nous voulons que l'archive contienne, à chaque instant, un ensemble de solutions non-dominées. Pour cela, nous utilisons le même processus d'archivage que celui de MOCA-I, afin de déterminer si une nouvelle solution doit être incluse dans l'archive. Tout d'abord, la solution est comparée avec toutes celles présentes dans l'archive ; si cette nouvelle solution est dominée par au moins une d'entre elles, le processus s'arrête et la nouvelle solution est éliminée. Sinon, s'il existe des solutions de l'archive dominées par cette nouvelle solution, celles-ci sont supprimées. S'il reste de la place dans l'archive, la nouvelle solution est ajoutée. La taille de l'archive est fixée à 100 après expérimentations préliminaires.

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Algorithme 8 : Biclustering par méta-heuristique - génération des solutions initiales

```
biclusters ← {};
while |biclusters| < nbBiclustersSouhaite do
    nbLignes ← random(min,max);
    nbColonnes ← random(min,max);
    choisir une ligne r au hasard;
    lignes ← r;
    colonnes ← colonnes avec valeurs sur r;
    while |lignes| < nbLignes AND |colonnes| > nbColonnes do
        choisir une ligne r' au hasard avec r' ∉ lignes;
        lignes ← lignes ∪ r';
        supprimer colonnes avec valeurs manquantes sur r';
    end
    if |lignes| == nbLignes AND |colonnes| == nbColonnes then
        bc ← nouveau bicluster(lignes,colonnes);
        ajouter bc aux biclusters;
    end
end
```

5.2.2.4 Sélection et critères d'arrêt

A chaque itération, certaines des solutions de l'archive sont sélectionnées et leur voisinage est généré. L'algorithme de DMLS laisse le choix entre deux mécanismes pour cette phase : sélectionner une solution au hasard, ou prendre l'ensemble des solutions de l'archive. Nous retenons la première option, qui est déjà utilisée dans MOCA-I et qui a été montrée plus performante en général [90]. Seules les solutions n'ayant pas été sélectionnées précédemment peuvent être choisies. Quand elle est sélectionnée, une solution est marquée comme *explorée* afin de stocker cette information.

Nous définissons également un ensemble de critères d'arrêt possibles :

— Convergence : l'algorithme s'arrête lorsque toutes les solutions de l'archive sont *explorées*.

— Temps : l'algorithme s'arrête après un temps t, à définir en paramètre.

— Itérations : l'algorithme s'arrête après k itérations, à définir en paramètre. Une itération correspond à un cycle *sélection - génération de voisinage - archivage*.

5.2.2.5 Opérateurs de voisinage

L'algorithme DMLS est, par définition, basé sur une recherche locale. Le voisinage d'une solution est l'ensemble des solutions créées par l'application exhaustive d'opérateurs de voisinage. Nous utilisons les quatre opérateurs suivantes, qui sont fréquemment employés dans les métaheuristiques pour le biclustering :

— ajoutLigne : une ligne extérieure au bicluster est ajoutée à celui-ci.

— ajoutColonne : une colonne extérieure au bicluster est ajoutée à celui-ci.

— supprimeLigne : une ligne est supprimée du bicluster.

— supprimeColonne : une colonne est supprimée du bicluster.

Par *application exhaustive*, nous voulons dire que chaque opérateur est appliqué à toutes les lignes et colonnes possibles. Par exemple, l'opérateur *ajouteLigne* permet de générer autant de voisins

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

qu'il y a de lignes en-dehors du bicluster. Une conséquence intéressante est que le voisinage de chaque solution contient toujours $|L|+|C|$ éléments, c'est-à-dire le nombre de lignes et de colonnes de la matrice de données. En effet, un bicluster avec davantage de lignes (respectivement colonnes) a moins de voisins générés par *ajouteLigne* (resp. *ajouteColonne*), mais davantage générés par *supprimeLigne* (resp. *supprimeColonne*). Le nombre total de voisins demeure constant à tout moment du processus d'optimisation, quelle que soit la solution considérée.

5.2.3 Expérimentations

5.2.3.1 Protocole expérimental

Nous comparons tout d'abord la méthode proposée, appelée LSBC (*Local Search for heterogeneous BiClustering*), avec plusieurs autres algorithmes de biclustering, afin d'établir sa validité et son efficacité. Le premier est l'algorithme de Cheng et Church (appelé CC par la suite) [31]. Celui-ci a été l'une des premières méthodes développées, et demeure un algorithme de référence dans les outils dédiés au biclustering. Pour ces expérimentations, nous utilisons l'implémentation disponible dans le package R 'biclust'. Plusieurs autres méthodes de biclustering sont disponibles dans cet outil, mais aucune autre que CC ne donne de résultats valables : ces autres méthodes ne trouvent soit aucun bicluster, soit un unique bicluster contenant la matrice de données complète. En observant les données utilisées par ces méthodes (par exemple dans l'outil FABIA [82]), nous suspectons que celles-ci supposent que les données sont normalisées (*log-normalisation*), ce qui n'est pas le cas dans les données qui nous intéressent. Deux autres algorithmes sont utilisés pour comparaison, issus du package Scikit-learn [131] : SpectralCoclustering [52] et SpectralBiclustering [147].

L'efficacité d'une méthode de biclustering peut être évaluée de plusieurs manières différentes. Un premier moyen est de comparer directement les résultats produits par un algorithme, en utilisant des mesures telles que la qualité et la taille. Un second moyen consiste à injecter dans les données un (ou plusieurs) bicluster(s) fortement homogène(s), et observer dans quelle mesure l'algorithme est capable de retrouver ces biclusters. Ces deux types d'évaluation se complètent bien, étant donné qu'ils se concentrent sur des propriétés différentes des méthodes : capacité à générer de grands biclusters de bonne qualité, et capacité à (re)trouver des biclusters présents à différents endroits dans la matrice de données.

Nous utilisons des jeux de données synthétiques comprenant 500 lignes et 100 colonnes, ces dimensions étant communes dans la littérature sur le sujet. Etant donné qu'aucun autre algorithme de biclustering ne supporte les données hétérogènes, nous utilisons des données purement numériques pour ces expérimentations. Pour les mêmes raisons, les jeux de données ne comportent aucune valeur manquante. Les valeurs pour chaque cellule de la matrice de données sont générées aléatoirement selon une distribution uniforme sur l'intervalle $[0, 100]$. Les biclusters injectés dans les données ont une taille fixe et sont composés de valeurs identiques dans chaque cellule. Le cas basique consiste à ajouter un unique bicluster dans les données, en remplacant les valeurs originales par celles du bicluster aux positions (lignes et colonnes) adéquates. Cependant, nous souhaitons aussi évaluer la capacité des méthodes de biclustering à traiter des situations plus complexes que ce cas de base. Par conséquent, pour chaque jeu de données, plusieurs variantes sont générées, en utilisant différents paramètres pour la création et l'insertion des biclusters :

— taille des biclusters : les petits biclusters sont en général plus difficiles à détecter que les grands, en partie parce que la taille est parfois un objectif direct des méthodes employées. Par conséquent, il arrive que des biclusters de petite taille mais très homogènes soient remplacés par des biclusters plus larges mais moins homogènes. Utiliser plusieurs tailles pour les biclusters injectés permet d'évaluer la sensibilité des méthodes à ce type de comporte-

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

ment. Les valeurs choisies pour ce paramètres sont {20x5, 50x10, 100x15}, notées "nombre de lignes X nombre de colonnes".

— niveau de bruit : le cas idéal est celui où toutes les valeurs du bicluster sont parfaitement identiques. Dans ce cas, les mesures de qualité internes telles que la variance et l'homogénéité sont optimales et l'algorithme converge plus facilement vers ces biclusters. Cependant, les données, surtout réelles, ne sont jamais dans cette situation idéale. Par conséquent, nous souhaitons évaluer la résilience des algorithmes de biclustering aux imperfections dans les données, et leur capacité à trouver des biclusters imparfaits, où les valeurs varient légèrement. Afin d'y parvenir, les valeurs des biclusters sont bruitées par l'ajout d'une valeur aléatoire générée suivant une distribution normale de moyenne zéro. L'écart-type de cette distribution détermine le niveau de bruit. Les valeurs choisies pour ce paramètre sont $\{0, 2, 5, 10\}$.

Il est à noter que d'autres paramètres peuvent être modifiés. Le premier concerne le nombre de biclusters injectés : nous avons réalisé des expérimentations avec 1, 3 et 5 biclusters respectivement. Le second concerne le niveau de recouvrement entre les biclusters : nous avons réalisé des expérimentations avec 0% et 20% de recouvrement (c'est-à-dire que 20% des lignes et colonnes de chaque bicluster appartiennent aussi à un autre bicluster). La Table 5.2 présente un résumé des différents paramètres et des valeurs choisies pour ceux-ci lors des expérimentations. Nous présentons dans la suite les résultats correspondant au cas avec 5 biclusters et 20% de recouvrement, ce qui correspond au cas le plus difficile et la plus intéressant pour les méthodes de biclustering.

Paramètre	**Valeurs**	**Remarque**
Taille des biclusters	20x5, 50x10, 100x15	
Niveau de bruit	0, 2, 5, 10	
Nombre de biclusters	1, 3, 5	Résultats présentés pour 5 uniquement
Niveau de recouvrement	0%, 20%	Résultats présentés pour 20% uniquement

TABLE 5.2 – Paramètres utilisés pour l'insertion des biclusters dans les données synthétiques : paramètre, valeurs et remarque éventuelle

Nous générons 20 jeux de données synthétiques, dans lesquels les biclusters sont injectés. Pour chaque jeu de données, 12 variantes sont générées, une pour chaque configuration de paramètres possibles (*taille des biclusters* et *niveau de bruit*). Cela correspond à 240 jeux de données au total. L'algorithme CC est déterministe, donc une seule exécution sur chaque jeu de données est suffisante. Cependant, LSBC est stochastique, aussi nous faisons 10 exécutions sur chaque jeu de données et prenons la valeur moyenne sur celles-ci. Toutes les expérimentations sont faites sur la même machine, sous Ubuntu 14.0.1, avec quatre cœurs à 3.3GHz et 4 Go de RAM. L'implémentation de LSBC est faite en C++, gcc version 4.8.2.

5.2.3.2 Résultats

Dans le reste de cette section, nous présentons donc les résultats pour les deux aspects considérés : qualité des biclusters générés, et capacité à retrouver les biclusters artificiellement injectés.

Comparaison des biclusters La première partie des expérimentations implique donc de comparer directement la qualité des résultats obtenus par LSBC, CC, SpectralCoclustering et SpectralBiclustering. Il est à noter que CC produit un nombre fixe de biclusters (par défaut 20, dans l'implémentation utilisée), tandis que LSBC en produit un nombre variable. Compte tenu de cela et pour assurer une comparaison juste, nous ne considérons que les 20 biclusters de meilleure

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

qualité pour LSBC. La justification derrière ce choix est que CC génère elle aussi 20 biclusters de la meilleure qualité possible. Nous remarquons cependant que les conclusions des expérimentations ne changent pas si nous considérons l'intégralité des biclusters générés par LSBC, et non seulement les 20 meilleurs. SpectralCoclustering et SpectralBiclustering génèrent également un nombre fixe de biclusters. Cependant, la valeur par défaut pour ces deux algorithmes est de 3 biclusters. Ce nombre ne permet pas à ces algorithmes de gérer correctement les cas où 5 biclusters sont insérés dans les données. Nous choisissons de fixer ce paramètre à 20 biclusters, comme pour CC. Deux mesures sont utilisées : qualité (à minimiser), et taille (à maximiser). La Table 5.3 présente la moyenne et l'écart-type des résultats des algorithmes considérés, sur les 20 jeux de données (et sur les 10 exécutions pour LSBC), pour chacune des 12 configurations de paramètres (*taille des biclusters* et *niveau de bruit*). Dans les tables, la notation $nXsY$ correspond à un jeu de données avec le niveau de bruit X et la taille Y (où *s0* est 20x5, *s1* est 50x10, et *s2* est 100x15).

	Qualité (à minimiser)				Taille (à maximiser)			
	LSBC	CC	SpectralCo	SpectralBi	LSBC	CC	SpectralCo	SpectralBi
n0s0	**0.04 ± 0.01**	5.01 ± 3.05	5.42 ± 0.9	5.48 ± 1.32	**3228 ± 790**	6 ± 2	132 ± 77	146 ± 105
n0s1	**0.04 ± 0.01**	3.54 ± 3.71	5.59 ± 1.0	5.25 ± 0.97	**3070 ± 741**	38 ± 65	144 ± 98	150 ± 101
n0s2	**0.04 ± 0.01**	3.37 ± 4.15	5.54 ± 1.37	4.94 ± 1.16	**3091 ± 800**	268 ± 380	162 ± 132	148 ± 124
n2s0	**0.05 ± 0.02**	5.27 ± 3.54	5.44 ± 0.88	5.79 ± 1.15	**3304 ± 1082**	6 ± 1	131 ± 77	127 ± 87
n2s1	**0.04 ± 0.02**	2.98 ± 3.56	5.52 ± 0.98	5.54 ± 1.02	**3084 ± 1006**	26 ± 30	143 ± 98	135 ± 88
n2s2	**0.03 ± 0.01**	0.49 ± 1.11	5.48 ± 1.22	5.90 ± 1.94	**2914 ± 934**	117 ± 69	157 ± 117	116 ± 89
n5s0	**0.05 ± 0.02**	5.63 ± 3.47	5.46 ± 0.95	5.53 ± 1.34	**3380 ± 1015**	6 ± 1	131 ± 76	136 ± 82
n5s1	**0.04 ± 0.01**	2.53 ± 3.02	5.52 ± 0.83	5.68 ± 1.11	**3263 ± 833**	11 ± 5	142 ± 94	127 ± 87
n5s2	**0.04 ± 0.01**	0.78 ± 0.26	5.56 ± 1.45	5.74 ± 1.62	**3124 ± 806**	21 ± 5	155 ± 132	121 ± 95
n10s0	**0.04 ± 0.01**	5.12 ± 3.19	5.39 ± 0.8	5.0 ± 1.0	**3277 ± 849**	6 ± 1	130 ± 74	168 ± 107
n10s1	**0.04 ± 0.01**	3.79 ± 3.05	5.56 ± 1.02	5.96 ± 1.4	**3323 ± 887**	7 ± 2	139 ± 84	120 ± 88
n10s2	**0.04 ± 0.02**	1.73 ± 0.86	5.59 ± 1.22	5.09 ± 0.98	**3533 ± 993**	10 ± 2	153 ± 115	158 ± 109

TABLE 5.3 – Comparaison de LSBC et CC - qualité et taille des biclusters générés. $nXsY$ correspond à un jeu de données avec le niveau de bruit X et la taille Y. La meilleure valeur est indiquée en gras

Ces résultats montrent assez clairement que LSBC génère des biclusters qui sont à la fois plus grands et de meilleure qualité que les autres algorithmes. Une observation intéressante est que les résultats pour CC s'améliorent lorsque les biclusters injectés sont plus grands (les cas $nXs2$, avec 100 lignes et 15 colonnes dans chaque bicluster). Cela correspond au comportement classique des algorithmes de biclustering, qui trouvent plus facilement les biclusters plus grands. D'un autre côté, les résultats pour LSBC, SpectralCoclustering et SpectralBiclustering ont tendance à rester similaires quel que soit le type de biclusters injectés dans les données.

Comparaison de la capacité à redécouvrir les biclusters Dans la seconde partie des expérimentations, nous utilisons deux mesures [21] pour évaluer l'efficacité de chacun des deux algorithmes à retrouver les biclusters injectés dans les données :

— U : la proportion du bicluster injecté présent dans le bicluster découvert par l'algorithme (à maximiser).

— E : la proportion du bicluster découvert qui n'appartient pas au bicluster injecté (à minimiser).

Plus formellement, nous définissons U et E comme indiqué dans l'équation ci-dessous :

$$U = \frac{|bc_{disc} \cap bc_{inj}|}{|bc_{inj}|} \qquad E = \frac{|bc_{disc}| - |bc_{disc} \cap bc_{inj}|}{|bc_{disc}|} \tag{5.2}$$

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

où bc_{disc} sont les biclusters découverts par l'algorithme, et bc_{inj} sont les biclusters injectés dans les données.

Pour chaque bicluster caché, nous ne considérons, parmi les biclusters découverts par l'algorithme, que celui ayant la valeur pour U la plus élevée pour ce bicluster. Quand plusieurs biclusters cachés sont injectés, le bicluster le plus proche de chaque bicluster caché est retenu, et les moyennes pour U et E sont calculées. Ainsi, si par exemple l'algorithme retrouve parfaitement quatre biclusters sur les cinq injectés, la valeur U moyenne vaut $\frac{4}{5} = 0.8$. La Table 5.4 présente les moyennes et écart-types sur les 20 jeux de données pour chaque configuration de paramètres, comme précédemment, pour 5 bicluster injectés avec 20% de recouvrement.

	U				E			
	LSBC	CC	SpectralCo	SpectralBi	LSBC	CC	SpectralCo	SpectralBi
n0s0	**0.11 ± 0.09**	0.01 ± 0.04	0.02 ± 0.01	0.02 ± 0.01	0.89 ± 0.04	**0.4 ± 0.2**	0.75 ± 0.04	0.48 ± 0.25
n0s1	0.17 ± 0.13	**0.22 ± 0.19**	0.03 ± 0.03	0.01 ± 0.01	0.68 ± 0.16	**0.14 ± 0.24**	0.51 ± 0.10	0.61 ± 0.12
n0s2	0.1 ± 0.12	**0.57 ± 0.16**	0.01 ± 0.01	0.01 ± 0.01	0.61 ± 0.27	**0.01 ± 0.09**	0.44 ± 0.12	0.47 ± 0.11
n2s0	**0.12 ± 0.08**	0.01 ± 0.03	0.01 ± 0.01	0.02 ± 0.01	0.88 ± 0.05	**0.39 ± 0.2**	0.74 ± 0.04	0.61 ± 0.11
n2s1	**0.15 ± 0.13**	0.11 ± 0.08	0.03 ± 0.03	0.02 ± 0.02	0.69 ± 0.16	**0.12 ± 0.23**	0.52 ± 0.15	0.51 ± 0.09
n2s2	0.11 ± 0.11	**0.13 ± 0.05**	0.02 ± 0.01	0.01 ± 0.01	0.58 ± 0.26	0.03 ± 0.13	0.39 ± 0.08	**0.22 ± 0.2**
n5s0	**0.11 ± 0.07**	0.01 ± 0.01	0.02 ± 0.01	0.0 ± 0.0	0.88 ± 0.05	**0.42 ± 0.16**	0.63 ± 0.17	0.58 ± 0.12
n5s1	**0.16 ± 0.13**	0.02 ± 0.02	0.03 ± 0.02	0.03 ± 0.01	0.71 ± 0.14	**0.13 ± 0.21**	0.58 ± 0.16	0.52 ± 0.16
n5s2	**0.11 ± 0.12**	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	0.59 ± 0.26	**0.17 ± 0.28**	0.4 ± 0.02	0.32 ± 0.18
n10s0	**0.1 ± 0.09**	0.01 ± 0.01	0.03 ± 0.01	0.02 ± 0.0	0.89 ± 0.04	**0.43 ± 0.19**	0.66 ± 0.06	0.72 ± 0.05
n10s1	**0.16 ± 0.11**	0.01 ± 0.01	0.03 ± 0.02	0.01 ± 0.01	0.71 ± 0.14	**0.19 ± 0.23**	0.52 ± 0.17	0.52 ± 0.08
n10s2	**0.12 ± 0.13**	0.01 ± 0.01	0.01 ± 0.01	0.02 ± 0.02	0.62 ± 0.25	**0.21 ± 0.33**	0.35 ± 0.16	0.3 ± 0.23

TABLE 5.4 – Comparaison de LSBC et CC - recouvrement des biclusters injectés. $nXsY$ correspond à un jeu de données avec le niveau de bruit X et la taille Y

Les résultats ici sont plus nuancés et complexes que pour la comparaison directe des qualités et tailles. SpectralCoclustering et SpectralBiclustering produisent des résultats de faible qualité pour tous les jeux de données, ne redécouvrant jamais de part significative des biclusters injectés. CC produit de bons résultats dans les cas les plus simples, avec de grands biclusters et aucun ou peu de bruit ($n0s1$ et $n0s2$, $n0s1$ et $n0s2$ dans une moindre mesure). CC échoue néanmoins à retrouver ne serait-ce qu'une partie des biclusters dans les autres cas. D'un autre côté, LSBC génère dans tous les cas des biclusters contenant une partie, petite mais remarquable, des biclusters injectés (U entre 10% et 20%). Cependant, elle ne parvient pas à trouver plus que cela, même dans les cas les plus simples. Afin de comprendre ces résultats, il convient de regarder plus en détail les biclusters générés par LSBC. Tous les biclusters présents dans une archive à la fin de LSBC sont des variantes l'un de l'autre, avec quelques lignes ou colonnes ajoutées ou supprimées. Cet ensemble de solutions forme effectivement un front Pareto, avec des biclusters de grande taille et faible qualité, et vice versa. Ce comportement est aisé à comprendre quand on considère le fonctionnement de LSBC. Une solution de l'archive est explorée ; ses voisins forment un ensemble de solutions non-dominées qui inondent l'archive et étouffent rapidement toute diversité. Pour des jeux de données dans lesquels des biclusters artificiels sont injectés, les résultats convergent vers un de ces biclusters. Ce comportement est fréquent parmi les méthodes à population, et explique les valeurs obtenues pour U, entre 0.1 et 0.2. Comme mentionné précédemment, avec cinq biclusters injectés, le score maximum possible pour U est de 0.2 si la méthode ne retrouve qu'un des cinq biclusters.En d'autres termes et pour résumer, LSBC est performante pour trouver un bicluster (et ses variantes), mais peu performante pour trouver davantage que cela. Compte tenu de cette limitation importante, nous proposons dans la section suivante une seconde approche.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

5.2.3.3 Discussion

L'approche par méta-heuristique proposée dans cette section présente des avantages intéressants, notamment de par sa capacité à générer des biclusters optimisant à la fois la qualité (homogénéité) et la taille. Cependant, le problème principal de cette approche est sa tendance à ne conserver dans l'archive qu'un ensemble de variantes (très proches entre elles) dérivées d'un unique bicluster. Ce comportement est naturel compte tenu de l'algorithme d'optimisation utilisé ; et les solutions conservées dans l'archive sont effectivement les meilleurs biclusters qu'il est possible de trouver sur les données. Cela pose néanmoins problème dans le cadre des applications pour lesquelles nous développons cette méthode de biclustering. En effet, l'objectif général est de découvrir un ensemble de profils patient dans les données. Aussi une méthode ne renvoyant qu'un unique profil (bicluster) ainsi qu'une pléthore de variantes n'est pas adaptée, en l'état, à ce que nous souhaitons faire. Afin de corriger ce problème, nous avons tenté plusieurs modifications.

Une première idée assez naturelle consiste à forcer une certaine diversité dans l'archive. Nous avons tout d'abord tenté de relaxer la contrainte de dominance Pareto pour l'insertion de solutions dans l'archive, en utilisant une ϵ-dominance à la place et en augmentant la taille de l'archive. Nous avons ensuite modifié les conditions d'insertion, en obligeant les nouvelles solutions à être différentes, c'est-à-dire à avoir un niveau minimum de non-recouvrement avec les biclusters déjà présents. Ces modifications n'amènent pas d'amélioration probante des résultats du point de vue de la diversité, et la détermination des paramètres est très délicate et dépendante des données.

Une seconde idée consiste à utiliser une liste d'attributs à ignorer au cours du processus de biclustering. Ces attributs ne pourront pas être utilisés pour former des biclusters. En se basant sur ce mécanisme, nous avons élaboré une version itérative de la méthode proposée précédemment. Lorsque le biclustering est terminé, l'archive des solutions est scannée, et les attributs revenant le plus souvent dans les solutions sont ajoutés à la liste des attributs à ignorer. Le biclustering est ensuite relancé avec cette liste en paramètre. De cette manière, l'algorithme explore itérativement des régions différentes de l'espace de recherche. Cette méthode apporte une certaine diversité dans les solutions de l'archive finale (après plusieurs exécutions et mises à jour de la liste). Cependant, en procédant ainsi, nous limitons la capacité de l'algorithme à générer des biclusters utilisant les mêmes attributs, ce qui était un prérequis fort de la méthode.

Pour résumer, les différentes modifications que nous avons essayé d'apporter n'ont pas permis de surmonter ce problème de diversité des solutions. Ces modifications peuvent soit affecter négativement la qualité des solutions générées, soit nécessiter un paramétrage complexe et fortement dépendant des données. Une dernière modification est envisageable pour résoudre ce problème de diversité, qui consisterait à considérer un ensemble de biclusters, et non plus un unique bicluster, comme une solution de l'algorithme d'optimisation. Un tel changement est similaire à celui qui a été opéré pour MOCA-I, où l'utilisation d'ensembles de règles comme solutions a permis d'amener une meilleure diversité dans les solutions [89]. Cela nécessiterait d'apporter des changements en profondeur à LSBC (mesure de diversité, sélection, voisinage, etc.), aussi nous choisissons de laisser cette modification comme perspective possible.

Lors du développement de cette première approche, nous avons observé que la méthode employée pour l'initialisation des solutions a un effet très important sur la qualité des solutions finales. Une méthode basique d'initialisation consiste à choisir au hasard un ensemble de lignes et de colonnes pour chaque bicluster. Une méthode un peu moins basique consiste à limiter le choix de telle sorte qu'aucun bicluster ne compte de valeur manquante. Une méthode plus élaborée consiste à sélectionner une colonne puis ajouter d'autres colonnes ayant des valeurs "proches". Chaque évolution dans la méthode d'initialisation amène une amélioration significative des résultats finaux. Pour ces raisons, nous proposons de nous concentrer, dans une seconde approche, sur la définition d'une heuristique constructive pour le biclustering, inspirée des améliorations

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

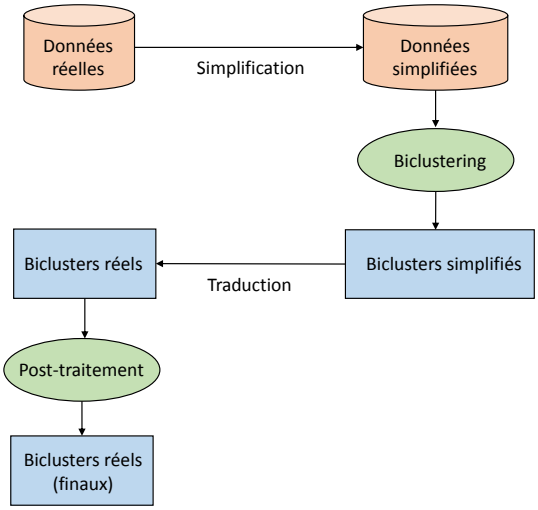


FIGURE 5.5 – Vue d'ensemble de l'algorithme de biclustering par heuristique proposé, HBC

apportées à la méthode d'initialisation des biclusters.

5.3 Seconde approche : résolution par heuristique constructive

Dans cette section, nous présentons donc la seconde méthode proposée. Dans un premier temps, nous détaillons les différents éléments de l'algorithme : heuristique constructive, transformation des données en pré- et post-traitement, amélioration des biclusters construits et analyse de la complexité. Nous présentons et discutons ensuite des résultats d'expérimentations permettant de valider l'efficacité de cette méthode.

5.3.1 Méthode proposée

5.3.1.1 Vue d'ensemble

Nous présentons tout d'abord une vue d'ensemble de l'algorithme de biclustering proposé, nommé HBC (pour *Heterogeneous BiClustering*), dans la Figure 5.5. Les différents composants de celle-ci sont décrits plus en détails dans le reste de cette section, mais nous donnons ici un bref résumé de leurs objectifs respectifs :

— Simplification : les données subissent une phase de pré-traitement au cours de laquelle elles sont simplifiées afin de faciliter le processus de biclustering

— Biclustering : l'étape principale, dans laquelle une heuristique constructive est utilisée pour former un ensemble de biclusters.

— Traduction : lors de la phase de biclustering, les biclusters sont formés sur les données simplifiées. Ici, ces biclusters sont transposés sur les données originales.

— Post-traitement : après traduction sur les données réelles, les biclusters sont évalués (qualité). Divers mécanismes permettent de filtrer, d'ordonner et de compléter les biclusters.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

5.3.1.2 Heuristique constructive

Nous proposons une méthode de biclustering par heuristique, qui profite de la nature particulière des biclusters que nous souhaitons découvrir (valeurs homogènes en colonnes), ainsi que de l'aspect souvent creux des matrices de données utilisées. L'idée centrale est de construire un bicluster en ajoutant, à chaque étape, la colonne ayant le plus de valeurs identiques pour les lignes incluses dans le bicluster, puis en supprimant les lignes dans lesquelles les valeurs sont différentes. Construire un bicluster implique donc de choisir une colonne (attribut) de départ, et d'y ajouter progressivement d'autres colonnes en supprimant des lignes. Afin de couvrir l'ensemble des données, ce processus est itéré pour chaque valeur et chaque attribut possible comme point de départ. Cela assure que même des biclusters petits mais homogènes soient découverts. Cela signifie aussi qu'un même bicluster peut être trouvé plusieurs fois ; ce problème est aisément traité en phase de post-traitement. Le fonctionnement exact de l'heuristique est décrit dans l'Algorithme 9. La Figure 5.6 présente un exemple de formation d'un bicluster par cette heuristique. Le bicluster est formé par la première colonne (attribut binaire *bin1*) et des lignes comportant la valeur 1. L'algorithme ajoute ensuite la colonne *num1* et supprime la dernière ligne pour conserver l'homogénéité, puis la colonne *symb1* sans avoir besoin de supprimer de ligne. A partir de là, il n'est plus possible d'ajouter de colonne sans réduire la taille totale du bicluster, donc le processus s'arrête et le bicluster ainsi formé est enregistré.

Algorithme 9 : Fonctionnement de HBC - heuristique constructive pour le biclustering

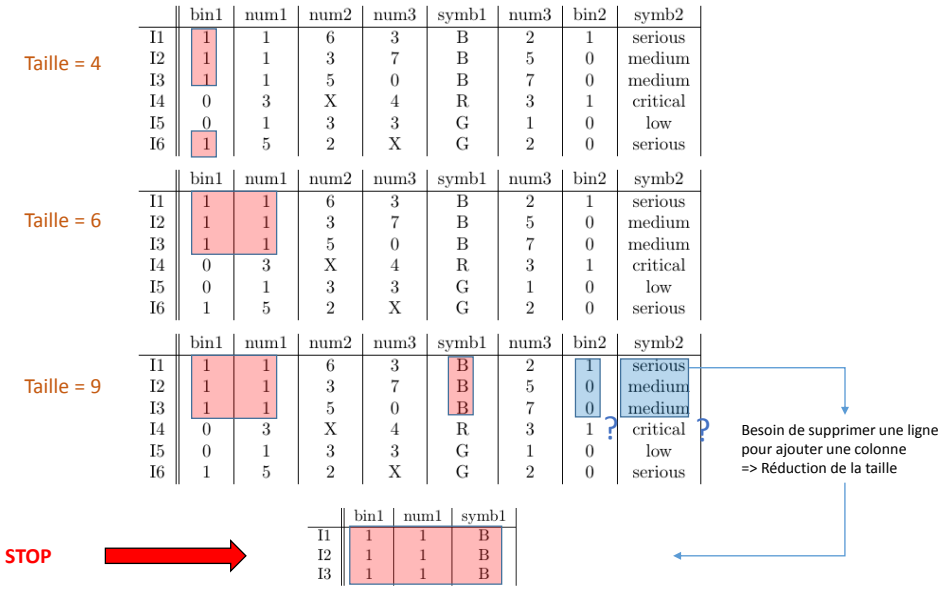
```
construire la matrice de données simplifiée SD à partir des données réelles S;
foreach colonne C dans SD do
    foreach valeur possible V dans C do
        R ← lignes dans C ayant la valeur V;
        créer nouveau bicluster B ← (R, C);
        while (NOT(stop) OU nbColonnes(B) < nbColonnesMinimum) do
            foreach colonne C_j ≠ C do
                val_j ← valeurs dans C_j sur les lignes R;
                H_j ← count(valeur la plus fréquente parmi val_j);
            end
            C_best ← C_j avec max(H_j);
            taille_t ← taille(B);
            ajouter colonne C_best à B;
            supprimer les lignes pour que B reste parfaitement homogène;
            taille_{t+1} ← taille(B);
            if taille_{t+1} < taille_t then
                stop = true;
            end
        end
        ajouter B aux biclus;
    end
end
vraisBiclus ← traduire biclus sur données réelles D;
post-traitement(vraisBiclus);
return vraisBiclus;
```

Ce principe induit la nécessité de disposer d'un ensemble fini de valeurs possibles pour chaque

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017



	bin1	num1	num2	num3	symb1	num3	bin2	symb2
I1	1	1	6	3	B	2	1	serious
I2	1	1	3	7	B	5	0	medium
I3	1	1	5	0	B	7	0	medium
I4	0	3	X	4	R	3	1	critical
I5	0	1	3	3	G	1	0	low
I6	1	5	2	X	G	2	0	serious

	bin1	num1	num2	num3	symb1	num3	bin2	symb2
I1	1	1	6	3	B	2	1	serious
I2	1	1	3	7	B	5	0	medium
I3	1	1	5	0	B	7	0	medium
I4	0	3	X	4	R	3	1	critical
I5	0	1	3	3	G	1	0	low
I6	1	5	2	X	G	2	0	serious

	bin1	num1	num2	num3	symb1	num3	bin2	symb2
I1	1	1	6	3	B	2	1	serious
I2	1	1	3	7	B	5	0	medium
I3	1	1	5	0	B	7	0	medium
I4	0	3	X	4	R	3	1	critical
I5	0	1	3	3	G	1	0	low
I6	1	5	2	X	G	2	0	serious

	bin1	num1	symb1
I1	1	1	B
I2	1	1	B
I3	1	1	B

FIGURE 5.6 – Exemple de formation d'un bicluster par HBC : ajout de la colonne *num1* (et suppression de la ligne $I6$), puis ajout de la colonne *symb*1 (sans suppression de ligne). A ce point, la construction du bicluster s'arrête car il est impossible d'ajouter une colonne sans supprimer une ligne, ce qui réduirait la taille globale du bicluster

attribut, car cela facilite le comptage des valeurs "identiques" dans une colonne candidate à l'insertion. Parmi les types d'attributs que nous considérons pour le moment, seuls les attributs numériques requièrent une transformation, étant donné qu'ils peuvent potentiellement prendre un nombre infini de valeurs différentes. A l'inverse, les attributs binaires ou symboliques disposent déjà d'une liste finie de valeurs possibles. Nous choisissons de simplifier les attributs numériques par discrétisation. La discrétisation présente également l'avantage d'améliorer souvent les résultats obtenus dans une tâche de fouille de données. Ce comportement a été observé dans diverses études [29], ainsi que dans la première partie de ce travail sur les règles de classification (voir Chapitre 3). Comme nous l'avons vu précédemment, le choix d'une méthode de discrétisation adéquate peut s'avérer complexe. Dans le contexte présent, nous sommes néanmoins limités aux méthodes de discrétisation non-supervisées, car le biclustering est en lui-même un processus non-supervisé. En effet, nous ne disposons pas d'un label unique pour chaque instance qui aiderait à définir un partitionnement approprié. Les méthodes de discrétisation non-supervisées sont moins nombreuses que les méthodes supervisées, et en général plus basiques et moins performantes [66]. Nous choisissons une méthode standard de partition en intervalles de longueur égale, ou *equal-width binning*, avec 10 intervalles. Cette méthode comporte certaines limitations, et est notamment très sensible aux données extrêmes ou aberrantes. Des expérimentations préliminaires ont cependant montré que cette méthode permet à l'algorithme de produire des biclusters de meilleure qualité que d'autres méthodes de discrétisation standard : partition en intervalles de fréquence égale (*equal-frequency binning*) et discrétisation par clustering k-means.

Après cette phase de pré-traitement, l'heuristique construit un ensemble de biclusters "parfaits" sur les données simplifiées, c'est-à-dire des biclusters dans lesquels, pour chaque colonne, toutes les valeurs sont identiques. A noter que ces biclusters ne sont "parfaits" que sur les données

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

simplifiées, pas sur les données réelles. En un sens, le seuil pour considérer des valeurs numériques identiques est contrôlé par le processus de discrétisation. Une fois construits, les biclusters sont traduits sur les données réelles et ajustées en phase de post-traitement. Les données simplifiées ne sont utilisées que dans la première phase du processus de biclustering ; l'évaluation de la qualité et toutes les opérations ultérieures sont menées sur les biclusters réels (*i.e.* sur données réelles).

5.3.1.3 Post-traitement

La phase de post-traitement regroupe un ensemble d'opérations pouvant être appliquées aux biclusters formés par l'heuristique. En effet, l'un des problèmes des méthodes non-supervisées est qu'elles génèrent des résultats sous une forme quelque peu déstructurée. L'algorithme de biclustering proposé ci-dessus ne fait pas exception : un nombre conséquent de biclusters sont générés, potentiellement autant que de valeurs possibles pour l'ensemble des attributs, soit souvent plusieurs centaines ou milliers de biclusters. L'exploitation et l'évaluation de ces résultats "bruts" dépend fortement du type d'information que l'on souhaite extraire des données. Par conséquent, la plupart des procédures présentées ici sont optionnelles et à sélectionner en fonction des besoins. Cependant, les deux mécanismes suivants sont toujours appliqués, car ils sont utiles quelle que soit l'application considérée :

— Calcul de la qualité de chaque bicluster, sur les données réelles. Cette fonction ajoute une information utile sur l'homogénéité des biclusters, et elle est nécessaire pour d'autres procédures.

— Suppression des doublons. Comme mentionné précédemment, l'algorithme peut redécouvrir plusieurs fois un même bicluster. Ces doublons sont supprimés en post-traitement, car ils n'ont aucun intérêt. Leur suppression agit comme correction d'un défaut mineur dans le processus de création de biclusters.

Les autres procédures, optionnelles, sont les suivantes. Leur objectif commun est de réduire le nombre de biclusters, afin de rendre les résultats plus lisibles et plus facilement utilisables. Certaines de ces procédures comportent un paramètre qu'il peut être utile d'ajuster en fonction des cas d'étude, mais même les valeurs par défaut permettent déjà d'alléger l'ensemble des résultats.

— Suppression des biclusters ayant trop peu de lignes. La valeur par défaut est de 15 lignes.

— Suppression des biclusters ayant trop peu de colonnes. Cette méthode permet de forcer l'ajout d'au minimum *nbColonnesMinimum* colonnes dans chaque bicluster, même si cela réduit la taille globale du bicluster. Cela permet de s'assurer que le processus ne s'arrête pas dès le début, sans ajouter aucune colonne. Des tests préliminaires ont révélé que les biclusters qui arrêtent leur expansion dès que cette contrainte est levée (*i.e.* quand le bicluster contient *nbColonnesMinimum* colonnes), sont en général moins intéressants. Par conséquent, ces biclusters sont supprimés par cette procédure.

— Tri des biclusters par qualité décroissante, conservation des k meilleurs résultats uniquement.

— Tri des biclusters par taille décroissante, conservation des k meilleurs résultats uniquement.

5.3.1.4 Construction de biclusters imparfaits

L'idée centrale de la méthode proposée est de construire des biclusters parfaitement homogènes (en colonnes) sur les données simplifiées, puis de les traduire sur les données réelles, ce qui permet de récupérer les variations masquées par le processus de discrétisation (ou autre simplification). Cependant, un problème restant est que ce système ne montre aucune tolérance aux valeurs manquantes ou aberrantes qui peuvent être présentes dans les données. En effet, une ligne

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

est supprimée dès qu'une de ses valeurs n'est pas identique à celles de sa colonne respective. Ce comportement quelque peu extrême a été choisi afin d'accélérer le processus et d'éviter de devoir fixer des paramètres additionnels tels que le "seuil de tolérance aux erreurs". Cela peut néanmoins mener à des résultats sous-optimaux, car une ligne parfaitement homogène avec le reste du bicluster peut être écartée pour une seule valeur manquante, par exemple. Ce problème peut également provenir de la discrétisation : une valeur peut être proche des autres sur les données réelles, mais correspondre à une valeur discrétisée différente (surtout pour les valeurs proches des limites des intervalles de la discrétisation).

Nous proposons ici un mécanisme permettant d'atténuer ce problème, qui doit être exécuté après le calcul de la qualité des biclusters et la suppression des doublons. Ce mécanisme, détaillé dans l'Algorithme 10, ajoute des lignes à un bicluster tant que cela ne déteriore pas le ratio de sa *qualité* (à minimiser) sur sa *taille* (à maximiser).

Algorithme 10 : Procédure pour la formation de biclusters imparfaits par ajout de lignes

foreach *bicluster* B **do**
  foreach *ligne* L *non-incluse dans* B **do**
    $B_L \leftarrow B \cup L$;
    $q_{BL} \leftarrow \text{qualité}(B_L)$;
  end
  trier les lignes L_i par qualité croissante;
  $r \leftarrow \dfrac{\text{qualité}(B)}{\text{taille}(B)}$;
  repeat
    $B \leftarrow B \cup L_i$;
    $r' \leftarrow \dfrac{\text{qualité}(B)}{\text{taille}(B)}$;
  until $r' \leq r$;
end

Nous choisissons ce ratio comme critère d'arrêt pour deux raisons. La première est que ces deux mesures, qualité et taille, sont par nature concurrentes, et souvent utilisées dans les algorithmes de biclustering multi-objectif, comme nous l'avons vu dans le Chapitre 6. Utiliser ce ratio permet d'également de ne pas avoir à fixer de paramètre pour contrôler la dégradation de la qualité tolérée. A noter que dans cette procédure, les lignes sont triées une seule fois, au début. Cela signifie que les lignes sont ajoutées au bicluster dans un ordre dépendant de la variation de qualité que chacune apporte au bicluster de départ. A mesure que le bicluster inclut de nouvelles lignes, les qualités changent et l'ordre idéal peut devenir différent de l'ordre original. Ainsi, pour garantir l'optimalité de la procédure, il faudrait recalculer les qualités pour les lignes restantes et l'ordre de ces lignes qui en découle, après chaque inclusion de ligne. Procéder ainsi est beaucoup plus coûteux en termes de temps de calcul. De plus, des expérimentations ont révélé que l'ordre ne change qu'assez peu au fil des itérations, aussi le recalcul systématique n'apporterait-il au mieux que des gains de qualité très minimes. Par conséquent, nous nous cantonnons à la version basique, où l'ordre est calculé une seule fois au début de la procédure. Ce mécanisme pourrait également être appliqué aux colonnes au lieu des lignes. Cependant, il n'y a, conceptuellement parlant, aucune raison de la faire, étant donné que l'heuristique de biclustering procède déjà par ajout itératif de colonnes.

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

5.3.1.5 Stockage des données et complexité

La volumétrie des données hospitalières peut poser des problèmes au niveau du stockage des données ou du temps d'exécution de l'algorithme de biclustering. Nous exposons ici la complexité en espace et en temps de HBC afin de vérifier théoriquement sa scalabilité.

Complexité en espace HBC vise à traiter des matrices de données à la fois très larges et très creuses (nombreuses valeurs manquantes). De telles matrices ne peuvent raisonnablement pas être stockées sous forme d'un tableau à deux dimensions. Nous utilisons à la place un format de stockage compressé, CSC (pour *Compressed Sparse Column*) [24], qui garantit un accès en colonnes rapide. Cela permet d'accélérer le processus de biclustering, qui procède par ajout de colonnes. Ce format garantit une complexité en espace de l'ordre de $O(2n)$, où n est le nombre de valeurs non-manquantes dans la matrice de données.

Complexité en temps L'opération centrale de l'heuristique décrite plus haut est l'ajout d'une colonne à un bicluster en formation (et la suppression de lignes afin de garder le nouveau bicluster homogène). Le coût de cette opération est le même que de chercher quelles lignes de la colonne à ajouter sont présentes dans le bicluster en formation. Cela se traduit par une complexité dans le pire des cas en $O(i.log(i))$, étant donné que la recherche d'un élément dans une liste ordonnée est de l'ordre de $O(log(i))$, où i est le nombre de lignes de la matrice de données. A chaque étape, la meilleure colonne est ajoutée, ce qui signifie que toutes les colonnes doivent être essayées afin de déterminer laquelle est la plus adaptée (celle qui nécessite de supprimer le moins de lignes). Par conséquent, $O(j^2)$ opérations d'ajout de colonne sont effectuées lors de la construction de chaque bicluster, où j est le nombre de colonnes dans la matrice de données. Tout ce processus est exécuté pour chaque colonne et chaque valeur possible comme point de départ, donc $O(jk)$ itérations, où k est le nombre de valeurs possibles pour un attribut. Cela donne une complexité totale dans le pire des cas de $O(i.log(i).j^3.k)$. Cependant, en pratique, bien moins que $O(j^2)$ opérations d'ajout de colonne sont exécutées, étant donné que cette méthode atteint en général la condition d'arrêt (diminution de la taille du bicluster en formation) après avoir ajouté une dizaine de colonnes environ. Une autre observation intéressante est que le temps d'exécution n'augmente pas de manière linéaire avec le nombre de valeurs possibles k pour chaque attribut. En effet, les détails d'implémentation de la méthode font que k a en pratique un effet très marginal sur le temps d'exécution total. En fonction des données, il arrive parfois que ce paramètre ait même un effet inverse : un jeu de données binaires ($k = 2$ pour chaque attribut) est parfois plus long à traiter qu'un jeu de données numériques ($k = 10$). Par conséquent, augmenter le nombre de valeurs possibles pour les attributs symboliques pour mieux coller à la réalité, ou le nombre d'intervalles de discrétisation pour affiner les biclusters, ne pose pas de problème au niveau du temps d'exécution.

En pratique, le temps d'exécution augmente de manière à peu près linéaire avec le volume effectif de données, calculé par la formule suivante :

$$\text{Volume} = \text{nombre de lignes} \times \text{nombre de colonnes} \times \% \text{ données non-manquantes} \tag{5.3}$$

Un autre avantage de cette méthode est qu'il est possible d'estimer le temps d'exécution total à partir du temps d'exécution pour les premiers attributs, étant donné que l'algorithme traite chaque colonne comme point de départ successivement.

5.3.2 Expérimentations

Dans cette section, nous comparons la méthode proposée, HBC, avec les trois algorithmes déjà utilisés pour comparaison dans la section précédente : CC, SpectralCoclustering et Spec-

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

	Qualité (à minimiser)				Taille (à maximiser)			
	HBC	CC	SpectralCo	SpectralBi	HBC	CC	SpectralCo	SpectralBi
n0s0	**2.39 ± 0.79**	5.01 ± 3.05	5.42 ± 0.9	5.48 ± 1.32	**221 ± 76**	6 ± 2	132 ± 77	146 ± 105
n0s1	**0.40 ± 0.08**	3.54 ± 3.71	5.59 ± 1.0	5.25 ± 0.97	**262 ± 97**	38 ± 65	144 ± 98	150 ± 101
n0s2	**0.23 ± 0.05**	3.37 ± 4.15	5.54 ± 1.37	4.94 ± 1.16	**677 ± 274**	268 ± 380	162 ± 132	148 ± 124
n2s0	**2.50 ± 0.71**	5.27 ± 3.54	5.44 ± 0.88	5.79 ± 1.15	**226 ± 67**	6 ± 1	131 ± 77	127 ± 87
n2s1	**0.67 ± 0.12**	2.98 ± 3.56	5.52 ± 0.98	5.54 ± 1.02	**409 ± 74**	26 ± 30	143 ± 98	135 ± 88
n2s2	**0.37 ± 0.07**	0.49 ± 1.11	5.48 ± 1.22	5.90 ± 1.94	**955 ± 323**	117 ± 69	157 ± 117	116 ± 89
n5s0	**2.71 ± 0.48**	5.63 ± 3.47	5.46 ± 0.95	5.53 ± 1.34	**243 ± 60**	6 ± 1	131 ± 76	136 ± 82
n5s1	**1.09 ± 0.19**	2.53 ± 3.02	5.52 ± 0.83	5.68 ± 1.11	**389 ± 88**	11 ± 5	142 ± 94	127 ± 87
n5s2	**0.66 ± 0.11**	0.78 ± 0.26	5.56 ± 1.45	5.74 ± 1.62	**850 ± 306**	21 ± 5	155 ± 132	121 ± 95
n10s0	**2.94 ± 0.10**	5.12 ± 3.19	5.39 ± 0.8	5.0 ± 1.0	**261 ± 19**	6 ± 1	130 ± 74	168 ± 107
n10s1	**2.00 ± 0.34**	3.79 ± 3.05	5.56 ± 1.02	5.96 ± 1.4	**397 ± 104**	7 ± 2	139 ± 84	120 ± 88
n10s2	**1.14 ± 0.12**	1.73 ± 0.86	5.59 ± 1.22	5.09 ± 0.98	**702 ± 185**	10 ± 2	153 ± 115	158 ± 109

TABLE 5.5 – Qualité et taille moyennes des biclusters

	U				E			
	HBC	CC	SpectralCo	SpectralBi	HBC	CC	SpectralCo	SpectralBi
n0s0	**0.92 ± 0.09**	0.01 ± 0.04	0.02 ± 0.01	0.02 ± 0.01	**0.02 ± 0.05**	0.4 ± 0.2	0.75 ± 0.04	0.48 ± 0.25
n0s1	**0.92 ± 0.07**	0.22 ± 0.19	0.03 ± 0.03	0.01 ± 0.01	**0.0 ± 0.0**	0.14 ± 0.24	0.51 ± 0.10	0.61 ± 0.12
n0s2	**0.91 ± 0.07**	0.57 ± 0.16	0.01 ± 0.01	0.01 ± 0.01	**0.0 ± 0.0**	0.01 ± 0.09	0.44 ± 0.12	0.47 ± 0.11
n2s0	**0.74 ± 0.36**	0.01 ± 0.03	0.01 ± 0.01	0.02 ± 0.01	**0.16 ± 0.26**	0.39 ± 0.2	0.74 ± 0.04	0.61 ± 0.11
n2s1	**0.86 ± 0.16**	0.11 ± 0.08	0.03 ± 0.03	0.02 ± 0.02	**0.01 ± 0.01**	0.12 ± 0.23	0.52 ± 0.15	0.51 ± 0.09
n2s2	**0.84 ± 0.19**	0.13 ± 0.05	0.02 ± 0.01	0.01 ± 0.01	**0.01 ± 0.02**	0.03 ± 0.13	0.39 ± 0.08	0.22 ± 0.2
n5s0	**0.61 ± 0.42**	0.01 ± 0.01	0.02 ± 0.01	0.0 ± 0.0	**0.29 ± 0.31**	0.42 ± 0.16	0.63 ± 0.17	0.58 ± 0.12
n5s1	**0.77 ± 0.31**	0.02 ± 0.02	0.03 ± 0.02	0.03 ± 0.01	**0.05 ± 0.15**	0.13 ± 0.21	0.58 ± 0.16	0.52 ± 0.16
n5s2	**0.69 ± 0.29**	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	**0.01 ± 0.04**	0.17 ± 0.28	0.40 ± 0.02	0.32 ± 0.18
n10s0	**0.07 ± 0.14**	0.01 ± 0.01	0.03 ± 0.01	0.02 ± 0.0	0.71 ± 0.13	**0.43 ± 0.19**	0.66 ± 0.06	0.72 ± 0.05
n10s1	**0.57 ± 0.30**	0.01 ± 0.01	0.03 ± 0.02	0.01 ± 0.01	**0.11 ± 0.22**	0.19 ± 0.23	0.52 ± 0.17	0.52 ± 0.08
n10s2	**0.4 ± 0.17**	0.01 ± 0.01	0.01 ± 0.01	0.02 ± 0.02	**0.01 ± 0.05**	0.21 ± 0.33	0.35 ± 0.16	0.30 ± 0.23

TABLE 5.6 – Mesures de redécouverte des biclusters injectés U et E

tralBiclustering. Le protocole expérimental est similaire à celui du chapitre précédent. Etant donné que tous les algorithmes comparés ne contiennent aucun élément stochastique, une seule exécution pour chaque variante est suffisante. Le calcul des mesures U et E est effectué de la même manière que précédemment, en considérant pour chaque bicluster injecté le bicluster trouvé par l'algorithme qui en est le plus proche. Les Tables 5.5 et 5.6 présentent les résultats obtenus par ces quatre méthodes pour les mesures de qualité et taille des biclusters, et les mesures de redécouverte des biclusters injectés U et E, respectivement. Comme précédemment, les tables présentent les résultats pour 5 biclusters injectés, avec 20% de recouvrement.

Nous constatons que HBC produit dans tous les cas des biclusters plus grands et de meilleure qualité que les autres algorithmes. Une autre observation sur ces résultats pour U est que HBC parvient à retrouver au moins une partie de tous les biclusters injectés pour toutes les variantes considérées, à l'exception de *n10s0* (bruit élevé, petits biclusters), qui est le cas le plus difficile. Les résultats pour la mesure E montrent qu'assez peu d'éléments externes aux vrais biclusters sont ajoutés. De manière générale, HBC parvient à retrouver une portion significative des biclusters injectés (mesure U élevée) et à les isoler convenablement du reste des données (mesure E basse). Un test statistique de Wilcoxon indique que HBC donne de meilleurs résultats de manière significative sur les quatre mesures (qualité, taille, U et E), avec des $p - values < 0.05$. Tous les algorithmes comparés ont un temps d'exécution inférieur à une minute sur les jeux de données considérés.

Nous n'avons pas inclus dans ces tableaux les résultats obtenus par LSBC (l'approche par méta-heuristique du chapitre précédent), car le manque de diversité dans les résultats de celle-ci

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

la disqualifie. Cette méthode génère des biclusters en moyenne plus grands et de meilleure qualité que HBC, mais donne des résultats très médiocres au niveau de la redécouverte des biclusters injectés.

5.3.3 Discussion

Dans cette section, nous avons motivé l'adoption d'une nouvelle approche pour le biclustering sur données hétérogènes, afin de corriger les lacunes de la première approche. Cette nouvelle méthode se base sur une phase de simplification des données en pré-traitement, une heuristique constructive et une phase de post-traitement pour évaluer, trier, filtrer et compléter les biclusters générés. Cette méthode permet de couvrir l'ensemble des données et de détecter des biclusters plus petits mais homogènes, ce que ne permettait pas la précédente. Nous avons proposé un ensemble de procédures de post-traitement pour adapter la méthode aux besoins. Des expérimentations sur données synthétiques ont révélé que cette méthode génère rapidement des résultats de bonne qualité, et permet de retrouver les biclusters artificiels injectés dans presque tous les cas. Des tests statistiques montrent que cette méthode, HBC, donne des résultats significativement meilleurs que CC, qui utilise également un paradigme d'heuristique constructive.

5.4 Conclusion

Dans ce chapitre, nous avons présenté le biclustering et justifié l'utilité de son application à la fouille de données hospitalières. Nous avons exposé les différents problèmes à résoudre afin de pouvoir traiter ce type de données, notamment liés à leur hétérogénéité. Dans une première approche, nous avons proposé une modélisation du biclustering sous forme d'un problème d'optimisation multi-objectif, incluant une mesure de qualité pour biclusters sur données hétérogènes, ainsi qu'une méta-heuristique basée sur une recherche locale et ses différents composants (opérateurs de voisinage, etc.). Cette première méthode génère des biclusters de très bonne qualité, mais souffre d'un problème de diversité qui l'empêche de générer des solutions notablement différentes les unes des autres. A partir de ces observations, nous avons présenté une seconde approche, appelée HBC, basée sur une heuristique constructive et une simplification des données en pré-traitement. Des expérimentations indiquent que cette méthode donne également des résultats de bonne qualité, et est capable de couvrir des régions très diverses de la matrice de données. Cette méthode forme donc une bonne base pour la gestion des données hétérogènes, mais nécessite une extension à la gestion des données temporelles, très présentes parmi les données hospitalières. Le chapitre suivant présente cette extension, ainsi que des premières expérimentations sur des données réelles. Un cas d'étude plus complet est présenté dans le Chapitre 8.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Chapitre 6

Biclustering sur données temporelles

Dans le chapitre précédent, nous avons présenté l'algorithme HBC et son application à plusieurs types de données non-temporelles : numériques, symboliques, ordinales et binaires. Dans ce chapitre, nous détaillons l'extension de cet algorithme à deux types de données temporelles : les *événements temporels* et les *séries temporelles*. Dans une première section, nous nous intéressons aux événements temporels, que nous avons déjà intégré au sein de l'algorithme MOSC dans le Chapitre 4. La seconde section se concentre sur les séries temporelles, et plus particulièrement les séries temporelles *irrégulières*, un cas particulier plus adapté aux données hospitalières que nous souhaitons pouvoir traiter. La structure de checune de ces sections met en exergue les différents éléments à concevoir afin d'étendre HBC au nouveau type de données considéré : mesure de qualité, transformation des données en pré-traitement et post-traitement.

6.1 Gestion des événements temporels

Dans cette section, nous rappelons d'abord les caractéristiques des événements temporels que nous souhaitons utiliser comme données dans HBC. Nous motivons l'utilisation de ces données et présentons un modèle permettant de les intégrer dans HBC, ainsi qu'une mesure de qualité adaptée à ce nouveau type de données. Ensuite, nous apportons quelques modifications dans HBC pour la formation de biclusters comportant des événements temporels. Après cela, nous présentons des expérimentations validant l'efficacité de l'approche choisie : comparaison avec une méthode existante de référence, scalabilité sur données larges et creuses, et exemples de biclusters trouvés sur données réelles. Nous concluons cette partie avec une discussion de la méthode et des résultats, et la présentation de perspectives d'amélioration.

6.1.1 Motivations et modélisation

Nous nous intéressons dans cette partie aux données temporelles telles que décrites dans le chapitre 4 sur MOSC (classification de séquences temporelles), c'est-à-dire aux *événements temporels*. Pour ceux-ci, au lieu de savoir uniquement que l'événement s'est produit, nous savons également quand il s'est produit. Comme expliqué dans le Chapitre 4, exploiter ces données temporelles présente un réel intérêt d'un point de vue métier, car les résultats obtenus sur de telles données peuvent représenter plus précisément la réalité. Ce type d'attribut ne saurait être traité comme un simple attribut binaire, car cela impliquerait de ne pas utiliser l'information supplémentaire apportée par la date (et l'heure) associée à chaque événement. Par ailleurs, bien que la date soit une valeur numérique, celle-ci ne devrait pas être traitée de la même manière que les attributs numériques. En effet, nous estimons que la date associé à un événement n'apporte

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

pas, en tant que tel, suffisamment d'information pour être utile. Par exemple, le fait qu'un événement E se produise un jour chez un patient A, et un autre jour chez un patient B, ne signifie rien en soi. Une alternative serait de considérer le temps écoulé depuis le premier événement pour chaque patient, au lieu du temps absolu. Une telle méthode permet d'atténuer quelque peu le problème, mais ne saurait s'adapter aux fortes disparités entre les parcours de soins de chaque patient. En effet, procéder ainsi suppose qu'en général, un même événement se produit à des instants fixes du parcours de soins, par exemple "deux heures après l'admission". Cela n'est que rarement le cas dans la réalité, aussi raisonner sur les temps de cette manière ne paraît pas adéquat.

Par conséquent, nous choisissons une seconde approche, dans laquelle nous nous concentrons sur l'ordre relatif des événements se produisant pour un patient, et non plus sur les dates auxquelles ceux-ci se produisent. Dans le contexte du biclustering, cela signifie que nous considérons que des patients sont similaires pour un sous-ensemble d'événements si ces événement se produisent dans le même ordre chez ces patients. La Figure 6.1 illustre cette notion de transformation des temps en positions relatives dans les séquences d'événements, ainsi que la formation d'un bicluster sur un sous-ensemble d'événements ordonnés de manière identique entre eux.

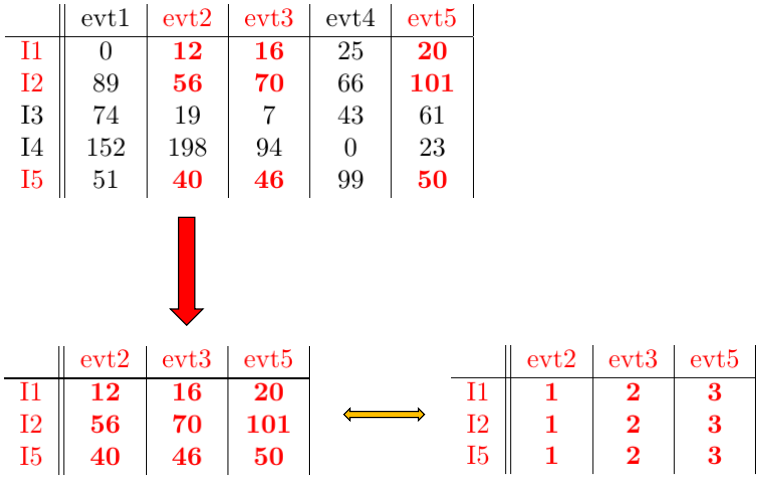


FIGURE 6.1 – Extraction d'un bicluster pour des données temporelles. Le bicluster extrait ici comporte trois lignes I1, I2 et I5 pour lesquelles l'ordre des événements evt2, evt3 et evt5 est identique. Cet ordre est représenté dans le tableau en bas à droite.

Comme montré sur cet exemple, seul l'ordre relatif des événements au sein du sous-ensemble doit être identique, et non l'ordre relatif pour l'ensemble des attributs événements. Dans l'instance $I2$, les événements $evt1$ et $evt4$ (hors bicluster) se produisent entre $evt3$ et $evt5$ (qui font partie du bicluster). L'ordre relatif au sein du bicluster n'est cependant pas altéré, aussi considérons-nous $I2$ identique à $I1$ et $I5$ pour les trois événements considérés. En cela, la modélisation choisie est proche de celle adoptée pour la classification de séquences temporelles (voir Chapitre 4), où l'inclusion d'événements supplémentaires n'invalide pas la similarité de deux séquences, tant que l'ordre de celles-ci est respecté. L'idée de considérer l'ordre des événements provient du travail pionnier sur les *Order Preserving SubMatrices* (OPSM) [16]. De nombreux autres travaux sur le biclustering impliquent une forme de temporalité dans les données, notamment pour les applications en génomique, mais OPSM formalise ce problème plus précisément et introduit la

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

transformation des données que nous avons reprise ici pour la modélisation.

Cette tolérance aux variations entre séquences complètes, ainsi que les contraintes plus souples sur les dates, rendent cette approche plus flexible que si les dates exactes étaient utilisés et non les séquences. Cette approche utilise également davantage les relations entre attributs (ordre relatif des événements), là où la première considère chaque attribut individuellement. Cela est davantage en phase avec le principe du biclustering, qui vise entre autres à détecter des corrélations entre les colonnes. D'un point de vue applicatif, nous espérons que l'exploitation des données temporelles permette de passer d'un bicluster décrivant "un groupe de patients chez qui les événements E_1 et E_2 se sont produits", à un bicluster pour "un groupe de patients chez qui E_1 *puis* E_2 se sont produits". De tels biclusters sont plus informatifs et plus faciles à interpréter.

6.1.2 Mesure de qualité

Nous nous concentrons donc sur l'ordre relatif des événements temporels dans une séquence. Par conséquent, l'approche choisie dans le chapitre précédent pour le calcul de la qualité d'un bicluster doit être modifiée pour les attributs temporels. En effet, nous avons choisi de mesurer la qualité totale d'un bicluster comme la somme des qualités individuelles de chaque colonne du bicluster. Cependant, nous avons vu qu'un attribut temporel n'a de sens que par rapport aux autres attributs temporels présents dans le bicluster. Il est donc impossible de calculer une qualité pour chaque attribut temporel individuellement. A la place, nous nous inspirons ici aussi des travaux réalisés sur OPSM, d'une étude qui définit une mesure de qualité pour ce problème comme "un score pour le degré de préservation de l'ordre" [20]. L'idée ici est de calculer la mesure classique de *mean-squared residue score* [31], en utilisant les rangs respectifs des événements comme valeurs. Cette mesure est définie ainsi :

$$q(R,C) = \frac{1}{|R| \times |C|} \sum_{i \in R, j \in C} (e_{ij} - e_{iC} - e_{Rj} + e_{RC})^2 \tag{6.1}$$

avec :

$$e_{iC} = \frac{1}{|C|} \sum_{j \in C} e_{ij}, \quad e_{Rj} = \frac{1}{|R|} \sum_{i \in R} e_{ij} \tag{6.2}$$

$$e_{RC} = \frac{1}{|R| \times |C|} \sum_{i \in R, j \in C} e_{ij} \tag{6.3}$$

R et C sont les lignes et les colonnes du bicluster, respectivement. e_{ij} est l'élément de la matrice situé sur la ligne i et la colonne j. e_{RC} est la moyenne de tous les éléments du bicluster. Dans l'étude présentant cette mesure de qualité, les auteurs montrent que celle-ci est cohérente pour évaluer le degré de préservation de l'ordre dans un bicluster [20]. La mesure est notamment idéale (*i.e.* égale à zéro) quand l'ordre est identique pour toutes les lignes, ce qui correspond à ce que nous recherchons. Pour illustrer cette mesure, nous reprenons l'exemple de la Figure 6.1 et calculons la qualité du bicluster à partir de l'ordre des événements qu'il contient, rappelé dans la Figure 6.2.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

	evt2	evt3	evt5
I1	**1**	**2**	**3**
I2	**1**	**2**	**3**
I5	**1**	**2**	**3**

FIGURE 6.2 – Ordre des événements composant le bicluster dont nous cherchons à calculer la qualité. Les positions des événements sont utilisées comme valeurs pour le calcul de qualité.

$$
\begin{aligned}
q(R,C) &= \frac{1}{|R| \times |C|} \sum_{i \in R, j \in C} (e_{ij} - e_{iC} - e_{Rj} + e_{RC})^2 \\
&= \frac{1}{|R| \times |C|} [(e_{11} - e_{1C} - e_{R1} + e_{RC})^2 + (e_{12} - e_{1C} - e_{R2} + e_{RC})^2 + (e_{13} - e_{1C} - e_{R3} + e_{RC})^2 \\
&+ (e_{21} - e_{2C} - e_{R1} + e_{RC})^2 + (e_{22} - e_{2C} - e_{R2} + e_{RC})^2 + (e_{23} - e_{2C} - e_{R3} + e_{RC})^2 \\
&+ (e_{31} - e_{3C} - e_{R1} + e_{RC})^2 + (e_{32} - e_{3C} - e_{R2} + e_{RC})^2 + (e_{33} - e_{3C} - e_{R3} + e_{RC})^2]
\end{aligned}
$$

avec

$$
\begin{aligned}
e_{RC} &= \frac{1}{|R|.|C|} \sum_{i \in R, j \in C} e_{ij} \\
&= \frac{1}{3 \times 3} [1 + 2 + 3 + 1 + 2 + 3 + 1 + 2 + 3] = 2
\end{aligned}
$$

et

$$
\begin{aligned}
e_{1C} = e_{2C} = e_{3C} &= \frac{1}{3}[1 + 2 + 3] = 2 \\
e_{R1} &= \frac{1}{3}[1 + 1 + 1] = 1 \\
e_{R2} &= \frac{1}{3}[2 + 2 + 2] = 2 \\
e_{R3} &= \frac{1}{3}[3 + 3 + 3] = 3
\end{aligned}
$$

d'où

$$
\begin{aligned}
q(R,C) &= \frac{1}{9}[(1 - 2 - 1 + 2)^2 + (2 - 2 - 2 + 2)^2 + (3 - 2 - 3 + 2)^2 \\
&+ (1 - 2 - 1 + 2)^2 + (2 - 2 - 2 + 2)^2 + (3 - 2 - 3 + 2)^2 \\
&+ (1 - 2 - 1 + 2)^2 + (2 - 2 - 2 + 2)^2 + (3 - 2 - 3 + 2)^2] \\
&= 0
\end{aligned}
$$

En utilisant la mesure proposée, la qualité du bicluster vaut 0, qui est la valeur optimale. Ce résultat correspond à ce qui était escompté, étant donné que le bicluster est parfaitement homogène, c'est-à-dire que l'ordre des événements est identique sur chaque ligne. Cette mesure de qualité est calculée sur l'ensemble des attributs temporels du bicluster, et définit la qualité individuelle de chaque colonne correspondant à un attribut temporel.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

6.1.3 Adaptations de l'algorithme

Comme nous l'avons vu dans le chapitre précédent, certains types de données nécessitent une phase de pré-traitement (e.g. discrétisation), afin de les ramener à un ensemble fini de valeurs possibles. Une telle phase semble être nécessaire pour les attributs temporels, car ceux-ci sont représentés par des valeurs numériques. Cela n'est cependant pas nécessaire, et il est plus efficace de modifier légèrement l'algorithme de formation des biclusters pour le traitement des données temporelles. Pour les autres types d'attributs, l'algorithme recherche la valeur présente sur le plus grand nombre de lignes dans le bicluster en formation. Pour les données temporelles, nous ne considérons pas la valeur d'un attribut en elle-même, mais plutôt l'ensemble des attributs temporels et la séquence d'événements qu'ils représentent. Ainsi, pour ces attributs, l'algorithme calcule pour chaque ligne du bicluster l'ordre des événements après ajout du nouvel attribut. Les lignes conservées sont celles correspondant à la séquence d'événements la plus fréquente. Procéder de cette manière implique certes de modifier le cadre général de l'algorithme de biclustering, mais cela permet d'éviter des phases de transformation des données temporelles avant et après formation des biclusters. A noter que le calcul de l'ordre de la séquence d'événements sur chaque ligne est moins coûteux que le tri d'une liste de valeurs numériques. En effet, il suffit de calculer la position à laquelle le nouvel événement doit être inséré, étant donné que toutes les séquences présentes dans le bicluster en formation ont le même ordre. Cette opération se fait en temps linéaire sur le nombre d'attributs temporels présents dans le bicluster.

6.1.4 Expérimentations

Afin de valider l'approche proposée pour la gestion des événements temporels, nous comparons tout d'abord HBC avec un algorithme de biclustering conçu pour les données temporelles, OPSM. Nous évaluons ensuite l'efficacité et la scalabilité de HBC sur des données synthétiques volumineuses et creuses. Enfin, nous appliquons HBC à des jeux de données hospitalières et présentons des exemples de biclusters trouvés sur ces données.

6.1.4.1 Comparaison avec OPSM

Dans un premier temps, les performances de l'algorithme HBC pour données temporelles sont comparées à celles de l'algorithme OPSM (*Order Preserving SubMatrices*) [16]. Pour ce dernier, l'implémentation utilisée est celle disponible dans l'outil BicAT, une plate-forme logicielle pour le biclustering [14]. Dans ces expérimentations, HBC utilise la méthode de post-traitement pour formation de biclusters imparfaits. Le protocole expérimental est similaire à celui utilisé dans le chapitre précédent. Les données utilisées sont des données synthétiques avec 500 lignes et 100 colonnes, avec 100% d'attributs temporels, et 100% de données présentes car OPSM ne supporte pas les données manquantes. 5 biclusters sont injectés dans les jeux de données, selon les mêmes paramètres que dans le chapitre précédent (taille et niveau de bruit). Les données temporelles sont bruitées de la même manière que les données numériques, avec ajout d'une valeur aléatoire choisie selon une distribution normale de moyenne zéro et de variance correspondant au niveau de bruit choisi. Les Tables 6.1 et 6.2 montrent les performances respectives de HBC et OPSM, pour la qualité q telle que définie à la Section 6.1.2 et la taille des biclusters formés, ainsi que la capacité des algorithmes à retrouver les biclusters injectés (mesures U et E), respectivement.

La Table 6.1 montre que HBC et OPSM génèrent tous deux des biclusters très homogènes (qualité proche de zéro). Dans le cas de HBC, cela s'explique aisément par le fait que l'algorithme génère par définition des biclusters parfaitement homogènes. Les imperfections sont amenées par la procédure d'ajout de lignes en post-traitement. Nous remarquons également que HBC tend à créer des biclusters de taille plus importante que OPSM, et ce, sur l'ensemble des jeux de

© 2017 Tous droits réservés. lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

	Qualité		Taille	
	HBC	OPSM	HBC	OPSM
n0s0	**0.0 ± 0.0**	**0.0 ± 0.0**	**201 ± 13**	176 ± 187
n0s1	**0.0 ± 0.0**	**0.0 ± 0.0**	**343 ± 52**	243 ± 186
n0s2	**0.0 ± 0.0**	**0.0 ± 0.0**	**1049 ± 317**	450 ± 236
n2s0	**0.0 ± 0.0**	0.01 ± 0.01	**192 ± 12**	166 ± 192
n2s1	0.06 ± 0.13	**0.03 ± 0.04**	**313 ± 32**	219 ± 204
n2s2	**0.0 ± 0.0**	0.0 ± 0.04	**560 ± 70**	297 ± 262
n5s0	0.01 ± 0.01	**0.0 ± 0.01**	**188 ± 13**	168 ± 190
n5s1	0.03 ± 0.06	**0.01 ± 0.01**	**248 ± 22**	208 ± 209
n5s2	0.03 ± 0.09	**0.01 ± 0.01**	**402 ± 32**	257 ± 243
n10s0	0.0 ± 0.01	**0.0 ± 0.0**	**191 ± 12**	183 ± 197
n10s1	0.01 ± 0.04	**0.01 ± 0.01**	**228 ± 13**	203 ± 206
n10s2	0.05 ± 0.06	**0.0 ± 0.01**	**320 ± 24**	235 ± 235
n15s0	**0.0 ± 0.0**	**0.0 ± 0.0**	**190 ± 12**	167 ± 191
n15s1	0.01 ± 0.04	**0.0 ± 0.0**	**235 ± 9**	179 ± 206
n15s2	**0.01 ± 0.01**	0.02 ± 0.03	**295 ± 11**	212 ± 232
n20s0	0.0 ± 0.01	**0.0 ± 0.0**	**191 ± 11**	169 ± 189
n20s1	**0.0 ± 0.0**	0.0 ± 0.01	**230 ± 15**	179 ± 203
n20s2	0.02 ± 0.05	**0.01 ± 0.01**	**286 ± 18**	230 ± 237

TABLE 6.1 – Données temporelles synthétiques - qualité q et taille des biclusters pour HBC et OPSM

	U		E	
	HBC	OPSM	HBC	OPSM
n0s0	**0.44 ± 0.21**	0.19 ± 0.11	**0.27 ± 0.18**	0.44 ± 0.25
n0s1	**0.44 ± 0.18**	0.19 ± 0.08	**0.03 ± 0.04**	0.22 ± 0.16
n0s2	**0.84 ± 0.08**	0.18 ± 0.10	**0.0 ± 0.0**	0.25 ± 0.3
n1s0	**0.22 ± 0.07**	0.13 ± 0.08	**0.41 ± 0.1**	0.46 ± 0.17
n1s1	**0.28 ± 0.01**	0.17 ± 0.09	**0.09 ± 0.01**	0.32 ± 0.18
n1s2	**0.19 ± 0.01**	0.13 ± 0.05	**0.05 ± 0.01**	0.13 ± 0.07
n2s0	**0.17 ± 0.06**	0.1 ± 0.07	**0.44 ± 0.09**	0.59 ± 0.1
n2s1	**0.12 ± 0.02**	0.11 ± 0.05	**0.24 ± 0.06**	0.32 ± 0.07
n2s2	**0.12 ± 0.01**	0.08 ± 0.04	**0.09 ± 0.01**	0.31 ± 0.16
n3s0	**0.18 ± 0.05**	0.1 ± 0.07	**0.43 ± 0.08**	0.47 ± 0.24
n3s1	**0.09 ± 0.01**	0.08 ± 0.03	**0.29 ± 0.03**	0.41 ± 0.08
n3s2	0.06 ± 0.01	**0.06 ± 0.03**	**0.18 ± 0.04**	0.35 ± 0.18
n4s0	**0.17 ± 0.04**	0.09 ± 0.07	**0.43 ± 0.08**	0.62 ± 0.09
n4s1	**0.09 ± 0.01**	0.05 ± 0.04	**0.30 ± 0.01**	0.32 ± 0.09
n4s2	**0.06 ± 0.01**	0.05 ± 0.02	**0.18 ± 0.04**	0.41 ± 0.14
n5s0	**0.16 ± 0.04**	0.09 ± 0.07	**0.44 ± 0.07**	0.62 ± 0.09
n5s1	**0.09 ± 0.01**	0.06 ± 0.04	**0.3 ± 0.02**	0.48 ± 0.18
n5s2	**0.05 ± 0.01**	0.05 ± 0.02	**0.23 ± 0.03**	0.33 ± 0.05

TABLE 6.2 – Données temporelles synthétiques - mesures de recouvrement des biclusters (U et E) pour HBC et OPSM

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

données considérés. La Table 6.2 montre que HBC est plus efficace pour retrouver des biclusters injectés dans les données, d'après les deux mesures considérées. Nous observons cependant que les performances des deux algorithmes se dégradent rapidement dès que du bruit est ajouté aux données. Cela s'explique par le fait que les biclusters injectés pour les données temporelles correspondent à une séquence $[t, t+1, t+2, t+3, \dots]$. Par conséquent, le moindre bruit peut facilement changer l'ordre des événements, car les valeurs originales sont très proches entre elles.

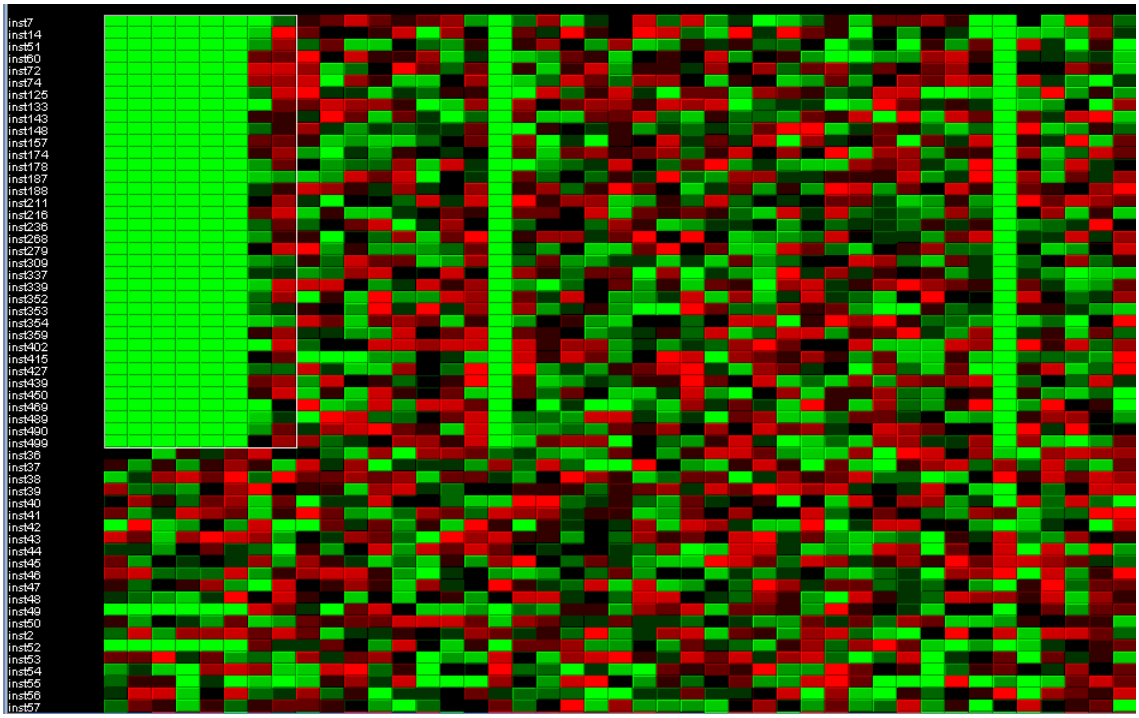


FIGURE 6.3 – Exemple de bicluster trouvé par OPSM dans BicAT, en haut à gauche sur l'image, sur un jeu de données $n0s1$ (pas de bruit, taille 50x10). On remarque que deux autres lignes (*inst49* et *inst52*) et colonnes, en vert, pourraient être ajoutées au bicluster mais ne le sont pas

La Figure 6.3 montre un exemple de bicluster généré par OPSM, via l'interface graphique intégrée dans BicAT. Une observation étonnante sur cette figure est que l'algorithme semble passer à côté de lignes et de colonnes parfaitement homogènes avec le reste du bicluster (en vert), sans raison apparente. OPSM génère un ensemble de biclusters contenant un nombre variable de lignes et de colonnes, mais ce comportement est présent dans chacun de ces biclusters. Cela explique les résultats assez médiocres pour la redécouverte des biclusters injectés (mesure U).

6.1.4.2 Scalabilité

Pour la comparaison avec OPSM, nous avons utilisé des données de taille assez réduite et sans valeurs manquantes, car OPSM est adapté à ce type de données. Cependant, nous cherchons davantage à pouvoir gérer des jeux de données plus larges, avec un grand nombre de valeurs manquantes ; ces caractéristiques étant présentes dans les données hospitalières réelles. Afin de valider les performances de HBC sur données temporelles de ce type, nous présentons ici les résultats obtenus par l'algorithme sur un ensemble de jeux de données synthétiques avec 100 000 lignes, 10 000 colonnes et 0.5% de valeurs présentes. Ces caractéristiques correspondent à

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

	Quality	Size	U	E
n0s0	0.0 ± 0.0	100 ± 0.0	1.0 ± 0.0	0.0 ± 0.0
n0s1	0.0 ± 0.0	500 ± 0.0	1.0 ± 0.0	0.0 ± 0.0
n0s2	0.0 ± 0.0	1500 ± 0.0	1.0 ± 0.0	0.0 ± 0.0
n2s0	0.0 ± 0.0	93 ± 6	0.92 ± 0.06	0.0 ± 0.0
n2s1	0.0 ± 0.0	414 ± 27	0.83 ± 0.05	0.0 ± 0.0
n2s2	0.0 ± 0.0	1139 ± 70	0.76 ± 0.04	0.0 ± 0.0
n5s0	0.0 ± 0.0	81 ± 9	0.81 ± 0.09	0.0 ± 0.0
n5s1	0.0 ± 0.0	324 ± 30	0.65 ± 0.06	0.0 ± 0.0
n5s2	0.0 ± 0.0	781 ± 64	0.52 ± 0.04	0.0 ± 0.0
n10s0	0.0 ± 0.0	69 ± 9	0.69 ± 0.08	0.0 ± 0.0
n10s1	0.0 ± 0.0	223 ± 26	0.45 ± 0.05	0.0 ± 0.0
n10s2	0.0 ± 0.0	483 ± 40	0.33 ± 0.02	0.0 ± 0.0
n15s0	0.0 ± 0.0	57 ± 9	0.57 ± 0.11	0.01 ± 0.1
n15s1	0.0 ± 0.0	174 ± 21	0.36 ± 0.04	0.0 ± 0.0
n15s2	0.0 ± 0.0	349 ± 32	0.24 ± 0.02	0.0 ± 0.0
n20s0	0.0 ± 0.0	50 ± 7	0.45 ± 0.16	0.1 ± 0.3
n20s1	0.0 ± 0.0	133 ± 14	0.28 ± 0.02	0.0 ± 0.0
n20s2	0.0 ± 0.0	268 ± 19	0.19 ± 0.01	0.0 ± 0.0

TABLE 6.3 – Performances de HBC - données temporelles pures, larges - qualité, taille et recouvrement des biclusters injectés

peu près aux jeux de données réels sur lesquels nous travaillons. Dans un premier temps, nous examinons les performances de l'algorithme lorsque tous les attributs sont temporels. Ensuite, nous étudions le cas de données mixtes, avec la répartition suivante : 20% numériques, 20% binaires, 20% symboliques, 40% temporelles. Ce second cas est présenté afin de rester en phase avec le but final de la méthode, qui est bien de gérer des données hétérogènes (dont temporelles). Nous conservons les mêmes valeurs pour les différentes variantes de chaque jeu de données (taille des biclusters, bruit, etc.). Cela signifie que l'algorithme doit rechercher des biclusters très petits (20 lignes et 5 colonnes) par rapport à la taille totale des données, ce qui constitue souvent un cas difficile pour les algorithmes de biclustering. Cette étude est intéressante car elle permet de valider la capacité de HBC à trouver de tels biclusters, qui peuvent représenter des groupes de patients intéressants dans les données réelles. Les Tables 6.3 et 6.4 présentent les résultats obtenus pour l'ensemble des quatre mesures considérées, pour les données temporelles "pures" et pour les données hétérogènes respectivement. Dans les tables, la notation $nXsY$ correspond à un jeu de données avec le niveau de bruit X et la taille Y (où *s0* est 20x5, *s1* est 50x10, et *s2* est 100x15). Nous n'utilisons que HBC pour ces expérimentations, car il n'existe, à notre connaissance, aucun autre algorithme adapté à ce type de données.

D'après les Tables 6.3 et 6.4, on observe que HBC génère dans tous les cas un ensemble de biclusters de très bonne qualité (proche de zéro). La taille moyenne dépend fortement de celle des biclusters injectés. On observe également que, pour toutes les variantes, HBC parvient à retrouver au moins une partie des 5 biclusters injectés (mesure U). De plus, E est le plus souvent faible (< 0.1), ce qui signifie que les biclusters sont bien isolés du reste des données, et que peu de données externes aux biclusters injectés sont ajoutées. Dans l'ensemble, ces résultats sont positifs et montrent l'efficacité de HBC pour le biclustering sur données larges, creuses, hétérogènes et temporelles. L'algorithme génère toujours des solutions de bonne qualité, et retrouve une partie significative des biclusters même dans les cas les plus difficiles (bruit et petits biclusters).

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

	Quality	Size	U	E
n0s0	0.0 ± 0.01	97 ± 7	0.97 ± 0.08	0.0 ± 0.0
n0s1	0.0 ± 0.0	486 ± 26	0.97 ± 0.07	0.0 ± 0.0
n0s2	0.0 ± 0.0	1469 ± 71	0.96 ± 0.09	0.0 ± 0.0
n2s0	0.05 ± 0.03	84 ± 11	0.85 ± 0.1	0.0 ± 0.0
n2s1	0.02 ± 0.02	353 ± 76	0.77 ± 0.10	0.0 ± 0.0
n2s2	0.01 ± 0.01	910 ± 270	0.65 ± 0.13	0.0 ± 0.0
n5s0	0.04 ± 0.05	69 ± 8	0.7 ± 0.08	0.0 ± 0.0
n5s1	0.01 ± 0.01	298 ± 37	0.56 ± 0.08	0.0 ± 0.0
n5s2	0.01 ± 0.01	689 ± 149	0.47 ± 0.06	0.0 ± 0.0
n10s0	0.05 ± 0.10	63 ± 11	0.63 ± 0.09	0.0 ± 0.0
n10s1	0.02 ± 0.02	235 ± 35	0.48 ± 0.05	0.0 ± 0.0
n10s2	0.01 ± 0.01	469 ± 208	0.38 ± 0.03	0.0 ± 0.0
n15s0	0.08 ± 0.13	59 ± 13	0.58 ± 0.09	0.0 ± 0.0
n15s1	0.02 ± 0.02	198 ± 37	0.41 ± 0.03	0.0 ± 0.0
n15s2	0.01 ± 0.01	350 ± 183	0.32 ± 0.02	0.0 ± 0.0
n20s0	0.08 ± 0.11	55 ± 13	0.54 ± 0.09	0.01 ± 0.1
n20s1	0.02 ± 0.02	166 ± 43	0.37 ± 0.03	0.0 ± 0.0
n20s2	0.01 ± 0.01	326 ± 151	0.3 ± 0.02	0.0 ± 0.0

TABLE 6.4 – Performances de HBC - données hétérogènes (dont temporelles), larges - qualité, taille et recouvrement des biclusters injectés

	Nb lignes	Nb colonnes	Nb présentes	% présentes
MRB	194 715	11 450	2 584 252	0.12%
PMSI	102 896	11 413	1 413 057	0.12%
PMSI_2013	49 231	7 941	746 566	0.19%

TABLE 6.5 – Données réelles - caractéristiques

6.1.4.3 Données réelles

Nous appliquons maintenant HBC à trois jeux de données réels extraits d'un système d'information hospitalier. MRB provient d'un cas d'étude sur les bactéries multi-résistantes, tandis que PMSI et PMSI_2013 sont plus génériques. Ces deux jeux de données contiennent presque exclusivement les données PMSI, ainsi que l'âge et le genre des patients. MRB contient également ces informations, ainsi que quelques dizaines d'attributs numériques et symboliques représentant les données additionnelles concernant le cas d'étude (antibiotiques notamment). La Table 6.5 présente les caractéristiques principales de ces trois jeux de données : nombre d'instances, nombre d'attributs, nombre et pourcentage de valeurs présentes. La Table 6.6 présente la qualité et taille moyenne des biclusters découverts sur chaque jeu de données. Le graphe représenté sur la Figure 6.4 montre les biclusters trouvés sur le jeu MRB selon leur qualité et taille respectives.

Une première observation sur ces résultats est que la qualité moyenne des biclusters formés est très bonne (proche de zéro), avec une variance assez réduite. D'un autre côté, la taille moyenne est importante, mais avec une variance plus élevée. Dans l'ensemble, cela signifie que les biclusters trouvés sont tous très homogènes, mais ont des tailles très différentes. Cette observation est supportée par le graphe de la Figure 6.4, qui montre que les biclusters ont le plus souvent une très bonne qualité mais des tailles très variées (d'où l'échelle logarithmique utilisée en ordonnée).

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

	Qualité	Taille	Runtime (s.)
MRB	0.05 ± 0.18	1446 ± 4388	5863
PMSI	0.03 ± 0.06	624 ± 1354	1047
PMSI_2013	0.01 ± 0.04	736 ± 1625	519

TABLE 6.6 – Données réelles - résultats

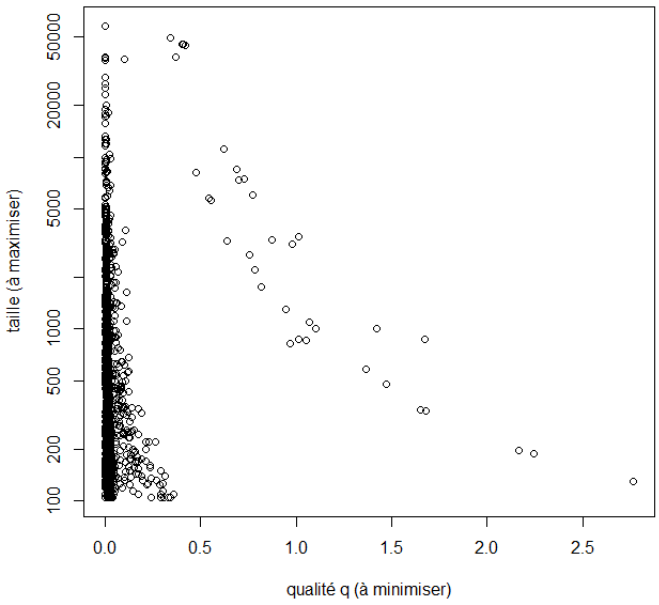


FIGURE 6.4 – Données réelles - qualité et taille des biclusters trouvés sur le jeu MRB

Exemples de biclusters Nous présentons ici quelques biclusters trouvés sur les jeux de données réels utilisés pour l'étude, et montrons que ces biclusters ont un sens d'un point de vue médical. Un premier exemple, extrait de MRB, contient 55 patients et les 4 attributs suivants :

— {Unité médicale 1} Maladies infectieuses et parasitiques
— {Unité médicale 1} Isolement prophylactique (préventive)
— {Unité médicale 1} Staphylocoque doré
— {Unité médicale 1} Agents résistants à la méthicilline

Celui-ci est assez simple à interpréter ; il regroupe les patients ayant contracté un staphylocoque résistant à la méthicilline, qui ont été placés en isolement pour éviter la propagation.

Un second bicluster, extrait de PMSI, contient 1102 lignes et 4 attributs :

— {Unité médicale 1} Diabète sucré
— {Unité médicale 1} Electrocardiographie sur au moins 12 dérivations
— {Unité médicale 1} Rétinographie en couleur ou en lumière monochromatique

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

— {Unité médicale 1} Examen de contrôle après d'autres traitements pour d'autres affectations

Ce bicluster décrit un ensemble de patients diabétiques. Une rétinographie est effectuée car le diabète est souvent associé à des problèmes occulaires ; l'électrocardiogramme car les maladies coronariennes sont les principales comorbidités associées au diabète. A noter que dans les deux exemples montrés jusqu'à présent, tous les événements sont considérés comme simultanés, car se produisant dans la même unité médicale.

Un dernier exemple est présenté pour montrer la capacité de HBC à trouver des biclusters présentant un réel aspect temporel. Ce bicluster est extrait de PMSI, et comporte 598 lignes et 3 attributs :

— {Unité médicale 1} Radiographie du thorax

— {Unité médicale 2} Maladies endocriniennes, nutrionnelles et métaboliques

— {Unité médicale 2} Maladies de l'appareil circulatoire

Dans ce bicluster, une radiographie est d'abord effectuée, puis les deux diagnostics dans une seconde unité médicale. Cela est cohérent, car une radiographie est souvent faite au début de la prise en charge, avant d'orienter le patient vers l'unité de soins la plus appropriée.

6.1.5 Conclusion

Dans cette section, nous avons détaillé l'extension du modèle général de HBC pour les *événements temporels*. Ces données sont du même type que celles qui nous intéressaient dans la partie précédente sur les règles de classification temporelle. L'approche par biclustering offre donc un éclairage différent sur ces données et permet de répondre à des problématiques plus exploratoires. Nous avons adopté dans ce chapitre une modélisation assez proche de celle employée pour la classification sur données temporelles, en nous concentrant sur les séquences d'événements plutôt que sur les moments d'occurrence de ces événements. Nous avons choisi de modifier légèrement l'algorithme de biclustering pour ces données temporelles, tout en gardant le même paradigme général de construction itérative de biclusters homogènes. La méthode proposée ne nécessite aucune transformation des données, n'effectue aucune opération réellement plus coûteuse que pour les autres types d'attributs, et génère des biclusters contenant des séquences d'événements parfaitement homogènes. L'algorithme HBC ainsi étendu a été favorablement comparé à OPSM, la méthode de référence pour le biclustering sur données temporelles. D'autres expérimentations montrent son adaptation aux matrices de données larges et creuses. L'application sur des jeux de données réels confirme la bonne qualité des résultats obtenus. Des exemples de biclusters réels montrent que les résultats générés sont cohérents d'un point de vue médical. La méthode proposée permet donc de répondre au problème de l'intégration des événements temporels dans HBC. Dans la section suivante, nous nous concentrons sur l'autre facette des données temporelles : les séries temporelles irrégulières.

6.2 Gestion des séries temporelles irrégulières

Dans la première section de ce chapitre, nous avons proposé une méthode permettant d'intégrer les événements temporels dans HBC. Dans cette seconde section, nous nous intéressons à une autre forme que peuvent prendre les données temporelles, à savoir les séries temporelles irrégulières. Nous cherchons à intégrer également ce type de données dans HBC. Dans la première partie de cette section, nous présentons les particularités de ce type de données, notamment par rapport aux séries temporelles classiques, ainsi que les techniques d'analyse adaptées à ces données spécifiques. Nous proposons ensuite une nouvelle mesure de similarité entre deux séries temporelles

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

irrégulières. La mesure proposée est utilisée à la fois pour le calcul de qualité des biclusters en sortie de HBC, et pour le pré-traitement de ces séries temporelles par clustering. Nous présentons donc ensuite différentes méthodes de clustering permettant de définir des groupes de séries temporelles similaires. Cela permet de ramener l'ensemble des séries temporelles à un ensemble de valeurs possibles finies, comme la discrétisation pour les valeurs numériques. Nous discutons ensuite de l'utilisation de ces deux éléments, mesure de similarité et clustering, pour l'intégration des séries temporelles irrégulières dans HBC. Enfin, nous présentons des expérimentations permettant de choisir la méthode de clustering la plus adaptée, et de valider l'efficacité de la mesure de similarité proposée par rapport à une autre mesure existante, Dynamic Time Warping [17]. Nous présentons également des résultats pour le clustering des séries temporelles sur un cas d'étude réel, qui montrent la cohérence des clusters générés. Nous concluons cette section par un résumé des différents éléments introduits et discutons de possibles futures améliorations.

6.2.1 Motivations et travaux existants

6.2.1.1 Motivations

Au long de son parcours de soins, un patient peut être amené à subir des mesures et examens répétés dans le temps, par exemple pour relever certaines constantes biologiques (tension, pouls, etc.). Ces données se présentent donc sous la formes de *séries temporelles*, avec une suite ordonnée de k valeurs $V = \{v_1, v_2, \ldots, v_k\}$ et de leurs temps correspondants $T = \{t_1, t_2, \ldots, t_k\}$ avec $\forall t_i \in T : t_{i+1} > t_i$. De nombreuses méthodes ont été développées pour l'analyse de séries temporelles, notamment pour des applications médicales telles que les électrocardiogrammes [167] ou électroencéphalogrammes [79]. Cependant, les données qui nous intéressent plus spécifiquement ici présentent certaines particularités qui les distinguent de ces cas classiques.

La première et plus importante spécificité est leur **irrégularité**. Dans les séries temporelles classiques, l'intervalle de temps entre deux mesures successives et constant, donc $\forall t_i, t_j \in T : t_{i+1} - t_i = t_{j+1} - t_j$. Dans le cas qui nous intéresse ici, cette égalité n'est généralement pas respectée. En effet, les mesures peuvent être réalisées de manière irrégulière, en fonction des besoins de la prise en charge médicale. Nous choisissons donc l'appellation *séries temporelles irrégulières* pour parler de celles-ci. Cette particularité est problématique, étant donné que la régularité entre les mesures est une condition nécessaire à l'application de nombreuses méthodes d'analyse des séries temporelles.

La seconde spécificité concerne la **longueur** de ces séries. Celles-ci sont souvent assez brèves, avec en général moins d'une dizaine de mesures et parfois aussi peu que deux ou trois. Or, la plupart des méthodes d'analyse de séries temporelles supposent implicitement un nombre plus important de mesures dans chaque série. En effet, certaines méthodes visent à retrouver la fonction permettant de décrire au mieux l'évolution d'une mesure, tandis que d'autres transforment les séries par "lissage" (transformée de Fourier, par exemple). Ces méthodes ne sont pas applicables dans le cas de séries de quelques points.

Une dernière spécificité, qui découle des précédentes, est que les séries sont de **longueur variable** ; toutes ne contiennent pas le même nombre de mesures. Les techniques d'analyse de séries temporelles supposent habituellement que les séries soient définies sur le même intervalle de temps et qu'elles possèdent le même nombre d'observations. Ces hypothèses ne sont jamais (ou très rarement) vérifiées dans notre cas.

Nous insistons sur le fait que le problème des séries temporelles irrégulières est différent de celui des séries temporelles. Les méthodes que nous étudions dans ce chapitre sont conçues pour supporter les spécificités mentionnées ci-dessus. Dans le cas où les séries ont un format plus classique et sont mieux formatées, nous recommandons d'appliquer les méthodes très performantes qui ont été développées pour ce problème plus général [73].

© 2017 Tous droits réservés. lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Comme nous l'avons vu dans les chapitres précédents, intégrer un nouveau type de données dans HBC nécessite deux éléments : une mesure de qualité pour un ensemble de valeurs de ce type, et une méthode de simplification des données pour le pré-traitement. Dans la suite de cette section, nous passons en revue les méthodes existantes pour répondre à ces deux problèmes.

6.2.1.2 Travaux existants - similarité

Un premier point à noter ici est qu'il n'existe, à notre connaissance, aucune mesure d'homogénéité pour un ensemble de séries temporelles, au même sens que la mesure de variance pour un ensemble de valeurs numériques par exemple. Par conséquent, nous nous concentrons plutôt ici sur le choix d'une mesure de similarité $d(S_1, S_2)$ entre deux séries S_1 et S_2. Une fois cette mesure choisie, la qualité q d'un ensemble de k séries peut être calculé par la distance moyenne entre deux séries sur tout l'ensemble :

$$q(\{S_1, ..., S_k\}) = \frac{1}{0.5 \times k \times (k-1)} \sum_{i \neq j} d(S_i, S_j) \tag{6.4}$$

De nombreuses mesures ont été développées pour la similarité de deux séries temporelles [54]. Cette étude distingue notamment deux types distincts de mesures :

— les mesures de type *lock-step* calculent la similarité en comparant le point à chaque position i dans la première série avec le point à la même position i dans la seconde série. Cette catégorie inclut les distances de Manhattan et euclidienne (et plus généralement toutes les distances de Minkowski).

— les mesures *élastiques* sont plus flexibles à ce niveau, étant donné qu'elles autorisent différentes formes d'étirement, compression et extension entre les deux séquences, au lieu d'un alignement strict et statique. Cette catégorie inclut les techniques de *Dynamic Time Warping* (DTW) [17] et de *Longest Common Sub-Sequence* (LCSS) [42, 164].

Le problème principal avec les mesures *lock-step* est leur forte sensibilité aux décalages dans le temps des séries comparées. Par exemple, deux séries $S_1 = \{5, 20, 25, 35\}$ et $S_2 = \{20, 25, 35\}$ sont très différentes d'après ces mesures, alors que S_2 est incluse dans S_1. La gestion des décalages temporels étant nécessaire dans le cas qui nous occupe, ces mesures ne conviennent donc pas à nos besoins. A l'inverse, les mesures *élastiques* apparaissent comme mieux adaptées, car conçues pour gérer ces décalages. Cependant, ces mesures présentent plusieurs limitations qui les rendent difficilement applicables. Une première limitation est qu'elles considèrent par défaut des séries temporelles classiques, donc régulières (intervalle de temps constant entre les mesures). Cela constitue un problème majeur pour DTW, qui utilise une fenêtre de temps pour accélérer la comparaison et améliorer les résultats. Dans le cas de séries irrégulières, cette fenêtre risque d'être en permanence mal adaptée à la situation, trop large ou trop étroite. Un autre problème est l'aspect paramétrique de ces méthodes. Compte tenu de la diversité des attributs de type *série temporelle irrégulière*, les paramètres optimaux varient probablement d'un attribut à l'autre. Cela rend le processus de détermination de ces paramètres plus complexe et laborieux. Un problème plus conceptuel tient au fait que ces méthodes ont été développées pour des problèmes très différents de celui qui nous occupe. Bien que l'objectif général reste le même (similarité de séries temporelles), les applications pour lesquelles sont conçues les méthodes influent sur les choix de modélisation. Même les méthodes génériques font le choix de se concentrer sur un sous-ensemble de déformations ou imperfections dans les séries, telles que l'ajustement pour l'amplitude ou le décalage temporel. Par exemple, LCSS est conçue pour supporter les valeurs aberrantes et les variations d'amplitude [164], car ces deux types de bruit sont communs dans les séries temporelles produites par des équipements électroniques. Par conséquent, il ne paraît pas forcément judicieux

© 2017 Tous droits réservés. lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

d'utiliser une mesure existante, car celle-ci a peu de chance de pouvoir s'adapter aux particularités d'un nouveau problème. Concevoir une mesure adaptée au problème qui nous intéresse semble être une meilleure option.

Comme signalé plus haut, nous nous concentrons sur les séries temporelles irrégulières, pour lesquelles aucune méthode dédiée d'analyse n'existe. Le fait que ces séries soient souvent courtes pose également certains problèmes, et invalide nombre de méthodes telles que la décomposition par ondelettes [132], l'analyse spectrale [136] ou les modèles de moyenne mouvante (*moving average* [168]). Ces *séries temporelles courtes* ont néanmoins attiré quelques contributions pour leur analyse. Une première étude intéressante propose un algorithme de classification floue et une mesure de distance entre séries courtes [124]. Cette méthode est bien adaptée à la détection de séries montrant une évolution similaire, mais la mesure de distance proposée est de type *lock-step*, et ne convient donc pas pour des séries potentiellement mal alignées. Une autre étude s'est concentrée sur le problème du clustering de séries courtes pour l'expression génomique [57]. Pour cela, les auteurs utilisent des modèles autorégressifs pour identifier des *profils d'expression* statistiquement significatifs, étendant en quelque sorte un travail précédent [139] pour les séries courtes. Cette méthode produit en général des résultats très cohérents en un temps très bref, et dispose d'une implémentation publique avec interface graphique [56]. Cependant, comme les autres applications conçues pour l'étude des données génomiques, celle-ci ne cherche que les similarités dans les évolutions, et non dans les valeurs. Un autre inconvénient est que cette méthode ne propose pas de mesure de similarité entre séries temporelles, ce qui est indispensable dans notre cas.

6.2.1.3 Travaux existants - clustering

L'algorithme HBC nécessite également, pour chaque attribut, un processus qui transforme chaque valeur originale en une valeur simplifiée, où l'ensemble des valeurs simplifiées doit être fini. Deux types d'approches existent pour simplifier une série temporelle en une valeur unique. La première consiste à réduire la dimensionnalité des séries par une suite de transformations simples. Par exemple, prendre la moyenne de toutes les mesures d'une série, puis appliquer une discrétisation pour obtenir un nombre fini de valeurs possibles. La seconde consiste à grouper les séries temporelles similaires dans des groupes par clustering ; les k clusters ainsi formés étant autant de valeurs simplifiées possibles pour les séries.

La réduction de dimensionnalité peut être réalisée par un ensemble de méthodes, telles que la transformée de Fourier discrète [4], la transformée par ondelettes discrète [30], la décomposition en valeurs singulières [103] ou l'agrégation par parties (*Piecewise Aggregate Approximation* [176]). Toutes ces méthodes partagent les mêmes limitations dans le contexte qui nous intéresse, étant donné qu'elles sont conçues pour des séries temporelles régulières classiques. Par conséquent, elles ne semblent pas applicables à notre problème, à moins de les transformer lourdement.

Le clustering de séries temporelles a suscité un intérêt certain, et de nombreuses études ont été réalisées sur le sujet. Cela s'explique aisément lorsqu'on considère l'omniprésence de ce type de données dans des domaines d'applications réels très variés (biologie, finance, sciences sociales, etc.). Pour la plupart de ces travaux, la contribution principale est la définition d'une mesure de distance entre les séries. Le processus de clustering utilisé est souvent classique : *k-means*, *k-medoids* ou clustering hiérarchique. Pour la définition de cette mesure de distance donc, certains travaux utilisent des outils statistiques tels que les cepstres [93], les distances autorégressives [37], les prédictions de lois de probabilité [7] ou les modèles mixtes [171]. Dans une autre étude, les auteurs utilisent un ensemble de caractéristiques pour décrire chaque série temporelle : périodicité, auto-similarité, asymétrie, etc. Cette approche est intéressante, car elle résout le problème des séries temporelles de longueurs inégales. Chaque série est donc représentée

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

par un vecteur de caractéristiques, et ceux-ci peuvent être regroupés en clusters par clustering hiérarchique [91] ou cartes auto-adaptatives [162]. Une autre étude sur cette même idée utilise des mesures de similarité probabilistes avec une représentation linéaire par parties des séries temporelles [97]. Le problème avec cette approche est que les caractéristiques utilisées pour décrire les séries ne sont pas applicables sur des séries courtes et irrégulières. Une direction de recherche différente proposait de se concentrer sur le clustering de sous-séquences des séries temporelles [130], ce qui est intéressant pour les séries de longueurs différentes. Cependant, l'intérêt de cette approche a été fortement remis en cause et prouvé peu concluant [96]. Une dernière idée consiste à calculer la distance deux-à-deux entre chaque paire de séries de l'ensemble, ici par LCSS (*Longest Common SubSequence*), puis d'appliquer un clustering hiérarchique sur la matrice des distances ainsi calculée [42, 164]. Cette idée est très intéressante dans notre cas, car elle permet d'utiliser la mesure de distance à la fois comme base du clustering et comme mesure de qualité (pour le biclustering en aval).

Cependant, nous constatons d'après cette revue du sujet qu'il n'existe pas réellement de méthode de calcul de similarité et de clustering pour les séries temporelles courtes et irrégulières qui nous intéressent.

6.2.2 Aperçu de la méthode proposée

La Figure 6.5 illustre le mécanisme complet de traitement des séries temporelles dans HBC, et qui contient les étapes présentées ci-dessous. Sur cette figure, les éléments indiqués en rouge sont les nouveaux éléments proposés, qui sont détaillés dans la suite de cette section.

- La **mesure de similarité** permet de calculer la similarité (ou distance), entre deux séries temporelles.
- Le calcul de cette mesure pour toutes les paires de séries donne la **matrice de similarité** pour l'ensemble de ces séries.
- La **méthode de clustering** permet d'obtenir un ensemble de **clusters de séries temporelles** à partir de la matrice de similarité.
- Les clusters auxquels les séries appartiennent forment les **données simplifiées** qui sont passées en entrée de l'algorithme de biclustering **HBC**.
- HBC produit un ensemble de **biclusters sur ces données simplifiées**, qui sont traduits en **biclusters sur les données originales**.
- La **mesure de similarité** permet de calculer la qualité, ou l'homogénéité, des ensembles de séries temporelles contenus dans les biclusters.

6.2.3 Mesure de similarité

La revue des travaux existants présentée dans le Section 6.2.1 révèle le manque d'une mesure de distance adaptée aux séries temporelles irrégulières courtes. Nous définissons ici les caractéristiques souhaitées d'une telle mesure, afin de motiver les choix de conception qui sont faits dans la création de celle-ci par la suite :

- Toutes les valeurs sont importantes : les valeurs considérées dans ces séries irrégulières ne sont pas relevées par des équipements électroniques (ECG, etc.), mais plutôt collectées à la main par le personnel médical (par exemple, prise de tension). Dans ces conditions, nous estimons que toutes les valeurs relevées sont valables et importantes, et qu'aucune valeur aberrante ne doit être présente, qui serait due à un dysfonctionnement matériel par exemple. Les erreurs humaines restent possibles, mais sont assez rares pour ne pas constituer un problème majeur.

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

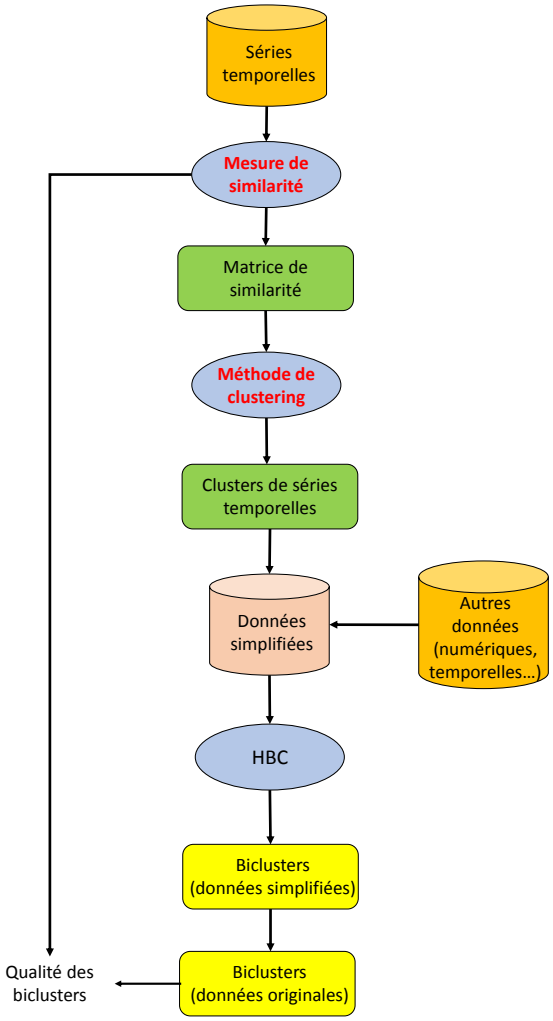


FIGURE 6.5 – Mécanisme de traitement des séries temporelles dans HBC : calcul de la matrice de similarité puis clustering sur cette matrice pour obtenir un ensemble fini de valeurs possibles, et utilisation de la mesure de similarité pour calculer la qualité des biclusters

— Les séries ont des longueurs différentes : nous voulons que la mesure puisse relever si une série correspond à une sous-partie d'une autre.

— Les dates auxquelles les mesures sont relevées sont importantes mais une certaine flexibilité est nécessaire. Comme dans DTW, nous souhaitons que la mesure soit élastique. Cependant, ces temps doivent être utilisés par la mesure afin d'affiner celle-ci. Cela doit permettre, par exemple, de détecter que deux séries présentent une suite de valeurs similaires, mais sur une période de temps plus brève. La mesure doit être capable de déterminer que de telles séries sont similaires.

Compte tenu de ces besoins, nous définissons une nouvelle mesure de distance deux-à-deux adaptée à ce type de séries temporelles. Le calcul de cette mesure pour deux séries S_1 et S_2 est détaillé dans l'Algorithme 11. La distance entre deux points (un point de chaque série) est calculée

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

en utilisant la distance euclidienne sur les deux dimensions de la *valeur* et du *temps*. Nous avons essayé d'utiliser la distance de Manhattan, mais celle-ci donnait des résultats inférieurs à ceux de la distance euclidienne. A noter que les temps sont alignés, de sorte que le premier élément de chaque série soit considéré comme se produisant à $t = 0$ pour cette série. La distance entre deux séries est simplement la somme des distances entre toutes les paires de points constituant les deux séries. Etant donné qu'on considère des séries de longueurs différentes, le problème consiste donc à définir le *matching* entre ces deux séries, à partir duquel est calculée la distance. Un point notable ici est que nous n'utilisons pas un matching *optimal*, mais le matching *minimal* pouvant exister entre deux séries. Cela signifie qu'au lieu de prendre le matching minimisant la somme des distances point à point, nous prenons celui qui maximise cette distance. Cela peut sembler contre-intuitif, mais ce choix a un sens quand on considère qu'une distance réduite pour cette mesure correspond au cas où deux séries ne sont "pas trop dissimilaires". Le choix de cette variante est motivé par les expérimentations présentées dans la Section 6.2.6, où les deux options sont comparées. Cette section présente également d'autres expérimentations, où cette mesure est comparée avec DTW.

Le calcul de cette mesure a une complexité temporelle exponentielle dans sa version naïve, mais un paradigme de programmation dynamique permet de réduire cette complexité à $O(ij)$ pour la première étape, et $O(j(i-j)^2)$ pour la seconde étape, où i et j sont les longueurs des deux séries, respectivement. Il est intéressant de constater que la complexité dépend principalement de la différence de taille entre les deux séries. La complexité de cette mesure est à peu près équivalente à celle de DTW, et demeure raisonnable pour des cas d'application réels. Le traitement des attributs *séries temporelles* est plus coûteux que pour d'autres attributs, mais cela est attendu, étant donné la nature plus complexe de ce type de données.

6.2.4 Clustering de séries temporelles

Nous avons noté précédemment qu'une méthode intéressante consiste à appliquer un processus de clustering sur la matrice de similarité d'un ensemble de séries temporelles. Les algorithmes de clustering du *k-medoids* et du *clustering hiérarchique* sont particulièrement adaptés pour cela, étant donné qu'ils utilisent justement les distances entre les différents éléments pour grouper ceux-ci. Ainsi, nous pouvons voir une synergie évidente avec la mesure de distance développée dans la Section 6.2.3, qui était nécessaire pour le calcul de qualité dans le cadre du biclustering. Par conséquent, nous choisissons d'utiliser la matrice de similarité déjà calculée comme base du clustering des séries temporelles. Le choix qui reste à faire concerne l'algorithme de clustering à utiliser. Nous détaillons dans la suite de cette section les deux options possibles à ce niveau : *k-medoids* ou clustering hiérarchique.

6.2.4.1 k-medoids

Nous avons tout d'abord implémenté l'algorithme du *k-medoids* classique (avec *Partitioning Around Medoids*) [95]. Cet algorithme fonctionne en choisissant au hasard k éléments parmi l'ensemble d'éléments à grouper. Ces k éléments sont appelés *médoïdes*, ou *centroïdes*. Ensuite, les autres éléments sont groupés en fonction du *médoïde* dont ils sont le plus proches, ce qui donne un premier groupement. Par la suite, ce groupement est modifié de manière itérative, en échangeant un médoïde avec un élément non-médoïde. La nouvelle configuration est conservée si son *coût* est inférieur à la précédente. Le *coût* est défini comme la somme des distances entre chaque élément et le médoïde duquel il est le plus proche. L'algorithme se termine lorsque aucune meilleure configuration n'est trouvée.

Après avoir implémenté l'algorithme *k-medoids* original, nous avons développé une variante de celui-ci. Cette variante est décrite en détail dans l'Algorithme 12 ; la partie modifiée par rapport

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Algorithme 11 : Calcul de la distance entre deux séries temporelles S_1 et S_2

```
Data : v1 = valeurs ordonnées pour S1
Data : v2 = valeurs ordonnées pour S2
Data : t1 = temps ordonnés pour S1
Data : t2 = temps ordonnés pour S2
begin
    tailleDiff ← taille(S1) − taille(S2);
    dist ← matrice avec taille(S1) lignes et taille(S2) colonnes;
    for i=0..taille(S1) do
        for j=0..taille(S2) do
            dist[i][j] ← sqrt((v1[i] − v2[j])² + (t1[i] − t2[j])²);
        end
    end
    for j=1..taille(S2)-1 do
        for i=j..j+tailleDiff do
            maxVal ← 0;
            for k=j-1..i-1 do
                if dist[k][j − 1] > maxVal then
                    maxVal ← dist[k][j − 1];
                end
            end
            dist[i][j] ← dist[i][j] + maxVal;
        end
    end
    resultat ← 0;
    for i=taille(S2)-1..taille(S1)-1 do
        if dist[i][taille(S2) − 1] > result then
            resultat ← dist[i][taille(S2) − 1];
        end
    end
    return resultat;
end
```

à la version originale est indiquée en gras. Dans l'algorithme classique, après changement d'un centroïde, seuls les éléments liés à ce médoïde sont réassignés au médoïde le plus proche. Dans la variante que nous proposons, tous les éléments sont réassignés, et le coût total de la nouvelle configuration est recalculé. Nous pensons que cette variante permet de mieux mesurer l'effet des modifications de la configuration. Procéder ainsi est naturellement plus long en termes de temps de calcul. Cependant, diverses expérimentations (présentées plus bas et dans la Section 6.2.6) montrent que cette variante amène à de meilleurs résultats pour le biclustering, qui travaille sur les données préparées par cette phase de clustering.

6.2.4.2 Clustering hiérarchique

L'autre méthode de clustering applicable à une matrice de similarité est le clustering hiérarchique [91]. Deux approches existent pour cette méthode : agglomérative ou divisive. Nous ne considérons pas la seconde option, étant donné que les algorithmes pour celle-ci ont une com-

© 2017 Tous droits réservés. lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Algorithme 12 : Algorithme de clustering *k-medoids* modifié

```
begin
    initialisation : sélection aléatoire de k centroïdes parmi les n éléments;
    associer chaque élément à son centroïde le plus proche;
    while le coût de la configuration diminue do
        foreach centroïde m do
            foreach élément non-médoïde o do
                échanger m et o;
                assigner chaque élément à leur nouveau médoïde;
                calculer le nouveau coût;
                if le coût augmente then
                    annuler l'échange entre m et o;
                end
            end
        end
    end
end
```

plexité (en temps) exponentielle sur le nombre d'éléments à grouper. L'approche agglomérative signifie que chaque élément forme initialement un cluster, et les clusters sont progressivement fusionnés en fonction de leur proximité. Un algorithme naïf pour cette approche a une complexité en $O(n^3)$, ce qui mène à des temps de calculs prohibitifs même pour des jeux de données assez réduits. Cependant, des variantes plus efficaces ont été développées, notamment les algorithmes SLINK [152] et CLINK [48], qui ont des complexités en $O(n^2)$ et $O(n^2 log(n))$, respectivement. Ces algorithmes sont fortement liés à la mesure de distance inter-clusters qu'ils utilisent, qui est le paramètre le plus important pour ces méthodes. Plusieurs options existent pour cette distance, parmi lesquelles les plus fréquemment utilisées sont les suivantes :

— $max\{d(x, y) : x \in A, y \in B\}$: la distance maximale entre deux éléments des clusters A et B. Utilisée dans CLINK, qui signifie *complete-linkage*.

— $min\{d(x, y) : x \in A, y \in B\}$: la distance minimale entre deux éléments des clusters A et B. Utilisée dans SLINK, qui signifie *single-linkage*.

— $\frac{1}{|A|.|B|} \sum_{x \in A} \sum_{y \in B} d(x, y)$: la distance moyenne entre deux éléments des clusters A et B. Parfois appelée *average-linkage*.

Nous avons implémenté SLINK et CLINK pour comparer leurs efficacités respectives sur le problème qui nous intéresse. Des tests préliminaires ont révélé un défaut majeur de l'algorithme SLINK, qui tend à produire des clusters en forme de chaîne, à l'opposé des formes plus sphériques/ovaloïdes auxquelles ont pourrait s'attendre. Ce comportement, souvent appelé *chaining effect*, a été largement observé dans la littérature sur le sujet. Une observation intéressante ici est que nous avons obtenu les meilleurs résultats en utilisant l'algorithme SLINK, mais avec la mesure de distance *complete-linkage* au lieu de *single-linkage*. Ceci est étonnant, étant donné que procéder ainsi ne garantit plus l'optimalité du clustering produit. Nous conservons néanmoins cette variante plus performante pour comparaison plus poussée.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

6.2.4.3 Comparaison

Nous cherchons à évaluer la qualité des clusterings produits par les différentes méthodes existantes. Pour cela, nous comparons plusieurs variantes de clustering hiérarchique et de *k-medoids* selon un ensemble de mesures de qualité pour le clustering (stabilité, distances intra-cluster et inter-clusters, etc.). Ces comparaisons sont détaillées en Annexe A, mais ne permettent pas d'aboutir à une conclusion définitive sur quelle méthode de clustering est la plus adaptée. Par conséquent, nous reportons le choix de la méthode la plus performante après d'autres tests visant à évaluer l'effet de ces méthodes sur le biclustering, présentés dans la Section 6.2.6.

6.2.5 Intégration dans HBC

Dans ce chapitre, nous avons jusqu'à présent développé une mesure de distance pour les séries temporelles irrégulières courtes, et présenté plusieurs méthodes de clustering possibles (le choix de la variante la plus adaptée sera fait d'après les résultats des expérimentations en Section 6.2.6). Il est possible d'appliquer ces outils directement aux séries temporelles afin de pouvoir les traiter comme un nouveau type de données dans HBC. Cependant, ce n'est pas la seule possibilité de modélisation, et la meilleure option à ce niveau dépend des données et de l'application considérées. Par exemple, les utilisations du biclustering en analyse de données génomiques considèrent des séries comme similaires si celles-ci présentent une évolution semblable [57], appelée *profil*. Pour illustrer, prenons deux séries $S_1 = \{10, 16, 12\}$ et $S_2 = \{90, 98, 94\}$. Bien qu'ayant des valeurs très différentes, S_1 et S_2 ont un *profil* très proche : augmentation, puis diminution, avec des variations absolues presque identiques.

Le problème, dans notre cas, est que les séries temporelles pour les données médicales peuvent concerner des mesures très différentes, allant des évaluation psychologiques aux tests d'endurance en passant par les analyses biologiques, entre autres. La modélisation utilisée pour l'analyse de données génomiques (évolution plutôt que valeurs) peut être adéquate pour certaines d'entre elles, quand l'information discriminante est contenue dans l'évolution des mesures. Dans d'autres cas, les valeurs en elles-mêmes sont plus intéressantes que leur évolution. Il est également possible que les deux aspects soient utiles. Déterminer quel aspect est le plus utile pour chaque mesure est une tâche ardue, même pour un expert. De plus, nous souhaitons que les outils proposés soient génériques, afin de pouvoir s'adapter à des cas d'utilisation divers. Par conséquent, nous choisissons d'inclure ces deux aspects, *valeurs* et *évolution*, afin de pouvoir les traiter simultanément. L'idée sous-jacente ici est de chercher des corrélations en utilisant ces deux aspects, sans a priori. Ce choix se traduit en pratique par la définition de deux nouveaux types d'attributs dans HBC :

- *timeseries_values* : les valeurs $v_1, v_2, \ldots, v_n$ prises par la mesure aux temps $t_1, t_2, ..., t_n$.
- *timeseries_evolution* : les évolutions des valeurs $e_1, e_2, \ldots, e_n$, où $e_i = \dfrac{v_{i+1} - v_i}{t_{i+1} - t_i}$, $\forall i \in [1, n-1]$.

Cette formule pour le calcul des vitesses d'évolution, déjà utilisée dans une étude précédente [124], permet de considérer également les temps auxquels sont réalisées les mesures. Cela rend la représentation des données plus fidèle à la réalité, en traduisant les irrégularités dans les séries. A noter que les deux types d'attributs sont séparés, mais les mêmes outils sont utilisés pour les traiter (mesure de qualité et clustering pour le pré-traitement).

6.2.6 Expérimentations

Dans cette section, nous présentons d'abord les résultats d'expérimentations sur jeux de données synthétiques, afin d'évaluer la performance de l'algorithme de biclustering HBC étendu à

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

la gestion des séries temporelles irrégulières. Ces expérimentations nous permettent de déterminer la mesure de distance et la méthode de clustering les mieux adaptées. Ensuite, nous comparons la méthode choisie avec une méthode existante, Dynamic Time Warping [17]. Enfin, nous présentons quelques exemples de résultats sur données réelles, et montrons que ceux-ci sont cohérents d'un point de vue médical.

6.2.6.1 Comparaison sur données synthétiques

Etant donné que les expérimentations réalisées dans la Section 6.2.4 n'ont pas permis d'identifier la méthode la plus efficace pour le clustering de séries temporelles, nous procédons ici différemment. Au lieu d'évaluer la qualité des clusterings, nous nous concentrons sur l'effet que ces clusterings ont sur les résultats finaux du processus de biclustering. En cela, la comparaison se rapproche de celle effectuée dans la partie précédente sur MOCA-I et ses extensions, où nous avons choisi la méthode de discrétisation qui permettait d'obtenir de meilleurs résultats de classification.

Pour cette partie, nous reprenons le protocole expérimental utilisé dans les chapitres précédents. Nous générons plusieurs jeux de données synthétiques (500 lignes, 100 colonnes) et injectons des biclusters dans ces données. Plusieurs variantes de chaque jeu de données sont générées, chacune correspondant à une configuration de paramètres pour ces biclusters : nombre, taille, niveau de bruit, overlapping. Comme précédemment, nous considérons ici les cas avec 5 biclusters injectés et 20% d'overlapping. Parmi les 100 colonnes présentes, 40% correspondent à des séries temporelles, les autres sont numériques, binaires et symboliques. De cette manière, nous restons dans un contexte de biclustering sur données hétérogènes. Une particularité liée aux données ici concerne la méthode de bruitage de celles-ci. Les séries temporelles sont en effet plus complexes que des données numériques par exemple, et il est possible de les altérer sous plusieurs aspects. Nous expérimentons avec deux types de bruit différents :

— première option : les valeurs et les temps d'une série sont bruités, de la même manière que pour les attributs numériques (distribution normale). Le bruit apporté aux temps est cependant limité de sorte que l'ordre des mesures ne soit pas modifié après bruitage. En d'autres termes, la relation $t_{i-1} < t_i < t_{i+1}$, où t_i est le temps associé au *i-ème* point d'une série, doit rester valable après bruitage.

— seconde option : identique à la première, mais la restriction sur les temps est levée. Les mesures peuvent ainsi changer de place dans la série.

Nous nous attendions à ce que la seconde option apporte des modifications plus importantes aux séries, et forme ainsi des cas plus difficiles. Cependant, les résultats ne montrent pas de différence notable de performance pour l'algorithme de biclustering entre ces deux options. Les Tables 6.7 et 6.8 présentent les valeurs moyennes pour la qualité et la taille des biclusters découverts, respectivement. Les Tables 6.9 et 6.10 présentent les résultats pour les deux mesures de redécouverte des biclusters injectés, U et E, respectivement. A noter que pour la Table 6.7, les deux groupes de méthodes (1-2-4 et 3-5) doivent être considérés séparément, car ces méthodes utilisent deux mesures de qualité différentes. Cette séparation n'est pas nécessaire pour les autres tables de résultats, mais nous conservons néanmoins le même ordre des méthodes (1-2-4-3-5) afin de rester cohérents.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

	Méthode 1	Méthode 2	Méthode 4	Méthode 3	Méthode 5
n0s0	0.283 ± 0.215	**0.129 ± 0.097**	0.508 ± 0.182	**0.095 ± 0.094**	0.53 ± 0.166
n0s1	0.186 ± 0.177	**0.082 ± 0.092**	0.328 ± 0.237	**0.073 ± 0.078**	0.339 ± 0.21
n0s2	0.013 ± 0.018	**0.001 ± 0.0**	0.024 ± 0.022	**0.001 ± 0.0**	0.018 ± 0.018
n2s0	0.293 ± 0.162	**0.149 ± 0.091**	0.551 ± 0.151	**0.139 ± 0.075**	0.542 ± 0.148
n2s1	0.248 ± 0.162	**0.135 ± 0.049**	0.364 ± 0.227	**0.115 ± 0.065**	0.373 ± 0.186
n2s2	**0.095 ± 0.037**	0.096 ± 0.053	**0.095 ± 0.07**	**0.086 ± 0.038**	0.135 ± 0.049
n5s0	0.356 ± 0.169	**0.154 ± 0.091**	0.541 ± 0.138	**0.208 ± 0.116**	0.575 ± 0.118
n5s1	0.267 ± 0.122	**0.206 ± 0.076**	0.431 ± 0.166	**0.183 ± 0.075**	0.408 ± 0.166
n5s2	0.093 ± 0.064	**0.056 ± 0.032**	0.141 ± 0.093	**0.132 ± 0.05**	0.186 ± 0.052
n10s0	0.352 ± 0.169	**0.135 ± 0.1**	0.508 ± 0.148	**0.138 ± 0.099**	0.569 ± 0.116
n10s1	0.279 ± 0.18	**0.205 ± 0.098**	0.488 ± 0.153	**0.227 ± 0.096**	0.435 ± 0.146
n10s2	0.081 ± 0.059	**0.041 ± 0.031**	0.203 ± 0.13	**0.079 ± 0.051**	0.18 ± 0.073
n15s0	0.328 ± 0.186	**0.139 ± 0.102**	0.551 ± 0.144	**0.151 ± 0.111**	0.574 ± 0.117
n15s1	0.341 ± 0.188	**0.204 ± 0.109**	0.513 ± 0.178	**0.25 ± 0.119**	0.462 ± 0.133
n15s2	0.097 ± 0.076	**0.045 ± 0.044**	0.269 ± 0.181	**0.044 ± 0.034**	0.181 ± 0.066
n20s0	0.321 ± 0.15	**0.148 ± 0.086**	0.566 ± 0.14	**0.149 ± 0.103**	0.582 ± 0.104
n20s1	0.268 ± 0.123	**0.234 ± 0.126**	0.493 ± 0.114	**0.238 ± 0.12**	0.485 ± 0.135
n20s2	0.169 ± 0.132	**0.077 ± 0.062**	0.319 ± 0.183	**0.06 ± 0.04**	0.199 ± 0.086

TABLE 6.7 – Qualité moyenne des biclusters formés

	Méthode 1	Méthode 2	Méthode 4	Méthode 3	Méthode 5
n0s0	170.1 ± 37.1	159.0 ± 41.1	199.8 ± 51.5	154.5 ± 31.6	**226.2 ± 70.6**
n0s1	415.7 ± 250.0	446.8 ± 257.9	389.2 ± 237.0	**481.5 ± 267.8**	366.0 ± 230.4
n0s2	954.0 ± 564.0	1016.7 ± 270.6	**1045.8 ± 556.8**	838.9 ± 307.1	1025.3 ± 560.8
n2s0	170.8 ± 39.4	158.5 ± 42.1	198.1 ± 43.5	150.8 ± 36.0	**229.3 ± 74.6**
n2s1	393.6 ± 178.2	384.2 ± 162.7	255.3 ± 92.3	**496.5 ± 249.0**	350.9 ± 216.9
n2s2	778.9 ± 474.2	733.4 ± 472.6	642.4 ± 324.0	**1180.9 ± 618.3**	1161.2 ± 573.8
n5s0	182.2 ± 39.5	168.5 ± 43.6	201.4 ± 48.6	159.1 ± 43.6	**230.4 ± 72.6**
n5s1	346.6 ± 138.7	319.2 ± 97.0	232.2 ± 76.3	**429.6 ± 212.0**	347.0 ± 186.2
n5s2	521.8 ± 256.1	447.8 ± 203.3	497.2 ± 258.0	804.2 ± 495.7	**993.7 ± 476.5**
n10s0	177.1 ± 35.7	158.0 ± 46.6	205.5 ± 54.5	157.5 ± 49.2	**238.8 ± 85.5**
n10s1	261.0 ± 113.9	269.3 ± 109.0	210.4 ± 62.0	**338.0 ± 114.7**	316.6 ± 138.5
n10s2	373.6 ± 155.3	304.1 ± 138.3	403.9 ± 189.9	441.1 ± 214.1	**714.6 ± 315.0**
n15s0	168.4 ± 37.8	161.7 ± 44.6	199.2 ± 44.3	163.0 ± 49.5	**247.5 ± 107.6**
n15s1	228.9 ± 97.7	233.1 ± 105.7	205.4 ± 51.9	263.2 ± 87.7	**281.9 ± 118.1**
n15s2	338.6 ± 128.3	277.0 ± 109.3	323.2 ± 144.4	306.5 ± 123.8	**473.8 ± 215.6**
n20s0	172.5 ± 32.3	161.2 ± 45.2	194.5 ± 41.0	157.5 ± 43.2	**228.8 ± 69.3**
n20s1	201.5 ± 68.4	222.0 ± 78.8	196.9 ± 37.6	227.3 ± 70.2	**239.4 ± 83.9**
n20s2	316.0 ± 129.7	268.3 ± 92.7	288.6 ± 149.9	290.2 ± 95.3	**432.4 ± 161.9**

TABLE 6.8 – Taille moyenne des biclusters formés

Ces résultats montrent qu'utiliser un clustering *k-medoids* modifié (méthodes 2 et 3), en prétraitement des séries temporelles, produit des biclusters de qualité notablement meilleure que le clustering hiérarchique (méthodes 4 et 5), pour les deux mesures de qualité considérées. La méthode 5 produit des biclusters en moyenne plus grands que les autres, mais la qualité de ceux-

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

	Méthode 1	Méthode 2	Méthode 4	Méthode 3	Méthode 5
n0s0	0.215 ± 0.392	0.213 ± 0.393	0.031 ± 0.029	**0.8 ± 0.4**	0.204 ± 0.398
n0s1	**1.0 ± 0.0**	**1.0 ± 0.0**	0.988 ± 0.016	**1.0 ± 0.0**	**1.0 ± 0.0**
n0s2	0.965 ± 0.031	**1.0 ± 0.0**	0.974 ± 0.025	**1.0 ± 0.0**	0.974 ± 0.032
n1s0	0.02 ± 0.018	0.215 ± 0.392	0.015 ± 0.031	**0.548 ± 0.435**	0.168 ± 0.316
n1s1	0.9 ± 0.0	0.92 ± 0.046	0.568 ± 0.301	**1.0 ± 0.0**	**1.0 ± 0.0**
n1s2	0.866 ± 0.084	0.807 ± 0.14	0.705 ± 0.106	**0.973 ± 0.053**	0.943 ± 0.069
n2s0	0.015 ± 0.018	0.014 ± 0.019	0.0 ± 0.0	**0.178 ± 0.311**	0.028 ± 0.056
n2s1	0.648 ± 0.111	0.635 ± 0.125	0.106 ± 0.178	**0.94 ± 0.08**	0.82 ± 0.074
n2s2	0.625 ± 0.091	0.579 ± 0.059	0.525 ± 0.102	**0.89 ± 0.072**	0.733 ± 0.119
n3s0	**0.02 ± 0.018**	0.018 ± 0.018	0.01 ± 0.012	0.018 ± 0.022	0.0 ± 0.0
n3s1	0.297 ± 0.244	0.323 ± 0.167	0.009 ± 0.005	**0.712 ± 0.079**	0.504 ± 0.238
n3s2	0.396 ± 0.067	0.351 ± 0.033	0.343 ± 0.058	**0.56 ± 0.124**	**0.56 ± 0.073**
n4s0	0.02 ± 0.016	0.025 ± 0.017	0.015 ± 0.013	0.022 ± 0.02	**0.038 ± 0.027**
n4s1	0.07 ± 0.14	0.076 ± 0.141	0.006 ± 0.003	**0.534 ± 0.288**	0.434 ± 0.258
n4s2	0.31 ± 0.08	0.275 ± 0.027	0.168 ± 0.129	0.348 ± 0.067	**0.428 ± 0.07**
n5s0	0.007 ± 0.011	0.003 ± 0.004	**0.018 ± 0.016**	0.001 ± 0.004	0.012 ± 0.014
n5s1	0.009 ± 0.013	0.006 ± 0.007	0.016 ± 0.008	**0.331 ± 0.279**	0.298 ± 0.155
n5s2	0.238 ± 0.12	0.191 ± 0.087	0.266 ± 0.048	0.287 ± 0.019	**0.389 ± 0.028**

TABLE 6.9 – Mesure de recouvrement U - part du bicluster injecté retrouvée

	Méthode 1	Méthode 2	Méthode 4	Méthode 3	Méthode 5
n0s0	0.669 ± 0.341	0.643 ± 0.321	0.785 ± 0.12	**0.198 ± 0.37**	0.658 ± 0.327
n0s1	0.005 ± 0.0	0.005 ± 0.0	**0.001 ± 0.002**	0.005 ± 0.0	0.003 ± 0.002
n0s2	0.0 ± 0.001	0.002 ± 0.0	0.0 ± 0.001	0.002 ± 0.001	**0.0 ± 0.0**
n1s0	0.82 ± 0.099	0.636 ± 0.318	0.824 ± 0.149	**0.331 ± 0.402**	0.649 ± 0.323
n1s1	**0.0 ± 0.0**	**0.0 ± 0.0**	0.171 ± 0.342	**0.0 ± 0.0**	**0.0 ± 0.0**
n1s2	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**
n2s0	0.827 ± 0.093	0.832 ± 0.091	0.803 ± 0.122	**0.629 ± 0.326**	0.797 ± 0.142
n2s1	0.014 ± 0.017	**0.0 ± 0.0**	0.509 ± 0.279	**0.0 ± 0.0**	**0.0 ± 0.0**
n2s2	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**
n3s0	0.82 ± 0.099	**0.788 ± 0.07**	0.835 ± 0.068	0.813 ± 0.101	0.843 ± 0.053
n3s1	0.384 ± 0.455	0.15 ± 0.3	0.689 ± 0.104	**0.0 ± 0.0**	0.119 ± 0.239
n3s2	**0.0 ± 0.0**	**0.0 ± 0.0**	0.004 ± 0.003	**0.0 ± 0.0**	0.001 ± 0.001
n4s0	0.787 ± 0.085	0.768 ± 0.072	0.808 ± 0.076	0.769 ± 0.083	**0.702 ± 0.088**
n4s1	0.794 ± 0.397	0.606 ± 0.316	0.726 ± 0.031	**0.166 ± 0.328**	0.186 ± 0.307
n4s2	**0.0 ± 0.0**	**0.0 ± 0.0**	0.206 ± 0.241	**0.0 ± 0.0**	0.003 ± 0.006
n5s0	0.841 ± 0.095	0.808 ± 0.137	**0.771 ± 0.052**	0.88 ± 0.078	0.805 ± 0.071
n5s1	0.636 ± 0.133	0.703 ± 0.08	0.65 ± 0.047	0.27 ± 0.334	**0.165 ± 0.288**
n5s2	0.065 ± 0.131	0.072 ± 0.145	0.012 ± 0.008	**0.0 ± 0.0**	**0.0 ± 0.0**

TABLE 6.10 – Mesure de recouvrement E - part de données externes au bicluster injecté

ci est nettement dégradée. Nous observons également que les méthodes 3 et 5, qui partagent la mesure de distance en *matching minimal*, sont les plus efficaces pour la mesure U (part des biclusters injectés retrouvée). A partir de l'ensemble de ces résultats, nous pouvons affirmer que la méthode 3, clustering k-medoids modifié avec *matching minimal*, est la mieux adaptée pour

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

	U		E	
	WM	DTW	WM	DTW
n0s0	**1.0 ± 0.0**	0.274 ± 0.27	**0.013 ± 0.0**	0.288 ± 0.219
n0s1	**1.0 ± 0.0**	0.709 ± 0.095	0.005 ± 0.0	**0.0 ± 0.0**
n0s2	**0.988 ± 0.014**	0.681 ± 0.108	0.001 ± 0.001	**0.0 ± 0.0**
n1s0	**0.96 ± 0.079**	0.272 ± 0.271	**0.008 ± 0.006**	0.348 ± 0.271
n1s1	**0.952 ± 0.037**	0.631 ± 0.136	**0.0 ± 0.0**	**0.0 ± 0.0**
n1s2	**0.893 ± 0.077**	0.557 ± 0.142	**0.0 ± 0.0**	**0.0 ± 0.0**
n2s0	**0.9 ± 0.063**	0.296 ± 0.281	**0.0 ± 0.0**	0.361 ± 0.25
n2s1	**0.911 ± 0.092**	0.506 ± 0.086	**0.0 ± 0.0**	**0.0 ± 0.0**
n2s2	**0.814 ± 0.132**	0.462 ± 0.079	**0.0 ± 0.0**	**0.0 ± 0.0**
n3s0	**0.744 ± 0.078**	0.229 ± 0.208	**0.003 ± 0.007**	0.385 ± 0.25
n3s1	**0.686 ± 0.064**	0.416 ± 0.079	**0.0 ± 0.0**	**0.0 ± 0.0**
n3s2	**0.654 ± 0.027**	0.379 ± 0.098	**0.0 ± 0.0**	**0.0 ± 0.0**
n4s0	**0.744 ± 0.074**	0.107 ± 0.013	**0.006 ± 0.008**	0.579 ± 0.165
n4s1	**0.638 ± 0.021**	0.428 ± 0.093	0.001 ± 0.003	**0.0 ± 0.0**
n4s2	**0.62 ± 0.037**	0.323 ± 0.083	0.002 ± 0.001	**0.0 ± 0.0**
n5s0	**0.704 ± 0.089**	0.107 ± 0.009	**0.01 ± 0.008**	0.65 ± 0.031
n5s1	**0.6 ± 0.0**	0.358 ± 0.105	0.006 ± 0.003	**0.0 ± 0.0**
n5s2	**0.6 ± 0.0**	0.308 ± 0.067	0.003 ± 0.0	**0.0 ± 0.0**

TABLE 6.11 – Recouvrement des biclusters injectés ; comparaison avec DTW

notre problème, que ce soit au niveau de la qualité des biclusters générés ou de son aptitude à retrouver des biclusters cachés.

6.2.6.2 Comparaison avec Dynamic Time Warping

Dans cette section, nous comparons la mesure de distance entre séries temporelles proposée avec une mesure existante, Dynamic Time Warping (DTW) [17]. Etant donné que la qualité des biclusters est par définition incomparable pour des méthodes utilisant une mesure de distance différente, nous ne considérons que leur capacité à retrouver les biclusters injectés dans les données. DTW n'étant pas conçue pour traiter les séries irrégulières, nous simplifions le problème en utilisant des séries régulières pour la comparaison des méthodes. Le processus de génération des données synthétiques est assez similaire à celui de la section précédente. Les seules différences sont donc que les mesures sont espacées uniformément dans le temps, et que le bruitage n'altère que les valeurs et pas les temps. Les séries sont néanmoins toujours courtes (<10 éléments) et de longueurs inégales. La Table 6.11 présente les résultats de la comparaison entre les deux méthodes suivantes :

— WM : mesure de distance entre séries telle qu'introduite dans la Section 6.2.3 avec *matching minimal*, clustering par k-medoids modifié ($k = 5$).

— DTW : mesure de distance entre séries par Dynamic Time Warping, clustering par k-medoids modifié ($k = 5$).

Ces résultats montrent que la mesure de distance proposée améliore notablement la capacité de l'algorithme de biclustering à retrouver les biclusters injectés, et ce pour l'ensemble des variantes considérées. DTW donne des résultats inférieurs, et montre une performance très faible dans les cas où les biclusters injectés sont petits (les cas $nXs0$, avec des biclusters de 20 lignes et 5 colonnes). Une observation intéressante ici est que la qualité des résultats se dégrade moins vite

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

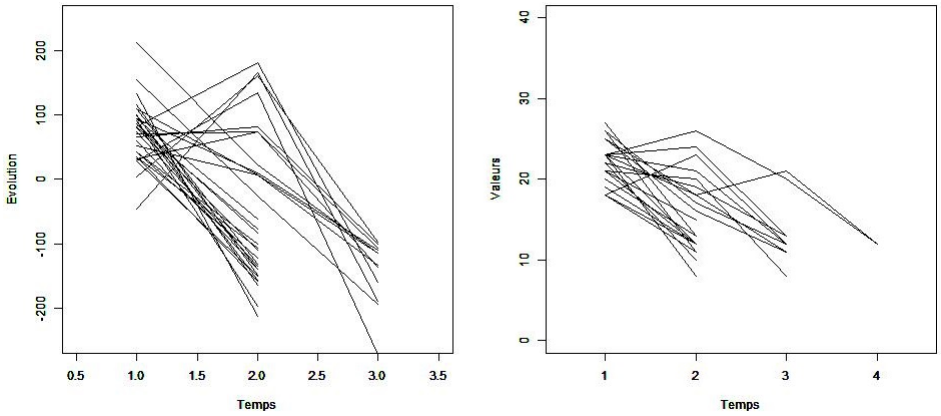


FIGURE 6.6 – Deux exemples de clusters de séries temporelles ; à gauche, un attribut de type *évolution*, à droite un attribut de type *valeurs*

que précédemment à mesure que le bruit augmente, et ce pour les deux méthodes. Cela est probablement dû au fait que seules les valeurs sont bruitées et pas les temps pour ces expérimentations, au contraire des précédentes.

6.2.6.3 Données réelles

Dans cette section, nous montrons quelques exemples de clusters de séries temporelles obtenus sur des données médicales réelles. Les données utilisées ici décrivent divers tests d'effort et questionnaires de qualité de vie pour des patients souffrant de troubles pulmonaires chroniques obstructifs (BPCO ou cancer du poumon, par exemple). Chaque patient subit un ensemble de tests à différentes étapes du processus de réhabilitation ; ces tests servent à évaluer l'efficacité du processus. Une particularité de ces données est que les tests sont administrés aux mêmes moments pour chaque patient : une première fois au début du traitement, puis à la fin, puis six mois et douze mois après la fin du traitement. Toutes ces évaluations ne sont pas toujours menées (pour diverses raisons) ; les séries temporelles dont nous disposons comportent donc entre deux et quatre points. Ainsi, ces séries présentent la plupart des caractéristiques que nous avons considérées dans la création de la méthode : brièveté, longueurs différentes, natures différentes. Bien que techniquement irrégulières (l'intervalle entre deux mesures n'est pas uniforme), ces séries ne sont cependant pas idéales pour illustrer la capacité de la méthode à gérer des séries irrégulières, mais elles demeurent intéressantes et correspondent à un cas d'étude réel. La Figure 6.6 présente deux exemples de clusters de séries temporelles trouvés dans ces données, pour deux attributs différents. Afin de faciliter la visualisation, les mesures sont considérées comme étant relevées à intervalles réguliers sur ces graphes.

Le graphe de gauche représente l'évolution des valeurs (attribut de type *timeseries_evolution*), tandis que celui de droite représente les valeurs elles-mêmes (type *timeseries_values*). Les deux graphes portent sur des attributs et clusters différents. Celui de gauche regroupe des patients qui ont montré une évolution positive au début (environ +100), puis négative après cela (environ -100). Cela correspond à une amélioration au cours du processus de réhabilitation, puis une dégradation une fois celui-ci terminé. Le second graphe regroupe des patients montrant une

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

dégradation générale pour la mesure considérée. Nous pouvons remarquer que chaque cluster tend à grouper des séries ayant la même tendance bien qu'étant de longueurs différentes. Ce comportement est précisément ce que nous souhaitions en conçevant la méthode de gestion des séries temporelles irrégulières.

6.2.7 Discussion

Dans cette section, nous avons défini le problème des séries temporelles irrégulières courtes et montré que les méthodes classiques d'analyse de séries temporelles ne s'appliquent pas à celles-ci. Nous avons introduit une nouvelle mesure pour calculer la distance entre deux telles séries. Des expérimentations ont montré que cette mesure est plus efficace que *Dynamic Time Warping*, qui est souvent utilisée pour les séries irrégulières. Nous avons comparé plusieurs variantes de deux algorithmes de clustering, k-medoids et clustering hiérarchique, et présenté une variante de k-medoids, plus coûteuse en temps mais plus stable. D'autres espérimentations sur données synthétiques ont montré que ce k-medoids modifié est mieux adapté lorsque l'objectif final est d'améliorer la qualité du biclustering qui utilise ces séries temporelles. Les deux éléments introduits dans ce chapitre, mesure de distance et clustering, sont intégrés dans l'algorithme HBC afin d'étendre son champ applicatif aux séries temporelles irrégulières. La modélisation proposée est conçue de manière à prendre en compte les deux aspects des séries temporelles, les deux axes qui permettent de grouper des séries similaires : valeurs proches ou évolution semblable. Ces éléments forment un processus de pré-traitement des séries temporelles pour utilisation dans le biclustering. Ce pré-traitement est plus long que pour des types de données plus simples (numériques, symboliques, etc.), mais les choix d'implémentation proposés permettent de conserver une complexité raisonnable (quadratique) afin de pouvoir s'adapter à la volumétrie de cas d'étude réels. Quelques exemples de clusters de séries temporelles trouvés sur des données réelles viennent illustrer et justifier le travail sur ces données.

6.3 Conclusion

Dans ce chapitre, nous avons étendu HBC afin de pouvoir gérer deux types de données temporelles souvent présentes dans les données hospitalières. Le premier type, appelé événements temporels, est similaire aux données temporelles gérées par MOSC pour la classification supervisée de séquences (voir Chapitre 4). Pour ces événements, nous avons proposé une méthode inspirée de la modélisation introduite par OPSM [16], qui considère l'ordre respectif des événements plutôt que la date à laquelle ils se produisent. Les biclusters ainsi formés contiennent donc un ensemble de séjours ayant une séquence d'événements en commun (dans un ordre identique). Les expérimentations réalisées montrent que HBC ainsi étendu est plus efficace que OPSM sur les données temporelles, et qu'il passe correctement à l'échelle sur des données réelles. Différents exemples de biclusters montrent que HBC est capable de découvrir des biclusters correspondant à des profils de patients variés et cohérents d'un point de vue médical. Le second type concerne les séries temporelles irrégulières, un cas particulier des séries temporelles rarement étudié dans la littérature. Pour celles-ci, nous avons proposé une nouvelle mesure de distance deux-à-deux, et montré que celle-ci est plus adéquate qu'une mesure existante (*Dynamic Time Warping*). Cette mesure est utilisée pour le calcul de qualité, ainsi que pour la simplification de ces séries par clustering *k-medoids*. Nous avons comparé différentes variantes de cette méthode et retenu celle qui donne les meilleures performances. Enfin, nous avons présenté des exemples graphiques de clusters de séries temporelles irrégulières, afin de justifier plus avant la validité de l'approche proposée. L'extension à ces deux types de données temporelles nous permet de traiter par HBC les données réelles des différents cas d'étude.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Chapitre 7

Implémentation - parallélisation et visualisation

Dans les deux chapitres précédents, nous avons proposé une méthode pour le biclustering sur données hétérogènes (HBC), ainsi que des extensions permettant de gérer les données temporelles. Dans ce chapitre, nous nous intéressons à des extensions et améliorations visant à faciliter un déploiement industriel futur de HBC. Dans un premier temps, nous explorons les possibilités de parallélisation de HBC. Dans un second temps, nous discutons des difficultés de la visualisation des biclusters sur données hétérogènes, et développons un outil apportant des réponses à certains de ces problèmes.

7.1 Parallélisation

Dans cette section, nous développons une version parallèle de HBC afin d'en garantir la scalabilité et de pouvoir tirer parti d'environnements de calcul multi-processeurs ou multi-cœurs. Dans une première section, nous motivons plus en détail la conception d'une version parallèle de l'algorithme. Nous proposons ensuite une première approche, utilisant MPI, adaptée aux systèmes à mémoire distribuée, puis une seconde version, utilisant OpenMP, adaptée aux systèmes à mémoire partagée. Nous présentons ensuite des expérimentations sur cette seconde version, qui est plus adaptée aux systèmes dont nous disposons, afin de valider l'accélération apportée par la parallélisation. Nous concluons ce chapitre par une discussion des approches proposées et la présentation de pistes d'amélioration.

7.1.1 Motivations

Dans le contexte qui nous intéresse ici, le parallélisme consiste à répartir le traitement des données sur différents processus exécutés simultanément, afin de réduire le temps d'exécution total. Un *processus* peut être une machine, un cœur ou un *thread* ; cet aspect sera davantage détaillé par la suite. Ce procédé a vu le jour dans les années 60, et fait l'objet de nombreux travaux et applications. En particulier, celui-ci est très utilisé en fouille de données, étant donné les volumes de données importants et les temps de calcul parfois longs que ces volumes impliquent. L'intérêt de *paralléliser* un programme donné dépend de trois aspects majeurs :

— le **temps d'exécution**, ou la **complexité** du programme : la parallélisation est souvent employée pour réduire le temps d'exécution d'un programme qui serait trop long à traiter de manière séquentielle. Cependant, les programmes dont l'exécution est très rapide peuvent également bénéficier de la parallélisation.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

— l'**accélération** qu'il est possible d'atteindre pour ce programme. Cela correspond au ratio entre le temps d'exécution en séquentiel et le temps d'exécution en parallèle du même programme. La loi d'Amdahl montre que l'accélération maximale atteignable (quel que soit le nombre de processus utilisés) est d'autant plus grande que la partie du programme qu'il est possible d'exécuter en parallèle est importante [9].

— la **scalabilité**, ou la capacité du programme à bénéficier de l'ajout de processus supplémentaires pour réduire davantage le temps d'exécution. Cette notion est formalisée dans la loi de Gustafson, qui montre que l'accélération dépend de la proportion du programme qui bénéficie de l'ajout de processus [71].

Nous examinons à présent les caractéristiques de l'algorithme de biclustering HBC, sous les trois aspects définis ci-dessus :

— complexité : HBC a une complexité théorique linéaire mais élevée (cubique sur le nombre de colonnes). Les expérimentations sur données réelles prennent souvent plusieurs heures, ce qui n'est pas problématique pour l'instant, mais pourrait le devenir si les volumes de données venaient à s'accroître. De plus, certaines applications réelles nécessitent une exécution rapide du biclustering. En particulier, la prédiction de codes PMSI implique l'exécution de HBC pour chaque courrier ; le nombre élevé de courriers (plusieurs centaines de milliers) rend la rapidité d'exécution indispensable, notamment pour un traitement des courriers au fil de l'eau par exemple.

— accélération : les seules parties de l'algorithme qui doivent être exécutées de manière séquentielle sont le chargement des données et une partie du post-traitement (clustering des résultats). L'ensemble du mécanisme de création de biclusters, qui représente la partie la plus coûteuse en temps d'exécution, est parallélisable, étant donné que chaque bicluster est généré indépendemment des autres.

— scalabilité : le point le plus important, qui a motivé le développement d'une version parallèle, est la scalabilité remarquable de HBC à l'ajout de processeurs. En effet, la procédure de création d'un bicluster (heuristique constructive) est exécutée de manière indépendante pour chaque attribut et chaque valeur possible des attributs. Les biclusters créés sont simplement ajoutés à une liste, sans aucun besoin d'interaction ou de communication entre les processus. Cela signifie qu'il est théoriquement possible d'atteindre une accélération linéaire sur le nombre de processus utilisés, jusqu'à une limite de P, où :

$$P = \sum_{a \in attributs} |\text{valeurs possibles}(a)| \tag{7.1}$$

Un tel comportement est appelé *embarrassingly parallel* : la programme se prête parfaitement à la conception d'une version parallèle. La suite de cette section détaille deux approches possibles et présente des expérimentations afin vérifier le comportement prévu de HBC en parallèle.

7.1.2 Première approche : MPI

Deux approches sont possibles pour concevoir une implémentation parallèle. La première est MPI (*Message Passing Interface*), qui est un ensemble de normes définissant des bibliothèques de fonctions de communication entre les éléments d'un système distribué. Ces protocoles de communication permettent d'exploiter un système distribué pour du calcul parallèle, en répartissant et synchronisant les opérations effectuées sur chaque machine ou processeur. Plusieurs implémentations de MPI sont disponibles ; nous utilisons ici MPICH, qui est l'une des plus populaires et entièrement libre. Les différences entre les implémentations portent principalement sur la syntaxe ou sur des détails qui sont sans conséquence hormis dans les cas extrêmes (supercalculateurs, parallélisation massive, etc.).

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Une caractéristique majeure de MPI est qu'elle est conçue pour la communication sur des systèmes à mémoire distribuée. Dans ce paradigme, chaque nœud de calcul (processus, cœur, machine...) dispose d'une mémoire propre, et l'accès à la mémoire des autres nœuds nécessite une communication par le réseau. La Figure 7.1 illustre ce type d'architecture, qui modélise les systèmes parallèles utilisant un réseau de machines interconnectées, tels que les grilles de calcul.

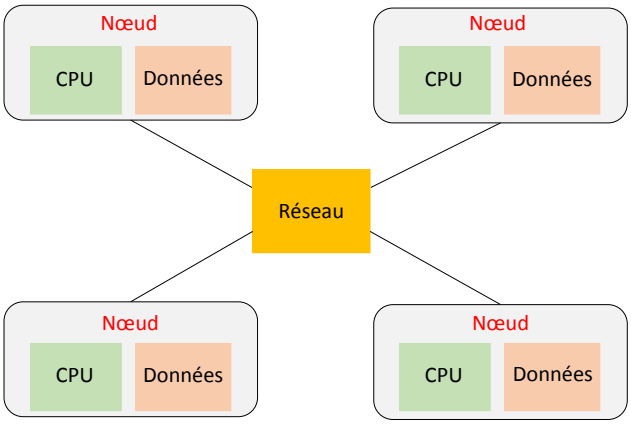


FIGURE 7.1 – Architecture de calcul parallèle à mémoire distribuée

Ce type d'architecture n'est pas particulièrement bien adapté à notre problème. En effet, la création d'un bicluster par HBC nécessite un accès à l'intégralité des données. Par conséquent, dans une architecture à mémoire distribuée, chaque nœud devrait charger dans sa mémoire propre l'ensemble des données. Dans un système à N nœuds de calcul, les données sont donc chargées N fois en mémoire. Cela peut s'avérer problématique lorsque le volume de données et/ou le nombre de nœuds augmente, mais reste une option réaliste dans certains cas (peu de données et/ou de nœuds utilisables). Par exemple, un jeu de données réel regroupant les données PMSI sur deux ans au GHICL occupe environ 600 MB de RAM, ce qui rend peu pratique la duplication massive des données en mémoire. Dans l'approche présentée ici, résumée dans la Figure 7.2, les données sont chargées sur chacun des N nœuds. La parallélisation se fait au niveau de la première boucle de l'algorithme de formation des biclusters : chaque nœud se voit attribuer un ensemble d'attributs et construit les biclusters formés en utilisant chacun de ces attributs comme colonne initiale. L'un des nœuds agit comme processus principal, vers lequel les biclusters construits sont envoyés. Ce processus effectue les opérations de post-traitement (clustering, tri) qui ne peuvent pas être parallélisées.

Afin de remédier au problème posé par la multiplication des données chargées en mémoire, nous proposons une seconde approche utilisant MPI. Ici, les données sont chargées dans un unique nœud. Celui-ci transmet les données aux autres nœuds en fonction de leurs besoins. En pratique, le nœud *maître* (ou *distributeur*) maintient un receveur qui écoute l'ensemble les nœuds *esclaves*. Lors de la réception d'un message, il envoie les données demandées et recrée un receveur afin de répondre à la requête suivante. En fin d'exécution, les nœuds *esclaves* transmettent les biclusters construits au nœud *maître*, qui effectue le post-traitement. La Figure 7.3 présente une vue d'ensemble de cette approche.

Dans la section suivante, nous nous intéressons à un autre type d'architecture parallèle, pour les systèmes à mémoire partagée.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

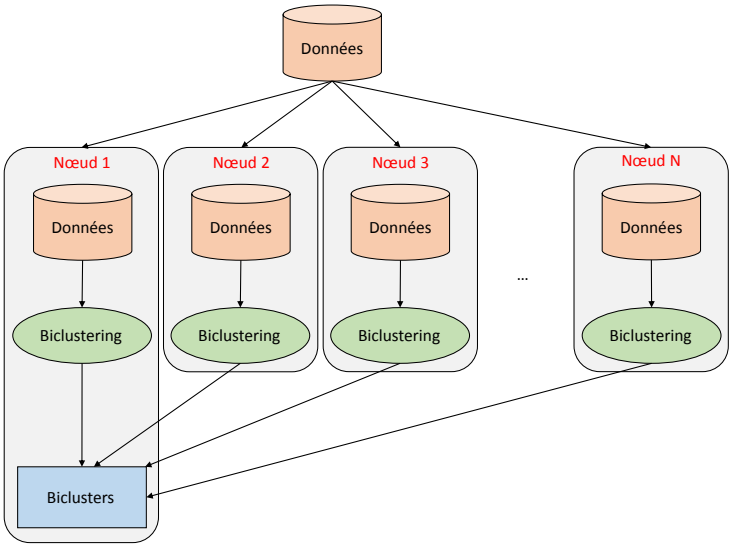


FIGURE 7.2 – HBC sur architecture à mémoire distribuée - première approche

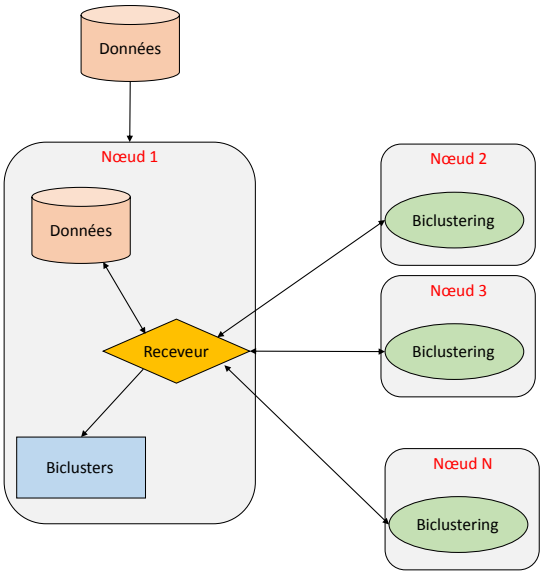


FIGURE 7.3 – HBC sur architecture à mémoire distribuée - seconde approche

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

7.1.3 Seconde approche : OpenMP

OpenMP est une interface de programmation (API) pour le calcul parallèle à mémoire partagée. Elle inclut un ensemble de directives, une bibliothèque logicielle et des variables d'environnement, qui changent le comportement du programme à l'exécution. La caractéristique principale d'OpenMP, qui la différencie de MPI, est le fait qu'elle soit conçue pour les systèmes à mémoire partagée. Cela signifie que les différents nœuds de calcul partagent un même espace mémoire, comme illustré par la Figure 7.4.

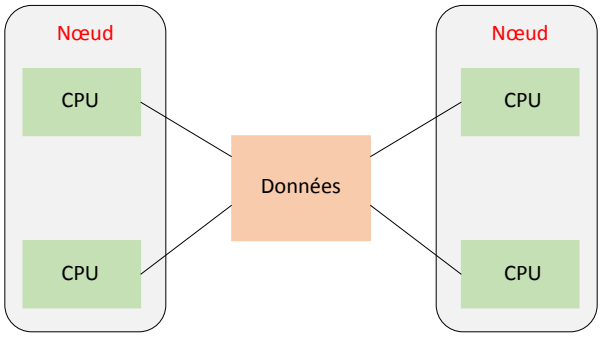


FIGURE 7.4 – Architecture de calcul parallèle à mémoire partagée

De par cette architecture, OpenMP semble mieux adaptée à nos besoins pour la parallélisation : les données sont conservées en mémoire à un endroit unique, et chaque nœud de calcul peut y accéder de manière simultanée. Ce système est d'autant mieux adapté que le processus de formation des biclusters ne modifie jamais la matrice de données, ce qui empêche tout risque de modification concurrente et dispense d'utiliser des mécanismes de synchronisation (sémaphores, etc.). Ainsi, nous pouvons implémenter la version parallèle simple décrite dans la Figure 7.5, où chacun des N nœuds est en charge de générer les biclusters construits à partir de $\frac{|attributs|}{N}$ attributs (comme colonne initiale du bicluster).

7.1.4 Expérimentations

Dans cette section, nous cherchons à déterminer en premier lieu laquelle de ces deux approches est la plus performante pour les environnements de calcul dont nous disposons pour cette étude, et quelle accélération il est possible d'atteindre par parallélisation de HBC.

Cependant, des tests préliminaires montrent que l'approche par MPI présente plusieurs problèmes qui limitent son efficacité. Un premier problème de cette approche est qu'un des nœuds est immobilisé pour servir de distributeur de données, et ne peut donc pas être utilisé pour construire de biclusters. Un second problème, plus important, est qu'en pratique la distribution des données agit comme un goulot d'étranglement qui ralentit considérablement la formation des biclusters sur les nœuds *esclaves*. En effet, ceux-ci envoient des requêtes pour obtenir des données à un rythme très élevé, que le distributeur ne peut pas suivre, même pour un nombre réduit de nœuds (deux ou trois). Ainsi, les expérimentations menées n'ont pas permis de relever une accélération significative en utilisant cette approche par rapport à une version purement séquentielle sur un

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

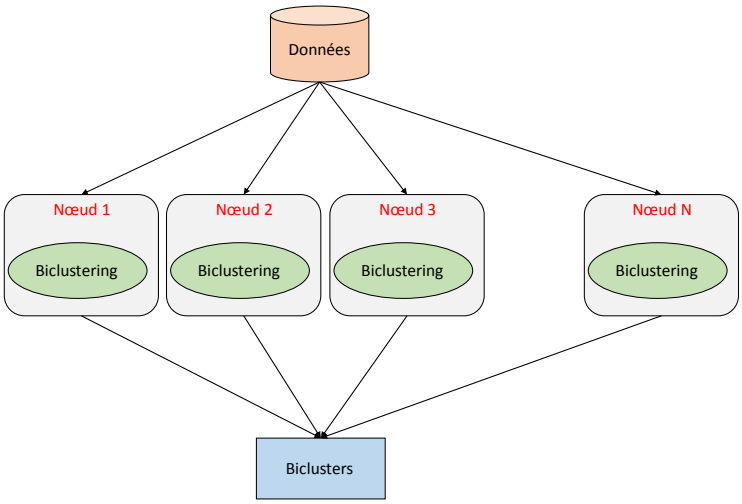


FIGURE 7.5 – HBC sur architecture à mémoire partagée

nœud unique. Il serait sans doute possible d'améliorer les performances de ce système en ajoutant des mécanismes de cache des données et en utilisant une partie de la mémoire disponible sur chaque nœud. Utiliser des techniques similaires à celles pour la répartition de charge sur les serveurs Web, ou pour la maximisation du *throughput* en calcul haute performance, est également envisageable. Il paraît cependant improbable d'atteindre l'accélération attendue, linéaire sur le nombre de nœuds utilisés, surtout lorsque celui-ci augmente.

Compte tenu de ces limitations, nous en déduisons que l'approche par MPI n'est pas adaptée pour cette étude. Pour cette raison, nous nous concentrons sur l'approche par OpenMP. Nous cherchons à valider la prédiction formulée précédemment, selon laquelle il est possible d'atteindre une accélération linéaire sur le nombre de nœuds utilisés dans une implémentation parallèle adéquate.

Pour les expérimentations, nous utilisons quatre jeux de données synthétiques, dont les dimensions sont décrites dans la Table 7.1. Ces jeux de données contiennent exclusivement des données binaires générées aléatoirement. A la différence des expérimentations présentées dans les chapitres précédents, aucun bicluster artificiel n'est injecté dans les données. En effet, nous ne nous intéressons pas à la performance de HBC, qui a déjà été validée, mais uniquement au temps d'exécution. Ainsi, les dimensions de ces jeux de données ont principalement été choisies afin de disposer de temps d'exécutions différents les uns des autres. La variation des différents paramètres permet également de valider (ou non) l'accélération sur différents profils de données.

	Nombre de lignes	Nombre de colonnes	% de données présentes
para1	1 000	10 000	10%
para2	1 000	10 000	25%
para3	10 000	1 000	10%
para4	10 000	1 000	25%

TABLE 7.1 – Informations jeux de données - tests parallélisation

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Nous exécutons HBC parallélisé avec OpenMP sur chaque jeu de données, en utilisant successivement 1, 2, 4 et 8 nœuds. Pour cela, nous utilisons une machine différente que précédemment. Celle-ci dispose de 2 cœurs avec 6 processeurs à 1.9 GHz chacun (donc 12 au total), 32GB de RAM, sous la distribution Arch Linux, avec gcc version 6.3.1-1. La Figure 7.6 présente l'évolution du temps d'exécution pour chaque jeu de données à mesure que le nombre de nœuds utilisés augmente. Les temps d'exécution pour les différents jeux de données étant très différents, le graphique utilise une échelle logarithmique en ordonnée.

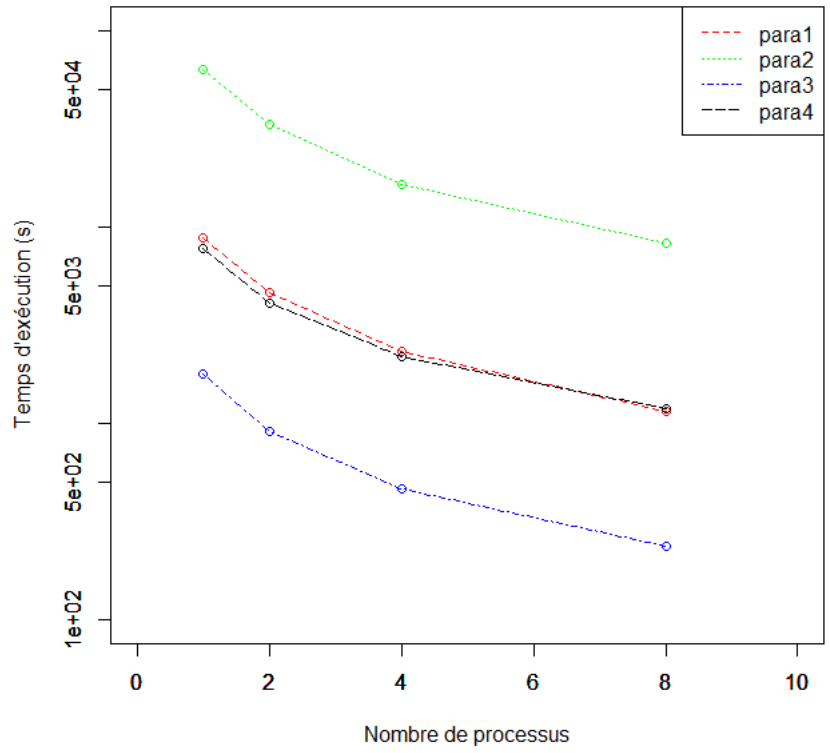


FIGURE 7.6 – HBC parallélisé avec OpenMP - temps d'exécution en fonction du nombre de processus utilisés, pour les quatre jeux de données détaillés dans la Table 7.1. Le temps d'exécution, en ordonnée, utilise une échelle logarithmique

D'après ces expérimentations, nous observons que le temps d'exécution décroît bien de manière linéaire à mesure que le nombre de nœuds augmente. L'implémentation parallèle avec OpenMP permet donc bien d'atteindre l'accélération souhaitée. Nous remarquons également que cette accélération est identique sur tous les jeux de données considérés.

7.1.5 Conclusion sur la parallélisation

Dans cette première section, nous avons exposé les avantages qu'apporte la parallélisation de HBC. Nous avons ensuite proposé deux versions parallèles, adaptées respectivement aux systèmes

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

à mémoire distribuée et à mémoire partagée. La seconde est davantage adaptée aux environnements de calcul disponibles pour cette étude. Nous avons donc mené des expérimentations sur cette version, permettant de mesurer l'accélération apportée par la parallélisation de l'heuristique de formation des biclusters qui forme le cœur de HBC. Les résultats de ces expérimentations montrent une accélération linéaire en fonction du nombre de processeurs utilisés, ce qui confirme les prédictions faites *a priori*.

Il convient d'apporter deux précisions par rapport aux versions parallèles de HBC qui ont été proposées dans ce chapitre. Tout d'abord, MPI et OpenMP peuvent fonctionner ensemble. MPI est conçue pour la communication entre machines (pour simplifier), et OpenMP pour le parallélisme sur les différents cœurs/processeurs d'une machine. L'architecture logicielle la mieux adaptée dépend donc du système disponible pour l'exécution du programme. Dans un environnement classique, avec une seule machine disposant de plusieurs cœurs, OpenMP est idéale. Cependant, pour un environnement de calcul plus conséquent, avec plusieurs machines de plusieurs cœurs chacune, il devient nécessaire de combiner les deux approches. Dans ce cas, les données sont chargées dans la mémoire partagée interne à chaque machine, à laquelle accèdent les différents cœurs de cette machine. La communication entre les machines se fait avec MPI, la répartition au sein de chaque machine se fait avec OpenMP.

Enfin, nous n'avons exposé ici que la parallélisation qu'il est possible de réaliser sur la formation des biclusters. Cette boucle est l'étape la plus coûteuse de HBC, et donc la plus intéressante à paralléliser. Cependant, il est possible d'appliquer le même traitement au pré-traitement des données. Cela n'est pas nécessaire pour les attributs numériques, car la discrétisation n'est pas coûteuse en temps d'exécution. Par contre, effectuer le clustering des séries temporelles irrégulières en parallèle permettrait d'en réduire considérablement le coût, d'autant plus qu'une variante plus complexe de l'algorithme du *k-medoids* est utilisée (voir Section 6.2.4).

7.2 Visualisation

Dans cette seconde section, nous discutons de la visualisation des résultats du biclustering dans le contexte qui nous intéresse : données hétérogènes, temporelles, et de grandes dimensions. Nous exposons tout d'abord en quoi ces spécificités des données posent de nouveaux problèmes pour la visualisation. Nous détaillons ensuite le fonctionnement de l'outil de visualisation développé, puis son intégration au sein d'une application avec une interface utilisateur. Enfin, nous concluons en présentant les perspectives d'améliorations pour l'application et les difficultés restant à résoudre pour la visualisation.

7.2.1 Motivations et difficultés

De nombreux logiciels de biclustering proposent un outil de visualisation des résultats. Par exemple, la Figure 7.7 montre un exemple de visualisation générée par le logiciel BicAT [14] utilisant l'algorithme OPSM [16], et la Figure 7.8 un résultat pour l'outil développé pour l'algorithme ISA [83].

Pour les données supportées par ces algorithmes, la visualisation des biclusters consiste en deux étapes assez simples :

— Réorganiser les lignes et les colonnes afin de former des blocs homogènes correspondant aux biclusters.

— Définir un code couleur pour les valeurs de la matrice de données, afin de mettre en exergue l'homogénéité des valeurs au sein de chaque bicluster par rapport au reste des données. Par exemple, l'outil ISA utilise une échelle de couleurs pour les valeurs numériques (voir Figure 7.8).

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

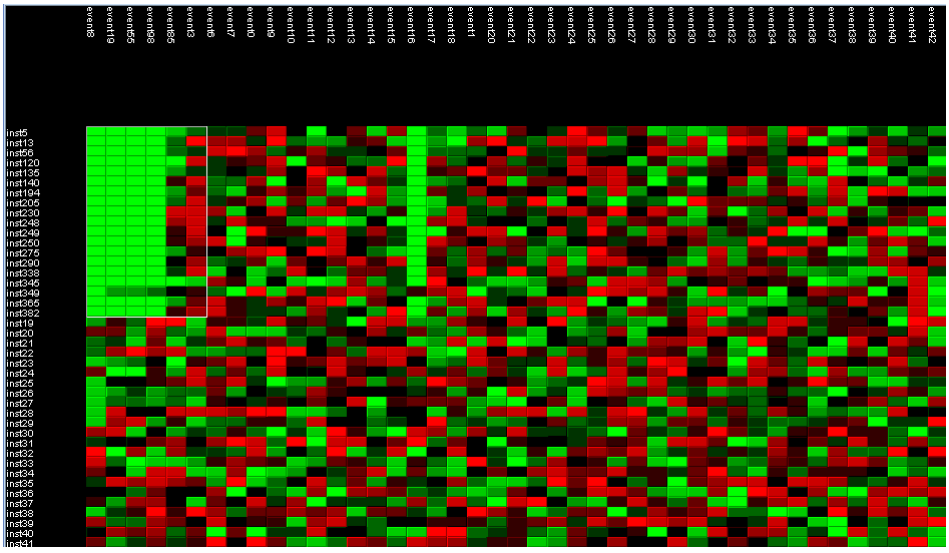


FIGURE 7.7 – Représentation visuelle d'un bicluster par l'outil BicAT, en haut à gauche de l'écran

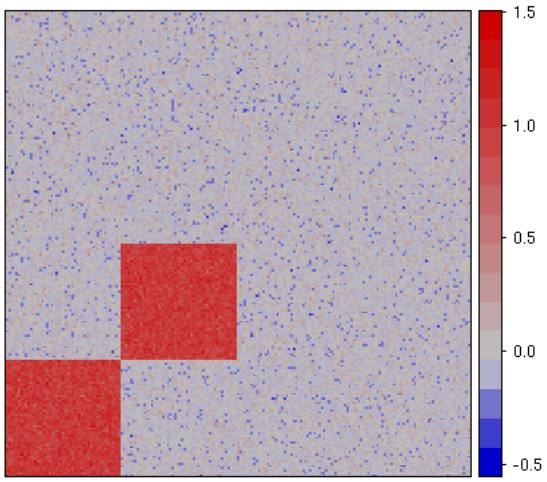


FIGURE 7.8 – Représentation visuelle de deux biclusters par l'outil ISA, dans une structure en damier

Cependant, les données qui nous intéressent dans le contexte de cette thèse posent plusieurs problèmes pour ces deux étapes :

— Tout d'abord, le volume des données pose un problème évident pour la visualisation. En effet, il est difficile de représenter l'ensemble des données si celles-ci comportent plusieurs dizaines de milliers de lignes ou de colonnes. De plus, pour de telles dimensions, représenter des biclusters comparativement beaucoup plus petits que la matrice de données (par

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

exemple, quelques dizaines de lignes/colonnes) s'avère compliqué en raison des proportions.

— Un deuxième problème majeur concerne l'hétérogénéité des données. Les outils classiques de biclustering considèrent des données homogènes, ce qui permet de définir une échelle de couleurs identique pour l'ensemble de la matrice, comme illustré par les Figures 7.7 et 7.8. Procéder ainsi est impossible pour les données hétérogènes, où chaque attribut couvre une plage de valeurs différente.

— Représenter les événements temporels pose également problème, en particulier compte tenu de l'utilisation qui en est faite dans HBC. En effet, HBC se concentre sur la séquentialité des événements et donc leur ordre relatif, ce qui implique une transformation des données par rapport aux valeurs de base (décrivant le temps d'occurrence des événements). Ce problème est décrit plus en détail dans la Section 7.2.2.

— Enfin, un dernier problème est que HBC autorise un recouvrement complet (sur lignes et colonnes) entre les biclusters découverts. La plupart des outils de biclustering, tels que ISA (voir Figure 7.8), recherchent des structures en damier dans les données, ce qui rend le réarrangement des lignes et colonnes évident. Ce n'est pas le cas pour HBC, où les biclusters peuvent se recouvrir sans contraintes. Dans ce contexte, il est impossible, sauf cas particuliers, de réordonner les lignes et les colonnes de telle sorte que chaque bicluster soit constitué d'un seul bloc, dès qu'on considère plus de deux biclusters à représenter simultanément.

7.2.2 Deux modes de visualisation : vue globale et vues détaillées

Nous présentons ici une méthode de visualisation prenant en compte les difficultés identifiées. Cette méthode n'apporte pas de solution définitive à tous ces problèmes, mais permet d'obtenir des représentations satisfaisantes compte tenu des conditions. L'outil développé prend en entrée le fichier de données et le fichier de résultats pour le biclustering, et produit en sortie des fichiers au format SVG (Scalable Vector Graphics) représentant les vues. Le format SVG permet d'obtenir des images scalables, faciles à créer car syntaxiquement proches du XML, et bien adaptées à une intégration dans des interfaces Web.

Un premier choix pour la représentation consiste à identifier deux vues différentes pour les biclusters générés par HBC :

— La **vue globale** vise à représenter l'ensemble des biclusters sur une même figure afin de montrer les recouvrements entre eux et la couverture de la matrice des données. Cette vue ne considère pas les valeurs, uniquement les cellules de la matrice et leur couverture par les biclusters.

— Pour chaque bicluster, une **vue détaillée** représente ce bicluster par rapport au reste des données, en considérant les valeurs, afin de montrer son homogénéité.

Procéder de la sorte permet de séparer le problème de recouvrement entre biclusters des autres problèmes. L'objectif de ce choix est de disposer d'une vue générale ressemblant à celle de la Figure 7.9 (reprise de la Figure 5.3 de la Section 5.1.2), et de vues détaillées similaires à celles produites par BicAT (voir Figure 7.7).

7.2.2.1 Vue globale

Nous présentons d'abord la vue globale, qui est la plus simple conceptuellement. Les deux problèmes qui se posent pour cette vue sont la taille des données et la représentation de biclusters avec recouvrement sur lignes et colonnes. Pour répondre au premier problème, nous choisissons

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

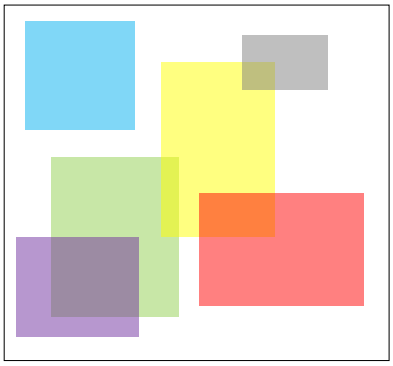


FIGURE 7.9 – Représentation souhaitée pour la vue générale : chaque bicluster est représenté par une couleur afin de visualiser la couverture de la matrice ; les valeurs sont ignorées

de ne représenter que les lignes et colonnes incluses dans au moins un bicluster. De cette manière, la vue globale vise surtout à représenter le recouvrement entre les biclusters. Le second problème paraît impossible à résoudre totalement ; comme mentionné précédemment, il n'est pas possible de représenter plus de deux biclusters simultanément sous forme de blocs uniques. Afin d'atténuer cela, le système proposé examine les lignes et les colonnes de l'ensemble des biclusters, afin d'identifier celles qui ne sont couvertes que par un seul bicluster (appelées L_1 et C_1 respectivement). Après cela, les lignes et colonnes sont réarrangées en groupant les L_1 et C_1 de manière contigue, afin d'obtenir une vue partiellement organisée sous forme d'une structure en damier, ce qui en améliore la lisibilité. Chaque bicluster est représenté par une couleur différente. La Figure 7.10 présente un exemple de vue globale, obtenue sur les données réelles déjà utilisées dans la Section 6.2.6 du chapitre précédent.

La vue est séparée en quatre parties distinctes découlant du processus de réorganisation des lignes et colonnes décrit ci-dessus. Ces zones sont délimitées par des lignes noires pour davantage de lisibilité :

1. Les cellules de la matrice de données pour lesquelles les lignes et les colonnes sont couvertes par plus d'un bicluster.
2. Les cellules pour lesquelles les lignes sont couvertes par plus d'un bicluster, mais les colonnes font partie de C_1 (couvertes par un seul bicluster).
3. Les cellules pour lesquelles les colonnes sont couvertes par plus d'un bicluster, mais les lignes font partie de L_1 (couvertes par un seul bicluster).
4. Les cellules pour lesquelles les lignes font partie de L_1 et les colonnes font partie de C_1, correspondant à une structure en damier.

7.2.2.2 Vues détaillées

L'outil de visualisation génère autant de vues détaillées (fichiers SVG) qu'il y a de biclusters (trouvés par HBC). Les problèmes afférents à cette vue sont le volume des données, leur hétérogénéité, et leur temporalité pour certains attributs.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

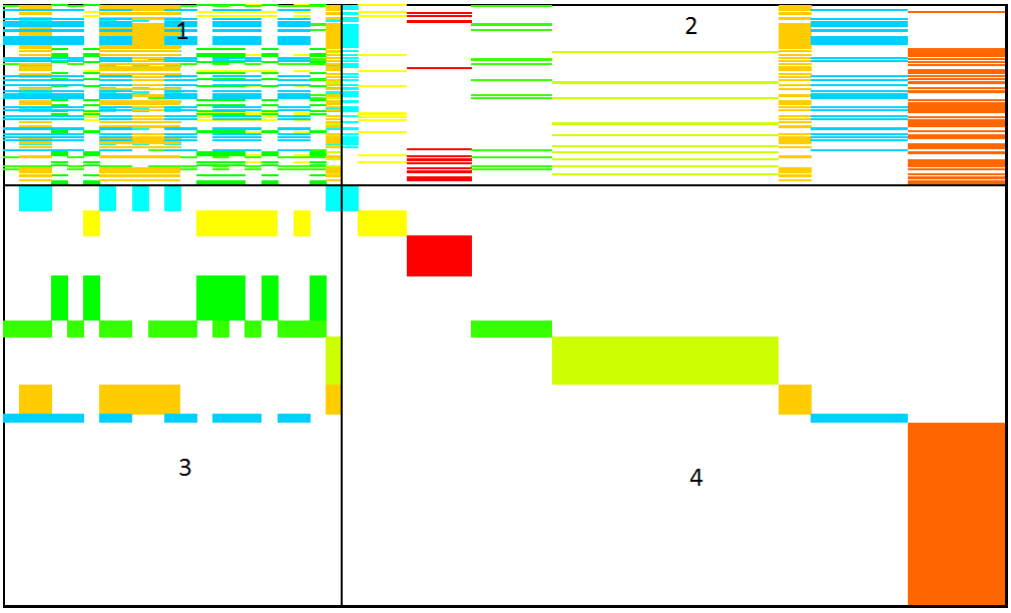


FIGURE 7.10 – Vue générale : représentation de l'ensemble des biclusters et des recouvrements entre eux. Chaque couleur représente un bicluster. Les quatre parties (1, 2, 3, 4) sont détaillées dans la Section 7.2.2.1

Pour ce qui est du volume, nous faisons le choix de représenter l'intégralité du bicluster, ainsi qu'une partie du reste de la matrice de données, de sorte que les lignes et colonnes du bicluster représentent au moins $p\%$ des lignes et colonnes de la vue. Par exemple, pour un bicluster de 100 lignes et 20 colonnes extrait d'une matrice à 15 000 lignes et 1 200 colonnes, si on fixe $p = 20\%$, la vue contient 500 lignes (dont $p = 20\%$ appartenant au bicluster, soit 100 lignes) et 100 colonnes (dont $p = 20\%$ appartenant au bicluster, soit 20 colonnes).

L'hétérogénéité des données pose problème car elle empêche d'utiliser une même échelle de couleurs représentant les valeurs pour l'ensemble des données de la matrice. Afin de contourner cette difficulté, l'outil de visualisation définit une échelle de couleurs pour chaque attribut en fonction de son type :

- Pour les attributs binaires, deux couleurs pré-définies sont utilisées pour les valeurs 0 et 1.
- Pour les attributs symboliques, un ensemble de couleurs différentes, une pour chaque valeur possible, sont utilisées.
- Pour les attributs séries temporelles et symboliques, l'outil utilise un fichier annexe indiquant le clustering de ces séries. Cela permet de ramener ces attributs à des attributs symboliques et d'utiliser le même traitement que pour ceux-ci, avec une couleur par cluster de séries.
- Pour les attributs numériques, l'outil définit une échelle de couleurs basée sur le spectre des couleurs RGB (du rouge au vert puis au bleu). Cette échelle est conçue de sorte que la moyenne des valeurs présentes dans le bicluster pour cet attribut corresponde à la couleur vert (code RGB #00FF00 en hexadécimal). Ce choix est fait afin d'avoir un point de repère similaire pour les attributs numériques présents dans le bicluster. De cette manière, il est attendu que les valeurs numériques du bicluster soient toutes représentées dans des

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

couleurs très proches du vert, ce qui permet de renforcer visuellement l'homogénéité des valeurs. Pour ce qui est des attributs numériques en dehors du bicluster, l'échelle de couleurs utilise l'intégralité du spectre RGB, où le rouge (#FF0000) représente la valeur minimale et le bleu (#0000FF) la valeur maximale.

La Figure 7.11 présente un exemple de bicluster, positionné en haut à gauche de la matrice globale, trouvé sur les mêmes données réelles que précédemment. Chaque colonne représente un attribut dont le nom est indiqué au-dessus de la colonne. Pour des raisons de confidentialité, les attributs ont cependant été remplacés par des chaînes de caractères aléatoires. La représentation obtenue permet d'évaluer visuellement la cohérence du bicluster. Une option de l'outil permet de choisir les attributs à afficher en priorité en plus des attributs du bicluster. Cette option a été utilisée sur la Figure 7.11 afin d'afficher les attributs ayant le plus de valeurs présentes. Les valeurs manquantes sont représentées en blanc sur les vues détaillées.

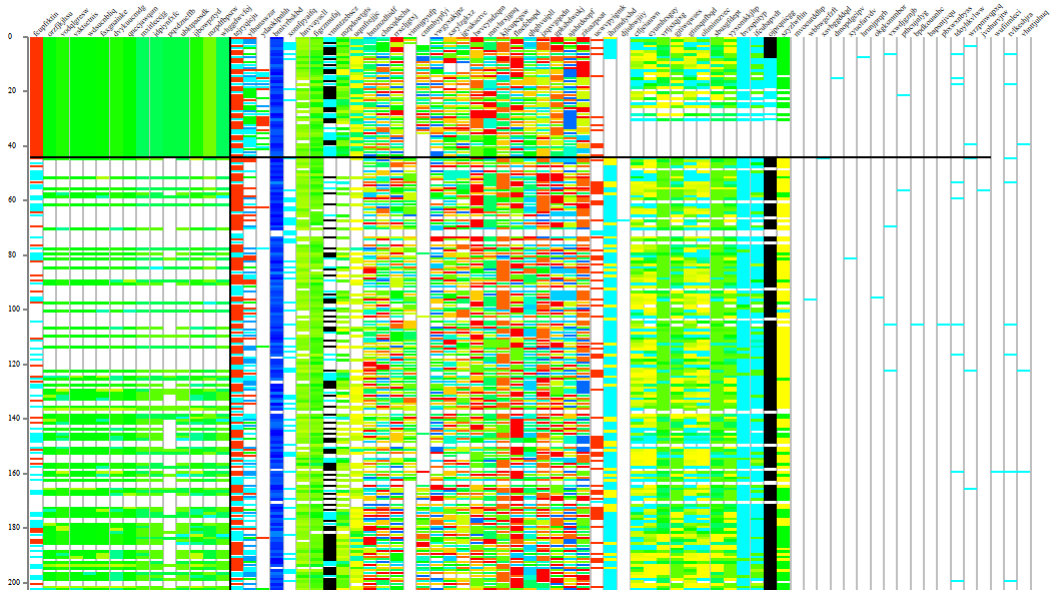


FIGURE 7.11 – Vue détaillée pour un bicluster, situé en haut à gauche et délimité par les lignes noires. Le bicluster contient 44 instances et 15 attributs, dont 14 numériques et 1 symbolique ; les couleurs représentent les valeurs de la matrice de données

Le dernier problème concerne la temporalité propre à certains attributs. Afin d'y répondre, le système développé considère l'ordre respectif des événements temporels et représente cet ordre par des nuances d'une couleur (du rouge clair au rouge foncé). La spécificité des données temporelles est qu'il convient d'examiner l'ensemble des attributs temporels afin d'évaluer la cohérence d'un bicluster, et non plus chaque colonne individuellement comme pour les autres types de données. La Figure 7.12 présente un exemple de bicluster contenant 6 instances et 4 attributs (sur données synthétiques). Parmi ces attributs, 3 sont temporels. Le système de représentation assure que les événements se produisant dans le même ordre sont représentés par la même suite de nuances de couleurs. Nous présentons un exemple sur données synthétiques, car la gestion des attributs temporels demeure problématique pour les cas réels, car le grand nombre d'attributs implique des nuances de couleurs très proches les unes des autres en raison de l'implémentation choisie.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

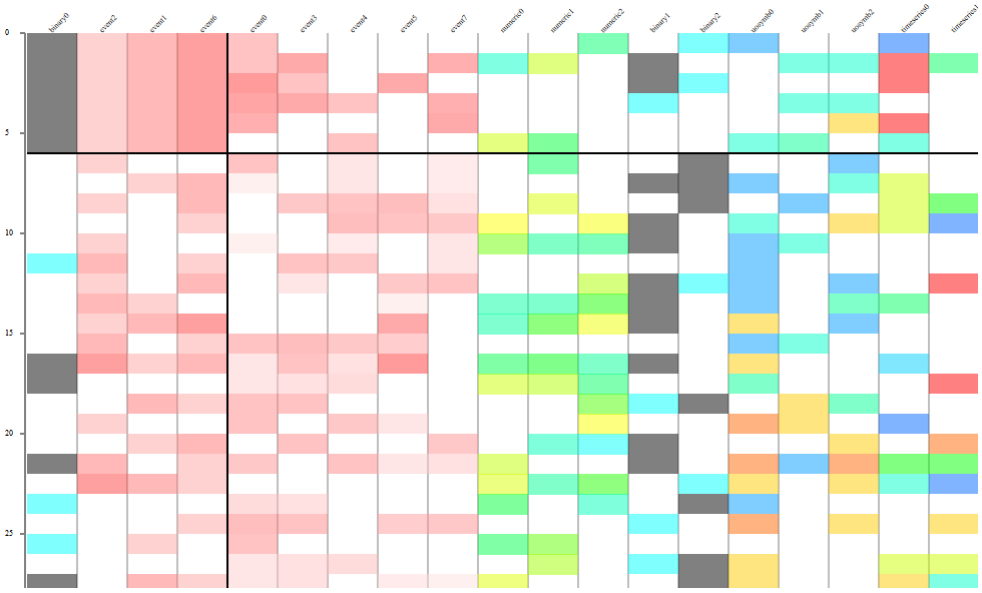


FIGURE 7.12 – Vue détaillée pour un bicluster contenant 1 attribut binaire et 3 attributs temporels. Ceux-ci sont groupés ensemble et dans leur ordre d'apparition afin de faciliter la visualisation

Une dernière fonctionnalité de l'outil développé est la possibilité de représenter la vue détaillée pour deux biclusters simultanément. Il est en effet toujours possible de représenter deux biclusters sous forme de blocs uniques en réorganisant les lignes et colonnes, quel que soit le niveau de recouvrement entre ces biclusters. La Figure 7.13 montre un exemple de vue détaillée pour deux biclusters, sur les données réelles utilisées précédemment et anonymisées.

Le fonctionnement de cet outil est présenté dans la Figure 7.14. L'outil, appelé VisuHBC, prend en entrée les fichiers de données et de résultats de HBC, ainsi que le fichier de clustering des séries temporelles et symboliques si les données en contiennent. Il crée un fichier SVG pour la vue globale, un fichier SVG par bicluster pour les vues détaillées, ainsi qu'un fichier SVG contenant les légendes, c'est-à-dire la signification des échelles de couleurs pour chaque attribut, pour utilisation dans l'application qui est décrite dans la suite de cette section.

7.2.3 Intégration dans une application

L'outil de visualisation VisuHBC a été intégré au sein d'une application fonctionnelle, afin de faciliter la présentation des résultats du biclustering et d'anticiper sur le travail futur d'industrialisation des outils développés lors de la thèse. La conception et le développement de cette application ont été réalisés en collaboration avec deux étudiants de Master 2 Informatique "E-services" de l'Université Lille 1, El Mehdi Jardani et Youssef Id Taleb. L'application reprend les standards de développement utilisés par la société Alicante, afin de favoriser la poursuite de son développement en interne : langage Java, frameworks Spring et JSF, gestion des bases de données par PostgreSQL.

La Figure 7.15 présente l'état d'avancement actuel de l'interface. Avant d'arriver sur cette interface, une page permet de charger les fichiers nécessaires à la visualisation (données, biclusters et fichiers SVG). Le périmètre fonctionnel de l'application est le suivant :

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

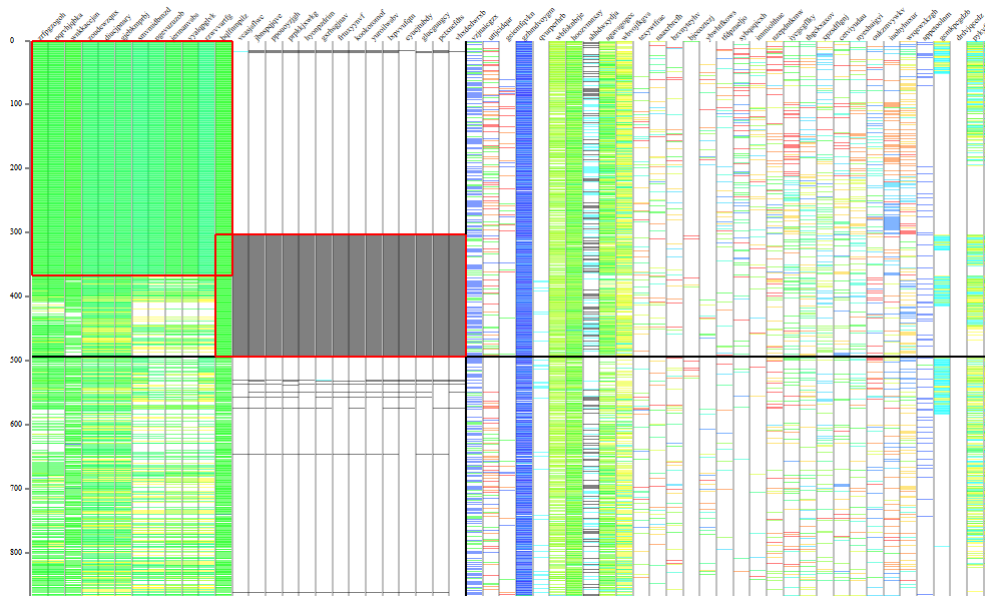


FIGURE 7.13 – Vue détaillée pour deux biclusters avec recouvrement, délimités par des lignes rouges (incluses dans la vue)

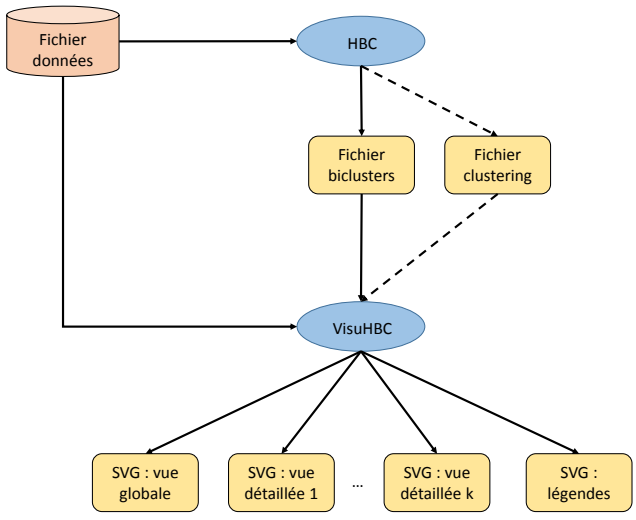


FIGURE 7.14 – Fonctionnement de l'outil de visualisation VisuHBC : fichiers d'entrée et de sortie. L'outil génère k vues détaillées, correspondant aux k biclusters découverts par HBC

— Affichage d'informations générales sur le jeu de données (en haut à droite) : nombre de lignes et de colonnes, nombre de biclusters.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

— Possibilité de changer l'affichage entre vue globale et vue détaillée (en haut à gauche). Une liste déroulante permet de sélectionner le bicluster à afficher pour la vue détaillée.

— Affichage d'informations sur le bicluster affiché actuellement : nombre de lignes et de colonnes, mesure de qualité, liste et type des attributs.

— Affichage de statistiques descriptives. Ce module n'est pas finalisé et n'est donc pas montré sur la figure. Ces statistiques décrivent, pour chaque attribut du bicluster affiché, la distribution des valeurs dans le bicluster par rapport à la distribution sur l'ensemble des valeurs. Le module utilise les fonctionnalités de la librairie PrimeFaces pour la création des graphiques. La légende décrivant la signification de l'échelle de couleurs pour chaque attribut est également affichée à cet endroit (en-dessous de la vue présentée dans la Figure 7.15).

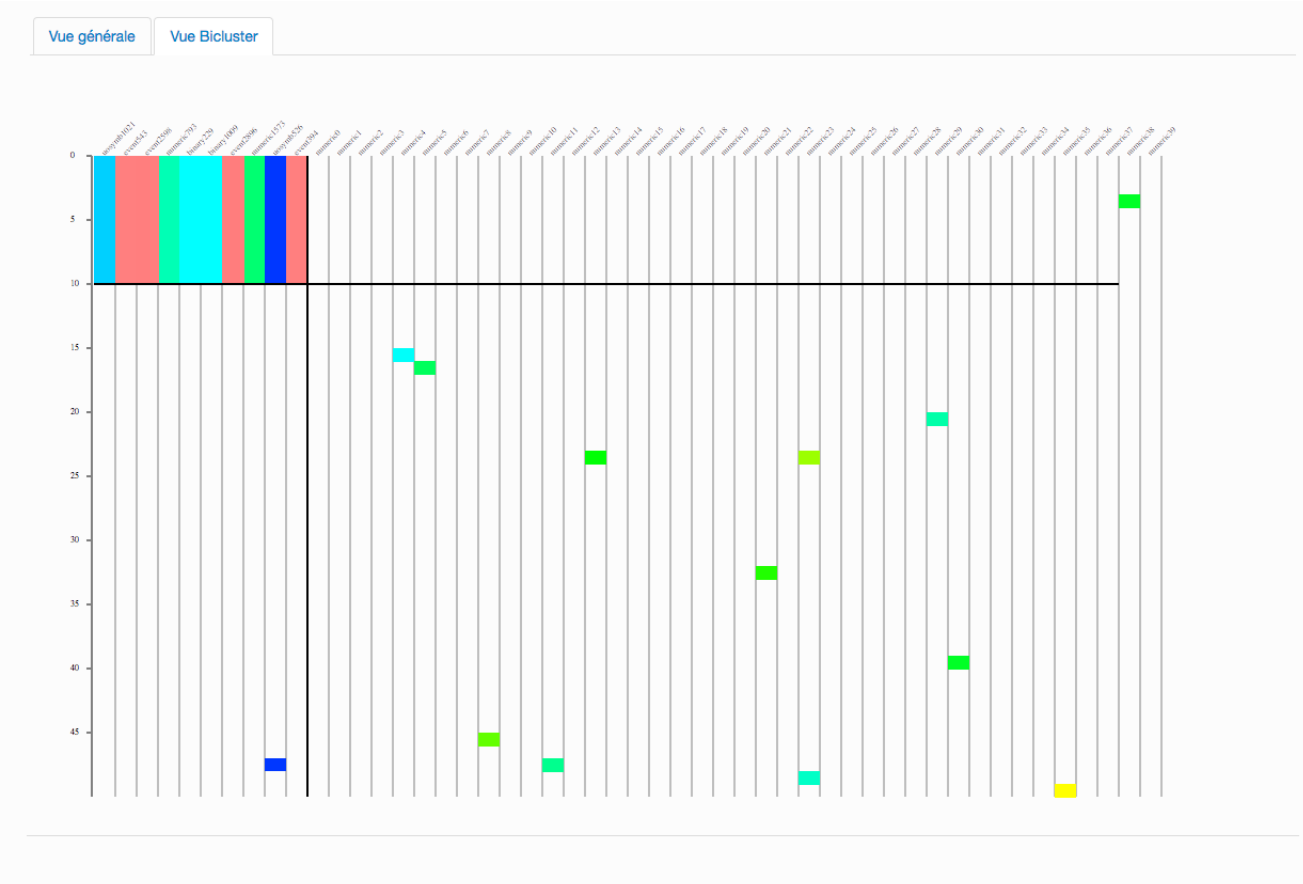


FIGURE 7.15 – Interface de l'application VisuHBC, affichant la vue détaillée pour un bicluster

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

7.2.4 Perspectives

L'approche proposée ici permet de surmonter en partie les problèmes liés à la visualisation des biclusters sur le type de données qui nous intéressent dans cette thèse. Les représentations générées sont cohérentes et permettent de mieux comprendre les résultats que les fichiers de résultats bruts produits en sortie de HBC. Cependant, de nombreuses pistes d'amélioration restent à explorer, tant pour la génération des vues que pour l'application liée.

Pour ce qui est des vues, le problème principal qui reste à surmonter concerne la représentation des événements temporels dans les vues détaillées, et notamment la scalabilité du processus à des données contenant plusieurs milliers d'attributs de ce type. D'autres perfectionnements moins importants à apporter concernent l'uniformisation des couleurs utilisées pour les différents types de données, et la correction de défauts de *clipping* entre les cellules. Pour la vue globale, poursuivre la réflexion sur une représentation plus adaptée aux besoins peut conduire à de nouvelles solutions qui évitent l'aspect chaotique provoqué par le recouvrement des biclusters.

Pour ce qui est de l'application, le développement le plus important concerne l'intégration de l'outil de génération des vues au sein de cette application. Cela permettrait de simplifier le processus de visualisation en utilisant directement les résultats de HBC pour générer et afficher les représentations de ces résultats. Cela permettrait également de générer dynamiquement des vues à partir de l'interface, en réglant les paramètres en fonction des besoins de l'utilisateur, par exemple une vue détaillée avec le recouvrement entre deux biclusters. Le fonctionnement souhaité est exposé dans la Figure 7.16 : les données et résultats de HBC sont chargés directement dans l'application, ce qui évite entre autres la gestion manuelle des fichiers SVG. Un développement plus ambitieux concerne la communication entre l'application et HBC afin d'intégrer toutes les étapes du biclustering dans un même processus.

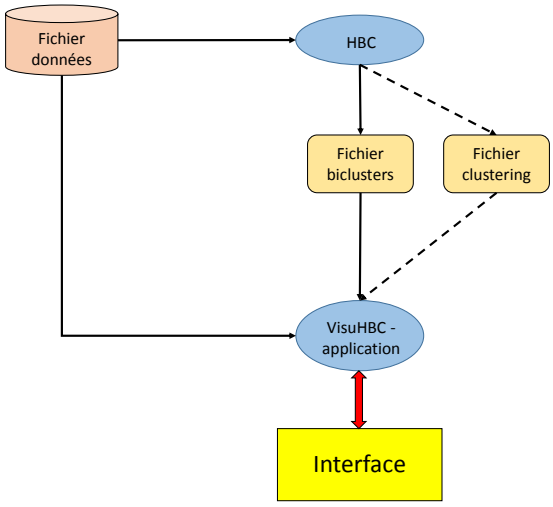


FIGURE 7.16 – Fonctionnement souhaité pour l'application VisuHBC : intégration de l'outil de génération de vues et de l'interface de visualisation

Plusieurs autres améliorations de l'application seraient également intéressantes à apporter.

— En premier lieu, le finalisation des graphiques décrivant la distribution des données pour chaque attribut, en particulier la conception d'un graphique adapté aux données temporelles, et la gestion des séries temporelles et symboliques.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

— Ensuite, l'interactivité de la vue. Pour l'instant, il s'agit d'un SVG simple intégré dans l'interface. Rendre cette image interactive, par exemple en affichant la légende d'un attribut quand l'utilisateur clique dessus, améliorerait l'ergonomie de l'application.

— Enfin, l'ajout d'un outil permettant d'effectuer des requêtes (type SQL) sur les données et d'afficher les résultats. Par exemple, il serait possible de rechercher et d'afficher les patients appartenant à un certain bicluster, et vérifiant certaines conditions sur des attributs.

En conclusion, l'outil développé dans cette section est pour le moment au stade de prototype, mais les résultats obtenus montrent qu'il est possible de résoudre en partie les problèmes posés par la visualisation des biclusters sur données hétérogènes et temporelles. L'intégration au sein d'une application fonctionnelle a été entamée et nécessite d'être poursuivie, en définissant de manière plus formelle les cas d'utilisation souhaités et en examinant leur faisabilité.

Nous disposons à présent d'un algorithme de biclustering, HBC, capable de gérer efficacement les données hétérogènes, y compris les données temporelles sous deux aspects différents. Cet algorithme est facilement parallélisable, et un outil complémentaire, VisuHBC, permet d'en représenter visuellement les résultats. Dans le chapitre suivant, nous appliquons HBC, développé dans les trois derniers chapitres, à un cas d'étude concret sur données hospitalières.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Chapitre 8

Cas d'étude - identification de parcours patients

Ce chapitre présente le travail réalisé sur un des cas d'étude du projet ClinMine. L'objectif de ce cas d'étude est de déterminer dans quelle mesure les patients ayant un profil similaire suivent un parcours de soins similaire. Dans une première section, nous exposons le problème posé par ce cas d'étude et motivons l'utilisation de HBC pour y répondre. Ensuite, nous détaillons les données utilisées pour cette étude, qui incluent un nouveau type de données, les *séries symboliques irrégulières*. Nous proposons une méthode de simplification pour ces données, qui peut être utilisée afin de gérer ces données dans HBC. Nous présentons ensuite le processus complet de fouille de données : préparation des données, application de la méthode et analyse des résultats. Nous concluons ce chapitre par la validation des objectifs du cas d'étude et la présentation de perspectives d'application à des cas d'étude plus précis et plus complets.

8.1 Présentation du problème

Le problème associé au cas d'étude qui nous intéresse ici, présent dans les objectifs du projet ClinMine, peut être formulé ainsi : **Les patients ayant des profils similaires suivent-ils des parcours de soins identiques ?**

Les motivations pour cette question ont déjà été présentées dans le Chapitre 1, Section 1.2.2.2. Il est possible de séparer la question en deux étapes :

— l'identification de groupes de patients similaires,

— la comparaison des parcours de soins au sein de chaque groupe.

La première étape correspond au type de tâches qui peuvent être résolues par le biclustering, ce qui justifie l'emploi de HBC. Pour la seconde étape, nous définissons le parcours de soins comme la séquence d'Unités Médicales (UM) qui forme le séjour. Il convient de noter que, dans les données, une instance correspond aux informations disponibles pour un *séjour*, et non pour un *patient*. Pour cette raison, le parcours de soins dont nous parlons ici est limité à la séquence d'UM au sein d'un même séjour, et n'inclut pas les éventuelles autres hospitalisations du patient concerné.

Ce cas d'étude présente les caractéristiques suivantes, qui en font un bon exemple de problématique métier réelle :

— Comme détaillé dans la section suivante, le problème à résoudre concerne un ensemble de données réelles, c'est-à-dire générées par l'activité de soins et extraites des bases de

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

données d'un hôpital (ou de plusieurs hôpitaux). De plus, ces données exhibent les spécificités propres aux données hospitalières : hétérogénéité, temporalité (événements et séries temporelles), provenance de plusieurs sources différentes.

— Corollairement au point précédent, ces données ont une réelle signification d'un point de vue métier, permettant de valider la cohérence médicale des résultats obtenus. Le problème permet également la découverte de nouvelles connaissances (via fouille de données), de cas particuliers, intéressants ou surprenants pour les experts médicaux.

— Le cas d'étude implique un volume de données conséquent, ce qui pose des contraintes d'efficacité et de scalabilité sur les méthodes de fouille de données utilisées. Idéalement, l'analyse de l'ensemble des données doit pouvoir se faire sur une configuration classique, comme un poste de travail ou un serveur d'application, sans nécessiter de recours à une grille de calcul par exemple. De plus, l'analyse doit se faire dans des temps raisonnables (quelques jours). Cela permet de valider les résultats et de corriger ou compléter les données, et ce de manière itérative, ce qui est habituel en fouille de données (voir Figure 1.7 dans le Chapitre 1 sur le Contexte).

8.2 Données utilisées

Afin de remplir les critères établis dans la section précédente, nous utilisons pour ce cas d'étude un ensemble de données extraites des systèmes d'information du GHICL, partenaire du projet ClinMine. Ces données couvrent deux ans d'activité, 2015 et 2016, et proviennent des trois hôpitaux du GHICL. Ces données sont composées des ensembles suivants :

— les données PMSI,

— les informations personnelles sur les patients : âge, sexe, date de naissance, code postal,

— les informations complémentaires sur les séjours : Unité Médicale (UM), Groupement Homogène de Malades (GHM), modes d'entrée et de sortie des UM,

— les résultats d'analyses biologiques.

Les **données PMSI** et **informations patients** ont déjà été utilisées précédemment, nous ne les présentons donc pas plus en détail. Les autres données disponibles nécessitent cependant une description plus approfondie.

Pour ce qui est des **informations complémentaires sur les séjours**, nous avons déjà présenté les Unités Médicales, qui correspondent aux différents services de soins d'un hôpital (neurologie, chirurgie, etc.). A la fin du séjour, à partir du Résumé de Sortie Standardisé (RSS), le Groupement Homogène de Malades (GHM) est déterminé. Le GHM correspond au "profil-type" du séjour, c'est-à-dire la raison principale qui a motivé l'hospitalisation du patient (par exemple, une appendicite ou un traumatisme crânien). En pratique, le GHM est un système de classification hiérarchique, formé des six caractères suivants :

— Deux chiffres formant un code décrivant le système fonctionnel concerné par le diagnostic principal. Par exemple, "01" désigne les affections du système nerveux et "02" les affections de l'œil.

— Une lettre décrivant la raison d'admission à l'hôpital, permettant surtout de différencier les consultations des hospitalisations.

— Deux chiffres décrivant plus précisément le diagnostic principal, en complément des deux premiers chiffres.

— Une lettre indiquant le niveau de sévérité du diagnostic ou le caractère chronique de l'affection.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Par exemple, le GHM 02C051 correspond aux "Interventions sur le cristallin avec ou sans vitrectomie, niveau 1". Dans le code, 02 correspond aux affections de l'œil, C à la nature chirurgicale de l'intervention, 05 est le code permettant de préciser l'affection, et 1 décrit le niveau de sévérité.

Les modes d'entrée ou de sortie d'UM indiquent le type de lieux de provenance ou de destination des patients par rapport à l'UM : domicile, autre UM, autre établissement hospitalier, etc. Ces deux informations utilisent en grande partie les mêmes codes ; un mode de sortie particulier existe cependant pour indiquer les patients décédés au cours de leur séjour dans l'UM.

Les **résultats d'analyses biologiques** indiquent pour chaque séjour les analyses biologiques qui ont été menées et les résultats. Ceux-ci peuvent se présenter sous forme d'une valeur numérique ou d'une chaîne de caractères, en fonction de l'analyse menée. Les valeurs numériques correspondent à des mesures biologiques, telles que la concentration en plasmocytes dans le sang, ou le dosage de paracétamol par exemple. Les valeurs textuelles peuvent correspondre à différents types d'information : noms de médicaments ou de pathologies, positionnement par rapport à une norme (valeurs *normale* et *anormale*), mesures qualitatives (par exemple *rares* ou *peu nombreux*), etc. Ces données contiennent également les délais entre les différentes analyses menées lors d'un séjour. Une même analyse peut être effectuée plusieurs fois au cours d'un même séjour.

En termes de volumétrie, les données contiennent 166 338 séjours et 18 184 attributs (dont 2 887 correspondant aux résultats d'analyses biologiques).

8.3 Extension aux séries symboliques

Dans la section précédente, nous avons présenté les données disponibles pour l'étude, et notamment les résultats d'analyses biologiques. Les valeurs de ces résultats peuvent être numériques ou textuelles, et font partie de séries temporelles car une même analyse peut être menée plusieurs fois. La présence de séquences de valeurs textuelles n'a pas été considérée jusqu'à présent. Dans cette section, nous motivons le développement d'une méthode de simplification pour ce type de données, afin de pouvoir les utiliser dans les outils de fouille de données développés dans cette thèse. Nous détaillons ensuite la méthode proposée à cet effet, puis présentons des expérimentations visant à valider, de manière qualitative, la cohérence des résultats de cette méthode.

8.3.1 Motivations

Dans le Chapitre 6, nous avons défini deux types de données permettant de décrire la temporalité dans les données :

— les *événements temporels* définissent les données contenant à la fois une information binaire (*l'événement se produit*) et un marqueur temporel (*l'événement se produit au temps t*).

— les *séries temporelles* définissent une suite ordonnée de paires *(valeur , temps)*. Celles-ci sont donc utilisées quand l'attribut concerné est défini sur un ensemble de valeurs numériques, et qu'il peut se produire plusieurs fois.

Il reste cependant un cas qui n'est pas couvert par ces deux types de données. Il arrive qu'un attribut se produise plusieurs fois, mais qu'il ne prenne pas de valeurs numériques. Les résultats d'analyses biologiques en sont un premier exemple. Une maladie peut également exister sous différentes formes, et évoluer d'une forme à l'autre. Un traitement médical peut aussi changer de type au cours du temps. Par exemple, le diabète peut être de type 1, de type 2 ou gestationnel. Le traitement peut se faire par injections d'insuline ou par voie orale, en fonction de la situation. Dans ce genre de cas, les valeurs que peut prendre l'attribut sont impossibles à ordonner. Il est donc impossible de modéliser une suite d'états pour un tel attribut comme une série temporelle.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Pour cette raison, nous proposons un nouveau type de données, appelé *série symbolique*, également défini comme une suite ordonnée de k valeurs $V = \{v_1, v_2, \ldots, v_k\}$ et de leurs temps correspondants $T = \{t_1, t_2, \ldots, t_k\}$ avec $\forall t_i \in T : t_{i+1} > t_i$. La différence avec les séries temporelles est qu'il n'existe aucune relation de type $v_i < v_j$ entre les valeurs, uniquement des relations de type $v_i = v_j$ ou $v_i \neq v_j$. Dans le reste de cette partie, nous définissons une méthode pour simplifier ce type de données, qui ne peuvent être traitées en l'état par les outils de fouille de données. Le processus de simplification proposé se veut générique, et peut donc être utilisé en pré-traitement de manière indépendante, par exemple avant utilisation par MOSC. Cependant, comme pour les séries temporelles décrites au Chapitre 6, nous gardons à l'esprit l'intégration de ce processus dans HBC.

8.3.2 Méthode choisie

Les *séries symboliques* qui nous intéressent ici ont de nombreux points communs avec les *séries symboliques* que nous avons étudiées précédemment. En particulier, ces séries sont, elles aussi, irrégulières et souvent courtes. Pour les séries temporelles, ces caractéristiques posent de nombreux problèmes et empêchent d'appliquer la plupart des méthodes existantes. Pour les séries symboliques, cependant, celles-ci ne posent pas de réel problème. Notamment, les modèles de Markov (en temps continu) paraissent bien adaptés à ce problème de modélisation d'une suite de changement d'états [10]. Par exemple, une étude utilise des chaînes (et champs aléatoires) de Markov caché(e)s pour mener une classification non-supervisée d'images radar [61]. Une telle approche permet d'obtenir la simplification des données nécessaire à HBC en pré-traitement : les séries sont rangées dans k classes, qui définissent donc un nombre fini de valeurs possibles (de la même manière que le clustering). Cependant, HBC requiert également une mesure de similarité entre valeurs, entre autres afin de calculer la qualité des biclusters. Une approche par modèles de Markov n'est pas idéale pour ce problème. En effet, s'il est aisé de mesurer la distance entre une série et une classe, en calculant la probabilité d'un modèle à générer cette série, il est plus difficile de calculer la distance entre deux séries avec ces outils.

Nous remarquons également que les séries symboliques sont très proches des séries temporelles. Pour ces raisons, nous choisissons d'utiliser une méthode de traitement similaire à celle développée pour les séries temporelles (Section 6.2), mais adaptée aux contraintes des séries symboliques. Nous reprenons donc le processus défini précédemment :

— Définition d'une mesure de similarité deux-à-deux

— Calcul de la matrice de similarité entre l'ensemble des séries

— Clustering sur cette matrice.

La mesure de similarité sert également pour le calcul de la qualité d'un ensemble de séries symboliques. Dans ce cadre, la seule réelle modification à apporter concerne la mesure de similarité. Pour les séries temporelles, la distance entre deux points (de deux séries différentes) est définie comme la distance euclidienne sur les deux dimensions de la valeur (numérique) et du temps. Pour les séries symboliques, cette distance, décrite dans l'Algorithme 13, est la différence entre les temps des deux points (normalisée sur l'écart maximum entre deux points de ces séries) si les valeurs sont identiques, et 1 si les valeurs sont différentes. De cette manière, nous tenons compte à la fois des valeurs et du temps, tout en adaptant la mesure aux valeurs symboliques.

8.3.3 Expérimentations

L'objectif de cette section est simplement de présenter quelques résultats préliminaires qui attestent que la méthode proposée génère des clusters de séries symboliques cohérents. Des ex-

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Algorithme 13 : Calcul de la similarité entre deux séries symboliques S_1 et S_2

```
Data : v1 = valeurs ordonnées pour S1
Data : v2 = valeurs ordonnées pour S2
Data : t1 = temps ordonnés pour S1
Data : t2 = temps ordonnés pour S2
begin
    sizeDiff ← size(S1) − size(S2);
    dist ← matrice avec size(S1) lignes et size(S2) colonnes;
    for i=0..size(S1) do
        for j=0..size(S2) do
            if v1[i]==v2[j] then
                dist[i][j] ← |t1[i]−t2[j]| / max(|t1[i]−t2[j]|);
            end
            else
                dist[i][j] ← 1;
            end
        end
    end
    for j=1..size(S2)-1 do
        for i=j..j+sizeDiff do
            maxVal ← 0;
            for k=j-1..i-1 do
                if dist[k][j − 1] > maxVal then
                    maxVal ← dist[k][j − 1];
                end
            end
            dist[i][j] ← dist[i][j] + maxVal;
        end
    end
    result ← 0;
    for i=size(S2)-1..size(S1)-1 do
        if dist[i][size(S2) − 1] > result then
            result ← dist[i][size(S2) − 1];
        end
    end
    return result;
end
```

périmentations plus conséquentes sont présentées dans le cas d'étude détaillé par la suite, qui inclut une large partie de séries symboliques.

Ici, nous nous intéressons à un ensemble de données pour un cas d'étude sur le diabète. Pour cela, nous disposons de données pour environ 10 000 patients, désignés par leur âge et par trois séries symboliques :

— *diabète_type* : le type de diabète, parmi {type1, type2, gestationnel, autre}.

— *diabète_complications* : les complications entraînées par la maladie, parmi {chroniques, aigues}.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

— *diabète_traitement* : le traitement prescrit pour la maladie, parmi {oral, insuline}.

Chacun de ces attributs peut également prendre une valeur supplémentaire, *NA*, quand aucune information n'est disponible pour cet attribut (donc absence, ou donnée non renseignée). Nous avons choisi ces données plutôt que celles du cas d'étude pour valider l'approche proposée pour les séries symboliques, car les résultats sur les données diabète sont plus aisés à interpréter, notamment pour ce qui est des changements d'état.

Nous présentons ci-dessous plusieurs exemples de clusters de séries symboliques obtenus sur ces données. Ces clusters sont représentés de la même manière que les séries symboliques dans la section précédente. Cette représentation n'est pas idéale, car elle nécessite de fixer un ordre arbitraire entre les différentes valeurs possibles pour représenter celles-ci comme différents niveaux sur l'axe des ordonnées. Par exemple, pour *diabète_complications*, nous fixons *NA* ⇔ -1, *chroniques* ⇔ 1 et *aigues* ⇔ 2. Ces choix ne correspondent pas à des réalités médicales, aussi convient-il de ne pas lire les graphes présentés comme des séries temporelles, mais davantage comme des suites de changements d'état.

Le graphe 8.1 montre un ensemble de séries pour l'attribut *diabète_type*, pour lesquelles des patients précédemment identifiés comme diabétiques passent à l'état *NA* (-1), probablement par manque d'information sur l'état de la maladie.

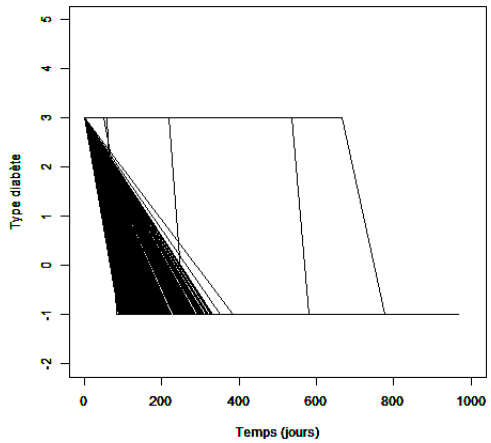


FIGURE 8.1 – Cluster de séries symboliques. Chaque série représente les changements d'état pour un patient, pour le type de diabète, au fil du temps. Les patients représentés ici passent d'un diabète type 1 (3 sur l'axe des ordonnées) à un état non renseigné (-1 sur l'axe des ordonnées)

Le graphe 8.2 montre un ensemble de séries pour l'attribut *diabète_traitement*, pour lesquelles les patients changent de traitement (transition de 2 à 1). Le graphe 8.2 montre également un ensemble de séries pour l'attribut *diabète_traitement*, pour lesquelles des patients précédemment non-traités (ou avec une information manquante concernant le traitement) passent à un état de traitement par insuline.

Il est intéressant de constater que les transitions d'état ne se font pas toutes au même moment. Cela est particulièrement remarquable sur la Figure 8.1, où certaines se produisent après une longue période à rester dans l'état initial. Cela illustre bien la capacité de la méthode à regrouper les séries présentant des sous-parties communes, ce qui est le but recherché. Ce comportement

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

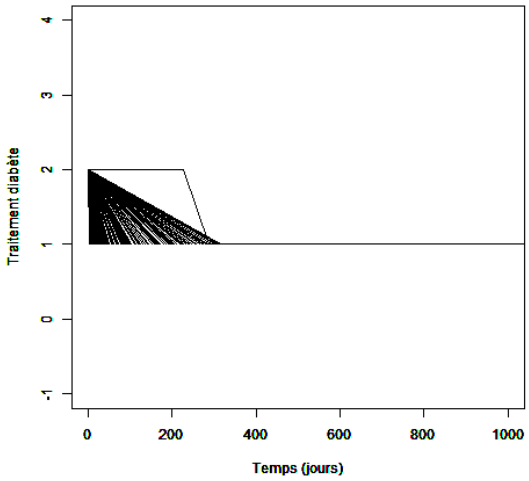


FIGURE 8.2 – Traitement du diabète (I)

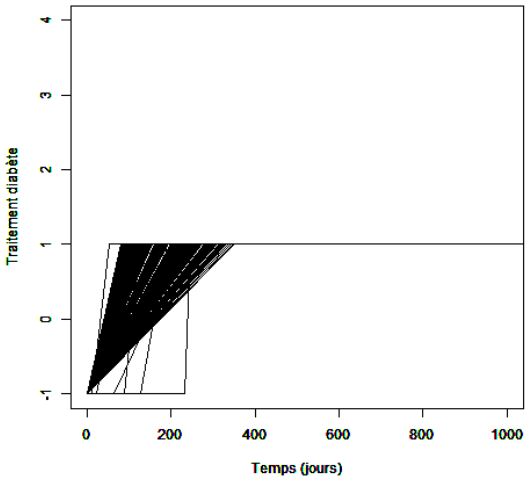


FIGURE 8.3 – Traitement du diabète (II)

avait déjà été illustré pour la méthode permettant de gérer les séries temporelles irrégulières, et est confirmé ici, les deux méthodes étant très similaires. A présent que le cas d'étude est défini et que nous sommes en mesure de gérer l'ensemble des données disponibles, nous présentons le processus de fouille de données, de la préparation des données à l'analyse des résultats.

8.4 Expérimentations et résultats

8.4.1 Préparation des données

Nous rappelons ici que la question à laquelle nous cherchons à répondre est la suivante : les patients ayant des profils similaires suivent-ils des parcours de soins identiques ? Nous avons vu que le biclustering permet d'obtenir des groupes de patients similaires, et que les parcours de soins correspondent à la séquence d'UM décrivant un séjour hospitalier. Pour ce qui est de la préparation des données, trois points importants sont à souligner.

Tout d'abord, nous choisissons d'utiliser les séquences d'UM uniquement après que les groupes de patients sont établis. Pour cette raison, ces séquences ne sont pas incluses dans les données utilisées pour le biclustering.

Ensuite, nous disposons du Groupement Homogène de Malades (GHM) associé à chaque séjour, qui décrit le motif d'hospitalisation pour ce séjour. Les GHM permettent donc d'effectuer un premier regroupement des séjours. Le GHM est un outil de classement hiérarchique, dont les différents caractères définissent différents niveaux de détail. Nous choisissons d'utiliser cette information complémentaire et de séparer l'ensemble des données disponibles en plusieurs jeux de données, chacun regroupant les patients ayant des GHM similaires. Ce processus de séparation est décrit visuellement dans la Figure 8.4. Nous ne cherchons pas à ce que les GHM dans chaque fichier ainsi créé soient rigoureusement identiques, mais plutôt qu'ils partagent les mêmes premiers caractères. Ainsi, nous créons deux ensembles de jeux de données, correspondant à deux niveaux de détail dans la séparation par GHM :

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

— Dans le premier, nous regroupons les séjours dont les deux premiers caractères du GHM sont identiques. Ces deux premiers caractères correspondent au système fonctionnel concerné par le diagnostic principal (œil, appareil respitatoire, système digestif, etc.).

— Dans le second, nous regroupons les séjours dont les cinq premiers caractères sont identiques. Ceux-ci correspondent au motif d'hospitalisation précis, avec uniquement le dernier caractère, indiquant la sévérité, pouvant varier d'un séjour à l'autre.

Pour chacun de ces modes de séparation, nous conservons uniquement les jeux de données (où chaque jeu correspond à GHM) contenant au moins 100 séjours. Nous obtenons ainsi 21 jeux de données pour la première séparation, et 241 pour la seconde. HBC est appliqué sur chacun de ces jeux de données. Nous utilisons la version parallèle de HBC avec 3 processus en parallèle, sur la même machine que pour les expérimentations des chapitres précédents (Dell T1700 quad-core à 3.30GHz, disposant de 4 GB de RAM, sous Ubuntu 14 avec gcc 4.8.2). L'exécution de HBC est de l'ordre de quelques secondes à quelques minutes pour chaque jeu de données. L'ensemble des résultats pour les deux modes de séparation est obtenu en moins de quatre heures, ce qui est tout à fait raisonnable par rapport au temps nécessaire pour l'agrégation et la préparation des données.

Le GHM permet de définir un premier regroupement des séjours, mais ne se substitue en aucun cas au processus de biclustering. Nous considérons le *profil patient* comme un ensemble de plusieurs informations ; le GHM seul ne permet donc pas de définir un profil patient. Des patients ayant le même GHM partagent un motif d'hospitalisation identique, mais peuvent présenter des caractéristiques très différentes qui les séparent en autant de groupes distincts (âge, comorbidités, diagnostics associés, etc.). Le regroupement par GHM facilite néanmoins l'analyse des données par biclustering, en empêchant la formation de biclusters sans intérêt compte tenu du cas d'étude. Par exemple, un bicluster (extrait de l'ensemble des données) regroupant des séjours concernant des patients masculins âgés admis en HdJ (Hospitalisation de Jour) et ayant subi une radiographie du thorax, est trop générique pour en tirer des informations intéressantes. Il est impossible d'identifier un parcours de soins type à partir de ces informations. A l'inverse, si ces patients ont tous été admis pour un motif similaire, identifier un parcours-type peut devenir possible selon les cas. De plus, le regroupement préalable par GHM favorise la formation de biclusters contenant des attributs plus rares et plus spécifiques à certains types de séjours. Enfin, ce regroupement a du sens d'un point de vue métier ; les experts médicaux raisonnent le plus souvent sur des types de séjours spécifiques (identifiés par le GHM ou le diagnostic principal) que sur l'ensemble des données.

Le dernier point concerne la gestion des séries temporelles (et séries symboliques). Nous observons que, pour chaque attribut de ces types, une majorité des séquences ne contiennent qu'un unique point. De telles séquences n'ont pas de réel aspect temporel. Nous choisissons donc de les considérer comme un attribut non-temporel (numérique ou symbolique) séparé, ce qui permet également d'alléger la phase de pré-traitement des séries temporelles (ou symboliques) en diminuant le nombre de séries dans chaque attribut.

8.4.2 Résultats

Nous présentons ici une vue d'ensemble des résultats obtenus, et utilisons ces observations pour déterminer les critères de sélection pour les biclusters à examiner plus en détail et à présenter à des experts médicaux.

HBC crée un nombre de biclusters dépendant directement du nombre d'attributs (et de valeurs possibles de ces attributs). Dans ce cas d'étude, en comptant l'ensemble des codes PMSI et des analyses biologiques, les jeux de données, 21 pour la première séparation et 241 pour la seconde, contiennent 18 184 attributs (le nombre d'instances varie d'un jeu à l'autre). Par conséquent,

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

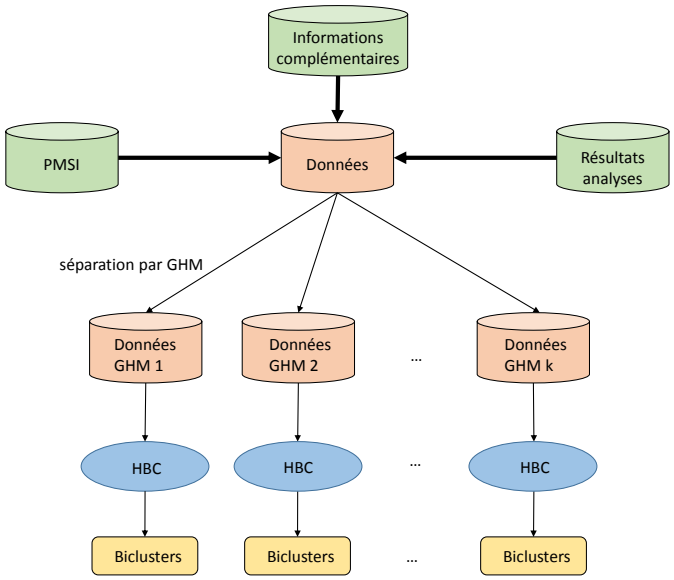


FIGURE 8.4 – Préparation et analyse des données pour le cas d'étude : agrégation à partir de différentes sources, séparation par GHM et biclustering

HBC produit un nombre important de biclusters, souvent plusieurs centaines à plusieurs milliers pour chaque jeu de données. Il est donc impossible en pratique de valider chaque bicluster. Par conséquent, il convient de filtrer ces résultats pour en extraire les biclusters les plus intéressants *a priori*. Une telle sélection des résultats est naturelle dans le cadre de la fouille de données, surtout pour une méthode non-supervisée telle que le biclustering. Il est attendu que tous les résultats ne soient pas utiles, et l'objectif consiste davantage à découvrir des connaissances utiles que d'expliquer tous les résultats obtenus.

Pour cela, nous nous recentrons sur le problème posé par le cas d'étude. Nous cherchons à comparer les parcours de soins au sein des biclusters découverts dans les données. Nous proposons donc le mécanisme de sélection suivant, appliqué à chaque bicluster afin de déterminer s'il est intéressant *a priori* pour le cas d'étude :

— Compter le nombre de séquences d'UM différentes sur l'ensemble des séjours du bicluster.

— Si ce nombre est inférieur à un certain seuil, conserver ce bicluster.

En procédant ainsi, nous sélectionnons les groupes de séjours pour lesquels le nombre de séquences d'UM différentes est peu élevé. Nous supposons que ces biclusters correspondent aux cas où la majorité des séjours partagent une même séquence d'UM, et où quelques séjours ont une séquence différente. De cette manière, nous estimons pouvoir valider deux aspects de l'approche :

— La capacité à trouver des groupes de patients ayant un parcours de soins similaire, donc la cohérence des groupes de patients ainsi formés.

— La présence de cas particuliers, c'est-à-dire de séjours ayant un parcours différent de la norme. Ces cas peuvent ensuite être soumis pour examen à des experts médicaux. Parmi ces cas, nous espérons découvrir des cas surprenants et ainsi apporter de nouvelles connaissances aux experts.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

8.4.3 Analyse et discussion

Nous appliquons le processus de sélection à l'ensemble des biclusters trouvés par HBC, pour tous les jeux de données des deux modes de séparation par GHM (2 et 5 caractères respectivement). Nous choisissons de conserver les biclusters ayant moins de cinq séquences d'UM différentes. Après sélection, nous obtenons plusieurs centaines de biclusters pour chaque mode de séparation, ce qui demeure impossible à analyser exhaustivement en pratique. Pour cette raison, nous présentons ici un sous-ensemble de ces résultats qui a été montré à des experts médicaux, avec les explications qui ont été trouvées, ou non, pour les cas particuliers (séquences déviant de la norme).

Bicluster 1 Le premier est un bicluster contenant 471 séjours (parmi les 3 323 du jeu de données) et les 4 attributs suivants :

— le code PMSI "Cataracte sénile nucléaire",
— décès = non,
— HC (Hospitalisation Complète) = non,
— sexe = M,

et pour lequel la répartition des séquences d'UM est la suivante :

— {*Ophtalmologie adultes, chirurgie ambulatoire*} : 470 séjours
— {*Ophtalmologie, hospitalisation complète*} : 1 séjour

Ce bicluster paraissait intéressant en raison de la présence d'un séjour en HC, malgré le fait que ce patient soit ressorti le jour de son arrivée. Cependant, après discussion avec les praticiens, ce cas se produit assez souvent, et correspond à un hébergement dans un autre service (ici, d'opthalmologie en HC) lorsque le service ambulatoire est plein ou fermé. Ce bicluster ne permet donc pas d'apporter de nouvelles connaissances, mais il reflète de manière exacte un aspect de l'activité hospitalière.

Bicluster 2 Un deuxième exemple est un bicluster contenant 583 séjours (parmi les 3 323 du jeu de données) et les 6 attributs suivants :

— le code PMSI "Anesthésie générale ou locorégionale complémentaire niveau 1",
— le code PMSI "Extraction extracapsulaire du cristallin par phakoémulsification",
— le code PMSI "Injection de substance inerte ou organique dans la chambre antérieure de l'oeil",
— le code PMSI "Cataracte sénile",
— décès = non,
— HC = non,

et pour lequel la répartition des séquences d'UM est la suivante :

— {*Ophtalmologie, chirurgie ambulatoire*} : 581 séjours
— {*Stomatologie, chirurgie ambulatoire*} : 1 séjour
— {*Gastrologie, hospitalisation de jour*} : 1 séjour

Ces résultats semblent étonnants, car les UM minoritaires ici (stomatologie et gastrologie) n'ont pas pour fonction d'effectuer des opérations de l'œil. Il est cependant possible de formuler une explication, en examinant les codes des UM concernées : 1746, 1745 et 1764. Ces numéros étant très proches, il est envisageable que ces anomalies dans les UM soient dues à un codage erroné. Cette explication est renforcée par la présence de plusieurs autres cas similaires parmi les biclusters.

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Bicluster 3 Un second exemple de ce type de cas concerne un bicluster contenant 67 séjours (parmi les 5 994 du jeu de données) et les 6 attributs suivants :

— le code PMSI "Changement d'une sonde d'urétérostomie cutanée",

— le code PMSI "Supplément pour injection peropératoire de produit de contraste radiologique dans les voies excrétrices urinaires pour prises de clichés radiologiques",

— le code PMSI "Tumeur maligne de la vessie",

— le code PMSI "Urétéropyélographie rétrograde [UPR]",

— décès = non,

— HC = non,

et pour lequel la répartition des séquences d'UM est la suivante :

— {*Urologie, chirurgie ambulatoire*} : 66 séjours

— {*Orthopédie, chirurgie ambulatoire*} : 1 séjour

Les UM concernées portent les codes 1744 et 1743, respectivement. Ceci qui renforce l'hypothèse d'un codage erroné, d'autant plus que tous ces cas proviennent du même hôpital (parmi les trois du GHICL).

Bicluster 4 Un autre bicluster contient 187 séjours (parmi les 983 du jeu de données) et les 4 attributs suivants :

— le code PMSI "Test d'évaluation et de catégorisation des déficiences",

— le code PMSI "Examen de contrôle après d'autres traitements pour d'autres affections",

— décès = non,

— HC = non,

et pour lequel la répartition des séquences d'UM est la suivante :

— {*Médecine physique et rééducation, HdJ*} : 152 séjours

— {*Gériatrie, HdJ*} : 32 séjours

— {*Dermatologie, HdJ*} : 3 séjours

Un test d'évaluation des déficiences paraît approprié dans les services de rééducation et de gériatrie, mais moins dans un service de dermatologie. Après discussion avec les experts médicaux, ce genre de cas vient confirmer leur expérience que le code pour ce test d'évaluation a tendance à être mal utilisé, dans des situations où il ne correspond pas à l'acte qui a été réalisé en pratique.

Bicluster 5 Un autre bicluster contient 1078 séjours (parmi les 24 735 du jeu de données) et les 5 attributs suivants :

— le code PMSI "Tumeur maligne de bronche ou du poumon",

— le code PMSI "Séance de chimiothérapie pour tumeur",

— décès = non,

— HC = non,

— sexe = M,

et pour lequel la répartition des séquences d'UM est la suivante :

— {*Oncologie, HdJ*} : 1070 séjours

— {*Hématologie, HdJ*} : 7 séjours

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

— {*Chirurgie digestive, HdJ*} : 1 séjour

Dans ce cas, la plupart des séjours passent en oncologie, ce qui est normal pour une séance de chimiothérapie. La présence de certains séjours en hématologie peut s'expliquer, car les séances s'accompagnent parfois de transfusions, qui relèvent du service d'hématologie. Le séjour qui passe en chirurgie digestive correspond à un cas d'implantation (par chirurgie) d'un *cathéter à chambre implantable*, un dispositif facilitant l'installation des cathéters, notamment utilisés pour la chimiothérapie. Ces cas particuliers correspondent donc à des réalités médicales, mais indiquent aussi probablement un oubli dans le codage des UM. En particulier, le patient passé en chirurgie digestive a vraisemblablement été transféré en oncologie pour chimiothérapie ensuite, mais la séquence d'UM ne le reflète pas.

Bicluster 6 Tous les exemples présentés jusqu'ici ne concernent que des séquences contenant une unique UM, en partie parce que ces cas sont plus faciles à interpréter. Le cas suivant contient des séquences d'UM plus longues, 33 séjours (parmi les 12 895 du jeu de données) et les 4 attributs suivants :

— le code PMSI "Dilatation intraluminale d'une artère du membre inférieur sans pose d'endoprothèse",

— le code PMSI "Supplément pour imagerie pour acte de radiologie interventionnelle",

— décès = non,

— HC = oui,

et pour lequel la répartition des séquences d'UM est la suivante :

— {*Chirurgie vasculaire, HC*} : 31 séjours

— {*Urgences, HC* ⇒ *Court séjour gériatrique, HC* ⇒ *Chirurgie vasculaire, HC*} : 1 séjour

— {*Urgences, HC* ⇒ *Neuro-vasculaire aigue* ⇒ *Cardiologie, HC* ⇒ *Chirurgie vasculaire, HC*} : 1 séjour

Cet ensemble de séjours correspond à des hospitalisations pour traitement chirurgical d'une artérite (rétrécissement du diamètre d'une artère), effectué dans une unité de chirurgie vasculaire. Le second code PMSI indique qu'une radiographie a été effectuée lors du séjour. Le premier cas particulier concerne un patient ayant été admis aux urgences, puis hébergé temporairement en gériatrie, probablement en raison de contraintes d'occupation des lits, avant de passer en chirurgie vasculaire pour opération. Le second cas particulier concerne un séjour atypique, avec de multiples complications. Après examen du dossier complet par des praticiens, le patient a été admis aux urgences pour un AVC, ce qui explique le transfert vers le service de neurologie vasculaire aigue. Le patient a ensuite subi une chute qui a provoqué une ischémie (diminution localisée de l'apport sanguin à un organe), et a donc été transféré au service de cardiologie. L'artérite semble s'être produite durant le séjour, ce qui a entraîné son transfert vers le service de chirurgie vasculaire.

8.5 Conclusions et perspectives

Dans cette partie, nous avons présenté un cas d'étude complet sur données réelles, avec les différentes étapes que cela implique : formulation du problème, extraction et préparation des données, fouille de données, analyse des résultats et extraction de nouvelles connaissances. Nous avons vu que l'application de HBC permet de répondre à ce problème de manière naturelle, en extrayant un ensemble de connaissances vraisemblables d'un point de vue médical. La nature non-supervisée du biclustering le rend plus complexe à évaluer qu'une approche prédictive. Au cours des travaux de cette thèse, nous avons exploré plusieurs applications pour HBC. Nous nous

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

sommes concentrés, dans le présent chapitre, sur un cas d'étude plus mature et complet qui fait aussi partie des objectifs du projet ClinMine.

Ces résultats, ainsi que leur analyse en compagnie d'experts médicaux, ont permis de soulever de nouvelles questions dans la continuité de ce cas d'étude :

— Tout d'abord, la séparation des données peut être modifiée afin de clarifier les résultats (biclusters) finaux. En première approche, nous avons séparé après les deuxième et cinquième caractères du code GHM respectivement. Séparer en fonction des trois premiers caractères peut avoir un intérêt, car le troisième permet notamment de séparer les séjours de médecine et de chirurgie. Ceux-ci sont intrinsèquement différents, aussi il est attendu que les séquences d'UM associées divergent, ce qui justifie une séparation au niveau des données.

— Au lieu de séparer en fonction du GHM, il est également possible de séparer en fonction du *diagnostic principal* (un code CIM-10 contenu dans le RSS). Cela produit une séparation plus détaillée, étant donné que le nombre de diagnostics principaux est bien plus important que le nombre de GHM (plusieurs milliers). Une telle séparation présente l'avantage de lever un certain nombre d'ambiguités et de faciliter l'interprétation des résultats. En effet, avec une séparation par GHM, il arrive que les biclusters contiennent des patients ayant un même diagnostic, mais qui est diagnostic principal pour certains séjours et diagnostic associé pour d'autres. Ces séjours concernent donc des patients ayant des motifs d'hospitalisation différents, pour lesquels il est vraisemblable que les séquences d'UM diffèrent sans que cela soit anormal. Séparer par diagnostic principal permet de résoudre ces problèmes en amont. Le problème de cette séparation est que tous les diagnostics principaux ne sont pas présents dans suffisamment de séjours pour former des jeux de données de taille suffisante (au moins 100 séjours, comme défini précédemment). Au plus le regroupement préalable est fin, au plus le nombre de jeux de données à écarter car trop petits est élevé. Cela amène à l'utilisation, au total, de moins de séjours pour l'extraction de connaissances.

— Il peut également être intéressant d'ajouter de nouvelles données à celles déjà utilisées (PMSI, analyses...). En particulier, la société Alicante travaille à l'extraction de termes à partir des courriers générés lors des séjours. Ces termes sont déjà utilisés dans d'autres cas d'étude, notamment pour la prédiction de codes PMSI à partir des courriers (voir Chapitre 1), et pourraient apporter des informations complémentaires utiles à la caractérisation des parcours de soins.

— Une suggestion des praticiens impliqués dans l'analyse des résultats est d'agréger les différents séjours d'un même patient, et donc de considérer, dans les données, une instance comme un patient et non plus comme un séjour. De cette manière, il est possible d'avoir une vue plus globale du parcours des patients au fil du temps. En pratique, cela permet de comparer le devenir des patients ayant été hospitalisés pour un même motif, par exemple afin de déterminer les facteurs permettant d'expliquer des complications ultérieures. Dans ce contexte, un premier cas intéressant d'après les experts concerne le suivi des patients ayant subi une opération du rachis (colonne vertébrale) et leurs éventuelles réhospitalisations futures. Un second cas concerne les patients ayant été victimes d'un AVC. Il arrive fréquemment que des complications se produisent après un tel événement, notamment des troubles cognitifs plus ou moins importants. Ce cas est d'un intérêt particulier, mais il est également complexe. En effet, les complications se produisent parfois jusqu'à cinq ou dix ans après l'AVC ; il serait donc utile de disposer des données pour une période de temps plus longue que deux ans. Il n'est cependant pas garanti que le patient effectue tous ses séjours dans le même hôpital sur plusieurs années. Pour cette raison, il est difficile de disposer

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

de l'intégralité des données sur les séjours d'un patient, le recoupement de données d'un hôpital à l'autre étant une tâche compliquée, notamment pour des raisons de confidentialité.

En conclusion, nous pouvons affirmer que ce cas d'étude permet de valider l'efficacité et l'utilité de la méthode de biclustering HBC, pour répondre à des problématiques intéressantes d'un point de vue métier sur des données hospitalières réelles. La discussion avec les experts médicaux a permis de valider la cohérence des résultats et d'identifier plusieurs cas particuliers apportant de nouvelles informations ou confirmant des hypothèses. Cela a également permis de formuler des questions plus précises sur les données, pour lesquelles il est possible d'apporter des réponses en utilisant un protocole similaire à celui suivi dans ce cas d'étude. Ces questions nécessitent de modifier la préparation des données, mais concernent des problèmes médicaux particulièrement intéressants. Ces nouveaux cas d'étude constituent donc autant de perspectives au travail mené dans ce chapitre.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Conclusion et perspectives

Nous rappelons ici les contributions principales de cette thèse, en les situant par rapport aux objectifs annoncés dans l'introduction et à ceux du projet ClinMine. Nous présentons ensuite les différentes perspectives de ces travaux, les différents axes de recherche et perspectives d'intégration dans des outils logiciels.

Contributions

La première partie de cette thèse s'intéresse au problème de la classification supervisée sur données numériques et temporelles. A cet effet, les contributions de cette partie sont les suivantes.

L'extension de l'algorithme MOCA-I (Multi-Objective Classification Algorithm for Imbalanced data) aux données numériques , appelée MOCA-IQ (Multi-Objective Classification Algorithm for Imbalanced and Quantitative data). Deux approches sont comparées : discrétisation des données en pré-traitement, et gestion au sein de l'algorithme de classification. Pour chacune de ces deux approches, plusieurs variantes sont proposées, afin de déterminer la plus efficace. La comparaison des variantes les plus performantes pour chaque méthode indique que l'approche par discrétisation permet à l'algorithme de classification de générer de meilleurs résultats. Cette approche est donc celle retenue dans MOCA-IQ. Plusieurs améliorations de l'algorithme de classification, indépendemment du type de données considéré, sont également proposées et évaluées. En particulier, une méthode alternative pour l'archivage des solutions est retenue, qui permet d'améliorer la qualité des ensembles de règles générés par MOCA-IQ.

L'extension de MOCA-IQ aux données temporelles , c'est-à-dire les informations pour lesquelles le temps (date et heure) d'occurrence est connu. Le nouvel algorithme, appelé MOSC (Multi-Objective Sequence Classification), permet d'utiliser des séquences d'événements temporels comme composants de règles de classification. De manière plus concrète, un nouveau type de données est introduit, qui permet de gérer ces séquences d'événements. Des opérateurs de voisinage adaptés à ce nouveau type de données et un algorithme de comparaison des séquences sont également proposées. Des restrictions de l'espace de recherche sont introduites, qui permettent de fortement limiter l'explosion combinatoire de la taille du voisinage. MOSC est appliqué à différents cas d'étude sur des ensembles de données hospitalières réelles ; les résultats montrent l'apport de la temporalité des données pour affiner les règles de classification.

La seconde partie s'intéresse au problème de la classification double non-supervisée, ou *biclustering*, sur le même type de données (hétérogènes, numériques et temporelles). Les contributions de cette partie sont les suivantes.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

La modélisation du biclustering sur données hétérogènes comme un problème d'optimisation combinatoire multi-objectif, et notamment la proposition d'une mesure de qualité pour les biclusters sur données hétérogènes. L'algorithme de biclustering proposé, LSBC, utilise une métaheuristique basée sur une recherche locale bi-objectif pour l'optimisation des biclusters sur les critères de qualité et de taille. Des expérimentations montrent la capacité de LSBC (Local Search for heterogeneous BiClustering) à générer des biclusters de grande taille et de bonne qualité, mais mettent en exergue le manque de diversité des résultats, qui est fortement préjudiciable dans le cadre de l'application aux données hospitalières.

Une heuristique efficace pour le biclustering sur données hétérogènes, appelée HBC (Heterogeneous BiClustering) basée sur une simplification des données en pré-traitement. HBC utilise la même mesure de qualité que LSBC, mais une approche différente, qui consiste à générer des biclusters parfaitement homogènes sur les données simplifiées. Cela permet, une fois les données retransformées en leurs valeurs initiales, d'obtenir des biclusters homogènes. Les expérimentations menées indiquent que HBC génère des biclusters de bonne qualité et de grande taille, et montrent la capacité de l'algorithme à explorer l'ensemble de la matrice de données.

L'extension de HBC aux données temporelles sous trois formes différentes : événements temporels, séries temporelles irrégulières et séries symboliques irrégulières. Les événements temporels sont identiques à ceux gérés par MOSC. Pour ceux-ci, HBC se concentre également sur l'aspect séquentiel des événements. Les séries temporelles irrégulières sont un cas particulier de séries temporelles. Celles-ci reflètent bien la réalité des séquences de valeurs produites par les hôpitaux, notamment par les laboratoires d'analyses biologiques : peu de valeurs, espacées de manière irrégulière dans le temps. Une nouvelle mesure de distance entre deux séries temporelles irrégulières est proposée, qui permet de définir une mesure de qualité pour les biclusters. Les séries temporelles sont simplifiées en pré-traitement par clustering. Les séries symboliques irrégulières sont similaires aux séries temporelles irrégulières, mais avec des valeurs symboliques et non numériques. Pour ce troisième type de données, un traitement proche de celui des séries temporelles est utilisé, avec une modification de la mesure de distance deux-à-deux.

L'implémentation d'une version parallèle de HBC Deux approches sont proposées, respectivement pour des architectures à mémoire distribuée et partagée. La seconde approche, mieux adaptée aux environnements de calcul disponibles pour le projet, permet d'obtenir une accélération linéaire sur le nombre de processeurs utilisés. HBC se prête particulièrement bien à la parallélisation, en particulier pour l'heuristique constructive de formation des biclusters, mais également pour certaines étapes coûteuses de pré-traitement des données.

La réalisation d'un outil de visualisation pour le biclustering adapté aux données hétérogènes, et son intégration dans une application utilisable. Cet outil permet de visualiser à la fois l'ensemble des biclusters, et chaque bicluster indépendemment par rapport au reste des données. Des mesures statistiques sur les valeurs des biclusters sont également intégrées à l'interface, afin de faciliter la compréhension des résultats.

L'application de HBC sur un cas d'étude réel complet : formulation du problème, agrégation de données provenant de différentes sources, préparation des données, application de la méthode et analyse des résultats avec l'aide d'experts médicaux. Le problème étudié est la comparaison des parcours de soins pour des groupes de patients présentant un profil similaire. Les résultats trouvés correspondent à des situations réelles, tant au niveau des tendances générales

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

que des cas particuliers. Parmi ces cas particuliers, certains apportent de nouvelles connaissances ou viennent confirmer des hypothèses.

L'ensemble de ces contributions répondent aux objectifs établis au début de ces travaux, qui visaient à développer un ensemble de méthodes de fouilles de données adaptées aux spécificités des données hospitalières. Le cas d'étude utilisé pour la validation de HBC correspond à un cas d'étude du projet ClinMine, et l'ensemble des travaux de cette thèse ont été menés en collaboration constante avec les différents partenaires du projet. L'utilisation future des méthodes développées dans des applications fonctionnelles a été anticipée, avec le développement d'une version parallèle ainsi que la conception d'une interface de visualisation pour le biclustering.

Perspectives

Au cours de ces travaux, plusieurs pistes d'amélioration ou nouveaux axes de recherche ont été identifiés mais n'ont pas été explorés car jugés moins prioritaires par rapport aux objectifs de la thèse.

Mieux exploiter les données temporelles dans MOSC Les travaux menés dans cette thèse ont montré l'intérêt d'exploiter les données temporelles. L'algorithme MOSC pourrait bénéficier d'une utilisation plus systématique des dates (et heures) d'occurrence exactes des événements temporels. Pour l'instant, les règles utilisent la séquentialité et l'ordre respectif de ces événements. Une méthode permettant de préciser les règles de classification à partir de ces informations a été proposée, mais n'est pas appliquée dans tous les cas, et ces précisions n'ont pour l'instant pas d'utilité autre qu'une description basique. Le perfectionnement de cette méthode et la conception d'autres utilisations pour les dates d'occurrence peuvent constituer des perspectives intéressantes pour certains cas d'étude. Une autre piste d'amélioration concerne les séquences à plusieurs niveaux de temporalité, par exemple pour représenter plusieurs séjours pour un patient. Un modèle pour deux niveaux de temporalité a été proposé, mais pas validé. La validation de ce modèle, ou son extension pour gérer un nombre arbitraire de niveaux de temporalité, forment des perspectives possibles. Dans l'ensemble, il est préférable que les axes de recherche explorés s'accordent aux besoins précis de futurs cas d'étude. Le modèle proposé dans MOSC permet de gérer la temporalité des données de manière générique ; les perfectionnements à apporter dépendent des besoins.

Intégrer MOSC dans un logiciel MOSC est inspiré de MOCA-I, dont il reprend le fonctionnement général, avec des extensions pour de nouveaux types de données. MOCA-I est actuellement intégré dans Opcyclin, un logiciel développé par Alicante, déployé sur plusieurs sites hospitaliers. Une perspective applicative consiste donc à remplacer MOCA-I par MOSC dans Opcyclin, afin de pouvoir gérer un spectre plus large de données.

Améliorer la mesure de qualité des biclusters En ce qui concerne le biclustering, une première perspective est d'améliorer la mesure de qualité proposée. Dans cette thèse, nous utilisons la somme pondérée de mesures de qualité propres à chaque type de données. Ces mesures sont individuellement cohérentes, mais une étude plus poussée de l'impact du type d'agrégation sur les résultats pourrait être intéressante. Le choix des mesures individuelles peut également être étudié plus en détail. Nous nous sommes limités ici aux mesures d'homogénéité utilisées dans les algorithmes de biclustering, mais d'autres mesures existent dans d'autres domaines, notamment en statistiques. Une étude sur ce point est en cours dans le cadre du projet ClinMine ; les conclusions pourront servir à améliorer la mesure de qualité proposée. Il est à noter que cette mesure

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

de qualité n'est pas fondamentale pour HBC, car elle n'est calculée qu'à la fin de la formation des biclusters, mais elle conserve néanmoins une utilité en post-traitement. Un autre point intéressant concerne la définition d'une mesure de qualité adaptée à des biclusters contenant des données manquantes. Une telle mesure n'est pas strictement requise dans HBC, mais peut ouvrir la voie au développement de nouvelles méthodes de biclustering pour des matrices de données *creuses*.

Améliorer la scalabilité de HBC pour les séries temporelles La gestion des séries temporelles (et symboliques) irrégulières dans HBC a nécessité le développement d'une mesure de similarité pour le pré-traitement de ces données. Une limitation de la méthode actuelle est qu'elle calcule une matrice de similarité, qui occupe un espace quadratique sur le nombre de séries. Cela pose problème pour les cas où de très nombreuses séries (>50,000) sont disponibles pour un attribut de ce type, et limite donc la scalabilité de la méthode sur de telles données. Cela peut également amener à repenser la méthode de clustering des séries temporelles, dont le temps d'exécution augmente fortement avec le nombre de séries et le nombre de clusters souhaités. La conception de méthodes plus efficaces et moins coûteuses en espace et en temps, pour ces deux éléments, constitue donc un axe de recherche intéressant.

Poursuivre le développement de l'outil de visualisation pour HBC Un premier outil de visualisation pour le biclustering a été proposé et intégré dans une application. Cet outil répond en partie aux problématiques posées par la visualisation de biclusters sur données hétérogènes, mais pourrait être amélioré ou repensé pour pallier les lacunes actuelles. En particulier, la visualisation de volumes de données conséquents, et de biclusters se recouvrant partiellement, forment deux pistes d'amélioration majeures. L'intégration dans l'application, notamment pour la communication entre l'interface et la génération des images, mérite également une attention particulière. Une perspective à plus long terme consisterait à intégrer l'ensemble du processus de biclustering, y compris la visualisation, dans une même application.

Appliquer HBC à de nouveaux cas d'étude Enfin, différents cas d'étude plus précis ont été identifiés à la fin du chapitre précédent, qui correspondent à de réelles problématiques médicales. Les expérimentations menées jusqu'à présent montrent que les méthodes développées pour la fouille de données hospitalières permettent de découvrir des résultats cohérents et parfois surprenants. L'application de ces méthodes aux nouveaux cas d'étude identifiés nécessite un travail supplémentaire au niveau de l'agrégation et de la préparation des données, mais doit permettre d'apporter des réponses à ces questions médicales. Les cas d'étude traités au cours de cette thèse répondent déjà aux objectifs du projet ClinMine, et il est naturel de chercher à poursuivre l'application à des cas d'étude plus précis.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Annexe A

Comparaison des méthodes de clustering pour séries temporelles irrégulières dans HBC

Nous cherchons à évaluer la qualité des clusterings produits par les différentes méthodes existantes (*k-medoids* et clustering hiérarchique). Pour cela, deux aspects doivent être considérés. Le premier aspect concerne, pour les méthodes stochastiques (*k-medoids*), la stabilité des clusterings obtenus, c'est-à-dire la similarité des clusters sur plusieurs exécutions indépendantes de l'algorithme avec différents médoïdes initiaux. Nous comparons les trois variantes suivantes, avec un nombre de clusters $k = 5$ dans chaque cas :

- Méthode 1 : algorithme *k-medoids* classique ; similarité entre séries temporelles calculée en utilisant un *matching optimal* (voir Section 6.2.3).
- Méthode 2 : algorithme *k-medoids* modifié ; similarité entre séries temporelles calculée en utilisant un *matching optimal*.
- Méthode 3 : algorithme *k-medoids* modifié ; similarité entre séries temporelles calculée en utilisant un *matching minimal*.

Nous générons dix jeux de données synthétiques (ds1 à ds10), chacun comportant 80 ensembles de 500 séries temporelles chacun. La longueur de ces séries varie de manière aléatoire entre 2 et 6 mesures. Pour chaque jeu de données et chaque méthode, 100 exécutions indépendantes du processus de clustering sont effectuées. Après cela, nous calculons la dissimilarité entre toutes les paires possibles de clusterings (parmi les 100 obtenus), ces résultats sont enregistrés dans une matrice de dissimilarité. La dissimilarité entre deux clusterings C_1 et C_2 est calculée en utilisant une distance de Hamming normalisée [109], qui est semblable à une fonction objectif 0-1 :

$$dissim(C1, C2) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{C1_i \neq C2_i\} \tag{A.1}$$

A noter que nous considérons la valeur minimale parmi tous les matchings possibles entre les clusters de C_1 et ceux de C_2. Un score de dissimilarité dans $[0, 1]$ est calculé pour chaque cluster. Par conséquent, le score de dissimilarité pour $k = 5$ clusters est dans $[0, 5]$. Une fois la matrice de dissimilarité remplie, nous calculons la moyenne sur l'ensemble des valeurs. Nous prenons ensuite la moyenne sur les 80 ensembles de séries temporelles pour chaque jeu de données. La dissimilarité ainsi agrégée pour chaque jeu de données et chaque variante du k-medoids est présentée dans la Table A.1. Ces résultats montrent que le k-medoids modifié tend à produire des résultats

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

	Méthode 1	Méthode 2	Méthode 3
	$k-medoids$ classique matching optimal	$k-medoids$ modifié matching optimal	$k-medoids$ modifié matching minimal
ds0	2.71±0.06	1.82±0.4	**0.77±0.31**
ds1	2.71±0.05	1.83±0.39	**0.76±0.35**
ds2	2.71±0.05	1.78±0.39	**0.77±0.26**
ds3	2.7±0.05	1.89±0.35	**0.79±0.36**
ds4	2.7±0.05	1.85±0.35	**0.67±0.33**
ds5	2.7±0.05	1.8±0.38	**0.77±0.26**
ds6	2.7±0.06	1.79±0.36	**0.71±0.32**
ds7	2.7±0.05	1.86±0.38	**0.77±0.3**
ds8	2.71±0.05	1.85±0.39	**0.77±0.33**
ds9	2.7±0.05	1.86±0.38	**0.75±0.25**

TABLE A.1 – Stabilité - mesure de dissimilarité pour les variantes de *k-medoids*

notablement plus stables que les autres variantes. L'utilisation d'un *matching minimal* pour la mesure de distance entre séries amène également une amélioration de la stabilité.

Au-delà de la stabilité, qui n'est applicable que pour les variantes du *k-medoids*, nous voulons également évaluer la qualité des clusterings générés. De nombreuses mesures ont été développées pour cette tâche, qui peuvent être groupées en trois catégories. Les mesures *intra-cluster* calculent la densité ou l'homogénéité des clusters. Les mesures *inter-cluster* calculent la distance entre clusters ; une distance plus élevée indique que les clusters sont bien délimités. Les mesures *composites* incluent les deux types précédents, et sont en général formées par le ratio d'une mesure *intra-cluster* sur une mesure *inter-cluster*. Pour cette étude, nous utilisons trois mesures *intra*, trois mesures *inter* et une mesure *composite* :

- intra_max : distance maximum entre deux points d'un même cluster (à minimiser).
- intra_avg : distance moyenne entre deux points d'un même cluster (à minimiser).
- intra_centre : distance moyenne d'un point au centroïde du cluster (à minimiser).
- inter_single : distance entre les deux points les plus proches de deux clusters différents (à maximiser).
- inter_complete : distance entre les deux points les plus éloignés de deux clusters différents (à maximiser).
- inter_centre : distance entre les centroïdes des clusters (à maximiser).
- composite : index Davies-Bouldin [44] (à minimiser). $DB = \frac{1}{n}\sum_{i=1}^{n} max_{j\neq i}\Big(\frac{\sigma_i+\sigma_j}{d(c_i,c_j)}\Big)$, où n est le nombre de clusters, c_x le centroïde du cluster x, σ_x est la distance moyenne des éléments de x au centroïde c_x, et $d(c_i,c_j)$ est la distance entre les centroïdes c_i et c_j.

En plus des trois variantes du k-medoids, nous évaluons deux variantes de clustering hiérarchique. Celles-ci n'ont pas été incluses dans la comparaison de la stabilité car elles ne sont pas stochastiques et produisent donc toujours les mêmes résultats.

- méthode 4 : clustering hiérarchique avec *matching optimal* pour la mesure de distance des séries temporelles.
- méthode 5 : clustering hiérarchique avec *matching minimal*.

Au cours de la comparaison des résultats, il convient de distinguer les méthodes utilisant un matching optimal ou minimal, car ce paramètre influe directement sur la mesure de qualité.

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

Pour cette raison, un premier groupe est formé par les méthodes 1, 2 et 4 (matching optimal), et une second groupe par les méthodes 3 et 5 (matching minimal). Les résultats sont présentés dans les Tables A.2 à A.8.

	matching optimal			matching minimal	
	Méthode 1	Méthode 2	Méthode 4	Méthode 3	Méthode 5
ds1	9356±1487	9567±1549	**9289±1528**	10140±1417	**9640±1473**
ds2	8909±1353	9112±1471	**8862±1369**	9740±1278	**9334±1290**
ds3	9077±1466	9325±1564	**8986±1462**	9849±1466	**9511±1370**
ds4	8890±1186	9129±1346	**8800±1268**	9771±1141	**9431±1160**
ds5	9218±1283	9441±1343	**9115±1348**	9986±1306	**9571±1277**
ds6	9361±1359	9582±1444	**9135±1354**	10098±1310	**9783±1339**
ds7	9341±1252	9590±1368	**9222±1375**	10240±1313	**9738±1341**
ds8	9088±1211	9325±1303	**9007±1213**	9858±1155	**9428±1162**
ds9	9050±1172	9248±1357	**8903±1203**	9752±1244	**9342±1294**
ds10	9235±1355	9519±1511	**9163±1395**	9916±1366	**9494±1350**

TABLE A.2 – Intra_max - distance maximale entre deux points

	Méthode 1	Méthode 2	Méthode 4	Méthode 3	Méthode 5
ds1	1483±284	**1480±282**	1488±290	2929±557	**2547±388**
ds2	1471±280	1474±282	**1469±285**	2911±558	**2517±377**
ds3	1493±288	1496±284	**1490±297**	2940±569	**2559±390**
ds4	**1506±279**	1514±276	1509±280	2966±559	**2570±383**
ds5	1490±275	**1487±278**	1491±281	2932±553	**2542±386**
ds6	1475±282	**1472±284**	1475±281	2915±562	**2548±388**
ds7	1539±286	**1536±283**	1536±296	2990±573	**2589±389**
ds8	1485±282	1495±282	**1481±284**	2948±558	**2549±378**
ds9	1488±276	**1482±278**	1485±276	2938±559	**2552±377**
ds10	**1455±273**	1458±274	1461±272	2904±552	**2523±372**

TABLE A.3 – Intra_avg - distance moyenne entre deux points

	Méthode 1	Méthode 2	Méthode 4	Méthode 3	Méthode 5
ds1	1150±264	937±290	**761±278**	2134±755	**1864±304**
ds2	1158±261	948±294	**732±260**	2126±768	**1849±317**
ds3	1161±262	935±293	**739±253**	2127±758	**1874±332**
ds4	1171±260	940±299	**756±265**	2141±763	**1863±341**
ds5	1176±265	957±293	**752±257**	2127±756	**1849±296**
ds6	1168±264	950±297	**754±272**	2117±753	**1829±285**
ds7	1181±263	958±302	**766±268**	2136±767	**1844±326**
ds8	1162±263	957±291	**744±258**	2131±759	**1852±344**
ds9	1157±259	936±296	**749±259**	2136±760	**1870±275**
ds10	1145±259	939±293	**748±260**	2133±746	**1817±290**

TABLE A.4 – Intra_centre - distance moyenne au centroïde

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

	Méthode 1	Méthode 2	Méthode 4	Méthode 3	Méthode 5
ds1	22.46±21.43	23.92±23.82	**37.03±34.82**	**607.89±129.19**	570.99±141.36
ds2	22.79±21.64	23.99±23.95	**36.22±34.37**	**609.01±134.13**	593.95±173.95
ds3	23.11±21.91	24.15±24.09	**36.92±34.62**	599.22±131.24	**601.01±168.75**
ds4	22.58±21.48	24.14±24.09	**36.14±34.07**	**618.96±131.84**	572.23±143.74
ds5	22.79±21.63	24.21±24.15	**36.77±34.8**	**611.37±132.65**	587.01±154.19
ds6	22.86±21.75	24.35±24.26	**36.68±34.36**	**611.14±130.9**	542.37±138.55
ds7	22.9±21.7	24.36±24.3	**36.49±34.3**	**615.71±131.66**	561.66±143.7
ds8	22.84±21.83	23.82±23.7	**36.74±34.8**	**625.41±142.66**	579.21±163.46
ds9	22.78±21.64	23.73±23.6	**36.43±34.26**	**608.57±128.76**	579.93±130.06
ds10	22.64±21.46	23.62±23.47	**36.73±34.75**	**610.38±145.21**	565.44±125.65

TABLE A.5 – Inter_single - distance entre les points les plus proches

	Méthode 1	Méthode 2	Méthode 4	Méthode 3	Méthode 5
ds1	8011±685	8809±648	**9911±938**	**8809±619**	8465±1004
ds2	8026±730	8788±630	**9868±893**	**8842±561**	8451±896
ds3	7950±645	8741±539	**9836±831**	**8726±487**	8367±828
ds4	7923±698	8664±525	**9749±885**	**8781±544**	8329±917
ds5	8001±678	8748±604	**9928±802**	**8850±530**	8547±892
ds6	7956±687	8751±546	**9907±921**	**8761±577**	8359±817
ds7	7912±657	8658±545	**9791±809**	**8820±566**	8383±912
ds8	8034±684	8772±552	**9924±897**	**8915±550**	8534±880
ds9	7932±719	8680±548	**9739±907**	**8797±610**	8390±931
ds10	8063±652	8868±537	**9993±947**	**8813±604**	8545±838

TABLE A.6 – Inter_complete - distance entre les points les plus éloignés

	Méthode 1	Méthode 2	Méthode 4	Méthode 3	Méthode 5
ds1	1189±270	**1273±399**	1268±524	**1849±874**	1534±346
ds2	1194±273	**1290±427**	1147±445	**1852±901**	1551±365
ds3	1205±270	**1294±396**	1233±479	**1838±878**	1536±349
ds4	1201±280	**1302±401**	1265±478	**1849±877**	1550±389
ds5	1204±276	**1295±421**	1207±456	**1843±884**	1512±328
ds6	1193±275	**1304±408**	1217±489	**1832±866**	1514±317
ds7	1198±280	**1295±407**	1220±421	**1814±868**	1511±363
ds8	1211±278	**1290±419**	1201±463	**1836±883**	1530±382
ds9	1188±274	**1280±403**	1256±481	**1845±881**	1538±303
ds10	1190±275	**1292±392**	1181±451	**1856±876**	1473±304

TABLE A.7 – Inter_centre - distance entre les centroïdes

Il est difficile de tirer des conclusions définitives de ces résultats, hormis le fait que le k-medoids original produit des clusterings de moindre qualité dans tous les cas. Parmi les méthodes utilisant un matching optimal, la méthode 4 (hiérarchique) donne de meilleurs résultats sur quatre des sept mesures, et la méthode 2 est meilleure sur deux des mesures. Parmi celles utilisant un matching minimal, les résultats sont encore plus partagés, avec une meilleure performance sur quatre mesures pour la méthode 3, et sur les trois restantes pour la 5. Une observation intéressante est que les algorithmes de clustering hiérarchique tendent à produire de meilleurs résultats sur

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

	Méthode 1	Méthode 2	Méthode 4	Méthode 3	Méthode 5
ds1	5.13±14.93	**0.85±0.44**	2.46±2.43	**1.71±0.45**	1.9±0.59
ds2	3.25±8.03	**0.85±0.41**	2.12±1.77	**1.74±0.47**	1.87±0.46
ds3	3.56±7.75	**0.85±0.45**	1.96±1.28	**1.72±0.46**	1.96±0.68
ds4	4.36±9.92	**0.89±0.48**	1.95±1.44	**1.72±0.46**	1.86±0.43
ds5	4.17±11.79	**0.83±0.4**	1.98±1.4	**1.73±0.47**	1.86±0.49
ds6	6.05±18.05	**0.87±0.48**	2.82±3.99	**1.72±0.43**	1.83±0.42
ds7	3.53±6.55	**0.85±0.45**	2.68±3.87	**1.76±0.46**	2.0±0.78
ds8	5.71±18.02	**0.87±0.46**	2.08±1.32	**1.76±0.46**	1.89±0.45
ds9	5.56±22.04	**0.83±0.41**	2.19±1.84	**1.72±0.48**	1.89±0.67
ds10	3.67±7.89	**0.86±0.46**	2.9±6.17	**1.7±0.45**	1.96±0.57

TABLE A.8 – Composite - index Davies-Bouldin

les mesures intra-cluster, tandis que les k-medoids sont en général meilleurs sur les mesures inter-cluster. Dans cette situation, on peut naturellement vouloir utiliser la mesure composite pour départager ces approches. D'après celle-ci, les méthodes k-medoids (2 et 3) donnent de meilleurs résultats que les autres. Cependant, cette mesure n'est pas toujours un indicateur objectif de la qualité d'un clustering. Par exemple, des expérimentations ont révélé que l'index DB d'un clustering contenant un fort *chaining effect* est très bon, alors même que quatre des cinq clusters n'incluent qu'un unique point chacun.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Bibliographie

[1] Classification commune des actes médicaux (CCAM). `http://www.ameli.fr/accueil-de-la-ccam/index.php/`.

[2] Classification internationale des maladies (CIM-10). `http://apps.who.int/classifications/icd10/browse/2008/fr`.

[3] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. On demand classification of data streams. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 503–508. ACM, 2004.

[4] Rakesh Agrawal, Christos Faloutsos, and Arun Swami. *Efficient similarity search in sequence databases*. Springer, 1993.

[5] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[6] Jesús Alcalá-Fdez, Luciano Sanchez, Salvador Garcia, Maria Jose del Jesus, Sebastian Ventura, Josep Maria Garrell, Jose Otero, Cristobal Romero, Jaume Bacardit, Victor M Rivas, et al. Keel : a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing*, 13(3) :307–318, 2009.

[7] Andrés M. Alonso, José Ramón Berrendero, Adolfo Hernández, and Ana Justel. Time series clustering based on forecast densities. *Computational Statistics & Data Analysis*, 51(2) :762–776, 2006.

[8] Faris Alqadah, Chandan K. Reddy, Junling Hu, and Hatim F. Alqadah. Biclustering neighborhood-based collaborative filtering method for top-n recommender systems. *Knowledge and Information Systems*, 44(2) :475–491, 2015.

[9] Gene M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pages 483–485. ACM, 1967.

[10] William J Anderson. *Continuous-time Markov chains : An applications-oriented approach*. Springer Science & Business Media, 2012.

[11] Cláudia M. Antunes and Arlindo L. Oliveira. Temporal data mining : An overview. In *KDD workshop on temporal data mining*, volume 1, page 13, 2001.

[12] N Baba. Convergence of a random optimization method for constrained optimization problems. *Journal of Optimization Theory and Applications*, 33(4) :451–461, 1981.

[13] Jaume Bacardit, Edmund K. Burke, and Natalio Krasnogor. Improving the scalability of rule-based evolutionary learning. *Memetic Computing*, 1(1) :55–67, 2009.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

[14] Simon Barkow, Stefan Bleuler, Amela Prelić, Philip Zimmermann, and Eckart Zitzler. BicAT : a biclustering analysis toolbox. *Bioinformatics*, 22(10) :1282–1283, 2006.

[15] Roberto Battiti, Mauro Brunato, and Franco Mascia. *Reactive search and intelligent optimization*, volume 45. Springer Science & Business Media, 2008.

[16] Amir Ben-Dor, Benny Chor, Richard Karp, and Zohar Yakhini. Discovering local structure in gene expression data : the order-preserving submatrix problem. *Journal of computational biology*, 10(3-4) :373–384, 2003.

[17] Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[18] Michael R. Berthold, Nicolas Cebron, Fabian Dill, Thomas R. Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Kilian Thiel, and Bernd Wiswedel. Knime-the konstanz information miner : version 2.0 and beyond. *AcM SIGKDD explorations Newsletter*, 11(1) :26–31, 2009.

[19] Nicola Beume, Boris Naujoks, and Michael Emmerich. SMS-EMOA : Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3) :1653–1669, 2007.

[20] Stefan Bleuler and Eckart Zitzler. Order preserving clustering over multiple time course experiments. In *Applications of Evolutionary Computing*, pages 33–43. Springer, 2005.

[21] Doruk Bozdağ, Ashwin S. Kumar, and Umit V. Catalyurek. Comparative analysis of biclustering algorithms. In *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology*, pages 265–274. ACM, 2010.

[22] Leo Breiman, Jerome Friedman, Charles J. Stone, and Richard A. Olshen. *Classification and regression trees*. CRC press, 1984.

[23] Jack Brimberg, Zvi Drezner, Nenad Mladenović, and Said Salhi. A new local search for continuous location problems. *European Journal of Operational Research*, 232(2) :256–265, 2014.

[24] Aydin Buluc, Jeremy T. Fineman, Matteo Frigo, John R. Gilbert, and Charles E. Leiserson. Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks. In *IN SPAA*, pages 233–244, 2009.

[25] Stanislav Busygin, Gerrit Jacobsen, Ewald Krämer, and Contentsoft Ag. Double conjugated clustering applied to leukemia microarray data. In *In 2nd SIAM ICDM, Workshop on clustering high dimensional data*. Citeseer, 2002.

[26] Stanislav Busygin, Oleg Prokopyev, and Panos M. Pardalos. Biclustering in data mining. *Computers & Operations Research*, 35(9) :2964–2987, 2008.

[27] José Caldas and Samuel Kaski. Bayesian biclustering with the plaid model. In *Machine Learning for Signal Processing, 2008. MLSP 2008. IEEE Workshop on*, pages 291–296. IEEE, 2008.

[28] Andrea Califano, Gustavo Stolovitzky, and Yuhai Tu. Analysis of gene expression microarrays for phenotype classification. In *Intelligent Systems for Molecular Biology*, volume 8, pages 75–85, 2000.

[29] Jason Catlett. On changing continuous attributes into ordered discrete attributes. In *European working session on learning*, pages 164–178. Springer, 1991.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

[30] Kin-Pong Chan and Ada Wai-Chee Fu. Efficient time series matching by wavelets. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 126–133. IEEE, 1999.

[31] Yizong Cheng and George M. Church. Biclustering of expression data. In *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*, volume 8, pages 93–103, 2000.

[32] Edward Choi, Mohammad Taha Bahadori, and Jimeng Sun. Doctor AI : Predicting clinical events via recurrent neural networks. *arXiv preprint arXiv :1511.05942*, 2015.

[33] Jonathan D. Cohen, Samuel M. McClure, and J. Yu Angela. Should i stay or should i go ? how the human brain manages the trade-off between exploitation and exploration. *Philosophical Transactions of the Royal Society of London B : Biological Sciences*, 362(1481) :933–942, 2007.

[34] William W. Cohen. Fast effective rule induction. In *Proceedings of the twelfth international conference on machine learning*, pages 115–123, 1995.

[35] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7 : A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011.

[36] Arthur L. Corcoran and Sandip Sen. Using real-valued genetic algorithms to evolve rule sets for classification. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 120–124. IEEE, 1994.

[37] Marcella Corduas and Domenico Piccolo. Time series clustering and classification by the autoregressive metric. *Computational Statistics & Data Analysis*, 52(4) :1860–1872, 2008.

[38] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3) :273–297, 1995.

[39] Paul Cotofrei and Kilian Stoffel. Classification rules+ time= temporal rules. In *International Conference on Computational Science*, pages 572–581. Springer, 2002.

[40] Hamish Cunningham, Yorick Wilks, and Robert J. Gaizauskas. Gate : a general architecture for text engineering. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 1057–1060. Association for Computational Linguistics, 1996.

[41] Ryan R. Curtin, James R. Cline, Neil P. Slagle, William B. March, Parikshit Ram, Nishant A. Mehta, and Alexander G. Gray. MLPACK : A scalable c++ machine learning library. *Journal of Machine Learning Research*, 14(Mar) :801–805, 2013.

[42] Gautam Das, Dimitrios Gunopulos, and Heikki Mannila. Finding similar time series. In *Principles of Data Mining and Knowledge Discovery*, pages 88–100. Springer, 1997.

[43] Gautam Das, King-Ip Lin, Heikki Mannila, Gopal Renganathan, and Padhraic Smyth. Rule discovery from time series. In *KDD*, volume 98, pages 16–22, 1998.

[44] David L. Davies and Donald W. Bouldin. A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (2) :224–227, 1979.

[45] Pablo A. D. de Castro, Fabrício O. de França, Hamilton M. Ferreira, and Fernando J. Von Zuben. Applying biclustering to text mining : an immune-inspired approach. In *Artificial Immune Systems*, pages 83–94. Springer, 2007.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

[46] Gilberto F. de Sousa, F. Cabral Lucidio dos Anjos, Luiz Satoru Ochi, and Fábio Protti. Hybrid metaheuristic for bicluster editing problem. *Electronic Notes in Discrete Mathematics*, 39 :35–42, 2012.

[47] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. A. M. T. Meyarivan. A fast and elitist multiobjective genetic algorithm : NSGA-II. *IEEE transactions on evolutionary computation*, 6(2) :182–197, 2002.

[48] Daniel Defays. An efficient algorithm for a complete link method. *The Computer Journal*, 20(4) :364–366, 1977.

[49] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan) :1–30, 2006.

[50] Janez Demšar, Blaž Zupan, Gregor Leban, and Tomaz Curk. Orange : From experimental machine learning to interactive data mining. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 537–539. Springer, 2004.

[51] Smitha Dharan and Achuthsankar S. Nair. Biclustering of gene expression data using reactive greedy randomized adaptive search procedure. *BMC bioinformatics*, 10(1) :1, 2009.

[52] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274. ACM, 2001.

[53] Chris Ding, Ya Zhang, Tao Li, and Stephen R. Holbrook. Biclustering protein complex interactions with a biclique finding algorithm. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 178–187. IEEE, 2006.

[54] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data : experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2) :1542–1552, 2008.

[55] James Dougherty, Ron Kohavi, Mehran Sahami, et al. Supervised and unsupervised discretization of continuous features. In *Machine learning : proceedings of the twelfth international conference*, volume 12, pages 194–202, 1995.

[56] Jason Ernst and Ziv Bar-Joseph. STEM : a tool for the analysis of short time series gene expression data. *BMC bioinformatics*, 7(1) :1, 2006.

[57] Jason Ernst, Gerard J. Nau, and Ziv Bar-Joseph. Clustering short time series gene expression data. *Bioinformatics*, 21(suppl 1) :i159–i168, 2005.

[58] Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *13th International Joint Conference on Uncertainty in Artificial Intelligence (IJCAI93)*, pages 1022–1029, 1993.

[59] Alberto Fernández, Salvador García, Julián Luengo, Ester Bernadó-Mansilla, and Francisco Herrera. Genetics-based machine learning for rule induction : state of the art, taxonomy, and comparative study. *IEEE Transactions on Evolutionary Computation*, 14(6) :913–941, 2010.

[60] Álvaro Fialho, Luis Da Costa, Marc Schoenauer, and Michèle Sebag. Analyzing bandit-based adaptive operator selection mechanisms. *Annals of Mathematics and Artificial Intelligence*, 60(1-2) :25–64, 2010.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

[61] Roger Fjortoft, Yves Delignon, Wojciech Pieczynski, Marc Sigelle, and Florence Tupin. Unsupervised classification of radar images using hidden markov chains and hidden markov random fields. *IEEE Transactions on geoscience and remote sensing*, 41(3) :675–686, 2003.

[62] Mark Fleischer. The measure of pareto optima applications to multi-objective metaheuristics. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 519–533. Springer, 2003.

[63] Christodoulos A. Floudas and Chrysanthos E. Gounaris. A review of recent advances in global optimization. *Journal of Global Optimization*, 45(1) :3–38, 2009.

[64] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, 32(200) :675–701, 1937.

[65] Wojciech Froelich and Alicja Wakulicz-Deja. Mining temporal medical data using adaptive fuzzy cognitive maps. In *2009 2nd Conference on Human System Interactions*, pages 16–23. IEEE, 2009.

[66] Sergio Garcia, Julian Luengo, José Antonio Sáez, Victor López, and Francisco Herrera. A survey of discretization techniques : Taxonomy and empirical analysis in supervised learning. *Knowledge and Data Engineering, IEEE Transactions on*, 25(4) :734–750, 2013.

[67] Lisa Getoor, Nir Friedman, Daphne Koller, and Benjamin Taskar. Learning probabilistic models of link structure. *Journal of Machine Learning Research*, 3(Dec) :679–707, 2002.

[68] Gad Getz, Erel Levine, and Eytan Domany. Coupled two-way clustering analysis of gene microarray data. *Proceedings of the National Academy of Sciences*, 97(22) :12079–12084, 2000.

[69] Gabriela Guimarães. The induction of temporal grammatical rules from multivariate time series. In *International Colloquium on Grammatical Inference*, pages 127–140. Springer, 2000.

[70] Lalit Gupta, Mark McAvoy, and James Phegley. Classification of temporal sequences via prediction using the simple recurrent neural network. *Pattern Recognition*, 33(10) :1759–1770, 2000.

[71] John L. Gustafson. Reevaluating amdahl's law. *Communications of the ACM*, 31(5) :532–533, 1988.

[72] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software : an update. *ACM SIGKDD explorations newsletter*, 11(1) :10–18, 2009.

[73] James Douglas Hamilton. *Time series analysis*, volume 2. Princeton university press Princeton, 1994.

[74] James A. Hanley and Barbara J. McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1) :29–36, 1982.

[75] Michael Pilegaard Hansen and Andrzej Jaszkiewicz. *Evaluating the quality of approximations to the non-dominated set*. IMM, Department of Mathematical Modelling, Technical Universityof Denmark, 1998.

[76] Nikolaus Hansen, Dirk V. Arnold, and Anne Auger. Evolution strategies. In *Springer Handbook of Computational Intelligence*, pages 871–898. Springer, 2015.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

[77] Pierre Hansen, Nenad Mladenović, and José A. Moreno Pérez. Variable neighbourhood search : methods and applications. *4OR*, 6(4) :319–360, 2008.

[78] John A. Hartigan. Direct clustering of a data matrix. *Journal of the american statistical association*, 67(337) :123–129, 1972.

[79] Ernst Haselsteiner and Gert Pfurtscheller. Using time-dependent neural networks for EEG classification. *IEEE transactions on rehabilitation engineering*, 8(4) :457–463, 2000.

[80] Rui Henriques and Sara C. Madeira. Bicnet : Flexible module discovery in large-scale biological networks using biclustering. *Algorithms for Molecular Biology*, 11(1) :1, 2016.

[81] Tin Kam Ho. Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 278–282. IEEE, 1995.

[82] Sepp Hochreiter, Ulrich Bodenhofer, Martin Heusel, Andreas Mayr, Andreas Mitterecker, Adetayo Kasim, Tatsiana Khamiakova, Suzy Van Sanden, Dan Lin, Willem Talloen, Luc Bijnens, Hinrich W. H. Göhlmann, Ziv Shkedy, and Djork-Arné Clevert. Fabia : factor analysis for bicluster acquisition. *Bioinformatics*, 26(12) :1520–1527, 2010.

[83] Jan Ihmels, Sven Bergmann, and Naama Barkai. Defining transcription modules using large-scale gene expression data. *Bioinformatics*, 20(13) :1993–2003, 2004.

[84] Ronald L. Iman and James M. Davenport. Approximations of the critical region of the fbietkan statistic. *Communications in Statistics-Theory and Methods*, 9(6) :571–595, 1980.

[85] Athirai A. Irissappane, Siwei Jiang, and Jie Zhang. A biclustering-based approach to filter dishonest advisors in multi-criteria e-marketplaces. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1385–1386. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[86] Hisao Ishibuchi, Yusuke Nojima, and Tsutomu Doi. Comparison between single-objective and multi-objective genetic algorithms : Performance comparison and performance measures. In *2006 IEEE International Conference on Evolutionary Computation*, pages 1143–1150. IEEE, 2006.

[87] Hisao Ishibuchi, Ken Nozaki, Naohisa Yamamoto, and Hideo Tanaka. Selecting fuzzy if-then rules for classification problems using genetic algorithms. *IEEE Transactions on fuzzy systems*, 3(3) :260–270, 1995.

[88] Hisao Ishibuchi, Noritaka Tsukamoto, and Yusuke Nojima. Iterative approach to indicator-based multiobjective optimization. In *2007 IEEE Congress on Evolutionary Computation*, pages 3967–3974. IEEE, 2007.

[89] Julie Jacques. *Classification sur données médicales à l'aide de méthodes d'optimisation et de datamining, appliquée au pré-screening dans les essais cliniques.* PhD thesis, Université de Lille 1, 2013.

[90] Julie Jacques, Julien Taillard, David Delerue, Laetitia Jourdan, and Clarisse Dhaenens. The benefits of using multi-objectivization for mining pittsburgh partial classification rules in imbalanced and discrete data. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 543–550. ACM, 2013.

[91] Stephen C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3) :241–254, 1967.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

[92] Laetitia Jourdan, Matthieu Basseur, and E.-G. Talbi. Hybridizing exact methods and metaheuristics : A taxonomy. *European Journal of Operational Research*, 199(3) :620–629, 2009.

[93] Konstantinos Kalpakis, Dhiral Gada, and Vasundhara Puttagunta. Distance measures for effective clustering of ARIMA time-series. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 273–280. IEEE, 2001.

[94] Adetayo Kasim, Ziv Shkedy, Sebastian Kaiser, Sepp Hochreiter, and Willem Talloen. *Applied Biclustering Methods for Big and High-Dimensional Data Using R*. CRC Press, 2016.

[95] Leonard Kaufman and Peter Rousseeuw. *Clustering by means of medoids*. North-Holland, 1987.

[96] Eamonn Keogh and Jessica Lin. Clustering of time-series subsequences is meaningless : implications for previous and future research. *Knowledge and information systems*, 8(2) :154–177, 2005.

[97] Eamonn J. Keogh and Padhraic Smyth. A probabilistic approach to fast pattern matching in time series databases. In *KDD*, volume 1997, pages 24–30, 1997.

[98] Scott Kirkpatrick, C. Daniel Gelatt, and Mario P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598) :671–680, 1983.

[99] Yuval Kluger, Ronen Basri, Joseph T. Chang, and Mark Gerstein. Spectral biclustering of microarray data : coclustering genes and conditions. *Genome research*, 13(4) :703–716, 2003.

[100] Joshua D. Knowles, Richard A. Watson, and David W. Corne. Reducing local optima in single-objective problems by multi-objectivization. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 269–283. Springer, 2001.

[101] Donald E. Knuth, James H. Jr. Morris, and Vaughan R Pratt. Fast pattern matching in strings. *SIAM journal on computing*, 6(2) :323–350, 1977.

[102] Daphne Koller and Mehran Sahami. Toward optimal feature selection. 1996.

[103] Flip Korn, Hosagrahar V. Jagadish, and Christos Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. *ACM SIGMOD Record*, 26(2) :289–300, 1997.

[104] Alexander Korytny, Klaris Riesenberg, Lisa Saidel-Odes, Fransisc Schlaeffer, and Abraham Borer. Bloodstream infections caused by multi-drug resistant proteus mirabilis : Epidemiology, risk factors and impact of multi-drug resistance. *Infectious Diseases*, 48(6) :428–431, 2016.

[105] Sandeep Kumar. *Classification and detection of computer intrusions*. PhD thesis, Purdue University, 1995.

[106] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields : Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289, 2001.

[107] Terran Lane and Carla E. Brodley. Temporal sequence learning and data reduction for anomaly detection. *ACM Transactions on Information and System Security (TISSEC)*, 2(3) :295–331, 1999.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

[108] Kevin J. Lang, Barak A. Pearlmutter, and Rodney A. Price. Results of the abbadingo one dfa learning competition and a new evidence-driven state merging algorithm. In *International Colloquium on Grammatical Inference*, pages 1–12. Springer, 1998.

[109] Tilman Lange, Volker Roth, Mikio L Braun, and Joachim M Buhmann. Stability-based validation of clustering solutions. *Neural computation*, 16(6) :1299–1323, 2004.

[110] Laura Lazzeroni and Art Owen. Plaid models for gene expression data. *Statistica sinica*, pages 61–86, 2002.

[111] Jean-Roger Le Gall, Stanley Lemeshow, and Fabienne Saulnier. A new simplified acute physiology score (SAPS II) based on a european/north american multicenter study. *Jama*, 270(24) :2957–2963, 1993.

[112] Neal Lesh, Mohammed J. Zaki, and Mitsunori Ogihara. Mining features for sequence classification. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 342–346. ACM, 1999.

[113] Fan Li, Min Li, Peng Guan, Shuang Ma, and Lei Cui. Mapping publication trends and identifying hot spots of research on internet health information seeking behavior : A quantitative and co-word biclustering analysis. *Journal of medical Internet research*, 17(3) :e81, 2015.

[114] Miqing Li, Shengxiang Yang, and Xiaohui Liu. Diversity comparison of pareto front approximations in many-objective optimization. *IEEE Transactions on Cybernetics*, 44(12) :2568–2584, 2014.

[115] M. Lichman. UCI machine learning repository, 2013.

[116] Arnaud Liefooghe, Jérémie Humeau, Salma Mesmoudi, Laetitia Jourdan, and El-Ghazali Talbi. On dominance-based multiobjective local search : design, implementation and experimental analysis on scheduling and traveling salesman problems. *Journal of Heuristics*, 18(2) :317–352, 2012.

[117] Xuan Liu, Pengzhu Zhang, and Dajun Zeng. Sequence matching for suspicious activity detection in anti-money laundering. In *International Conference on Intelligence and Security Informatics*, pages 50–61. Springer, 2008.

[118] R López. Open nn : An open source neural networks c++ library. Technical report, Technical report, Open Neural Network Library, 2014. http ://www. cimne. com/flood, 2015.

[119] B. L. W. H. Y. Ma. Integrating classification and association rule mining. In *Proceedings of the fourth international conference on knowledge discovery and data mining*, 1998.

[120] Sara C. Madeira and Arlindo L. Oliveira. A polynomial time biclustering algorithm for finding approximate expression patterns in gene expression time series. *Algorithms for Molecular Biology*, 4(1) :8, 2009.

[121] Brian W. Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2) :442–451, 1975.

[122] Ryszard S. Michalski. A theory and methodology of inductive learning. In *Machine learning*, pages 83–134. Springer, 1983.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

[123] Daniel Molina, Manuel Lozano, Ana M. Sánchez, and Francisco Herrera. Memetic algorithms based on local search chains for large scale continuous optimisation problems : Ma-ssw-chains. *Soft Computing*, 15(11) :2201–2220, 2011.

[124] Carla S. Möller-Levet, Frank Klawonn, Kwang-Hyun Cho, and Olaf Wolkenhauer. Fuzzy clustering of short time-series and unevenly distributed sampling points. In *Advances in Intelligent Data Analysis V*, pages 330–340. Springer, 2003.

[125] David W. Mount. Sequence and genome analysis. *Bioinformatics : Cold Spring Harbour Laboratory Press : Cold Spring Harbour*, 2, 2004.

[126] T. M. Murali and Simon Kasif. Extracting conserved gene expression motifs from gene expression data. In *Pacific symposium on biocomputing*, volume 8, pages 77–88, 2003.

[127] Peter Nemenyi. Distribution-free multiple comparisons. In *Biometrics*, volume 18, page 263, 1962.

[128] Juan A. Nepomuceno, Alicia Troncos, and Jesús S. Aguilar-Ruiz. Evolutionary metaheuristic for biclustering based on linear correlations among genes. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1143–1147. ACM, 2010.

[129] Juan A. Nepomuceno, Alicia Troncoso, and Jesús S. Aguilar-Ruiz. Biclustering of gene expression data by correlation-based scatter search. *BioData mining*, 4(1) :1, 2011.

[130] Tim Oates. Identifying distinctive subsequences in multivariate time series by clustering. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 322–326. ACM, 1999.

[131] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn : Machine learning in python. *Journal of Machine Learning Research*, 12(Oct) :2825–2830, 2011.

[132] Donald B. Percival and Andrew T. Walden. *Wavelet methods for time series analysis*, volume 4. Cambridge university press, 2006.

[133] Beatriz Pontes, Raúl Giráldez, and Jesús S Aguilar-Ruiz. Biclustering on expression data : A review. *Journal of biomedical informatics*, 57 :163–180, 2015.

[134] Beatriz Pontes, Ral Girldez, and Jess S. Aguilar-Ruiz. Quality measures for gene expression biclusters. *PloS one*, 10(3) :e0115497, 2015.

[135] Amela Prelić, Stefan Bleuler, Philip Zimmermann, Anja Wille, Peter Bühlmann, Wilhelm Gruissem, Lars Hennig, Lothar Thiele, and Eckart Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9) :1122–1129, 2006.

[136] Maurice Bertram Priestley. Spectral analysis and time series. 1981.

[137] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1) :81–106, 1986.

[138] Sridhar Ramaswamy, Sameer Mahajan, and Abraham Silberschatz. On the discovery of interesting patterns in association rules. In *VLDB*, volume 98, pages 368–379. Citeseer, 1998.

[139] Marco F. Ramoni, Paola Sebastiani, and Isaac S. Kohane. Cluster analysis of gene expression dynamics. *Proceedings of the National Academy of Sciences*, 99(14) :9121–9126, 2002.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

[140] David J. Reiss, Nitin S. Baliga, and Richard Bonneau. Integrated biclustering of heterogeneous genome-wide datasets for the inference of global regulatory networks. *BMC bioinformatics*, 7(1) :280, 2006.

[141] Jiadong Ren and Haiyan Tian. Sequential pattern mining with inaccurate event in temporal sequence. In *Networked Computing and Advanced Information Management, 2008. NCM'08. Fourth International Conference on*, volume 2, pages 659–664. IEEE, 2008.

[142] Brian D. Ripley. *Pattern recognition and neural networks*. Cambridge university press, 2007.

[143] Domingo S. Rodriguez-Baena, Antonio J. Perez-Pulido, and Jesus S. Aguilar. A biclustering algorithm for extracting bit-patterns from binary datasets. *Bioinformatics*, 27(19) :2738–2745, 2011.

[144] Cristóbal Romero, José Raúl Romero, Jose María Luna, and Sebastián Ventura. Mining rare association rules from e-learning data. In *Educational Data Mining 2010*, 2010.

[145] Pal Saetrom and Magnus Lie Hetland. Multiobjective evolution of temporal rules. In *Proc. 8th Scandinavian Conf. on Artificial Intelligence, SCAI. IOS Press*, 2003.

[146] M. S. Sarma. On the convergence of the baba and dorea random optimization methods. *Journal of Optimization Theory and Applications*, 66(2) :337–343, 1990.

[147] Mark Schena, Dari Shalon, Ronald W. Davis, and Patrick O. Brown. Quantitative monitoring of gene expression patterns with a complementary dna microarray. *Science*, 270(5235) :467, 1995.

[148] Khedidja Seridi, Laetitia Jourdan, and El-Ghazali Talbi. Multiobjective path relinking for biclustering : Application to microarray data. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 200–214. Springer, 2013.

[149] Fei Sha and Lawrence K. Saul. Large margin hidden markov models for automatic speech recognition. In *Advances in neural information processing systems*, pages 1249–1256, 2006.

[150] Juliet Popper Shaffer. Multiple hypothesis testing. *Annual review of psychology*, 46 :561, 1995.

[151] Qizheng Sheng, Yves Moreau, and Bart De Moor. Biclustering microarray data by gibbs sampling. *Bioinformatics*, 19(suppl 2) :ii196–ii205, 2003.

[152] Robin Sibson. SLINK : an optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1) :30–34, 1973.

[153] Francisco J. Solis and Roger J.-B. Wets. Minimization by random search techniques. *Mathematics of operations research*, 6(1) :19–30, 1981.

[154] Amos Tanay, Roded Sharan, and Ron Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(suppl 1) :S136–S144, 2002.

[155] Chun Tang, Li Zhang, Aidong Zhang, and Murali Ramanathan. Interrelated two-way clustering : an unsupervised approach for gene expression data analysis. In *Bioinformatics and Bioengineering Conference, 2001. Proceedings of the IEEE 2nd International Symposium on*, pages 41–48. IEEE, 2001.

[156] Henry S. Teng, Kaihu Chen, and Stephen C. Lu. Adaptive real-time anomaly detection using inductively generated sequential patterns. In *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on*, pages 278–284. IEEE, 1990.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

[157] Dirk Thierens. An adaptive pursuit strategy for allocating operator probabilities. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 1539–1546. ACM, 2005.

[158] Kai Ming Ting. An instance-weighting method to induce cost-sensitive trees. *IEEE Transactions on Knowledge and Data Engineering*, 14(3) :659–665, 2002.

[159] Miranda van Uitert, Wouter Meuleman, and Lodewyk Wessels. Biclustering sparse binary genomic data. *Journal of Computational Biology*, 15(10) :1329–1345, 2008.

[160] Tim Van den Bulcke. *Robust algorithms for inferring regulatory networks based on gene expression measurements and biological prior information.* PhD thesis, Universiteit Gent, 2009.

[161] CJ van Rijsbergen. Information retrieval, 1979.

[162] Juha Vesanto and Esa Alhoniemi. Clustering of the self-organizing map. *Neural Networks, IEEE Transactions on*, 11(3) :586–600, 2000.

[163] Ricardo Vilalta and Sheng Ma. Predicting rare events in temporal domains. In *Data Mining. IEEE International Conference on*, pages 474–481. IEEE, 2002.

[164] Michail Vlachos, George Kollios, and Dimitrios Gunopulos. Discovering similar multidimensional trajectories. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 673–684. IEEE, 2002.

[165] Shinya Watanabe and Kazutoshi Sakakibara. A multiobjectivization approach for vehicle routing problems. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 660–672. Springer, 2007.

[166] Li Wei and Eamonn Keogh. Semi-supervised time series classification. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 748–753. ACM, 2006.

[167] Li Wei and Eamonn Keogh. Semi-supervised time series classification. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 748–753. ACM, 2006.

[168] William Wu-Shyong Wei. *Time series analysis.* Addison-Wesley publ Reading, 1994.

[169] Lyndon While, Philip Hingston, Luigi Barone, and Simon Huband. A faster algorithm for calculating hypervolume. *IEEE transactions on evolutionary computation*, 10(1) :29–38, 2006.

[170] Zhengzheng Xing, Jian Pei, and Philip S. Yu. Early classification on time series. *Knowledge and information systems*, 31(1) :105–127, 2012.

[171] Yimin Xiong and Dit-Yan Yeung. Time series clustering with arma mixtures. *Pattern Recognition*, 37(8) :1675–1689, 2004.

[172] Oksana Yakhnenko, Adrian Silvescu, and Vasant Honavar. Discriminatively trained markov model for sequence classification. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 8–pp. IEEE, 2005.

[173] Junji Yamato, Jun Ohya, and Kenichiro Ishii. Recognizing human action in time-sequential images using hidden markov model. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*, pages 379–385. IEEE, 1992.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Maxence Vandromme, Lille 1, 2017

[174] Jiong Yang, Haixun Wang, Wei Wang, and Philip S. Yu. An improved biclustering method for analyzing gene expression profiles. *International Journal on Artificial Intelligence Tools*, 14(05) :771–789, 2005.

[175] Jiong Yang, Wei Wang, Haixun Wang, and Philip S. Yu. δ-clusters : Capturing subspace correlation in a large data set. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 517–528. IEEE, 2002.

[176] Byoung-Kee Yi and Christos Faloutsos. Fast time sequence indexing for arbitrary Lp norms. In *Proceedings of the 26th International Conference on Very Large Data Bases*, pages 1–24. VLDB, 2000.

[177] Jianhong Zhou and Ashfaq Khokhar. Parrescue : Scalable parallel algorithm and implementation for biclustering over large distributed datasets. In *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*, pages 21–21. IEEE, 2006.

[178] Eckart Zitzler and Simon Künzli. Indicator-based selection in multiobjective search. In *International Conference on Parallel Problem Solving from Nature*, pages 832–842. Springer, 2004.

[179] Eckart Zitzler, Marco Laumanns, Lothar Thiele, et al. Spea2 : Improving the strength pareto evolutionary algorithm. In *Eurogen*, volume 3242, pages 95–100, 2001.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr