No d'ordre 04-ISAL-0095 Année 2004

Thèse

Recherche automatique des fenêtres temporelles optimales des motifs séquentiels

Présentée devant
L'Institut National des Sciences Appliquées de Lyon

Pour obtenir
Le grade de docteur

École doctorale : Informatique et Information pour la Societé (EDIIS)
Spécialité : Informatique - Extraction des Connaissances à partir des Données (ECD)

par

Nicolas MÉGER
(ingénieur)

Soutenue le 15 décembre 2004
Jury

Jean-François BOULICAUT	Directeur	Maître de Conférences HDR à l'INSA de Lyon
Marie-Odile CORDIER	Examinatrice	Professeur à l'Université Rennes 1
Bruno CRÉMILLEUX	Examinateur	Maître de Conférences à l'Université de Caen
Dominique LAURENT	Rapporteur	Professeur à l'Université de Cergy-Pontoise
Pascal PONCELET	Rapporteur	Professeur à l'Ecole des Mines d'Alès
Christophe RIGOTTI	Directeur	Maître de Conférences à l'INSA de Lyon

Cette thèse a été préparée au Laboratoire d'InfoRmatique en Image et Systèmes d'information (FRE 2672 CNRS) de l'INSA de Lyon

Novembre 2003

INSTITUT NATIONAL DES SCIENCES APPLIQUEES DE LYON

***Directeur* : STORCK A.**

***Professeurs* :**

AMGHAR Y.	LIRIS
AUDISIO S.	PHYSICOCHIMIE INDUSTRIELLE
BABOT D.	CONT. NON DESTR. PAR RAYONNEMENTS IONISANTS
BABOUX J.C.	GEMPPM***
BALLAND B.	PHYSIQUE DE LA MATIERE
BAPTISTE P.	PRODUCTIQUE ET INFORMATIQUE DES SYSTEMES MANUFACTURIERS
BARBIER D.	PHYSIQUE DE LA MATIERE
BASKURT A.	LIRIS
BASTIDE J.P.	LAEPSI****
BAYADA G.	MECANIQUE DES CONTACTS
BENADDA B.	LAEPSI****
BETEMPS M.	AUTOMATIQUE INDUSTRIELLE
BIENNIER F.	PRODUCTIQUE ET INFORMATIQUE DES SYSTEMES MANUFACTURIERS
BLANCHARD J.M.	LAEPSI****
BOISSE P.	LAMCOS
BOISSON C.	VIBRATIONS-ACOUSTIQUE
BOIVIN M. (Prof. émérite)	MECANIQUE DES SOLIDES
BOTTA H.	UNITE DE RECHERCHE EN GENIE CIVIL - Développement Urbain
BOTTA-ZIMMERMANN M. (Mme)	UNITE DE RECHERCHE EN GENIE CIVIL - Développement Urbain
BOULAYE G. (Prof. émérite)	INFORMATIQUE
BOYER J.C.	MECANIQUE DES SOLIDES
BRAU J.	CENTRE DE THERMIQUE DE LYON - Thermique du bâtiment
BREMOND G.	PHYSIQUE DE LA MATIERE
BRISSAUD M.	GENIE ELECTRIQUE ET FERROELECTRICITE
BRUNET M.	MECANIQUE DES SOLIDES
BRUNIE L.	INGENIERIE DES SYSTEMES D'INFORMATION
BUFFIERE J-Y.	GEMPPM***
BUREAU J.C.	CEGELY*
CAMPAGNE J-P.	PRISMA
CAVAILLE J.Y.	GEMPPM***
CHAMPAGNE J-Y.	LMFA
CHANTE J.P.	CEGELY*- Composants de puissance et applications
CHOCAT B.	UNITE DE RECHERCHE EN GENIE CIVIL - Hydrologie urbaine
COMBESCURE A.	MECANIQUE DES CONTACTS
COURBON	GEMPPM
COUSIN M.	UNITE DE RECHERCHE EN GENIE CIVIL - Structures
DAUMAS F. (Mme)	CENTRE DE THERMIQUE DE LYON - Energétique et Thermique
DJERAN-MAIGRE I.	UNITE DE RECHERCHE EN GENIE CIVIL
DOUTHEAU A.	CHIMIE ORGANIQUE
DUBUY-MASSARD N.	ESCHIL
DUFOUR R.	MECANIQUE DES STRUCTURES
DUPUY J.C.	PHYSIQUE DE LA MATIERE
EMPTOZ H.	RECONNAISSANCE DE FORMES ET VISION
ESNOUF C.	GEMPPM***
EYRAUD L. (Prof. émérite)	GENIE ELECTRIQUE ET FERROELECTRICITE
FANTOZZI G.	GEMPPM***
FAVREL J.	PRODUCTIQUE ET INFORMATIQUE DES SYSTEMES MANUFACTURIERS
FAYARD J.M.	BIOLOGIE FONCTIONNELLE, INSECTES ET INTERACTIONS
FAYET M. (Prof. émérite)	MECANIQUE DES SOLIDES
FAZEKAS A.	GEMPPM
FERRARIS-BESSO G.	MECANIQUE DES STRUCTURES
FLAMAND L.	MECANIQUE DES CONTACTS
FLEURY E.	CITI
FLORY A.	INGENIERIE DES SYSTEMES D'INFORMATIONS
FOUGERES R.	GEMPPM***
FOUQUET F.	GEMPPM***
FRECON L. (Prof. émérite)	REGROUPEMENT DES ENSEIGNANTS CHERCHEURS ISOLES
GERARD J.F.	INGENIERIE DES MATERIAUX POLYMERES

GERMAIN P.	LAEPSI****
GIMENEZ G.	CREATIS**
GOBIN P.F. (Prof. émérite)	GEMPPM***
GONNARD P.	GENIE ELECTRIQUE ET FERROELECTRICITE
GONTRAND M.	PHYSIQUE DE LA MATIERE
GOUTTE R. (Prof. émérite)	CREATIS**
GOUJON L.	GEMPPM***
GOURDON R.	LAEPSI****.
GRANGE G. (Prof. émérite)	GENIE ELECTRIQUE ET FERROELECTRICITE
GUENIN G.	GEMPPM***
GUICHARDANT M.	BIOCHIMIE ET PHARMACOLOGIE
GUILLOT G.	PHYSIQUE DE LA MATIERE
GUINET A.	PRODUCTIQUE ET INFORMATIQUE DES SYSTEMES MANUFACTURIERS
GUYADER J.L.	VIBRATIONS-ACOUSTIQUE
GUYOMAR D.	GENIE ELECTRIQUE ET FERROELECTRICITE
HEIBIG A.	MATHEMATIQUE APPLIQUEES DE LYON
JACQUET-RICHARDET G.	MECANIQUE DES STRUCTURES
JAYET Y.	GEMPPM***
JOLION J.M.	RECONNAISSANCE DE FORMES ET VISION

Novembre 2003

JULLIEN J.F.	UNITE DE RECHERCHE EN GENIE CIVIL - Structures
JUTARD A. (Prof. émérite)	AUTOMATIQUE INDUSTRIELLE
KASTNER R.	UNITE DE RECHERCHE EN GENIE CIVIL - Géotechnique
KOULOUMDJIAN J. (Prof. émérite)	INGENIERIE DES SYSTEMES D'INFORMATION
LAGARDE M.	BIOCHIMIE ET PHARMACOLOGIE
LALANNE M. (Prof. émérite)	MECANIQUE DES STRUCTURES
LALLEMAND A.	CENTRE DE THERMIQUE DE LYON - Energétique et thermique
LALLEMAND M. (Mme)	CENTRE DE THERMIQUE DE LYON - Energétique et thermique
LAREAL P (Prof. émérite)	UNITE DE RECHERCHE EN GENIE CIVIL - Géotechnique
LAUGIER A. (Prof. émérite)	PHYSIQUE DE LA MATIERE
LAUGIER C.	BIOCHIMIE ET PHARMACOLOGIE
LAURINI R.	INFORMATIQUE EN IMAGE ET SYSTEMES D'INFORMATION
LEJEUNE P.	UNITE MICROBIOLOGIE ET GENETIQUE
LUBRECHT A.	MECANIQUE DES CONTACTS
MASSARD N.	INTERACTION COLLABORATIVE TELEFORMATION TELEACTIVITE
MAZILLE H. (Prof. émérite)	PHYSICOCHIMIE INDUSTRIELLE
MERLE P.	GEMPPM***
MERLIN J.	GEMPPM***
MIGNOTTE A. (Mle)	INGENIERIE, INFORMATIQUE INDUSTRIELLE
MILLET J.P.	PHYSICOCHIMIE INDUSTRIELLE
MIRAMOND M.	UNITE DE RECHERCHE EN GENIE CIVIL - Hydrologie urbaine
MOREL R. (Prof. émérite)	MECANIQUE DES FLUIDES ET D'ACOUSTIQUES
MOSZKOWICZ P.	LAEPSI****
NARDON P. (Prof. émérite)	BIOLOGIE FONCTIONNELLE, INSECTES ET INTERACTIONS
NAVARRO Alain (Prof. émérite)	LAEPSI****
NELIAS D.	LAMCOS
NIEL E.	AUTOMATIQUE INDUSTRIELLE
NORMAND B.	GEMPPM
NORTIER P.	DREP
ODET C.	CREATIS**
OTTERBEIN M. (Prof. émérite)	LAEPSI****
PARIZET E.	VIBRATIONS-ACOUSTIQUE
PASCAULT J.P.	INGENIERIE DES MATERIAUX POLYMERES
PAVIC G.	VIBRATIONS-ACOUSTIQUE
PECORARO S.	GEMPPM
PELLETIER J.M.	GEMPPM***
PERA J.	UNITE DE RECHERCHE EN GENIE CIVIL - Matériaux
PERRIAT P.	GEMPPM***
PERRIN J.	INTERACTION COLLABORATIVE TELEFORMATION TELEACTIVITE
PINARD P. (Prof. émérite)	PHYSIQUE DE LA MATIERE
PINON J.M.	INGENIERIE DES SYSTEMES D'INFORMATION
PONCET A.	PHYSIQUE DE LA MATIERE
POUSIN J.	MODELISATION MATHEMATIQUE ET CALCUL SCIENTIFIQUE
PREVOT P.	INTERACTION COLLABORATIVE TELEFORMATION TELEACTIVITE

PROST R.	CREATIS**
RAYNAUD M.	CENTRE DE THERMIQUE DE LYON - Transferts Interfaces et Matériaux
REDARCE H.	AUTOMATIQUE INDUSTRIELLE
RETIF J-M.	CEGELY*
REYNOUARD J.M.	UNITE DE RECHERCHE EN GENIE CIVIL - Structures
RICHARD C.	LGEF
RIGAL J.F.	MECANIQUE DES SOLIDES
RIEUTORD E. (Prof. émérite)	MECANIQUE DES FLUIDES
ROBERT-BAUDOUY J. (Mme) (Prof. émérite)	GENETIQUE MOLECULAIRE DES MICROORGANISMES
ROUBY D.	GEMPPM***
ROUX J.J.	CENTRE DE THERMIQUE DE LYON – Thermique de l'Habitat
RUBEL P.	INGENIERIE DES SYSTEMES D'INFORMATION
SACADURA J.F.	CENTRE DE THERMIQUE DE LYON - Transferts Interfaces et Matériaux
SAUTEREAU H.	INGENIERIE DES MATERIAUX POLYMERES
SCAVARDA S. (Prof. émérite)	AUTOMATIQUE INDUSTRIELLE
SOUIFI A.	PHYSIQUE DE LA MATIERE
SOUROUILLE J.L.	INGENIERIE INFORMATIQUE INDUSTRIELLE
THOMASSET D.	AUTOMATIQUE INDUSTRIELLE
THUDEROZ C.	ESCHIL – Equipe Sciences Humaines de l'Insa de Lyon
UBEDA S.	CENTRE D'INNOV. EN TELECOM ET INTEGRATION DE SERVICES
VELEX P.	MECANIQUE DES CONTACTS
VERMANDE P. (Prof émérite)	LAEPSI
VIGIER G.	GEMPPM***
VINCENT A.	GEMPPM***
VRAY D.	CREATIS**
VUILLERMOZ P.L. (Prof. émérite)	PHYSIQUE DE LA MATIERE

Directeurs de recherche C.N.R.S. **:**

BERTHIER Y.	MECANIQUE DES CONTACTS
CONDEMINE G.	UNITE MICROBIOLOGIE ET GENETIQUE
COTTE-PATAT N. (Mme)	UNITE MICROBIOLOGIE ET GENETIQUE
ESCUDIE D. (Mme)	CENTRE DE THERMIQUE DE LYON
FRANCIOSI P.	GEMPPM***
MANDRAND M.A. (Mme)	UNITE MICROBIOLOGIE ET GENETIQUE
POUSIN G.	BIOLOGIE ET PHARMACOLOGIE
ROCHE A.	INGENIERIE DES MATERIAUX POLYMERES
SEGUELA A.	GEMPPM***
VERGNE P.	LaMcos

Directeurs de recherche I.N.R.A. **:**

FEBVAY G.	BIOLOGIE FONCTIONNELLE, INSECTES ET INTERACTIONS
GRENIER S.	BIOLOGIE FONCTIONNELLE, INSECTES ET INTERACTIONS
RAHBE Y.	BIOLOGIE FONCTIONNELLE, INSECTES ET INTERACTIONS

Directeurs de recherche I.N.S.E.R.M. **:**

KOBAYASHI T.	PLM
PRIGENT A.F. (Mme)	BIOLOGIE ET PHARMACOLOGIE
MAGNIN I. (Mme)	CREATIS**

* ***CEGELY*** *CENTRE DE GENIE ELECTRIQUE DE LYON*
** ***CREATIS*** *CENTRE DE RECHERCHE ET D'APPLICATIONS EN TRAITEMENT DE L'IMAGE ET DU SIGNAL*
*** ***GEMPPM*** *GROUPE D'ETUDE METALLURGIE PHYSIQUE ET PHYSIQUE DES MATERIAUX*
**** ***LAEPSI*** *LABORATOIRE D'ANALYSE ENVIRONNEMENTALE DES PROCEDES ET SYSTEMES INDUSTRIELS*

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
	CHIMIE DE LYON	M. Denis SINOU Université Claude Bernard Lyon 1 Lab Synthèse Asymétrique UMR UCB/CNRS 5622 Bât 308 2ème étage 43 bd du 11 novembre 1918 69622 VILLEURBANNE Cedex Tél : 04.72.44.81.83 sinou@univ-lyon1.fr
E2MC	**ECONOMIE, ESPACE ET MODELISATION DES COMPORTEMENTS**	M. Alain BONNAFOUS Université Lyon 2 14 avenue Berthelot MRASH Laboratoire d' Economie des Transports 69363 LYON Cedex 07 Tél : 04.78.69.72.76 Alain.Bonnafous@mrash.fr
E.E.A.	**ELECTRONIQUE, ELECTROTECHNIQUE, AUTOMATIQUE**	M. Daniel BARBIER INSA DE LYON Laboratoire Physique de la Matière Bâtiment Blaise Pascal 69621 VILLEURBANNE Cedex Tél : 04.72.43.64.43 Daniel.Barbier@insa-lyon.fr
E2M2	**EVOLUTION, ECOSYSTEME, MICROBIOLOGIE, MODELISATION** http://biomserv.univ-lyon1.fr/E2M2	M. Jean Pierre FLANDROIS UMR 5558 Biométrie et Biologie Evolutive Equipe Dynamique des Populations Bactériennes Faculté de Médecine Lyon Sud Laboratoire de Bactériologie BP 1269600 OULLINS Tél : 04.78.86.31.50 Jean-Pierre.Flandrois@biomserv.univ-lyon1.fr
EDIIS	**INFORMATIQUE ET INFORMATION POUR LA SOCIETE** http://www.insa-lyon.fr/ediis	M. Lionel BRUNIE INSA DE LYON EDIIS Bâtiment Blaise Pascal 69621 VILLEURBANNE Cedex Tél : 04.72.43.60.55 lbrunie@if.insa lyon.fr
EDISS	**INTERDISCIPLINAIRE SCIENCES SANTE** http://www.ibcp.fr/ediss	M. Alain Jean COZZONE IBCP (UCBL1) 7 passage du Vercors 69367 LYON Cedex 07 Tél : 04.72.72.26.75 cozzone@ibcp.fr
	MATERIAUX DE LYON http://www.ec-lyon.fr/sites/edml	M. Jacques JOSEPH Ecole Centrale de Lyon Bât F7 Lab. Sciences et Techniques des Matériaux et des Surfaces 36 Avenue Guy de Collongue BP 163 69131 ECULLY Cedex Tél : 04.72.18.62.51 Jacques.Joseph@ec-lyon.fr
Math IF	**MATHEMATIQUES ET INFORMATIQUE FONDAMENTALE** http://www.ens-lyon.fr/MathIS	M. Franck WAGNER Université Claude Bernard Lyon1 Institut Girard Desargues UMR 5028 MATHEMATIQUES Bâtiment Doyen Jean Braconnier Bureau 101 Bis, 1er étage 69622 VILLEURBANNE Cedex Tél : 04.72.43.27.86 wagner@desargues.univ-lyon1.fr
MEGA	**MECANIQUE, ENERGETIQUE, GENIE CIVIL, ACOUSTIQUE** http://www.lmfa.ec-lyon.fr/autres/MEGA/index.html	M. François SIDOROFF Ecole Centrale de Lyon Lab. Tribologie et Dynamique des Systèmes Bât G8 36 avenue Guy de Collongue BP 163 69131 ECULLY Cedex Tél :04.72.18.62.14 Francois.Sidoroff@ec-lyon.fr

Remerciements

Je tiens à remercier Dominique Laurent (rapporteur), Pascal Poncelet (rapporteur), Bruno Crémilleux (examinateur) et Marie-Odile Cordier (examinatrice) pour le temps et le travail qu'ils ont consacré à l'évaluation de ce travail. Je remercie également Christophe Rigotti pour m'avoir guidé sur les chemins sur les chemins de la pensée formelle et pour la qualité de nos relations au travail, Jean-François Boulicaut pour avoir su créer les conditions nécessaires au développement enthousiaste de l'équipe data mining, Joanna Jung pour sa patience, sa compréhension et ses encouragements, Thomas Daurel pour m'avoir mis le pied à l'étrier, Cyrille Masson pour m'avoir aider à la fois au quotidien et lors de missions techniques très délicates, Artur Bykowski pour avoir su parrainer un petit Eurinsa, Claire Leschi pour son aide et ses patientes relectures, Marion Leleu pour son aide, ses judicieux conseils, et ses relectures, Francesco Pacchiani pour sa curiosité sans borne et son envie de science, Noël Lucas pour son agréable et instructrice coopération, Céline Robardet pour ses conseils et ses relectures, Jérémy Besson et Ruggero Pensa pour leur bonne humeur et leur influence positive, Cannella Canis pour son support quotidien et sans faille, et enfin mes parents et ma famille sans lequels les remerciements précédents n'auraient jamais eu l'occasion d'être lus.

Résumé

Ce mémoire concerne l'extraction sous contraintes de motifs dans une séquence d'événements. Les motifs extraits sont des règles d'épisodes. L'apport principal réside dans la détermination automatique de la fenêtre temporelle optimale de chaque règle d'épisodes. Nous proposons de n'extraire que les règles pour lesquelles il existe une telle fenêtre. Ces règles sont appelées FLM-règles. Nous présentons un algorithme, *WinMiner*, pour extraire les FLM-règles, sous les contraintes de support minimum, de confiance minimum, et de gap maximum. Les preuves de la correction de cet algorithme sont fournies. Nous proposons également une mesure d'intérêt dédiée qui permet de sélectionner les FLM-règles pour lesquelles il existe une forte dépendance entre corps et tête de règle. Deux applications de cet algorithme sont décrites. L'une concerne des données médicales tandis que l'autre a été réalisée sur des données sismiques.

Mots-Clés

Extraction des Connaisses dans les Données (ECD), fouille de données, séquences d'événements, règles d'épisodes, mesure d'intérêt, fenêtre optimale.

Abstract

This work addresses the problem of mining patterns under constraints in event sequences. Extracted patterns are episode rules. Our main contribution is an automatic search for optimal time window of each one of the episode rules. We propose to extract only rules having such an optimal time window. These rules are termed FLM-rules. We present an algorithm, *WinMiner*, that aims to extract FLM-rules, given a minimum support threshold, a minimum confidence threshold and a maximum gap constraint. Proofs of the correctness of this algorithm are supplied. We also propose a dedicated interest measure that aims to select FLM-rules such that their heads and bodies can be considered as dependant. Two applications are described. The first one is about mining medical datasets while the other one deals with seismic datasets.

Keywords

Knowledge Discovery in Databases (KDD), data mining, event sequences, episode rules, interest measure, optimal time window.

« En l'absence de toute autre preuve, la considération de mon pouce suffirait à me prouver l'existence de Dieu » , Isaac Newton.

Table des matières

Table des figures

Chapitre 1

Introduction

1.1 Extraction de connaissances à partir des données

Des tablettes d'argile aux bases de données, de la gestion des récoltes à la conquête spatiale, l'homme n'a cessé de créer des données, de les stocker, et de les utiliser. De nos jours, une très grande partie des activités humaines s'appuie sur ces tâches de collecte et d'utilisation de données. Le développement fulgurant des technologies liées aux capteurs, aux réseaux de télécommunications, aux calculateurs et aux bases de données fait qu'aujourd'hui, le volume de données collectées et stockées augmente constamment. Diverses formes de stockage et de manipulation des données ont donc vu le jour, telles que les bases de données hiérarchiques et réseau apparues dans les années 60, les bases de données relationnelles développées dans les années 70 et 80, et enfin, plus récemment, les bases de données orientées objet.

Généralement, les données stockées dans les bases de données peuvent être consultées à différents niveaux de granularité. Par exemple, il est possible d'opérer selon une lecture *en extension*. Ceci signifie que les requêtes posées au système ont pour résultat un/des objet(s) précis contenu(s) dans la base. À la requête « Quels sont les produits achetés par M. Durand cette année ? », le système répond « une serviette de bain, un maillot de bain et des lunettes de natation ». Il est également possible de poser la question suivante : « Quel est, par région, le prix moyen des serviettes de bain ? ». Ce type de requête nécessite quant à elle une consultation des données à différents niveaux d'agrégats. Dans ce cas, les résultats d'une requête ne sont pas les objets contenus dans la base mais les valeurs de fonctions d'évaluation appliquées à des sous-ensembles d'objets.

Dans les années 90 sont apparus les systèmes OLAP (On Line Analytical Pro-

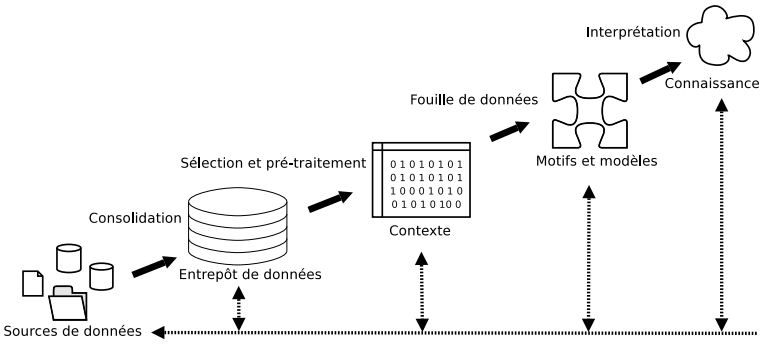


FIG. 1.1 – Le processus d'Extraction de Connaissances dans les Données.

cessing). Ceux-ci permettent, simultanément, de prendre en compte de très gros volumes de données, tels que ceux contenus dans les entrepôts de données, et de fournir à l'utilisateur le moyen de consulter ces données, en ligne, et à différents niveaux de granularité. En revanche, la consultation de ces données procède à chaque fois d'une démarche de test initiée par l'utilisateur. Autrement dit, le système peut exhiber un comportement, évaluer la pertinence d'une *loi*, d'une tendance, parfois à l'aide d'outils statistiques, mais cette *loi* doit d'abord être envisagée par l'utilisateur lui-même avant de la soumettre au système au travers d'une requête. Or, aujourd'hui, le besoin de l'utilisateur évolue vers une suggestion automatique, par le système, des lois ou des tendances contenues dans les données ; par exemple "si un client achète un maillot de bain, celui-ci achète une serviette de bain dans quatre-vingt-dix pour cent des cas". C'est ce besoin de *suggestion de lois* qui a donné naissance à une nouvelle discipline informatique, *l'Extraction de Connaissances dans les Données (ECD)*, ou *Knowledge Discovery in Databases (KDD)* ([Fay96, FPSS96a, IM96]). Bien évidemment, il reste possible, dans le cadre de l'ECD, de procéder selon une démarche de test, l'apport résidant alors dans l'utilisation de nouvelles formes de représentations des dépendances et/ou de techniques efficaces permettant de les extraire en abordant de très gros volumes de données.

L'ECD fédère des disciplines comme les statistiques, l'analyse de données, les bases de données, l'apprentissage automatique ou bien encore la visualisation ([FPSS96b]). L'ECD s'organise autour d'un processus itératif et interactif défini dans [FPSM91] comme *le processus non trivial d'extraction d'informations implicites, nouvelles, et potentiellement utiles à partir des données.* Ce processus, représenté figure 1.1, est itératif car les résultats d'une étape peuvent remettre en cause les traitements effectués durant les étapes précédentes ; et il est interactif car la qualité des résultats obtenus dépend en grande partie de l'intervention des utilisateurs finaux. Quatre étapes composent ce processus : *la consolidation*, *la*

sélection et le pré-traitement, *la fouille de données* et *l'interprétation*. De façon plus détaillée :

- *Consolidation* : cette première étape permet de regrouper et de mettre en forme des données d'origines diverses au sein d'une seule et même base de données. Les données peuvent provenir de différents systèmes de gestion de bases de données, de fichiers textes, ou bien même de notes manuscrites. Elles peuvent également être de natures très diverses, autant du point de vue qualitatif que quantitatif (e.g. cours d'indices boursiers, données de recensement, logs web, séquences d'achats, etc.). Les données rassemblées, par exemple au sein d'un *entrepôt de données*, sont également nettoyées (prise en compte des valeurs abbérantes et/ou manquantes) et codées selon un système uniforme.

- *Sélection et pré-traitement* : cette deuxième étape, permet, lors de la sélection, de choisir les données pertinentes à partir desquelles seront extraites les connaissances. Une fois les données sélectionnées, celles-ci sont pré-traitées. Il s'agit cette fois de représenter les données dans un format directement utilisable par les étapes suivantes du processus. On peut ainsi appliquer une opération de discrétisation sur les données. Par exemple, les cours d'indices boursiers peuvent être discrétisés en intervalles de valeurs. Dans le cas de variables discrètes, un travail de redéfinition des valeurs d'intervalles peut également être effectué (e.g. par union des intervalles de référence).

- *Fouille de données* : cette étape, également appelée *data mining*, est considérée comme l'articulation majeure du processus d'ECD. En effet, elle permet de passer des données à l'expression des relations et des lois qui les sous-tendent. Cette étape est considérée comme la plus difficile au point de vue algorithmique. En effet, les ressources en temps et mémoire utilisées lors de cette étape doivent rester raisonnables alors que les volumes de données traitées et les espaces de recherche sont très grands. L'utilisation active de contraintes (critères de sélection) lors de cette étape permet de diminuer les espaces de recherche, et par conséquence les ressources en temps et en mémoire, et de cibler au mieux les traitements effectués (e.g. contrainte de durée maximale entre deux achats d'une même séquence d'achats).

- *Interprétation* : cette dernière étape est celle lors de laquelle l'utilisateur final intervient le plus. Il lui revient en effet la tâche d'analyser les résultats obtenus à l'issue de la fouille de données et de les interpréter en fonction du domaine de connaissance mis en jeu. Il faut cependant garder à l'esprit que cela n'enlève rien à l'importance que revêt l'intervention de l'utilisateur final lors des précédentes étapes. En effet, si les résultats sont jugés insuffisants lors de l'interprétation, l'utilisateur doit pouvoir remettre en cause les étapes

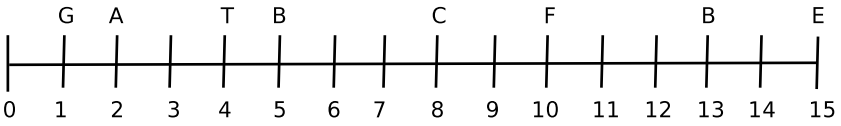


FIG. 1.2 – Exemple d'une séquence d'événements

qui selon lui ont mal été abordées, et à l'issue de chacune de ces étapes, il doit pouvoir porter le même regard critique.

1.2 Contribution

L'ensemble des travaux présentés dans ce mémoire a été réalisé dans le cadre du projet européen AEGIS (Ability Enlargement for Geophysicists and Information technology Specialists, IST-2000-26450). Ce projet a pour but de créer des synergies entre la communauté des géophysiciens et la communauté des informaticiens, et ce au travers de formations mutuelles et de recherches communes. Dans ce contexte, nous avons été amenés à collaborer avec des géophysiciens du laboratoire de géologie de l'ENS Paris dont l'un des besoins était l'exploration de données sismiques prenant la forme de longues séquences d'événements. Aussi, notre contribution concerne-t-elle la recherche de *règles d'épisodes* dans une longue séquence d'événements. Par exemple, une séquence contenant des milliers d'événements répartis sur des milliers d'unités temporelles est considérée comme longue. La figure 1.2 donne l'exemple d'une séquence d'événements. Par souci de clarté, la longueur de celle-ci a volontairement été limitée. Cette séquence contient de nombreux motifs, parmi lesquels la règle d'épisode $A \rightarrow B \Rightarrow C$. Cette règle signifie que « *lorsque A est suivi de B, alors C suit A et B* ». Il est possible d'évaluer le *support* de cette règle, i.e. le nombre de fois où il est possible d'observer A, suivi de B, lui-même suivi de C dans la séquence. Dans l'exemple de la figure 1.2, le *support* est de 1. En effet, on observe A suivi de B suivi de C une seule fois, A apparaissant à la date 2, B apparaissant à la date 5 et C apparaissant à la date 8. Il est également possible de définir la *confiance* de la règle, c'est-à-dire la probabilité d'observer C apparaître après A et B lorsque ceux-ci sont observés, A apparaissant avant B. Toujours dans l'exemple de la figure 1.2, la confiance est de $1/2$. Comme nous venons de le voir, A suivi de B suivi de C apparaît une seule fois. En revanche, A suivi de B apparaît deux fois. En effet, A apparaît à la date 2 et il est suivi de B à la date 5, mais ce même A, apparaissant à la date 2, est également suivi de B, qui cette fois-ci apparaît à la date 13.

Le support et la confiance permettent de cibler la recherche de règles d'épisodes

dans une longue séquence d'événements. Le support est également nécessaire en pratique afin de limiter la consommation de ressources en temps et en mémoire. Ainsi, l'utilisateur peut rechercher les règles d'épisodes ayant un support supérieur ou égal à un nombre σ et dont la confiance est supérieure ou égale à un nombre γ. Cette configuration est d'ailleurs considérée comme la configuration standard de recherche d'épisodes ; configuration dans le cadre de laquelle se place notre contribution.

En ce qui concerne la recherche de règles d'épisodes, deux approches ont été proposées ([MTV95, MT96, MTV97]). Le principe général de ces méthodes est tout d'abord de rechercher les *épisodes*, par exemple $A \rightarrow B$ (A est suivi de B), d'évaluer leurs propriétés tel que le support (i.e. le nombre de fois où A apparaît suivi de B dans la séquence), puis à partir de ces épisodes, de construire les règles d'épisodes, par exemple $A \rightarrow B \Rightarrow C$ (lorsque A est suivi de B, alors C suit A et B). Enfin, ces règles sont sélectionnées à partir de l'évaluation de leurs propriétés telle que le support et la confiance (on notera que le support de la règle $A \rightarrow B \Rightarrow C$ est égal au support de l'épisode $A \rightarrow B \rightarrow C$). Cependant, chacune des approches se différencie sur un point essentiel : la définition de l'*occurrence* d'un épisode. Cette définition répond à la question « Quand considère-t-on qu'un épisode apparaît dans la séquence d'événements ? ». Plus précisément, la première approche proposée, exprimée par la méthode WINEPI ([MTV95, MTV97]), est basée sur des occurrences d'épisodes définies grâce à une fenêtre glissante le long de la séquence d'événements. Chaque instance de la fenêtre glissante contenant un épisode est alors une occurrence. La deuxième approche, la méthode MINEPI ([MT96, MTV97]), fait quant à elle appel à la notion d'*occurrence minimale* des épisodes. L'occurrence est dite minimale lorsqu'elle ne contient pas elle-même une occurrence du même épisode. Par exemple, si l'on reprend la séquence représentée par la figure 1.2, et que l'on s'intéresse aux occurrences de l'épisode $A \rightarrow B$, alors l'occurrence telle que A apparaît à la date 2 et que B apparaît à la date 13 n'est pas une occurrence minimale. En effet, elle contient l'occurrence telle que A apparaît à la date 2 et que B apparaît à la date 5. En revanche, cette dernière occurrence est quant à elle une occurrence minimale.

Les deux approches présentées ont cependant un point commun : elles utilisent une contrainte de *fenêtre maximale*. Cette fenêtre maximale spécifie la durée de temps maximale entre le premier et le dernier événement des occurrences des épisodes à partir desquels sont construites les règles d'épisodes, ce qui de fait contraint les occurrences des règles d'épisodes elles-même. Cette contrainte est utile du point de vue de la consommation des ressources en temps et mémoire. En effet, si aucune contrainte de ce type n'est prise en compte, il devient possible de considérer un nombre énorme d'occurrences des épisodes, le cas limite étant l'occurrence d'un épisode dont le premier événement se situe au début de la séquence et dont le dernier événement se trouve à la fin de la séquence. Dans ce cas, le

nombre d'occurrences des épisodes devient tel que de très nombreux épisodes sont considérés comme fréquents (i.e. leur support est supérieur à un nombre σ spécifié par l'utilisateur) et donc comme utilisables pour construire des règles d'épisodes. Le nombre de ces épisodes peut alors être tel que l'exécution de l'algorithme ne peut pas être menée à son terme dans des conditions standards de consommation de ressources en temps et en mémoire. Autrement dit, dans cette situation, l'algorithme doit parcourir un espace d'épisodes très vaste, ce parcours étant d'autant plus coûteux en temps et en mémoire que l'espace à parcourir est grand, celui-ci étant lui-même d'autant plus étendu que le nombre d'épisodes potentiellement fréquents est élevé.

Cependant, spécifier une fenêtre maximale présente parfois un inconvénient. Ainsi, il existe plusieurs domaines d'application, e.g. la sismologie et la médecine, où la fenêtre peut être elle-même une information à extraire, au même titre que les épisodes et les règles d'épisodes. Cette fenêtre peut par exemple être la fenêtre maximale comme définie précédemment. Cela peut aussi être la *fenêtre optimale*, c'est-à-dire la fenêtre de temps sur laquelle une règle d'épisodes est considérée comme la plus *forte*, la plus *intéressante*. Plus précisément, cela peut être la fenêtre temporelle pour laquelle la propriété d'une règle est maximisée. Il est à noter que cette fenêtre peut évidemment varier d'une règle d'épisodes à une autre. À notre connaissance, il n'existe aucun algorithme complet capable d'extraire des règles d'épisodes sans utiliser une contrainte de fenêtre maximale (en plus des contraintes de support et de confiance), et ce sur des jeux de données non-triviaux. Une proposition a été avancée dans [CG03]. En l'occurrence, il s'agit de remplacer la contrainte de fenêtre maximale par une contrainte de *gap maximum* ; contrainte qui spécifie la durée maximale entre deux événements constituant l'occurrence d'un motif. Cette proposition est intéressante car elle permet aux motifs de s'étendre sur des intervalles de temps qui peuvent être plus larges que ceux spécifiés par une fenêtre maximale. En revanche, l'algorithme proposé est incomplet, ce que nous montrerons au chapitre 3. Néanmoins, ce travail suggère que la recherche de règles d'épisodes peut être effectuée en utilisant simplement une contrainte de gap maximum sans pour autant devoir utiliser la contrainte de fenêtre maximale. Cette contrainte de gap maximum est similaire à la contrainte de gap maximum utilisée dans le domaine de la recherche de motifs séquentiels dans des bases de séquences. Une base de séquences est une grosse collection de courtes séquences dans lesquelles on peut rechercher des motifs séquentiels comme, par exemple, des épisodes. Malheureusement, les algorithmes (e.g., [SA96, MCP98, Zak00, PHW02, LRBE03a]) créés pour explorer ce type de données ne peuvent pas s'appliquer au cas d'une longue séquence d'événements car la notion de support est différente dans ce contexte. En effet, dans une base de séquence, le support d'un motif correspond au nombre de séquences dans lesquelles apparaît au moins une fois le motif, alors que dans le cas d'une longue séquence d'événements, le support correspond au nombre d'occurrences du motif dans la séquence.

Notre contribution s'articule autour des points suivants. Tout d'abord, nous proposons un algorithme, *WinMiner*, qui permet d'extraire les règles d'épisodes satisfaisant aux contraintes de support minimum, de confiance minimum, de gap maximum et présentant un maximum local de confiance en fonction du temps, i.e. pour lesquelles il existe une fenêtre temporelle optimale. Les preuves de justesse et de complétude de cet algorithme sont présentées et diverses expériences relatives à la consommation en temps et mémoire sont également fournies. Les règles d'épisodes satisfaisant à l'ensemble des contraintes précédemment énumérées sont dénommées *FLM-règles*. Les FLM-règles tentent de répondre aux deux besoins exprimés par les géophysiciens : (1) fournir des motifs dont les mesures associées présentent un comportement singulier dans le temps, (2) aboutir à des résultats dont le volume ne dépasse pas l'entendement de l'utilisateur final. Nous proposons aussi une mesure d'intérêt, le *lift*[1], permettant de comparer la confiance observée de la règle à la confiance estimée en cas d'indépendance des parties droite et gauche de la règle. Cette mesure d'intérêt permet une sélection plus fine des règles intéressantes car elle peut, comme cela sera présenté au chapitre 3, être combinée aux critères de sélection que sont le support minimum, la confiance minimum et le gap maximum.

L'algorithme proposé a été testé sur différents jeux de données. Sur des jeux de données aléatoires synthétiques, aucun motif n'a été trouvé, ce qui est encourageant puisque un jeu aléatoire ne contient pas, par définition, de comportement particulier à exhiber. Pour ce qui est des jeux de données réels, des expériences ont été menées sur des données médicales à l'aide de *WinMiner* et ce dans le cadre du Discovery Challenge PKDD 2004. Ces expériences ont permis de (1) retrouver des connaissances considérées comme connues et acceptées en médecine, et (2), de fournir aux médecins une information qu'ils ne connaissaient pas déjà, i.e. les fenêtres de temps optimales pour des règles d'épisodes représentant une connaissance déjà validée dans le domaine. *WinMiner* a également été appliqué à des jeux de données sismiques. Ces expériences furent l'occasion d'utiliser la mesure d'intérêt nouvellement proposée, i.e. le lift, et, in fine, de suggérer aux géophysiciens des dépendances inconnues jugées potentiellement intéressantes.

1.3 Organisation du mémoire

Le chapitre 2 développe l'état de l'art en présentant les différentes techniques d'extraction de motifs sous contraintes dans les séquences d'événements, que ce soit dans le contexte d'une longue séquence d'événements ou d'une base de séquences. Le chapitre 3 présente l'algorithme *WinMiner* ainsi que son potentiel d'appli-

[1] Par référence à la mesure de lift utilisée dans le cadre de l'extraction de règles d'association (e.g., [SBM98, HTF01]).

cation en pratique. Plus précisément, il fournit les preuves de sa correction, et détaille l'implémentation de *WinMiner* ainsi que les résultats de tests relatifs à la consommation en temps et en mémoire. Enfin, ce même chapitre présente la nouvelle mesure d'intérêt proposée, le lift, ainsi que les divers paramètres additionnels disponibles permettant de cibler au mieux la recherche de règles d'épisodes. Le chapitre 4 décrit deux applications de *WinMiner* réalisées sur des données sismiques et médicales. Enfin, nous concluons avec le chapitre 5 et présentons les perspectives ouvertes par ce travail.

Chapitre 2

Définitions et état de l'art

Les données séquentielles sont des données ordonnées. La relation d'ordre établie sur ces données peut être temporelle ou spatiale. Les historiques d'erreur ou de connexion sur un réseau (e.g., [HKM+96, MTV97, ZLO98, Kle99]), les successions d'achats de clients ou de pages visitées d'un site Web (e.g., [CMS97, GRSS99, MPC99, KB00, Coo00, MA01]), les évolutions de cours boursiers, ou bien encore les mesures géophysiques sont des exemples de données temporelles (e.g., [BC96, BSWJL98, Pov99, HDY99, LB02a, LB02b]). Tout aussi variées sont les données spatiales. On pourra par exemple citer les chaînes de caractères, qui sont des successions de caractères alpha numériques, les séquences ADN, qui sont des successions de nucléotides, ou bien les séquences de protéines qui sont de successions d'acides aminés (e.g., [AHKV97, LAS97, PS00, BT01, PCGS02, GNPS03]). L'analyse de telles données passe le plus souvent par la recherche de motifs dont la définition peut être très différente, d'un domaine d'application à un autre, voire au sein d'un même domaine d'application. Une proposition de définition générale de ces motifs a été avancée dans [JKK99] et fait apparaître que les motifs recherchés sont généralement des successions d'éléments ou groupe d'éléments établies selon un ordre de nature très variable, et définies en fonction de contraintes tout aussi diverses (contraintes sur la nature des éléments, contraintes de durée entre les éléments, contraintes sur le nombre d'éléments, etc...). Par exemple, dans le cas de données décrivant la succession de pages visitées sur un site Web, on peut rechercher les successions de pages qui sont le plus fréquemment visitées dans un intervalle de temps donné. Sur des données spatiales telles que les séquences de protéines, les successions d'acides aminés qui apparaissent le plus souvent et ce, avec une certaine proximité spatiale, peuvent également faire l'objet d'une recherche exhaustive. Comme on peut le constater, la nature des données et le type des motifs recherchés dans ces données sont multiples, d'où la nécessité de structurer le domaine autant que cela est possible.

L'objectif de ce chapitre est donc de proposer au lecteur une vue structurée du domaine de recherche concernant l'extraction sous contraintes de motifs dans

les séquences d'événements. Comme cela a été présenté dans le chapitre 1, il existe deux contextes différents : d'une part, l'extraction de motifs dans une seule et longue séquence d'événements et d'autre part, l'extraction de motifs dans une base de séquences, c'est-à-dire une collection de séquences courtes. Le contexte des bases séquences a été générateur de très nombreuses contributions au cours de ces dix dernières années. Ces contributions permettent aujourd'hui, dans le domaine de l'extraction sous contraintes de motifs dans les séquences d'événements, de marquer la différence entre, d'une part, les diverses techniques de base d'extraction, et d'autre part, la prise en compte, au sein de ces techniques, des différentes contraintes possibles (autres que la fréquence d'apparition des motifs). Cette séparation est en revanche moins évidente dans le cadre de l'extraction de motifs dans une seule séquence d'événements, car les techniques d'extraction proposées ne peuvent se concevoir sans la prise en compte simultanée de certaines contraintes dont, par exemple, une contrainte spécifiant la distance temporelle/spatiale *autorisée* entre les éléments, i.e. une contrainte sur la taille des fenêtre temporelles utilisées. Contrairement aux bases de séquences dans lesquelles les séquences sont courtes, la recherche dans une longue séquence n'est envisageable qu'à la condition qu'il soit impossible d'associer n'importe quel élément apparaissant dans la séquence avec n'importe quel autre élément apparaissant dans la même séquence. En effet, si ce type de contrainte n'est pas posée, et s'il n'est pas utilisé au sein de l'algorithme pour limiter le nombre d'occurrences des motifs à envisager, alors l'espace des motifs possibles à analyser devient d'une taille telle que les ressources nécessaires (en temps et en mémoire) pour l'explorer dépassent de loin les capacités de calcul dont nous disposons.

Ce chapitre est organisé comme suit. Tout d'abord, nous présenterons le domaine des bases de séquences en introduisant les motifs calculés et les familles d'algorithmes disponibles. Puis, nous détaillerons le domaine de l'extraction de motifs dans une seule séquence. Nous présenterons ainsi les contributions majeures, et tenteront de faire ressortir d'une part les motifs calculés et, d'autre part, les contraintes qui sont utilisées et la façon dont elles participent à la définition même des algorithmes proposés.

2.1 Extraction de motifs dans une base de séquences

La recherche de motifs dans une base de séquence a pour origine une étude des habitudes de consommation des clients d'un magasin. Dans le cadre de cette étude, on dispose d'une base de séquences d'achats, chaque séquence comportant les achats d'un seul et même client. L'objectif recherché est donc l'identification des successions d'achats apparaissant le plus souvent dans cette base. Pour répondre à ce besoin applicatif, les algorithmes APrioriSome et AprioriAll ont été proposés en

1995 dans [AS95]. Avant d'en venir précisément aux divers algorithmes proposés, nous allons commencer par poser les définitions permettant de fixer le cadre général de l'extraction de ces motifs.

2.1.1 Définitions

Tout d'abord, nous allons définir l'objet sur lequel les extractions sont effectuées, les *bases de séquences*. Les *bases de séquences* sont des collections de *séquences d'événements*, qui elles-mêmes sont des successions d'*événements*.

De façon plus formelle :

Définition 1 ***(Événements)*** Soit $I = \{i_1, i_2, \ldots, i_m\}$, un ensemble de m symboles distincts appelés *items* et muni d'un ordre total. Un *événement* (ou *itemset*) de taille l est un ensemble non vide constitué par l items provenant de I. L'ensemble des événements est muni d'un ordre lexicographique défini à partir de l'ordre total des items.

Dans nos exemples nous utiliserons en général les lettres majuscules pour noter les items et l'ordre total sur ces items sera l'ordre alphabétique. Un événement, par exemple $\{F, G\}$, sera noté FG lorsque cela n'est pas ambigu afin d'alléger la notation.

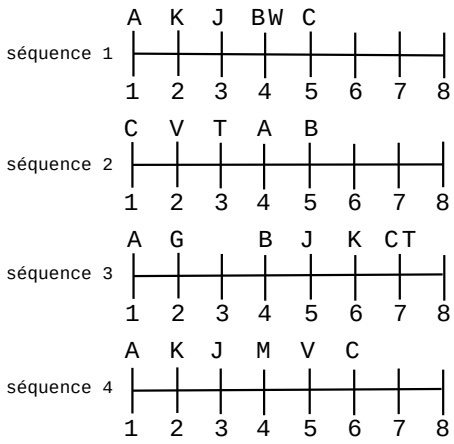
Définition 2 ***(Séquence d'événements)*** Une *séquence d'événements* (ou *séquence*) α de *longueur* L est une liste ordonnée composée de L événements $\alpha_1, \ldots, \alpha_L$, et représentée de la façon suivante : $\alpha_1 \rightarrow \alpha_2 \rightarrow ... \rightarrow \alpha_L$.

Par exemple, dans un contexte de séquences d'achats, la séquence C $\rightarrow$ A D $\rightarrow$ A signifie que le client a acheté l'item C lors d'un premier passage au magasin, puis les items A et D lors de son deuxième passage et enfin, l'item A lors d'un dernier passage.

Une base de séquence peut alors se définir de la façon suivante :

Définition 3 ***(Base de Séquences)*** Une base de séquences est un ensemble de paires (*sid*,*s*) où s représente une séquence et *sid* correspond à son identifiant. De plus, pour toute séquence s d'une base de séquences, chaque événement de s possède un identifiant, noté *eid*, qui correspond à sa date d'apparition.

On notera que la date *eid* peut représenter aussi bien une position temporelle que spatiale. Si l'on se replace dans le contexte originel des séquences d'achat, la position est alors une date. Une base de séquences peut être représentée sous la forme d'un ensemble de tuples $< sid, eid, items >$ avec *items* la liste des items composant l'événement situé à la position *eid* dans la séquence *sid*. Ainsi, si l'on



sid	eid	items
1	1	A
1	2	K
1	3	J
1	4	BW
1	5	C
2	1	C
2	2	V
2	3	T
2	4	A
2	5	B
3	1	A
3	2	G
3	4	B
3	5	J
3	6	K
3	7	CT
4	1	A
4	2	K
4	3	J
4	4	M
4	5	V
4	6	C

FIG. 2.1 – Exemple d'une base de séquences.

considère la base de séquences représentée en partie gauche de la figure 2.1, alors on peut construire sa représentation sous la forme du tableau donné en partie droite de cette même figure.

Définissons maintenant les objets recherchés dans un base de séquences, i.e. les *motifs*. Intuitivement, un motif est une liste d'items. Le nombre d'apparitions ou *occurrences* d'un motif est alors défini comme le nombre de séquences dans lesquelles le motif apparaît au moins une fois. Ce nombre est appelé *support*. L'ensemble des techniques d'extraction de motifs s'appuie sur cette notion de support afin de sélectionner et d'extraire les motifs dits *fréquents*, c'est-à-dire les motifs dont le support est supérieur ou égal à une valeur notée σ. Cette notion de support pose de fait une contrainte sur les motifs. L'utilisation active de cette contrainte est nécessaire aux différentes techniques d'extraction car elle permet de réduire l'espace des motifs envisagés durant le processus de calcul, processus qui sinon ne pourrait être mené à son terme dans des conditions standards du point de vue des ressources en temps et en mémoire. Autrement dit, les motifs de base recherchés sont les motifs fréquents. Formellement :

Définition 4 ***(Motif)*** Un *motif* est une séquence, et il est représenté de la façon suivante : $\alpha_1 \rightarrow \alpha_2 \rightarrow \ldots \rightarrow \alpha_n$.

Définition 5 ***(Sous-séquence)*** Une séquence (ou motif), de la forme $\alpha_1 \rightarrow \alpha_2 \rightarrow \ldots \rightarrow \alpha_n$ est appelée une *sous-séquence* (ou sous-motif) d'une séquence

$\beta_1 \rightarrow \beta_2 \rightarrow \ldots \rightarrow \beta_m$ s'il existe des entiers $1 \leq i_1 < i_2 < \ldots < i_n \leq m$ tels que $\alpha_1 \subseteq \beta_{i_1}$, $\alpha_2 \subseteq \beta_{i_2}$, ..., $\alpha_n \subseteq \beta_{i_n}$.

Définition 6 ***(Sur-séquence)*** Toute séquence (ou motif) β ayant pour sous-séquence une séquence α est une *sur-séquence* (ou sur-motif) de α.

Par exemple, si l'on considère la séquence $C \rightarrow AD \rightarrow A$, alors le motif $C \rightarrow A \rightarrow A$ en est une sous-séquence car $C \subseteq C$, $A \subseteq AD$ et $A \subseteq A$. De même, le motif $A \rightarrow A$ est une sous-séquence de $C \rightarrow AD \rightarrow A$ car $A \subseteq AD$ et $A \subseteq A$.

Définition 7 ***(Occurrence d'un motif)*** Si un motif z est sous-séquence d'une séquence s d'une base, on dit que z apparaît dans s. L'apparition d'un motif dans une séquence particulière d'une base est appelée *occurrence d'un motif* et correspond à une liste d'événements accompagnés de leurs identifiants dans cette séquence de la base. Une occurrence d'un motif $\alpha_1 \rightarrow \alpha_2 \rightarrow \ldots \rightarrow \alpha_n$ se représente de la façon suivante : $\alpha_1(eid(\alpha_1)) \rightarrow \alpha_2(eid(\alpha_2)) \rightarrow \ldots \rightarrow \alpha_n(eid(\alpha_n))$.

Par exemple, l'occurrence $A(1) \rightarrow B(4) \rightarrow C(5)$ représente une occurrence du motif $A \rightarrow B \rightarrow C$ dans la première séquence de la base de la figure 2.1.

Définition 8 ***(Motif Fréquent)*** Soient D une base de séquences et σ un entier positif appelé *seuil de support absolu*. Soit z un motif et $support(z)$ le nombre de séquences de D dans lequel il apparaît. Le motif z vérifie la contrainte de fréquence minimum dans une base de données D si z est une sous-séquence d'au moins σ séquences de D, i.e. si $support(z) \geq \sigma$. Dans ce mémoire, nous utiliserons aussi la notion de *seuil de support relatif*, correspondant au seuil absolu divisé par le nombre total de séquences de D.

Si l'on se reporte à la base exemple de la figure 2.1 le motif $A \rightarrow B$ est une sous-séquence des séquences 1, 2 et 3. Ainsi, si $\sigma = 3/4$ (seuil de support relatif), le motif $A \rightarrow B$ est considéré comme étant un motif fréquent.

Afin de pouvoir manipuler de façon aisée les motifs et leurs propriétés lors de la présentation des différentes techniques d'extraction, nous introduisons les définitions suivantes :

Définition 9 ***(k-motif, taille, préfixe et suffixe d'un motif)*** Un motif $\alpha_1 \rightarrow \alpha_2 \rightarrow \ldots \rightarrow \alpha_n$ est un *k-motif* si $\sum_{i=1\ldots n} |\alpha_i| = k$ (i.e., il est composé de k items) et k est aussi appelé *taille* du motif. Le *suffixe* d'un motif est le plus grand item (pour l'ordre total des items) contenu dans le dernier événement du motif. Le *préfixe* d'un motif est le motif privé de son suffixe.

Par exemple, le motif $A \rightarrow BC$ est un $3 - motif$, i.e. de taille 3, ayant pour préfixe $A \rightarrow B$ et pour suffixe C.

Définition 10 ***(Longueur et largeur d'un motif)***
La *longueur* d'un motif $\alpha_1 \rightarrow \alpha_2 \rightarrow \ldots \rightarrow \alpha_n$ est n (i.e., le nombre d'événements qu'il contient) et sa *largeur* est $max_{i=1\ldots n}|\alpha_i|$.

Si l'on reprend le motif $A \rightarrow BC$, sa longueur est de 2 et sa largeur est de 2.

2.1.2 Les différentes familles d'algorithmes

L'objectif de cette section est de présenter les différentes familles d'algorithmes permettant l'extraction de motifs fréquents dans les bases de séquences. Depuis le milieu des années 90, trois approches ont vu le jour : les approches de type *Apriori* (e.g., [AS95, SA96, MCP98, GRK99, Mas02]), les approches par *listes d'occurrences* (e.g., [Zak98, Zak00, Zak01, AFGY02, LRBE03b, LRBE03a]) et enfin les approches dites *par projections* (e.g., [HHMAC$^+$00, PHMAP01, GGK01, PHW02]). Avant d'exposer chacune de ces approches, il convient d'observer qu'elles ont pour point commun, comme nous l'avons déjà mentionné, l'utilisation active de la contrainte de support, utilisation qui permet de limiter le nombre de motifs envisagés par l'algorithme, et donc de limiter la consommation de ressources en temps et en mémoire. Plus précisément, le support est une contrainte *anti-monotone*. Autrement dit, si le motif $A \rightarrow B$ n'est pas fréquent, aucun des motifs ayant pour sous-motif $A \rightarrow B$ ne peut être fréquent. On peut donc éviter de considérer l'ensemble de ces motifs lors de la recherche, ce qui constitue un gain de temps non-négligeable. Cette stratégie est un exemple typique d'utilisation dite *active* d'une contrainte. D'autres contraintes sont également utilisées de façon active. Leur définition et leur utilisation sont l'objet de la section 2.1.3.

Un autre point à relever réside dans leurs structures même. En effet, et à l'exclusion des approches dites par projection, les algorithmes disponibles comportent généralement deux phases : une phase de génération des candidats et une phase de comptage des candidats. La première phase permet de constituer au préalable les motifs dont on peut espérer qu'ils apparaissent dans la base de séquence, tandis que la deuxième phase procède à l'établissement du support de ces motifs afin de savoir s'ils sont fréquents.

Approches de type Apriori

Les approches de type Apriori tirent leur nom du schéma algorithmique utilisé dans le cadre de l'extraction de motifs fréquents dans des matrices booléennes [AMS$^+$96, AS94]. Ces matrices contiennent des objets (les lignes) décrits selon des propriétés de type booléen (les colonnes). Les motifs pouvant être extraits sur ce type de données sont par exemples les *itemsets fréquents*, i.e. les sous-ensembles de propriétés, dont les valeurs sont toutes vraies simultanément pour un nombre minimum d'objet. Par exemple, les objets peuvent représenter chacun les achats d'une personne à un moment précis et les colonnes peuvent représenter

Algorithme 1 (Approche type APriori)
Entrée : *Une base de séquences et un support minimum* σ
Sortie : F, *ensemble de tous les motifs séquentiels fréquents*

1. $F_1 = \{i \in I | support(i) \geq \sigma \}$
2. $k = 1$
3. **while** $F_k \neq \emptyset$ **do**
4. C_{k+1} := *GENERATION_CANDIDATS*(F_k)
5. *COMPTER_SUPPORTS*(C_{k+1})
6. $F_{k+1} := \{s \in C_{k+1} \mid support(s) \geq \sigma \}$
7. $k := k + 1$
8. **od**
9. **output** $F = \bigcup_{1 \leq j < k} F_j$

l'éventail des produits disponibles dans le commerce considéré. Un itemset fréquent correspond alors à un ensemble de produits achetés ensemble, et ce lors d'un certain nombre d'achats (i.e un certain nombre de lignes), ce nombre étant supérieur à un seuil prédéfini par l'utilisateur.

Pour en revenir au contexte des bases de séquences, les techniques d'extraction (e.g., [AS95, SA96, MCP98, GRK99, Mas02]) adoptant une approche de type Apriori ont en commun la façon dont l'espace des motifs est exploré. L'exploration se fait en *largeur*, c'est-à-dire que les motifs de taille $k+1$ ne sont considérés qu'après avoir exploré tous les motifs de taille k. À ce principe d'exploration se rajoute l'utilisation active de propriété d'anti-monotonicité du support. Ainsi, une fois les motifs de taille k comptés, des motifs dits *candidats* de taille $k+1$ sont générés à partir des motifs de taille k qui sont fréquents ; ces motifs candidats étant alors les seuls à être comptés au niveau $k+1$. En effet, aucun motif fréquent ne peut contenir de sous-motif qui ne soit pas fréquent. Ce type d'approche nécessite donc une passe complète pour le comptage des motifs candidats à chaque niveau k. Ce schéma algorithmique, basé sur un parcours en largeur et sur l'exploitation de l'anti-monotonicité de la contrainte de support minimum, constitue également l'apport principal des algorithmes proposés dans [AS94, AMS$^+$96]. Ce principe algorithmique a par ailleurs été décrit dans [MT97] qui resitue l'ensemble dans un cadre plus général.

Les approches de type Apriori peuvent être formulées à l'aide de l'algorithme 1.

Si on analyse cet algorithme, trois phases sont à distinguer. La première (ligne 1) permet d'amorcer l'algorithme. Lors de cette phase, une passe complète est effectuée sur la base de séquences afin de comptabiliser le support de tous les items i de I. Les items dont le support est supérieur au support minimum σ seront retenus

pour former l'ensemble F_1, c'est-à-dire l'ensemble des 1-motifs fréquents. Ce n'est qu'à partir de là que l'on peut alors envisager l'itération basée sur la succession des phases de génération des candidats (ligne 4) et de comptage des supports (ligne 5) dont voici le détail :

- $GENERATION\_CANDIDATS$: cette étape permet la création des motifs candidats de taille $k+1$ (C_{k+1}) à partir de l'ensemble des motifs fréquents de taille k (F_k). Un exemple de ce type de génération est détaillé ci-après lors de la présentation de l'algorithme GSP (Generalized Sequential Patterns).
- $COMPTER\_SUPPORTS$: cette étape permet d'établir le support des motifs candidats. Pour cela, le processus va effectuer une passe sur la base et va incrémenter les supports de tous les motifs candidats contenus dans C_{k+1} (chaque fois que le processus détermine qu'une séquence contient un motif candidat, le support de celui-ci est incrémenté de 1).

Par la suite, le nouvel ensemble de motifs fréquents F_{k+1} est obtenu par sélection des motifs de C_{k+1} dont le support est supérieur ou égal à σ (ligne 6). Le processus s'arrête lorsque F_{k+1} est vide (il n'est plus possible de générer de candidats, ligne 3). L'union des F_k constitue alors F, l'ensemble des motifs fréquents. C'est cet ensemble qui est donné comme résultat, ligne 9. Le déroulement de cet algorithme est illustré ci-après par un exemple d'exécution de l'algorithme GSP sur la base de séquences représentée par la figure 2.1.

Un algorithme tout à fait représentatif de l'approche Apriori est l'algorithme GSP (Generalized Sequential Patterns). Celui-ci a été introduit dans [SA96] et permet, en plus de la contrainte de fréquence, de prendre en compte certaines contraintes de temps, aspects que nous détaillerons ci-après, dans la section 2.1.3. Les apports de GSP par rapport à l'algorithme abstrait présenté auparavant reposent d'une part sur le mode de génération des candidats et d'autre part sur la structure utilisée pour représenter les candidats et les compter dans les séquences.

La génération des candidats se fait de la façon suivante : deux motifs z_1 et z_2 de longueur k peuvent générer un motif de longueur $k+1$ si lorsque l'on retire le premier item à z_1, on obtient un motif identique à z_2 privé de son dernier item. Autrement dit, z_1 est étendu à l'aide du suffixe de z_2. Le motif z généré est alors tel que l'item ajouté à z_1 est un événement à part entière (événement composé d'un seul item) s'il constituait un événement également à part entière dans z_2 ; sinon l'item est ajouté comme dernier item au dernier événement de z_1. Ainsi $C \rightarrow AD$ et $AD \rightarrow A$ permettront de générer le motif $C \rightarrow AD \rightarrow A$ alors que $C \rightarrow AD$ et ADE permettront de générer le motif $C \rightarrow ADE$. En ce qui concerne le cas particulier de la génération des motifs de taille 2, une génération différente est employée. En effet, il est impossible de retirer des items à des motifs de taille 1 pour construire des motifs de taille 2. On procède alors ainsi : pour tout couple (z_1, z_2), avec $z_1 \neq z_2$ les motifs $z_1 \rightarrow z_2$ et $z_1 z_2$ sont générés. Si $z_1 = z_2$, alors seul

le motif $z_1 \rightarrow z_2$ est généré.

Venons-en maintenant à la structure utilisée par GSP pour stocker les candidats. Il s'agit d'un arbre de hachage dont un exemple est donné par la figure 2.2. Dans cet arbre, un noeud peut être soit une table de hachage dont chaque entrée pointe vers un noeud de niveau immédiatement supérieur (noeuds *intérieurs*), soit une liste de motifs (*feuilles*). Pour un motif candidat donné, GSP parcourt l'arbre, appliquant à chaque niveau p (le niveau de la racine est 1) une fonction de hachage sur le $p$ème item du motif, et ce de façon à atteindre une feuille dans laquelle il puisse insérer le motif. Si l'ajout dans cette feuille du motif porte le nombre de motifs contenus dans la feuille à niveau supérieur à une limite précisée au préalable, alors la feuille est scindée en deux ou plusieurs feuilles, l'ensemble de celles-ci reprenant l'intégralité des motifs contenus dans la feuille mère. Quant à celle-ci, elle est transformée en noeud intérieur pointant sur les feuilles nouvellement générées.

Lors de l'analyse d'une séquence d, la fonction de hachage est appliquée à toutes les occurrences des items de d, les occurrences des items étant considérées une à une. Puis cette procédure est appliquée récursivement à toutes les occurrences des items qui suivent les occurrences considérées lors de la précédente application de la fonction de hachage. Cette approche permet ainsi de ne pas considérer les motifs candidats qui ne commencent pas par un item de la séquence. Lorsque les applications successives de la fonction de hachage désignent une feuille, les candidats contenus dans cette feuille sont ajoutés à la liste des motifs susceptibles d'apparaître dans la séquence d. Ainsi, pour chaque séquence, on construit un sous-ensemble des candidats issus de la phase de génération ; chaque élément de ce sous-ensemble étant susceptible d'apparaître dans la séquence. Ce n'est qu'alors que GSP vérifie si ces candidats, d'une certaine façon *confirmés*, apparaissent vraiment dans la séquence en question et que les supports de candidats sont augmentés s'ils sont effectivement trouvés comme appartenant à la séquence.

Reprenons la base de séquence représentée par la figure 2.1. Supposons que seuls les motifs séquentiels ayant un support relatif supérieur ou égal à 3/4 soient à extraire. GSP va d'abord effectuer une première passe afin d'identifier et de sélectionner les items apparaissant au moins dans trois séquences sur quatre. Seuls les items A, B, C, J, et K sont retenus. On a donc $F_1 = \{A, B, C, J, K\}$. GSP génère alors les candidats de taille 2 (C_2) selon le principe décrit précédemment. C'est ainsi que tous les appariements possibles entre les items sont testés, que ce soit pour générer des motifs de longueur 2 et de largeur 1 (par exemple $A \rightarrow B$) ou bien pour générer des motifs de longueur 1 et de largeur 2 (par exemple AB). L'ensemble de ces candidats est représentée dans l'arbre de hachage de la figure 2.2 pour lequel le nombre maximal d'éléments par feuille est fixé à 11.

Puis, GSP analyse, comme cela a précédemment été évoqué, chacune des séquences de la base, et ce afin de restreindre si possible, pour chaque séquence, les motifs candidats. La première et la troisième séquence permettent, chacune, d'atteindre toutes les feuilles de l'arbre. Ainsi, pour ces deux séquences, GSP va

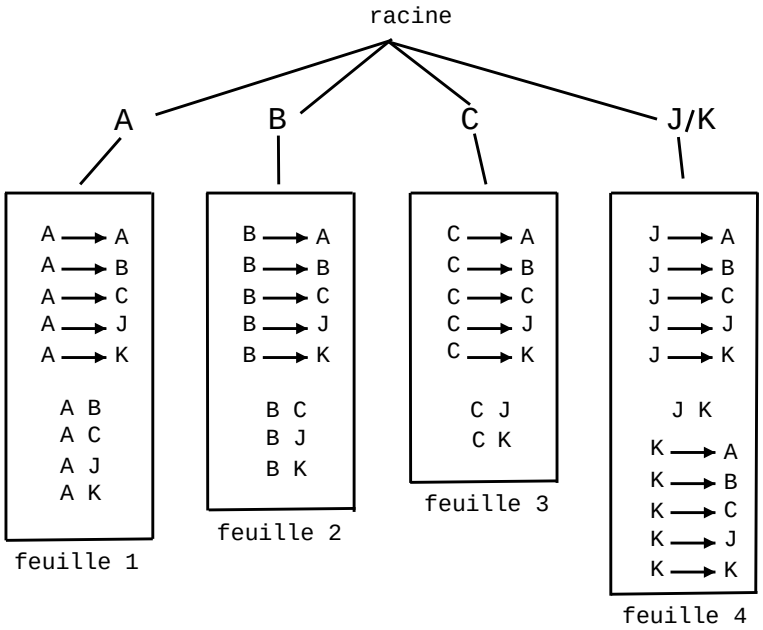


FIG. 2.2 – Exemple d'une arbre de hachage pour des motifs candidats de taille 2.

devoir rechercher tous les candidats possibles. En revanche, la deuxième séquence ne permet d'atteindre que les feuilles 1, 2 et 3. GSP pourra donc ne rechercher que les candidats des feuilles 1, 2, et 3. Il en est de même pour la quatrième séquence pour laquelle seuls les motifs candidats des feuilles 1, 3, 4 devront être recherchés. Au final, les motifs $A \rightarrow B$, $A \rightarrow C$, $A \rightarrow J$, $A \rightarrow K$, $J \rightarrow C$, et $K \rightarrow C$ sont identifiés comme fréquents, chacun ayant un support relatif de 3/4. Ces motifs constituent F_2. Seuls ces motifs pourront donner naissance à des candidats de taille supérieure. Selon la procédure précédemment exposée, les motifs $A \rightarrow J$, $A \rightarrow K$, $J \rightarrow C$, et $K \rightarrow C$ peuvent être utilisés afin de former C_3. C_3 contient les candidats suivants : $A \rightarrow J \rightarrow C$ et $A \rightarrow K \rightarrow C$. L'arbre permettant de stocker C_3 est représenté par la figure 2.3.

GSP parcourt à nouveau chacune des séquences. Les séquences 1, 2, 3 et 4 permettent, chacune, d'atteindre l'unique feuille de l'arbre. GSP va donc, pour toutes les séquences, rechercher les deux motifs candidats. À l'issue de cette recherche, $A \rightarrow J \rightarrow C$ et $A \rightarrow K \rightarrow C$ se révèlent être tous deux fréquents, avec un support de 3/4. Ils appartiennent donc à F_3. En revanche, il n'est pas possible de générer de nouveaux motifs candidats à partir de ces motifs (C_4 est donc vide), car si l'on enlève le premier item de l'un et le dernier item de l'autre, il est, dans tous les cas, impossible d'obtenir un même sous-motif ; ce qui signe l'arrêt de l'exécution de GSP.

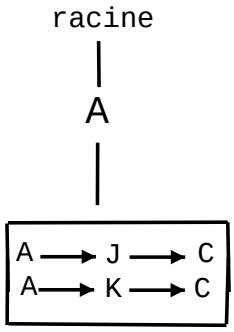


FIG. 2.3 – Exemple d'un arbre de hachage pour des motifs candidats de taille 3.

En ce qui concerne les différentes optimisations pouvant être apportées aux algorithmes de la famille Apriori, et outre l'utilisation d'un arbre de hachage proposée dans [SA96], celles-ci peuvent être envisagées par exemple au niveau même de la génération des candidats. Elles procèdent alors généralement de l'utilisation active de contraintes autres que la contrainte de support, par exemple une contrainte de longueur, contrainte qui permet de spécifier le nombre d'éléments composant un motif. Cet aspect sera approfondi dans la section 2.1.3.

Approches par listes d'occurrence

Les approches de type Apriori sont très consommatrices en accès disque. En effet, chaque phase de comptage, aussi optimisée soit-elle, déclenche une lecture de toute la base de séquences. Une solution (e.g., [Zak98, Zak00, Zak01, AFGY02, LRBE03b, LRBE03a]) consiste alors à stocker les informations de la base de séquences en mémoire vive, sous la forme de *listes d'occurrences*. Ces listes contiennent, comme leur nom l'indique, les positions des occurrences des différents motifs dans la base de séquences. Les structures utilisées pour représenter ces listes vont de simples tableaux ou listes d'entiers à des représentations booléennes sophistiquées comme dans le cas de l'algorithme SPAM [AFGY02].

Ci-après, l'algorithme abstrait 2 donne le principe général du fonctionnement des algorithmes basés sur les listes d'occurrences. La version proposée est une version en largeur, c'est-à-dire que l'espace des motifs est parcouru par niveau, un niveau k représentant l'ensemble des k-motifs candidats.

L'algorithme 2 est amorcé par la construction des listes (notées $IdList$) des 1-motifs fréquents, c'est à dire les 1-motifs dont le support est supérieur à σ. Cette opération (ligne 1) est effectuée lors d'une seule et même passe sur la base de séquences. L'algorithme s'articule alors autour de deux opérations principales

Algorithme 2 (Listes d'occurrences)
Entrée : *Une base de séquences et un support minimum* σ
Sortie : F, *ensemble de tous les motifs séquentiels fréquents*

1. Effectue une passe sur la base pour calculer :
  - $F_1 = \{i \in I | support(i) \geq \sigma \}$
  - $IdList(z)$ *pour tous les motifs* z *de* F_1
2. **let** $k := 1$
3. **while** $F_k \neq \emptyset$ **do**
4. **let** $F_{k+1} := \emptyset$
5. **for all** $z_1 \in F_k$ **do**
6. **for all** $z_2 \in F_k$ **do**
7. **if** z_1 *et* z_2 *ont le même préfixe* **then**
8. **for all** z *obtenu par* $fusion(z_1, z_2)$ **do**
9. *Construit* $IdList(z)$ *par* $jointure(IdList(z_1), IdList(z_2))$.
10. *Utilise* $IdList(z)$ *pour déterminer si* z *est fréquent.*
11. **if** z *est fréquent* **then**
12. $F_{k+1} := F_{k+1} \cup \{z\}$
13. **fi**
14. **od**
15. **fi**
16. **od**
17. **od**
18. $k := k + 1$
19. **od**
20. **output** $\bigcup_{1 \leq j < k} F_j$

(plus de détails sur ces opérations seront apportés lors de la description de l'algorithme SPADE) :

– l'opération de *fusion* : cette opération correspond à la phase de génération des candidats des algorithmes de la famille Apriori. Plus précisément, l'objet de cette opération est de former tous les motif candidats z possibles de taille $k + 1$ à partir de deux motifs fréquents z_1 et z_2, de taille k, et ayant, par exemple, le même préfixe (e.g., [Zak01]). Dans ce cas, si l'on considère les motifs dits *générateurs* $A \rightarrow B$ et $A \rightarrow C$, on peut alors former, parmi d'autres, le motif candidat $A \rightarrow B \rightarrow C$.

– l'opération de *jointure* : pour un motif candidat z, l'opération de jointure permet de construire sa liste d'occurrences $IdList(z)$ à partir des listes d'oc-

currences de z_1 et z_2, respectivement $IdList(z_1)$ et $IdList(z_2)$.

Ainsi, si l'on dispose de motifs fréquents de taille k (F_k, ligne 3), l'algorithme va tout d'abord générer les motifs candidats de taille $k+1$ en utilisant l'ensemble des combinaisons possibles (ligne 5 et 6) entre les éléments de F_k ayant le même préfixe (ligne 7) et en appliquant pour chaque combinaison (z_1, z_2) l'opération de $fusion$ (ligne 8). Puis, pour chaque motif z candidat obtenu pour z_1 et z_2, la liste d'occurrences $IdList(z)$ de z est calculée grâce à l'opération de $jointure$ (ligne 9). Si le nombre d'occurrences de z dans $IdList(s)$ est supérieur à σ (ligne 10), alors z est fréquent et il est ajouté à F_{k+1} (ligne 11 et 12). Si F_{k+1} n'est pas vide, alors la boucle « while »(ligne 3) sera exécutée de nouveau afin de calculer F_{k+2}. Si F_{k+1} est vide, alors l'algorithme s'arrête en donnant comme résultat l'ensemble des motifs fréquents (ligne 20). On notera par ailleurs que l'opération de jointure suivie de l'opération de comptage des occurrences du motif dans la liste est un traitement correspondant à la phase de comptage des supports des algorithmes de la famille Apriori.

Un parcours en profondeur est également possible. Le principe de ce type de parcours est d'explorer, pour tout α fréquent de taille k, tout à tour et de façon récursive, tous les k+1-motifs β tel que α soit un sous motif de β (prise en compte active de l'anti-monotonicité). Ce principe est appliqué de façon récursive à partir de tous les motifs de taille 1. Ce type de parcours est intéressant car il ne nécessite pas d'avoir en mémoire l'ensemble d'un niveau, ce qui peut parfois être prohibitif, surtout dans les cas des premiers niveaux. Pour une discussion plus avancée sur le choix du type de parcours, que ce soit en largeur ou en profondeur, le lecteur pourra consulter [Zak98, Zak01].

Nous allons maintenant étudier l'exemple de l'algorithme SPADE [Zak98, Zak01] en détaillant les opérations de $fusion$ et de $jointure$ qui sont utilisées.

L'opération de $fusion$ distingue deux types de motifs : les motifs *séquences* et les motifs *événements*. Les motifs séquences sont des motifs dont le suffixe est un item qui forme à lui seul le dernier événement, tandis que les motifs événements sont des motifs pour lesquels le suffixe et le préfixe possèdent des items apparaissant dans le même et dernier événement. Cette distinction entre motifs événements et motifs séquences intervient dès la prise en compte de la nature des motifs z_1 et z_2 (motifs séquences ou motifs événements). En effet, de la nature des motifs z_1 et z_2 dépend la nature du motif candidat z. Et de la nature du motif z dépend le type de jointure appliquée par la suite aux listes d'occurrences de z_1 et de z_2. De façon détaillée, l'opération de fusion gère les cas suivants :

- si z_1 et z_2 sont des *motifs événements* (cas 1) : z_1 et z_2 sont de la forme $z_1 = ps_1$ et $z_2 = ps_2$. Si $s_1 \neq s_2$, le motif obtenu par $fusion(z_1, z_2)$ correspond au motif $z = ps_1s_2$. z est alors un motif événement.

- si z_1 est un *motif événement* et z_2 un *motif séquence* (cas 2) : z_1 et z_2 sont alors de la forme $z_1 = ps_1$ et $z_2 = p \rightarrow s_2$. Le motif obtenu par $fusion(z_1, z_2)$ correspond au motif séquence $z = ps_1 \rightarrow s_2$. z est alors un motif séquentiel.

- si z_1 et z_2 sont des *motifs séquences* : z_1 et z_2 sont de la forme $z_1 = p \rightarrow s_1$ et $z_2 = p \rightarrow s_2$. Si $s_1 \neq s_2$, trois motifs sont générés par $fusion(z_1, z_2)$:
  - (cas 3) le motif événement $z = p \rightarrow s_1 s_2$. z est alors un motif événement.
  - (cas 4) le motif séquence $z = p \rightarrow s_1 \rightarrow s_2$. z est alors un motif séquence.
  - (cas 5) le motif séquence $z = p \rightarrow s_2 \rightarrow s_1$. z est alors un motif événement.
  - (cas 6) si $s_1 = s_2$, c'est-à-dire que $z_1 = z_2 = p \rightarrow s_1$ (cas 6), alors seul le motif séquentiel $z = p \rightarrow s_1 \rightarrow s_1$ est généré. z est alors un motif séquence.

Par exemple, si l'on dispose des motifs fréquents $A \rightarrow B$ et $A \rightarrow C$, alors l'opération va générer le motif $A \rightarrow BC$ (cas 3), le motif $A \rightarrow B \rightarrow C$ (cas 4) et le motif $A \rightarrow C \rightarrow B$ (cas 5).

Pour ce qui est du cas particulier de la génération des motifs de taille 2, celle-ci s'effectue en considérant les items comme des motifs séquentiels dont le préfixe correspond à la séquence vide. Les cas de génération qui s'appliquent sont alors les cas 3, 4 et 5 si $s_1 \neq s_2$ et le cas 6 si $s_1 = s_2$.

Pour ce qui est des opérations de *jointure*, il existe deux types de jointures : une jointure dite *temporelle* et une jointure dite *équivalente*. La jointure temporelle s'applique si l'on veut générer un motif séquence tandis que la jointure équivalente s'applique si l'on veut générer un motif événement. De façon informelle, la jointure temporelle crée des occurrences de z à partir des listes d'occurrences de z_1 et de z_2 en s'assurant que le suffixe de z_1 précède le suffixe de z_2. Quant à la jointure équivalente, celle-ci crée les occurrences de z à partir des listes d'occurrences de z_1 et de z_2 de sorte que leurs suffixes respectifs apparaissent en même temps dans la séquence.

Les listes d'occurrences qui sont employées et générées par les opérations de jointure permettent de décrire les occurrences des motifs à l'aide d'un identifiant de séquence *sid* et d'un identifiant *eid* de la position du suffixe sur la séquence. Cette description est nécessaire et suffisante pour générer tous les motifs séquentiels, comme l'a démontré Mohammed Zaki dans [Zak01]. La figure 2.4 donne des exemples de listes d'occurrences.

D'un point de vue plus calculatoire, les opérations de jointure entre les listes d'occurrences d'un motifs z_1 ($IdList(z_1)$) et les listes d'occurrences d'un motif z_2 ($IdList(z_2)$) peuvent alors être définies comme suit :

- **jointure temporelle** : Pour chaque occurrence $\langle sid_1, e_1 \rangle$ provenant de

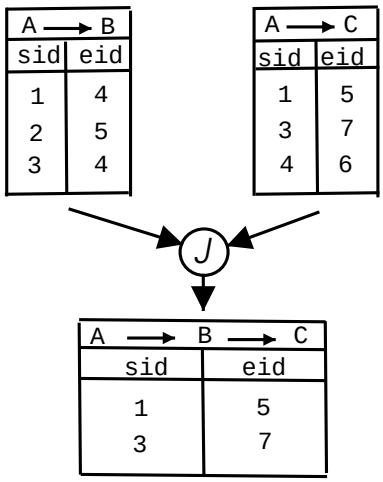


FIG. 2.4 – Exemple d'une opération de jointure temporelle.

$IdList(z_1)$ et pour chaque occurrence $\langle sid_2, e_2\rangle$ provenant de $IdList(z_2)$, cette jointure vérifie que $\langle sid_1, e_1\rangle$ représente une occurrence y_1 qui précède l'occurrence y_2 représentée par $\langle sid_2, e_2\rangle$. Celui-ci se traduit par $sid_1 = sid_2$ et $e_1 < e_2$, et signifie que les événements de y_1 et de y_2 forment une occurrence z. Le couple $\langle sid_1, e_2\rangle$ est alors ajouté à $IdList(z)$.

– **jointure équivalente :** Pour chaque couple $\langle sid_1, e_1\rangle$ représentant une occurrence de z_1 et $\langle sid_2, e_2\rangle$ représentant une occurrence de z_2, la jointure équivalente vérifie que le couple $\langle sid_1, e_1\rangle$ représente une occurrence y_1 qui se termine à la même date que l'occurrence y_2 représentée par le couple $\langle sid_2, e_2\rangle$ (i.e., $sid_1 = sid_2$ et $e_1 = e_2$). Si tel est le cas, les occurrences y_1 et y_2 forment une occurrence de z et le couple $\langle sid_1, e_1\rangle$ est alors ajouté à $IdList(z)$.

La figure 2.4 donne un exemple de jointure temporelle pour la base de séquences représentée par la figure 2.1. Cette jointure est effectuée à partir de 2 motifs appartenant à F_2 (support minimal fixé à 3/4), $A \rightarrow B$ et $A \rightarrow C$, et a pour objectif l'obtention de la liste des occurrences du motif candidat de C_3, $A \rightarrow B \rightarrow C$. La liste obtenue ne comporte que deux occurrences. Autrement dit, $A \rightarrow B \rightarrow C$ n'appartient pas à F_3 et ne pourra donc pas être utilisé au cours des opérations de fusion ultérieures.

Cet exemple d'opérations de jointure amène la remarque suivante : l'efficacité des opérations de jointure dépend grandement de la représentation choisie pour l'occurrence d'un motif dans sa liste. Plusieurs contributions à ce propos

ont d'ailleurs été proposées. Ainsi, dans [AFGY02], l'algorithme SPAM utilise des listes représentant les occurrences sous forme booléenne, ce qui permet de réduire la consommation en terme de ressources mémoires. De plus, les opérations de jointure peuvent alors ramenées à des opérations binaires très efficaces. Une autre approche à également été proposée par Marion Leleu et al. dans [LRBE03b] avec l'algorithme GoSpade. Cette fois-ci, les listes d'occurrences sont compressées de la façon suivante : si le dernier item d'un motif apparaît plusieurs fois dans une séquence de façon immédiate et consécutive sur un intervalle de temps donné, alors, plutôt que de noter chaque occurrence, c'est à dire chaque position du suffixe *eid*, on note l'intervalle de temps dont la date de départ est la date d'apparition du suffixe de la première occurrence, et dont la date de fin est la date d'apparition du suffixe de la dernière occurrence consécutive. Il a été montré dans [Lel04] qu'une telle représentation permet bien de générer tous les motifs fréquents. Ce type de compression est particulièrement bien adapté aux jeux de données contenant beaucoup de répétitions, ce qui est le cas de nombreux jeux de données, et permet de réduire la consommation en mémoire de manière significative.

Approche par projections

L'approche dite par *projections* rassemble les algorithmes proposés dans [HHMAC+00, PHMAP01, GGK01, PHW02]. Cette approche, tout comme l'approche par listes d'occurrences, a pour objet de réduire les coûts dûs au comptage du support des motifs candidats. Elle a également pour objectif de réduire la phase de génération des candidats. Deux idées sont alors mises en avant : (1) réduire la taille de la base de données et (2) éviter d'envisager des motifs n'existant pas dans la base.

La solution proposée réside dans le concept de *base projetée*. Une base projetée est un sous-ensemble d'une base initiale. Travailler sur de telles bases permet d'accélérer le comptage car la taille des bases projetées est réduite par rapport à la taille de la base initiale. La projection choisie peut par exemple être faite selon le préfixe des motifs à découvrir. C'est la solution retenue par l'algorithme PrefixSpan présenté dans [PHMAP01]. De plus, lors de la projection, ne sont gardés que les items susceptibles d'appartenir à des occurrences de motifs fréquents. La découverte des motifs fréquents s'effectue alors sur la base de ces items, en ne considérant que les items fréquents, ce qui permet d'éviter d'envisager des motifs qui ne pourraient pas de toute façon être fréquents ou apparaître dans la base de séquences. Ce type d'approche s'inspire directement des contributions faites dans le domaine de l'extraction de motifs fréquents dans des données non-séquentielles [HPY00, AAP01].

L'algorithme abstrait des approches par projection est donné par l'algorithme 3. Par souci de clarté, celui-ci ne couvre que le cas de la génération de motifs de lar-

geur 1, c'est-à-dire les motifs ne contenant qu'un seul item par événement. Cet algorithme est un algorithme récursif. Son exécution démarre un processus en trois étapes : l'amorçage, l'exploration et l'arrêt. L'amorçage est l'étape lors de laquelle tous les motifs fréquents de taille 1 sont extraits. Pour cela une passe sur toute la base de séquences initiale est requise. Puis, pour chaque 1-motif fréquent est calculée une base projetée. À partir de ces bases projetées commence alors l'étape d'exploration des sur-motifs. Pour chacune des bases projetées, une passe est effectuée afin d'établir quels sont les items fréquents contenus dans cette base. La découverte de ces items fréquents permet alors de découvrir directement les motifs fréquents de taille 2. À nouveau, pour chacun des motifs fréquents de taille 2, les bases projetées des 1-motifs fréquents sont projetées, les items fréquents sont identifiés, et les motifs fréquents de taille 3 sont extraits, et ce ainsi de suite jusqu'à ce que plus aucun item fréquent ne soit trouvé ; ce qui marque l'arrêt du processus.

Algorithme 3 (Approche par projections)
Entrée : *Une base de séquences D_z correspondant à une base projetée suivant un motif z, le motif courant z et un support minimum σ*
Sortie : *F, ensemble de tous les motifs séquentiels fréquents*
Premier appel : *GENERATION(D,$\emptyset$,σ)*
Appel récursif : *GENERATION($D_{|z}$,z,σ)*
Fonction appelée : *PROJECTION($D_{|z}$,z'). Effectue la projection de la base $D_{|z}$ suivant le motif z'*

$F := \emptyset$
$suffixeFreq = \{i \in I | support(i) \geq \sigma\ A \rightarrow B\ sur\ D_{|z}\ \}$
1. **for all** $i \in suffixeFreq$ **do**
2. $z' := z \rightarrow i$
3. $D_{|z'} := PROJECTION(D_{|z},z')$
4. $F := F \cup \{\ z'\ \} \cup GENERATION(D_{|z'},z',\sigma)$
5. **od**
6. **output** F

Dans [PHMAP01], un algorithme ce type, PrefixSpan, est présenté. PrefixSpan utilise des projections qui fonctionnent de la façon suivante : lorsque un motif α est identifié comme fréquent, alors la base est projetée selon ce motif, c'est-à-dire que la base est réduite aux sous-séquences des séquences initiales qui débutent sur les premiers items suivant les occurrences du motif fréquent α. À partir de cette projection, les items fréquents contenus dans la base projetée sont identifiés. Il est alors possible de former à l'aide ces items fréquents, les motifs dont le préfixe est α et dont le suffixe est un des items fréquents identifiés. On parle alors de projection

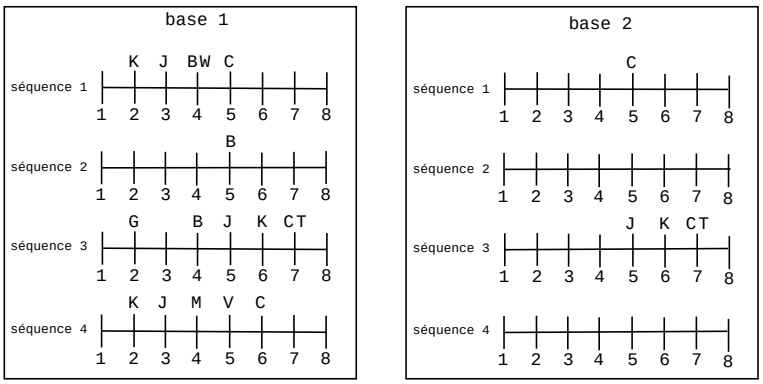


FIG. 2.5 – Exemples de bases projetées.

selon un préfixe, car le motif selon lequel la base est projetée est le motif qui sera le préfixe des motifs qu'il sera possible de découvrir dans la base nouvellement projetée. Comme on peut le constater, le phase de génération des candidats est réduite à son strict minimum. En effet, lors d'une projection selon un préfixe α, on envisage de pouvoir trouver des motifs de préfixe α et ce, sans les spécifier de quelque manière que ce soit. Ce sont les items fréquents dans la base projetée selon α, s'ils existent, qui, préciseront directement ces sur-motifs qui de surcroît seront déjà connus comme fréquents.

Toujours par souci de clarté, nous allons maintenant dérouler, en partie, l'algorithme PrefixSpan en ne nous intéressant qu'aux motifs dont les événements sont formés d'un seul item. Pour cela, considérons un support minimal relatif $\sigma = 3/4$, la base exemple de la figure 2.1 ainsi que la figure 2.5. La phase d'amorçage de l'algorithme fait apparaître que seuls les items A, B, C, J, K sont fréquents. Considérons maintenant la projection de la base de séquence initiale (figure 2.1) selon A. On obtient alors la base 1 de la figure 2.5. Cette base contient trois items fréquents qui sont B, C, J et K dont les supports sont tous égaux à 3/4. Par conséquent, les motifs $A \rightarrow B$, $A \rightarrow C$, $A \rightarrow J$ et $A \rightarrow K$ sont identifiés comme étant fréquents, leurs supports respectifs étant tous de 3/4. Considérons maintenant la projection de la base 1 selon $A \rightarrow B$. Celle-ci ne contient aucun item fréquent (cf. base 2 de la figure 2.5, ce qui signifie qu'aucun des sur-motifs de $A \rightarrow B$ n'est fréquent. L'algorithme considère alors la projection selon le motif suivant, c'est à dire $A \rightarrow C$, et poursuit sont exploration récursive de l'espace des motifs.

2.1.3 Gestion des contraintes

Les représentants des divers algorithmes abstraits présentés précédemment fonctionnent tous avec la contrainte de support qui impose que les motifs extraits soient fréquents. L'utilisation de cette seule contrainte peut parfois rencontrer ses limites, à la fois du point de vue de l'application et du point de vue technique.

Du point de vue strictement applicatif, extraire des motifs fréquents ne constitue pas toujours une réponse intéressante. En effet, de très nombreux motifs fréquents sont parfois obtenus, et parmi ceux-ci, peu se révèlent intéressants pour l'utilisateur. Tout cela sans compter l'effort que celui-ci doit fournir pour isoler ces quelques motifs intéressants parmi les autres motifs. Ainsi, spécifier à priori les motifs recherchés (c'est-à-dire *contraindre les motifs*), par exemple en utilisant une partie de la connaissance du domaine d'application de l'utilisateur, permet à la fois d'obtenir des résultats plus pertinents par rapport aux besoins de ce dernier, et à la fois de produire un résultat dont le volume n'est pas démesuré par rapport à la capacité d'analyse et d'interprétation de ce même utilisateur.

Du point de vue technique, une contrainte peut être gérée de façon passive ou de façon active. Le mode passif consiste par exemple à générer tous les motifs fréquents puis à ne retenir que ceux qui satisfont à une certaine contrainte, telle qu'une contrainte de durée maximale entre le premier événement et le dernier événement d'un motif. Si l'on considère la consommation des ressources en temps et mémoire, ceci est peu rentable, car la phase dite de *post-processing*, dont le but est de sélectionner les motifs satisfaisant la contrainte additionnelle, ne fera qu'ajouter une consommation supplémentaire en ressources de calcul. Au contraire, le mode actif, en intégrant au plus tôt la prise en compte de la contrainte lors d'extraction des motifs, permet de concentrer au plus vite les efforts de calcul sur les motifs susceptibles de satisfaire la contrainte et de réduire ainsi les ressources de calcul nécessaires.

Bien évidemment, il est possible de combiner plusieurs contraintes à la fois. On obtient alors un résultat d'autant plus concis, et ce pour une consommation des ressources d'autant plus réduite (en mode actif). Toutefois, il existe certaines contraintes pour lesquelles on ne connaît pas pour l'instant de techniques satisfaisantes permettant de les gérer de façon active sur le plan technique ; ce qui n'enlève rien à l'intérêt qu'elles peuvent représenter sur le plan applicatif.

Nous allons maintenant examiner de façon plus précise les contraintes, sur le plan applicatif et sur le plan technique.

Du point de vue applicatif, et sans vouloir prétendre à l'exhaustivité, les contraintes principalement utilisées dans la littérature (e.g., [SA96, MCP98, Zak00, PHW02, LRBE03a]) autres que la contrainte de support peuvent se répartir en 9 catégories :

– Catégorie 1 : une contrainte sur les *items* spécifie les items qui doivent apparaître ou non dans les motifs. On parle alors de contrainte *d'inclusion* et de contrainte *d'exclusion*. Par exemple, le motif $A \rightarrow J \rightarrow C$ satisfait à

la contrainte d'inclusion de l'item A. Il satisfait également à la contrainte d'exclusion de l'item K.

- Catégorie 2 : une contrainte de *longueur* spécifie la longueur, exacte, maximale ou minimale des motifs. Cette longueur peut par exemple être définie comme le nombre d'événements composant le motif ou comme le nombre d'items distincts composant le motif. Par exemple, si l'on impose que les motifs extraits ne doivent pas contenir plus de 3 événements, alors le motif $A \rightarrow J \rightarrow C$ peut être retenu.

- Catégorie 3 : une contrainte de *largeur* spécifie le nombre exact, maximal ou minimal d'items qui peuvent composer les événements formant les motifs. Par exemple, le motif $A \rightarrow J \rightarrow C$ satisfait à une contrainte de largeur maximale de 1.

- Catégorie 4 : une contrainte dite *super-pattern* est une contrainte qui pose comme condition que le motif contraint contienne un ensemble précis de sous-motifs. Par exemple, le motif $A \rightarrow J \rightarrow C$ satisfait à une contrainte imposant que le motif $J \rightarrow C$ soit un sous-motif des motifs extraits.

- Catégorie 5 : dans certaines applications, les items peuvent être associés à des valeurs. Par exemple, si l'on considère une séquence d'achats, chaque item peut être associé à la valeur du produit auquel il fait référence. Dans ce contexte, on peut alors utiliser les contraintes *d'agrégat simples*. Celles-ci portent sur un agrégat d'items dont la fonction d'agrégation peut être par exemple la somme minimale ou maximale (pour des items à valeurs positives), le maximum, et le minimum. Par exemple, si l'on découvre le motif $A \rightarrow J \rightarrow C$, alors les produits A, J et C sont achetés selon la séquence précédemment indiquée. Si le produit A vaut 10, si le produit J vaut 5 et si le produit C vaut 7, alors le motif $A \rightarrow J \rightarrow C$ satisfait à la contrainte imposant que la valeur maximum des valeurs d'achat soit inférieure ou égale à 10.

- Catégorie 6 : les contraintes *d'agrégat complexes* (*tough aggregate constraints*) portent sur un agrégat d'items dont la fonction d'agrégation peut être par exemple la somme minimale ou maximale (pour des items à valeurs positives et négatives), ou la moyenne. Par exemple, la séquence d'achat précédemment détaillée satisfait à une contrainte imposant que la moyenne des achats du motif soit inférieure à 10.

- Catégorie 7 : les contraintes *d'expression régulière* spécifient la forme syntaxique des motifs pouvant être formés à partir des items en utilisant des opérateurs tels que la disjonction ou la fermeture de Kleene. Un motif satisfait ce type de contrainte si il est accepté par un automate fini corres-

pondant à l'expression régulière. Par exemple, les motif $A \rightarrow J \rightarrow C$, $A \rightarrow A \rightarrow J \rightarrow C$ et $A \rightarrow J \rightarrow D$ satisfont à la contrainte d'expression régulière $(A*)J(C|D)$ (i.e. le motif commence par 0 ou plusieurs A, suivi(s) d'un J lui-même suivi de C ou D).

– Catégorie 8 : une contrainte de *durée* spécifie la durée minimale ou maximale entre le premier et le dernier événement des occurrences du motif séquentiel. Par exemple, l'occurrence du motif $A \rightarrow J \rightarrow C$ dans la première séquence de la base de séquence de la figure 2.1, commençant à la date 1 et finissant à la date 5, satisfait à une contrainte de durée minimale de trois unités temporelles.

– Catégorie 9 : une contrainte de *gap* spécifie la durée minimale ou maximale entre deux événements consécutifs du motif séquentiel. Par exemple, l'occurrence du motif $A \rightarrow J \rightarrow C$ dans la première séquence de la base de séquence de la figure 2.1, commençant à la date 1 et finissant à la date 5, satisfait à une contrainte de gap maximum de trois unités temporelles.

De façon plus générale, les catégories de contraintes précédemment évoquées peuvent être regroupées dans les trois classes suivantes :

– Les contraintes *syntaxiques* regroupent : les contraintes sur les items, sur la longueur, sur la largeur, les contraintes super-pattern, les contraintes d'expression régulière. Ces contraintes s'appliquent sur les motifs eux-mêmes. Leur prise en compte est donc essentiellement faite au niveau de l'étape de génération des motifs candidats.

– Les contraintes *temporelles* rassemblent : les contraintes de durée et de gap. Ces contraintes s'appliquent sur les occurrences des motifs. Leur prise en compte s'effectue au niveau de l'étape de comptage des motifs candidats.

– Les contraintes *d'agrégat simples et complexes* : tout comme pour les contraintes syntaxiques, elles s'appliquent sur les motifs eux-mêmes.

Il est à noter que d'autres contraintes existent bien que moins souvent mentionnées. Par exemple, l'algorithme SLPMiner [SK02] exploite une contrainte de support minimum variable selon la taille des motifs. Plus la tailles des motifs augmente, plus le support diminue. Cette contrainte s'appuie sur l'intuition selon laquelle plus un motif est long, plus il est rare ; ce qui n'enlève pas pour autant l'intérêt que l'on peut avoir à découvrir un tel motif. Ainsi, la contrainte de support est-elle envisagée de façon à ce que plus un motif est long, moins la contrainte qui lui est imposée soit restrictive.

Sur le plan technique, les contraintes sont généralement caractérisées selon les propriétés de *monotonie*, d'*anti-monotonie* et de *succinteness*. L'utilisation active d'une contrainte munie d'une de ces propriétés permet de réduire l'espace de recherche en évitant d'explorer les sous-espaces qui ne peuvent pas contenir de motifs satisfaisant la contrainte.

Une contrainte est *anti-monotone* si pour tout motif la satisfaisant, tous les sous-motifs qu'il contient satisfont également la contrainte. Une contrainte est *monotone* si pour tout motif la satisfaisant, tous les sur-motifs le contenant satisfont également la contrainte. Une contrainte est dite *succinte* si la spécification qui la définit permet de générer directement les motifs la satisfaisant.

Un exemple de contrainte monotone est la contrainte de longueur. Si cette contrainte pose que la longueur d'un motif doit être supérieure ou égale à k, alors les sur-motifs d'un $k-motif$ respectent aussi cette contrainte. Cette contrainte est également une contrainte succincte car sa spécification permet de générer directement les motifs la satisfaisant. La contrainte de support minimal est par exemple une contrainte anti-monotone. En effet, tous les sous-motifs d'un motifs fréquent sont fréquents. Il en est de même pour la contrainte d'exclusion d'un item.

Pour une caractérisation complète (selon les propriétés de monotonie, d'anti-monotonie, et de succinteness) de l'ensemble des contraintes précédemment citées, le lecteur pourra se référer à [PHW02]. Cette caractérisation fait apparaître que les contraintes d'expression régulière et les contraintes complexes d'agrégat ne sont ni anti-monotones, ni monotones, ni succinctes.

Afin d'intégrer les contraintes d'expression régulière dans un cadre permettant d'exhiber, pour toutes les contraintes (à l'exception des contraintes complexes d'agrégat), des propriétés les rendant utilisables en mode actif, Pei et al. propose dans [PHW02] de caractériser les contraintes selon la propriété dite *prefix-monotone*. Pour cela, il faut redéfinir la notion de préfixe de la façon suivante :

Définition 11 ***(préfixe)***
Soit un motif $\alpha = X_1 \rightarrow ... \rightarrow X_n$ tel que $\forall i = 1, .., n$, X_i soit un événement. Le motif $\beta = X_1 \rightarrow ... \rightarrow X_k \rightarrow Y$, avec Y un événement, est un préfixe de α si :
- $k < n$,
- $Y \subseteq X_{k+1}$,
- $\forall y \in Y$ et $\forall z \in (X_{k+1} \setminus Y)$, $y \prec z$, avec $\prec$ l'ordre établi sur les items.

Une contrainte *prefix anti-monotonic* est alors une contrainte telle que si le motif α satisfait la contrainte, alors tous les préfixes de α la satisfont également.

Une contrainte *prefix monotonic* est une contrainte telle que si un motif α satisfait la contrainte, alors tous les motifs ayant pour préfixe α la satisfont également.

Une contrainte est *prefix monotone* si elle est *prefix anti-monotonic* ou *prefix monotonic*.

De façon triviale, on a qu'une contrainte anti-monotone est une contrainte prefix anti-monotonic, et qu'une contrainte monotone est une contrainte prefix

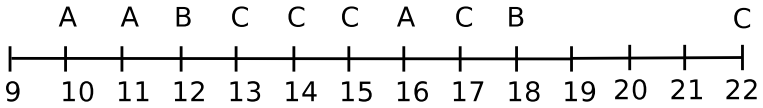


FIG. 2.6 – Exemple d'une séquence d'événements.

monotonic. En revanche, une contrainte succincte n'est pas forcément prefix monotone.

À l'exception des contraintes d'agrégat complexes, toutes les contraintes précédemment citées sont prefix-monotones et Pei et al. [PHW02] ont proposé un cadre général permettant leur prise en compte de façon active lors de l'extraction de motifs dans les bases de séquences.

2.2 Extraction de motifs dans une séquence d'événements

L'extraction de motifs dans une séquence d'événements concerne de longues séquences d'événements, c'est-à-dire des séquences qui peuvent s'étendre sur plusieurs dizaines voire centaines de milliers d'unités temporelles sur lesquelles sont dispersés plusieurs milliers d'événements. On retrouve ces séquences dans différents domaines d'application, que ce soit dans un contexte de gestion de réseaux de télécommunications [HKM+96] ou dans le cadre du projet européen AEGIS (projet au sein duquel ce travail de thèse a été effectué, cf. chapitre 1) avec l'analyse de données sismiques. Un exemple simplifié d'une telle séquence est donné par la figure 2.6. L'essentiel des apports dans ce domaine a été produit par Manilla et al. [MTV95, MT96, MTV97]. Aussi, la formalisation proposée dans [MTV97] sera-t-elle utilisée par la suite.

2.2.1 Définitions

Tout d'abord, définissons l'objet sur lequel s'effectue l'extraction, c'est-à-dire la *séquence d'événements*.

Définition 12 ***(Événement)*** Soit E un ensemble de *types d'événements*. Un *événement* est alors défini par la paire (A, t) avec $A \in E$ et, t, un entier définissant la date d'occurrence d'un événement.

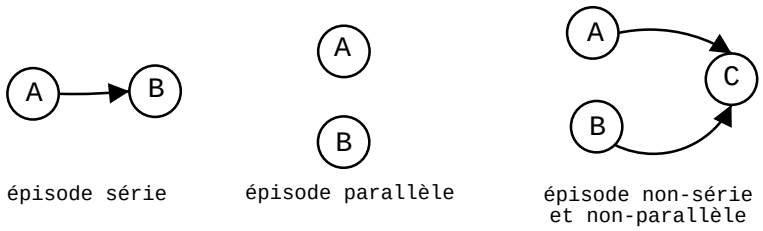


FIG. 2.7 – Les différents types d'épisodes.

Définition 13 ***(Séquence d'événements)*** Une *séquence d'événements* est le triplet (s, T_s, T_e) avec T_s un entier représentant la date à laquelle commence la séquence d'événements, T_e un entier, tel que $T_e \geq T_s$, représentant la date à laquelle finit la séquence d'événements, et $s =< (A_1, t_1), (A_2, t_2), ..., (A_n, t_n) >$, une *séquence ordonnée d'événements* telle que :

- $\forall i = 1, ..., n$, $A_i \in E$,
- $\forall i = 1, ..., n-1$, $t_i \leq t_{i+1}$,
- $\forall i = 1, ..., n$, $T_s \leq t_i < T_e$.

Par exemple, la séquence d'événements de la figure 2.6 peut se noter comme suit :
$(< (A, 10), (A, 11), (B, 12), (C, 13), (C, 14), (C, 15), (A, 16), (C, 17), (B, 18), (C, 22) >, 9, 22)$

Définissons maintenant les objets qui sont extraits des séquences d'événements. Tout d'abord considérons les *épisodes*. Si l'on considère la séquence exemple de la figure 2.6, on peut observer que le motif « *l'événement A est suivi de l'événement B* », noté $A \rightarrow B$, apparaît quelques fois. Ce type de motif est appelé épisode. Un épisode est une collection partiellement ordonnée de types d'événements qui peut être comparée à un graphe acyclique où les noeuds représentent les types d'événements et ou les arcs représentent l'ordre entre les types d'événements. Selon la nature de l'ordre utilisé, qui sera interprété comme un ordre de précédence temporelle, l'ensemble des épisodes peut être partitionné en trois classes : les épisodes dits *série*, les épisodes *parallèles* et les épisodes non-série et non-parallèles. La figure 2.7 illustre cette partition en donnant pour chaque classe une instance possible.

Plus formellement :

Définition 14 ***(Épisode, épisode série, épisode parallèle, taille d'un épisode)***

Un *épisode* e est un triplet $(V, \leq, g)$ où V est un ensemble de noeuds, $\leq$ est une relation d'ordre partiel et $g : V \mapsto E$, une fonction associant à chaque noeud un type d'événement. La nature de l'épisode dépend alors de la nature de l'ordre $\leq$ utilisé :

- si l'ordre est total, c'est-à-dire que l'on a $x \leq y$ ou $y \leq x$, pour tout x et y appartenant à V, alors l'épisode est un épisode *série*,
- si l'ordre est trivial, c'est-à-dire $x \not\leq y$, pour tout x et y appartenant à V tels que $x \neq y$, alors l'épisode est un épisode *parallèle*.

Soit $e = (V, \leq, g)$ un épisode. La *taille* de e, notée $|e|$, est $|V|$.

Par exemple, si l'on considère l'analyse de logs web, on peut s'intéresser aux trois types d'épisodes introduits :

- des épisodes séries : e.g. $p2 \rightarrow p3 \rightarrow p4$ qui signifie que les pages 2, 3 et 4 sont visitées l'une après l'autre, la page 2 étant visitée avant la page 3, cette dernière étant visitée avant la page 4.

- des épisodes parallèles : e.g. $(p2, p3, p4)$ qui signifie que les pages 2, 3 et 4 sont visitées sans pour autant spécifier un quelconque ordre de visite entre ces pages.

- des épisodes non-séries et non-parallèles : e.g. $(p2, p3) \rightarrow p4$ qui signifie que les pages 2 et 3 sont visitées avant la page 4 et ce sans que ne soit spécifié aucun ordre de visite entre les pages 2 et 3.

Les deux classes d'épisodes utilisées en pratique sont les épisodes séries et les épisodes parallèles. Pour ce qui est des épisodes non-séries et non-parallèles, nous n'avons pas trouvé trace de leur utilisation dans la bibliographie, bien que, comme le montre l'exemple précédent, ceux-ci pourraient se révéler intéressants d'un point de vue applicatif.

Les algorithmes des méthodes MINEPI et WINEPI présentées ci-après sont définis pour fonctionner avec les différentes classes d'épisodes mentionnées précédemment. En revanche, les illustrations que nous donnerons de ces algorithmes seront faites sur la base d'épisodes séries car notre contribution repose sur l'utilisation de tels épisodes.

Définition 15 ***(Sous-épisode, sur-épisode)*** Soit un épisode $\beta = (V, \leq, g)$ et un épisode $\alpha = (V', \leq', g')$. β est un *sous-épisode* de α (noté $\beta \preceq \alpha$), s'il existe une fonction injective $m : V' \mapsto V$ telle que, $\forall v, w \in V'$:

- $g'(v) = g(m(v))$,
- si $v \leq' w$, alors $m(v) \leq m(w)$.

On écrira $\beta \prec \alpha$ si $\beta \preceq \alpha$ et $\alpha \not\preceq \beta$.
Si $\beta \preceq \alpha$, alors α est un *super-épisode* de β.

Informellement, si β est une sous-épisode de α, alors l'épisode β est contenu dans l'épisode α. Par exemple, $A \rightarrow C$ est sous-épisode de $A \rightarrow B \rightarrow C$ qui lui-même est un super-épisode de $A \rightarrow C$.

Enfin, à partir des épisodes, on définit les *règles d'épisodes*, tout comme l'on définit les *règles d'association* à partir des *itemsets* pour extraire des motifs fréquents dans les matrices booléennes (cf. [AS95]).

Définition 16 ***(Règle d'épisodes)***
Une *règle d'épisodes* est une expression $\beta \Rightarrow \alpha$ avec α et β des épisodes tels que β soit un sous-episode de α.

2.2.2 Méthode WINEPI

Dans [MTV97], Manilla et al. proposent deux façons de rechercher les épisodes. Elles ont pour point commun la sélection des épisodes (resp. règles d'épisodes) dits *fréquents*, c'est à dire les épisodes (resp. règles d'épisodes) qui apparaissent suffisamment souvent dans une séquence d'événements. Que signifie le verbe « apparaître » dans cette dernière phrase ? La réponse a cette question est multiple, d'où une proposition double dans [MTV97]. La méthode WINEPI [MTV95, MTV97] instancie une des ces réponses. Celle-ci propose de définir l'*occurrence* (ou *apparition*) d'un épisode dans une séquence d'événements comme étant une *fenêtre* contenant au moins une fois l'épisode.

Formellement, sur une séquence d'événement, on définit les *sous-séquences d'événements*, ou *fenêtres*, de la façon suivante :

Définition 17 ***(Fenêtre, largeur d'une fenêtre)*** Une *fenêtre* $f = (w, t_s, t_e)$ est une séquence d'événements extraite d'une autre séquence d'événements $seq = (s, T_s, T_e)$ telle que :
- $t_s < T_e$ et $t_e > T_s$,
- w est composé de paires (A, t) de s telles que $t_s \leq t < t_e$.

On notera que dans cette définition, les événements contenus dans une fenêtre sont strictement antérieurs à la date de fin de fenêtre.

La *largeur de la fenêtre* est définie comme la quantité $t_e - t_s$ et elle est notée $width(f)$. L'ensemble de toutes les fenêtres de largeur win d'une séquence d'événements seq est noté $W(seq, win)$.

La séquence d'événements de la figure 2.6 contient plusieurs fenêtre dont par exemple : $f_{ex} = (< (A, 10), (A, 11), (B, 12) >, 10, 13)$. Sa largeur est $width(f_{ex}) = 3$.

Cette notion de fenêtre permet de définir la notion d'*occurrence d'un épisode*.

Définition 18 ***(Occurrence d'un épisode)***

Un épisode $e = (V, \leq, g)$ apparaît dans une fenêtre $f = (w, t_s, t_e)$ s'il existe une fonction injective $h : V \mapsto w$ qui associe un noeud $n \in V$ à un événement $(A, t) \in f$ telle que :

- $A = g(n)$,
- $\forall\ n_1, n_2 \in V$ tels que $n_1 \neq n_2$ et $n_1 \preceq n_2$, on a $t_1 < t_2$ avec $h(n_1) = (A_1, t_1)$ et $h(n_2) = (A_2, t_2)$.

f est alors une *occurrence* de e.

Par exemple, f_{ex} est une occurrence de l'épisode $A \rightarrow A$.

Une fois l'occurrence d'un épisode définie, il est alors possible de spécifier la notion de *support*. Cette dernière permet d'évaluer si un épisode apparaît *souvent* ou non dans une séquence d'événements en calculant le ratio entre le nombre de fenêtres contenant un épisode donné et le nombre total de fenêtres d'une séquence d'événements ; et ce pour une largeur de fenêtre fixée.

Définition 19 ***(Support d'un épisode)*** Le *support* d'un épisode e, dans une séquence d'événements seq, considérant des fenêtres de largeur win, est défini comme suit :
$sp(e, seq, win) = |\{f \in W(seq, win)\} \mid e\ apparait\ dans\ w|/|W(seq, win)|$

Par exemple, dans la séquence seq, $sp(A \rightarrow B \rightarrow C, seq, 4) = 2/16 = 1/8$. En effet, $A \rightarrow B \rightarrow C$ apparaît dans 2 fenêtres différentes. Plus précisément, il apparaît deux fois dans la fenêtre $(< (A, 10), (A, 11), (B, 12), (C, 13) >, 10, 14)$ et une fois dans la fenêtre $(< (A, 11), (B, 12), (C, 13), (C, 14) >, 11, 15)$. De plus, il existe 16 fenêtres de taille 4 : $(<>, 6, 10)$, $(< (A, 10) >, 7, 11)$, $(< (A, 10), (A, 11) >, 8, 12)$, $(< (A, 10), (A, 11), (B, 12) >, 9, 13)$, $(< (A, 10), (A, 11), (B, 12), (C, 13) >, 10, 14)$, $(< (A, 11), (B, 12), (C, 13), (C, 14) >, 11, 15)$, $(< (B, 12), (C, 13), (C, 14), (C, 15) >, 12, 16)$, $(< (C, 13), (C, 14), (C, 15), (A, 16) >, 13, 17)$, $(< (C, 14), (C, 15), (A, 16), (C, 17) >, 14, 18)$, $(< (C, 15), (A, 16), (C, 17), (B, 18) >, 15, 19)$, $(< (A, 16), (C, 17), (B, 18) >, 16, 20)$, $(< (C, 17), (B, 18) >, 17, 21)$, $(< (B, 18) >, 18, 22)$, $(< (C, 22) >, 19, 23)$, $(< (C, 22) >, 20, 24)$, et $(< (C, 22) >, 21, 25)$.

De même, on peut définir le support d'une règle d'épisodes.

Définition 20 ***(Support d'une règle d'épisode)*** Pour une règle d'épisode $\beta \Rightarrow \alpha$, observée dans une séquence d'événements seq, selon des fenêtres de largeur win, le *support* est définie par :
$sp(\beta \Rightarrow \alpha, seq, win) = sp(\alpha, seq, win)$

Par exemple, dans la séquence seq, $sp(A \rightarrow B \Rightarrow A \rightarrow B \rightarrow C, seq, 4) = sp(A \rightarrow B \rightarrow C, seq, 4) = 1/8$.

Cette définition du support peut alors conduire, pour ce qui est des règles d'épisodes, à la définition de la *confiance*. Si on considère la règle $\beta \Rightarrow \alpha$, alors la confiance exprime la probabilité conditionnelle d'observer α dans une fenêtre sachant que β apparaît dans cette fenêtre.

Définition 21 ***(Confiance d'une règle d'épisodes)*** Pour une règle d'épisode $\beta \Rightarrow \alpha$, observée dans une séquence d'événements seq, selon des fenêtres de largeur win, la *confiance* est définie comme suit :
$cf(\beta \Rightarrow \alpha, seq, win) = sp(\alpha, seq, win)/sp(\beta, seq, win)$.

Si l'on reprend l'exemple de la figure 2.6 : $cf(A \rightarrow B \Rightarrow A \rightarrow B \rightarrow C, seq, 4) = \frac{2/16}{5/16} = 2/5$.

Les motifs calculés avec WINEPI sont les règles dont le support est supérieur ou égale à un seuil donné σ (notion de règle *fréquente*) et dont la confiance est supérieure ou égale à une seuil donné γ (notion de règle *confiante*).

Algorithmes

Le principe de WINEPI peut s'expliciter à partir des deux algorithmes abstraits suivants (algorithme 4 et algorithme 5). Le premier, l'algorithme 4, spécifie la façon dont sont extraits les *épisodes fréquents*, c'est-à-dire les épisodes dont le support est supérieur σ. À partir de ces épisodes fréquents et de leurs supports respectifs, l'algorithme 5 se charge de constituer les différentes règles possibles, et de sélectionner les règles *confiantes*, c'est à dire les règles dont la confiance est supérieure ou égal à γ. Ces règles étant constituées par des épisodes fréquents, elles sont également fréquentes.

En ce qui concerne le premier algorithme, celui-ci fait apparaître deux phases, une phase de génération des épisodes candidats (lignes 1 et 6), et une phase de comptage (ligne 4) permettant d'évaluer le support des dits-candidats et de déterminer les épisodes fréquents. L'algorithme fonctionne en largeur, c'est à dire que les épisodes candidats d'une taille donnée sont tous générés, et leurs supports évalués, avant de générer et d'évaluer le support des épisodes candidats de taille immédiatement supérieure. L'algorithme commence par les épisodes les

Algorithme 4 (WINEPI - Algorithme 1)
Entrée : *E, un ensemble de types d'événements, une séquence d'événements s sur E, une taille de fenêtre win, un support minimal σ et une classe d'épisode ϵ.*
Sortie : *La collection $F(s, win, \sigma)$ des épisodes fréquents.*

1. $C_1 := \{\alpha \in \epsilon \mid |\alpha| = 1\}$
2. $l := 1$
3. **while** $C_l \neq \emptyset$ **do**
4. *calcul de* $F_l := \{\alpha | \alpha \in C_l \wedge sp(\alpha, s, win) \geq \sigma\}$
5. $l := l + 1$
6. *calcul de* $C_l = \{\alpha \in \epsilon \mid |\alpha| = l \wedge \forall \beta, \beta \prec \alpha \wedge |\beta| < l \Rightarrow \beta \in F_{|\beta|}\}$
7. **od**
8. **output** $\bigcup_{1 \leq i \leq l} F_i$

plus généraux, i.e. les épisodes de taille 1 (ligne 1), et progresse pas à pas vers les épisodes de taille supérieure en utilisant de façon active la contrainte de support qui est une contrainte anti-monotone. Ainsi, lorsque un épisode e n'est pas fréquent, aucun épisode candidat ne sera formé à l'aide de cet épisode e, car aucun sur-épisode de e ne peut être fréquent. Ceci permet d'éviter la génération de certains candidats et peut être tout à fait comparé dans le principe aux approches de type Apriori présentées en section 2.1.2. La différence principale réside alors dans la définition des motifs calculés et le mode d'évaluation de la fréquence de ces motifs. L'algorithme s'arrête lorsque plus aucun candidat ne peut être généré (ligne 3).

Pour en revenir à l'évaluation du support des épisodes candidats, celle-ci fait appel, par définition, à une taille de fenêtre *win*. Cela permet de fixer la durée maximale sur laquelle peut s'étendre l'apparition d'un motif dans la séquence, ce qui de fait peut être considéré comme une contrainte sur les occurrences des épisodes, et donc une contrainte sur les règles d'épisodes. Cette contrainte permet de réduire, pour un épisode candidat, le nombre d'occurrences de celui-ci, ce qui a pour effet de diminuer le nombre d'épisodes fréquents trouvés à l'issue d'une itération, et donc de réduire le nombre d'épisodes candidats dont il faut vérifier la fréquence lors de l'itération suivante. Enfin, on notera que l'évaluation des supports des épisodes candidats requiert, quelque soit la taille de ceux-ci (i.e. quelque soit l'itération), une passe complète sur la base de données.

L'algorithme 5 détaille la reconnaissance des règles fréquentes et confiantes. Cette reconnaissance s'effectue en deux phases :

– génération d'une règle d'épisode candidate : les règles d'épisodes candidates sont formées à partir d'une paire d'épisodes fréquents (lignes 1 et 2), tels que l'un soit le sous-épisode de l'autre (ligne 2). On notera que si un épisode α est fréquent, alors tous ses sous-épisodes β le sont aussi (propriété d'anti-monotonicité). Ainsi, tout comme α, β appartient de fait à l'ensemble des épisodes fréquents fournis en entrée à l'algorithme. Cette construction des règles assure par définition que les règles candidates produites sont fréquentes.

– vérification de la contrainte de confiance : une fois la règle d'épisode candidate produite, l'algorithme s'assure (ligne 3) que la confiance de cette règle est supérieure à la limite γ fixée par l'utilisateur. Si la règle est confiante, alors celle-ci est sortie comme résultat de l'algorithme (ligne 4).

Algorithme 5 (WINEPI - Algorithme 2)
Entrée : *Un seuil minimum de confiance γ, la collection $F(s, win, \sigma)$ des épisodes fréquents sur une séquence s, pour une taille de fenêtre win.*
Sortie : *Les règles d'épisodes contenues dans s dont le support est supérieur à σ et dont la confiance est supérieure à γ*

1. **for all** $\alpha \in F(s, win, \sigma)$ **do**
2. **for all** $\beta \prec \alpha$ **do**
3. **if** $sp(\alpha)/sp(\beta) \geq \gamma$ **then**
4. **output** *la règle $\beta \Rightarrow \alpha$, sa confiance $sp(\alpha)/sp(\beta)$ et son support $sp(\alpha)$*
5. **fi**
6. **od**
7. **od**

Reprenons maintenant l'exemple de la séquence s (cf. 2.6) et envisageons le déroulement de la méthode MINEPI pour une largeur de fenêtre de 4, un support minimum de 5/16 et une confiance minimum de 1/2. L'algorithme 4 est tout d'abord utilisé. Les épisodes ne contenant qu'un événement sont recherchés. On trouve l'épisode A (dont le support est 9/16), l'épisode B (dont le support est 8/16) et l'épisode C (dont le support est 11/16). Ces épisodes étant tous fréquents, les motifs candidats générés sont basés sur A, B et C. Les occurrences de ces motifs candidats sont alors recherchées et les supports respectifs sont établis. On a alors : $sp(A \rightarrow A, s, 4) = 3/16$, $sp(A \rightarrow B, s, 4) = 5/16$, $sp(A \rightarrow C, s, 4) = 5/16$, $sp(B \rightarrow A, s, 4) = 0/16$, $sp(B \rightarrow B, s, 4) = 0/16$, $sp(B \rightarrow C, s, 4) = 3/16$, $sp(C \rightarrow A, s, 4) = 3/16$, $sp(C \rightarrow B, s, 4) = 3/16$, et $sp(C \rightarrow C, s, 4) = 5/16$.

Seuls $A \rightarrow B$, $A \rightarrow C$ et $C \rightarrow C$ sont donc identifiés comme fréquents. C'est à partir de ces épisodes que sont générés les candidats de taille supérieure (ligne 6 de l'algorithme 4) tel que $A \rightarrow C \rightarrow B$. Puis les occurrences de ces candidats sont recherchés. Aucun candidat n'étant identifié comme fréquent, l'algorithme 4 s'arrête et renvoie l'ensemble des épisodes fréquents (ainsi que leurs supports respectifs) qui sont ensuite utilisés en entrée de l'algorithme 5. Pour chaque épisode fréquent α, l'algorithme 5 construit toutes les règles possibles à partir de sous-épisodes fréquents de α et évalue la confiance de ces règles. C'est ainsi que la confiance de la règle $A \Rightarrow A \rightarrow B$ et de la règle $A \Rightarrow A \rightarrow C$ est évaluée à 5/9. La confiance de la règle $C \Rightarrow C \rightarrow C$ est quant à elle évaluée à 5/11. Seules sont donc retenues les règles $A \Rightarrow A \rightarrow B$ et $A \Rightarrow A \rightarrow C$ comme étant fréquentes et confiantes. Ce sont ces règles qui sont sorties comme résultat de l'algorithme 5 (ligne 4).

2.2.3 Méthode MINEPI

L'algorithme MINEPI repose sur une définition d'occurrence des épisodes différente de celle utilisée par WINEPI. Cette fois-ci, une occurrence d'un épisode n'est pas une fenêtre f quelconque de taille $width(f)$ telle que ce même épisode y apparaisse. En effet, la fenêtre f doit être, dans ce cas, telle qu'il n'y ait pas d'autre fenêtre f' plus petite à l'intérieur de f dans laquelle apparaisse à nouveau l'épisode. De plus, l'occurrence n'est pas la fenêtre $f = (w, t_s, t_e)$ à proprement parler, mais l'intervalle noté $[t_s, te)$. Cet intervalle est appelé *occurrence minimale*.

De façon formelle :

Définition 22 ***(Occurrence minimale)***
Une *occurrence minimale* d'un épisode α est un intervalle $[t_s, t_e)$ tel que :
- α apparaît dans la fenêtre $f = (w, t_s, t_e)$,
- α n'apparaît dans aucune des fenêtres $f' = (w', t'_s, t'_e)$ telle que $t_s \leq t'_s$, $t'_e \leq t_e$ et que $width(f') < width(f)$.

Si l'on observe l'exemple donné par la figure 2.6, l'épisode $A \rightarrow B$ apparaît deux fois dans la séquence au sens des occurrences minimales (occurrence $[11, 13)$ et occurrence $[16, 19)$), alors qu'en se basant sur la définition d'une occurrence utilisée par WINEPI, on trouve 5 occurrences pour une taille de fenêtre de 4. La définition d'occurrence minimale d'un épisode semble plus intuitive car plus proche du sens commun.

L'ensemble des occurrences minimales d'un épisode α, noté $mo(\alpha)$, est défini comme suit : $mo(\alpha) = \{[t_s, t_e) \mid [t_s, t_e) est\ une\ occurence\ minimale\ de\ \alpha\}$.

Le *support* d'un épisode peut alors être défini. Il s'agit d'une mesure dont le sens est proche de celui de la fréquence à la différence près que le support est une mesure absolue alors que la fréquence est une mesure relative

Définition 23 ***(Support d'un épisode)***
Soit α un épisode et seq une séquence d'événements. Le support de α est défini par :
$supp(\alpha, seq) = |mo(\alpha)|$

Tout comme précédemment, on peut considérer la recherche de règles d'épisodes. Dans le cadre de MINEPI, celles-ci sont envisagées sous une forme $\beta_{[win_1]} \Rightarrow \alpha_{[win_2]}$, avec β un sous-épisode de α et win_1, win_2, deux entiers. De façon informelle, cette règle peut être lue de la façon suivante : si β a une occurrence minimale $[t_s, t_e)$ telle que $t_s - t_e \leq win_1$ alors l'épisode α a une occurrence minimale $[t'_s, t'_e)$ telle que $t'_e - t_s \leq win_2$.

Pour ce type de règle, on peut à nouveau définir la confiance :

Définition 24 ***(Confiance d'une règle d'épisodes)***
Soit $mo_{win_1}(\beta) = \{[t_s, t_e) \in mo(\beta) | t_e - t_s \leq win_1\}$ et soit $occ(\alpha, [u_s, u_e)) = vrai$ si et seulement s'il existe une occurrence minimale $[u'_s, u'_e)$ telle que $u_s \leq u'_s$ et $u'_e \leq u_e$; alors le ratio suivant exprime la *confiance* de la règle $\beta_{[win_1]} \Rightarrow \alpha_{[win_2]}$: $\frac{|\{[t_s,t_e) \in mo_{win_1}(\beta) | occ(\alpha,[t_s,t_s+win_2))\}|}{|mo_{win_1}(\beta)|}$

MINEPI a pour objectif de calculer les règles du type $\beta_{[win_1]} \Rightarrow \alpha_{[win_2]}$ qui sont fréquentes (c'est-à-dire dont le support de l'épisode α est supérieur à un seuil σ) et confiantes (c'est-à-dire dont la confiance est supérieure à un seuil γ).

Algorithmes

La méthode MINEPI se compose, tout comme la méthode WINEPI, de deux algorithmes principaux. Le premier algorithme (algorithme 6) permet d'établir les épisodes fréquents qui sont par la suite utilisés par le second algorithme (algorithme 7) afin d'identifier les règles confiantes et fréquentes.

L'algorithme présenté ci-après est, dans ses grandes lignes, pratiquement identique à l'algorithme 4 proposé dans le cadre la méthode WINEPI, à une exception près : le paramètre de taille de fenêtre win n'est plus fourni en entrée de l'algorithme. En effet, l'occurrence d'un épisode définie dans le cadre de MINEPI ne fait plus appel à la notion de taille de fenêtre.

À nouveau, on retrouve l'étape de génération des candidats (ligne 6), et l'étape de comptage des supports permettant de déterminer les épisodes fréquents (ligne 4). Le comptage des supports nécessite une seule passe sur la séquence. En effet, lors d'une première et unique passe, $mo(\alpha)$ est évalué pour tout épisode α tel que

Algorithme 6 (MINEPI - Algorithme 1)
Entrée : *E, un ensemble de types d'événements, une séquence d'événements s sur E, une fréquence minimale σ et une classe d'épisode ϵ.*
Sortie : *La collection $F(s, \sigma)$ des épisodes fréquents.*

1. $C_1 := \{\alpha \in \epsilon || \alpha| = 1\}$
2. $l := 1$
3. **while** $C_l \neq \emptyset$ **do**
4. *calcul de* $F_l := \{\alpha \in C_l \mid |mo(\alpha)| \geq \sigma\}$
5. $l := l + 1$
6. *calcul de* $C_l = \{\alpha \in \epsilon \mid |\alpha| = l \wedge \forall \beta, \beta \prec \alpha \wedge |\beta| < l \Rightarrow \beta \in F_{|\beta|} \}$
7. **od**
8. **output** $\bigcup_{1 \leq i \leq l} F_i$

$|\alpha| = 1$. Lors des itérations suivantes, les occurrences d'un épisode candidat α sont alors calculées à partir des occurrences minimales des épisodes fréquents de taille $|\alpha| - 1$. Dans le cas d'épisode séries, le calcul des occurrences d'un motif candidat α est effectué en :

- (1) : choisissant α_1 et α_2 dans $F_{|\alpha-1|}$ tels que α_1 contienne tous les événements composant α excepté le dernier et que α_2 contienne tous les événements composant α excepté le premier.
- (2) : effectuant une jointure temporelle entre les occurrences minimales de α_1 et les occurrences minimales de α_2. Cette jointure est définie ainsi : $mo(\alpha) = \{[t_s, u_e) \mid \exists\, [t_s, t_e) \in mo(\alpha_1) \wedge [u_s, u_e) \in mo(\alpha_2)$ tels que $t_s < u_s \wedge t_e < u_e$ et que $\forall\, [t'_s, t'_e) \in mo(\alpha_1), t'_s > t_s \Rightarrow t'_e > u_e$ et que $\forall\, [u'_s, u'_e) \in mo(\alpha_2), u'_e < u_e \Rightarrow u'_s > u_s\}$.

Ce calcul de $mo(\alpha)$ est tout à fait dans l'esprit des jointures temporelles utilisées dans le cas des listes d'occurrences proposé par Zaki dans [Zak01] et présentées dans la section 2.1.2.

Une fois les épisodes fréquents identifiés, l'algorithme suivant est employé afin d'établir les règles fréquentes et confiantes de la forme $\beta_{[win_1]} \Rightarrow \alpha_{[win_2]}$, avec β un sous-épisode de α, et win_1, win_2 des tailles de fenêtres.

Cet algorithme est très proche de l'algorithme 5. La différence principale réside dans le fait que le support de chaque épisode n'est pas connu mais qu'il doit être établi (lignes 3 à 9) à partir des listes d'occurrences des épisodes constituant les règles d'épisodes et ce, en fonction de la définition 24. Une fois les supports établis pour une règle d'épisode donnée, alors la confiance de la règle est évaluée, et si celle-ci est supérieure ou égale au seuil γ et que le support de la règle est supérieur ou égal au seuil σ (ligne 10), alors la règle d'épisode est sortie comme résultat.

Algorithme 7 (MINEPI - Algorithme 2)
Entrée : *La collection $F(s,\sigma)$ des épisodes fréquents et leurs occurrences minimales sur une séquence s, des tailles de fenêtre win_1 et win_2, et un seuil de support minimum σ et un seuil de confiance minimum γ.*
Sortie : *Les règles d'épisodes contenues dans s de la forme $\beta_{[win_1]} \Rightarrow \alpha_{[win_2]}$, dont le support est supérieur à σ, et dont la confiance est supérieure à γ.*

1. **for all** $\alpha \in F(s,\sigma)$ **do**
2. **for all** $\beta \prec \alpha$ **do**
3. $support_\beta = 0$ *;* $support_\alpha = 0$
4. **for all** $[t_s, t_e) \in mo(\beta) | t_e - t_s \leq win_1$ **do**
5. $support_\beta := support_\beta + 1$
6. **if** $\exists [u_s, ue) \in mo(\alpha) | t_s \leq u_s \ \wedge \ u_e - t_s \leq win_2$ **then**
7. $support_\alpha := support_\alpha + 1$
8. **fi**
9. **od**
10. **if** $support_\alpha / support_\beta \geq \gamma \wedge support\alpha \geq \sigma$ **then**
11. **output** *la règle* $\beta_{win_1} \Rightarrow \alpha_{win_2}$*, sa confiance* $\frac{support_\alpha}{support_\beta}$*, et son support* $support\alpha$
12. **fi**
13. **od**
14. **od**

On notera que les contraintes de fenêtre imposée par la définition d'une confiance d'une règle (win_1 et win_2 sont ici gérées de façon passive). Dans la pratique, lorsque on extrait les occurrences minimales nécessaires à la génération des règles, on se limite aux occurrences de largeur inférieure ou égale à win_2 (utilisation active de la contrainte win_2).

Si l'on applique la méthode MINEPI à la séquence d'événements de la figure 2.6, l'algorithme 6 est tout d'abord enclenché. À l'issue de la première passe (ligne 1) sur la base de séquences, les épisodes ne comportant qu'un événement sont identifiés et leurs occurrences minimales sont relevées. Ainsi A a pour occurrences minimales $[10,11), [11,12)$, et $[16,17)$. B a pour occurrences minimales $[12,13)$, et $[18,19)$. C a pour occurrences minimales $[13,14), [14,15), [15,16), [17,18)$ et $[22,23)$. Si le support minimal des épisodes est fixé à 3, seuls A, et C sont fréquents. C'est donc à partir de ceux-ci que sont générés les candidats. Puis, en appliquant la jointure temporelle décrite précédemment les listes d'occurrences

minimales des épisodes candidats sont calculées et les supports respectifs sont établis. C'est ainsi que l'on obtient l'information sur les candidats selon laquelle le support de $A \rightarrow A$ est 2 (occurrences minimales $[10, 12)$ et $[11, 17)$), le support de $A \rightarrow C$ est 2 (occurrences minimales $[11, 14)$ et $[16, 18)$), le support de $C \rightarrow A$ est 1 (occurrences minimales $[15, 17)$), le support de $C \rightarrow C$ est 4 (occurrences minimales $[13, 15), [14, 16), [15, 18)$ et $[17, 23)$). Seul l'épisode $C \rightarrow C$ est maintenant considéré comme fréquent. À partir de celui-ci on peut générer le candidat $C \rightarrow C \rightarrow C$. Par jointure temporelle, on obtient alors les occurrences $[13, 16)$, $[14, 18)$, et $[15, 23)$. $C \rightarrow C \rightarrow C$ est donc fréquent. Le motif $C \rightarrow C \rightarrow C \rightarrow C$ est alors généré comme candidat. Par jointure, on obtient les occurrences $[13, 18)$ et $[14, 23)$. $C \rightarrow C \rightarrow C \rightarrow C$ n'est donc pas fréquent et l'algorithme 6 renvoie l'ensemble des épisodes fréquents identifiés ainsi que leurs listes respectives d'occurrences (ligne 8) à l'algorithme 7 qui va tester l'ensemble des règles pouvant être construites à partir de ces épisodes fréquents. Si l'on fixe $win_1 = 2$ et $win_2 = 3$ et que l'on considère la règle $C \rightarrow C \Rightarrow C \rightarrow C \rightarrow C$, alors seules les occurrences $[13, 15)$ et $[14, 16)$ sont retenues pour l'épisode $C \rightarrow C$ et seule l'occurrence $[13, 16)$ est retenue pour l'épisode $C \rightarrow C \rightarrow C$. Ainsi la confiance de la règle $C \rightarrow C \Rightarrow C \rightarrow C \rightarrow C$ est évaluée à $1/2$. Si la confiance minimale est de $1/2$, alors cette règle est ajoutée au résultat (ligne 11). L'algorithme 7 procède ainsi pour toutes les combinaisons possibles d'épisodes fréquents. Ce n'est qu'une fois l'ensemble des combinaisons explorées que celui-ci parvient à son terme.

2.3 Conclusion

Qu'il s'agisse de l'analyse de bases de séquences d'événements ou de l'analyse d'une longue séquence d'événements, les données utilisées revêtent la forme d'une liste ordonnée de symboles, chaque symbole étant associé à une date d'apparition, laquelle doit être supérieure ou égale à la date d'apparition du symbole précédent. Pour ce qui est des motifs recherchés dans ces données, ceux-ci sont identiques pour ce qui concerne les motifs séquentiels dans les bases de séquences et les épisodes série dans une longue séquence d'événements. Dans les deux cas, il s'agit de listes ordonnées de symboles. En ce qui concerne les épisodes, série ou non, un autre type de motif, la règle d'épisode, est également utilisé afin de faire apparaître de façon plus précise la connexion entre les divers éléments, et ce au moyen de la mesure de confiance.

Pour ce qui est des divers algorithmes disponibles, ceux-ci doivent tous affronter des espaces de motifs candidats conséquents. Par exemple, si la taille des motifs n'est pas limitée, le nombre de motifs candidats devient infini. En effet, à un motif candidat précis, il est toujours possible de rajouter un événement afin de former un nouveau motif candidat. Afin de gérer les espaces de motifs candidats, diverses contraintes sont utilisées afin de restreindre ces espaces, et donc de réduire, si

possible, les coûts de calcul, et de cerner, au sens applicatif, les motifs considérés comme les plus intéressants. La contrainte utilisée par tous les algorithmes est la contrainte dite de fréquence. Celle-ci impose que les motifs recherchés « apparaissent suffisamment ». Elle permet à l'ensemble des algorithmes détaillés dans ce chapitre de limiter le nombre de motifs candidats. Pour cela, la génération de motifs candidats est basée sur l'utilisation de motifs identifiés comme fréquents (un motif ne pouvant pas être fréquent si l'un des sous-motifs le composant n'est pas fréquent). Toujours en ce qui concerne les gains en terme de coûts de calcul, certains des algorithmes tentent de tirer parti d'une représentation interne et concise de la base de donnée afin de réduire au maximum les accès disques aux données. C'est le cas des algorithmes SPADE [Zak01], CSPADE [Zak00], GO-SPADE [LRBE03b], GO-SPEC [LRBE03a], PrefixSpan [PHMAP01], [PHW02], et de l'algorithme MINEPI [MT96, MTV97]. En particulier, on notera que l'algorithme SPADE [Zak01] et l'algorithme MINEPI [MT96, MTV97] utilisent une représentation interne des données similaire, alors que l'un concerne l'analyse d'une base de séquences (SPADE) et que l'autre permet l'analyse d'une seule et longue séquence d'événements (MINEPI).

Pour en revenir aux diverses contraintes gérées par les différents algorithmes proposés, on notera que les contraintes de temps semblent être celles ayant reçu le plus d'attention. Qu'il s'agisse de l'algorithme PSP [MCP98], de CSPADE [Zak00], de GO-SPEC [LRBE03a], de Prefixgrowth [PHW02], de MINEPI [MT96, MTV97] ou de l'algorithme WINEPI [MTV95, MTV97], ceux-ci utilisent en effet des contraintes temporelles (contraintes de fenêtre et/ou de gap maximum). Plus précisément, et alors que tous les algorithmes précédemment cités permettant l'analyse de bases de séquences assurent la prise en compte de la contrainte de gap maximum, il apparaît que les algorithmes explorant une seule et longue séquence d'événements (WINEPI et MINEPI) ne gèrent pas cette contrainte mais utilisent plutôt la contrainte de fenêtre maximum. Le chapitre 3 propose notamment une façon de prendre en compte cette contrainte, et ce de manière active pour rechercher des règles d'épisodes dans une longue séquence d'événements.

Chapitre 3

Recherche de fenêtres optimales

3.1 Motivation

De nombreux jeux de données prennent, éventuellement après une discrétisation appropriée, la forme d'une longue séquence d'événements, chaque événement étant décrit par une date d'apparition et par un type d'événement. C'est notamment le cas des données sismiques que nous avons eu à traiter dans le cadre de ce travail de thèse (e.g., [BDMR03]) effectué au sein du projet AEGIS. Les motifs les plus utilisés pour appréhender ce type de jeux de données sont les règles d'épisodes (cf. section 2.2).

Le problème standard d'extraction de règles d'épisodes est de trouver toutes les règles satisfaisant à des contraintes de fréquence et de confiance données. Comme nous l'avons vu au chapitre 2, il existe deux approches pour trouver de telles règles. La première, proposée et utilisée dans [MTV95, MTV97] est la méthode WINEPI, méthode basée sur la recherche des occurrences des épisodes à l'aide d'une fenêtre glissante le long de la séquence. La deuxième, introduite dans [MT96, MTV97], est la méthode MINEPI. Celle-ci est basée sur la notion *d'occurrence minimale*. Les deux techniques ont été conçues pour être utilisées en employant une contrainte de taille de fenêtre maximum, contrainte qui spécifie la distance (temporelle ou spatiale) maximale entre le premier et le dernier événement des occurrences des motifs extraits. Plus précisément, dans le cas de WINEPI, l'algorithme utilise une seule contrainte de taille de fenêtre, et il doit être à nouveau exécuté si l'utilisateur souhaite faire une extraction avec une taille de fenêtre différente. Dans le cas de MINEPI, une contrainte de taille de fenêtre maximale doit aussi être utilisée en pratique afin de réduire l'espace de recherche. En revanche, MINEPI permet

d'extraire des règles pour des tailles de fenêtre inférieures à la taille spécifiée pour la fenêtre maximale.

À notre connaissance, il n'existe, à ce jour, aucun algorithme complet capable d'extraire des règles d'épisodes sur des jeux de données non triviaux sans spécifier au minimum une contrainte de taille de fenêtre maximale (en plus d'une contrainte de fréquence et de confiance). Or, certaines applications telle que la recherche de dépendances entre événements sismiques, sont des applications pour lesquelles la taille de fenêtre n'est pas connue à l'avance. De plus, la taille de fenêtre intéressante peut varier d'une règle d'épisodes à une autre. En effet, un même jeu de données peut « contenir » des phénomènes dont les durées ont des valeurs différentes. Pour prendre en compte ce type d'applications, il a récemment été proposé dans [CG03] d'utiliser une contrainte similaire à la contrainte de gap maximum. Cette contrainte permet aux occurrences des motifs de s'étendre sur des intervalles d'autant plus larges que les motifs recherchés sont longs, ces intervalles n'étant pas directement limités par une taille de fenêtre maximum. Cette contrainte est analogue à la contrainte de gap maximum prise en compte par les algorithmes (e.g., [SA96, Zak00, LRBE03a]) conçus pour la recherche de motifs séquentiels dans une base de séquence (cf. section 2.1.3). Cependant, comme nous le montrerons dans la section 3.2.3, l'algorithme proposé dans [CG03] est incomplet. En revanche, la contribution faite dans [CG03] reste intéressante car elle suggère que la recherche des règles d'épisodes pourrait, en pratique, être faite en utilisant une contrainte de gap maximum. Nous proposons dans ce chapitre, un algorithme juste et complet pour réaliser de telles extractions.

Pour autant que nous sachions, et bien que l'expert en exprime souvent le besoin, aucune mesure d'intérêt dédiée permettant de compléter l'information apportée par les mesures de support et de confiance n'a été proposée dans le cadre de l'extraction de règles d'épisodes, alors que cela a été fait dans des domaines similaires tel que la recherche de règles d'association (e.g., [GGP98, BA99]) ou dans un cadre plus général [FGLS98]. Pour cela, nous définissons une mesure d'intérêt, le *lift* [1], permettant d'évaluer le rapport entre la confiance observée sur la séquence d'événements et la *confiance attendue* d'une règle. Cette dernière est calculée sous l'hypothèse d'indépendance du corps et de la tête des règles. Le lift permet ainsi de sélectionner les règles dont on peut considérer que leur confiance est sensiblement plus élevée que la confiance à laquelle nous aurions pu nous attendre en supposant l'indépendance entre corps et têtes de règles.

Afin de fournir une indication concernant la durée des dépendances potentielles extraites sont la forme de règles d'épisodes, nous proposons de rechercher, pour chaque règle d'épisode, la taille de la fenêtre temporelle optimale. Celle-ci est définie comme étant la plus petite taille de fenêtre correspondant à un maximum

[1] Nom employé par référence à la mesure de lift utilisée dans le cas des règles d'association (e.g., [HTF01]).

local de confiance (i.e. la confiance est localement plus basse pour les tailles de fenêtres inférieures et supérieures). Nous proposons notamment une détermination efficace de ces premiers maximums locaux de confiance basée sur l'utilisation d'un calcul incrémental de confiance pour les différentes tailles de fenêtre. Ce calcul s'appuie sur la notion d'occurrence minimale, utilisée par MINEPI, pour lesquelles le support peut être défini sans recourir à une notion de taille fenêtre, contrairement aux occurrences utilisées par WINEPI. Enfin, pour éviter de submerger l'utilisateur avec de grandes collections de règles extraites, nous proposons de n'extraire que les règles présentant une telle singularité temporelle (i.e. les règles pour lesquelles il existe au moins une fenêtre localement optimale). Ces règles sont appelées *FLM-règles*.

La suite de ce chapitre est organisée comme suit. La section suivante donne des définitions préliminaires et montre en quoi l'algorithme présenté dans [CG03] est incomplet. La section 3.3 introduit l'algorithme *WinMiner* qui permet de gérer la contrainte de gap maximum et de trouver les règles d'épisodes présentant un maximum local de confiance. Cette section présente également les preuves de justesse et de complétude de *WinMiner* ainsi que des expériences réalisées sur des jeux de données synthétiques. Enfin, la section 3.3.4 définit la mesure de *lift* que nous utilisons et présente différents paramètres additionnels pris en compte par notre implémentation de *WinMiner*. Les détails de deux applications de *WinMiner* seront quant à eux fournis au chapitre 4. Ces applications concernent des données sismiques utilisées dans le cadre du projet AEGIS ainsi que des données médicales proposées par le Discovery Challenge ECML/PKDD 2004. L'application aux données médicales permet de montrer que l'approche proposée peut être utilisée dans des domaines différents de la recherche de dépendances entre événements sismiques.

3.2 Règles d'épisodes et fenêtres optimales

3.2.1 Définitions préliminaires

Dans cette section, nous reprenons les notions standards de séquence d'événements, d'épisode, et d'occurrence minimale utilisées dans [MTV97] (cf. section 2.2). Nous en donnons des définitions équivalentes appropriées à notre présentation. La seule différence notable réside dans le fait que la notion d'occurrence que nous utilisons présente la nécessité, pour les dites occurrences, de satisfaire à la contrainte de gap maximum.

Définition 25 ***(événement)*** Soit E un ensemble de *types d'événement*. Un *événement* est défini par une paire (e, t) où $e \in E$ et $t \in \mathbb{N}$. La valeur t précise la date à laquelle *l'événement* apparaît.

Définition 26 ***(séquence ordonnée d'événements)*** Une *séquence ordonnée d'événements* est un tuple $s = \langle (e_1, t_1), (e_2, t_2), ..., (e_n, t_n) \rangle$ tel que $\forall i \in \{1, ..., n\}, e_i \in E \wedge\ t_i \in \mathbb{N}$, et $\forall i \in \{1, ..., n-1\}, t_i \leq t_{i+1}$.

Définition 27 ***(opérateur $\sqsubseteq$)*** Soit α et β deux séquences ordonnées d'événements. α est une sous-séquence de β, noté $\alpha \sqsubseteq \beta$, si et seulement si $\alpha = \beta$ ou si α peut être obtenue en enlevant certains éléments de β.

Définition 28 ***(séquence d'événements)*** Une *séquence d'événements* S est un triplet (s, T_s, T_e), où s est une séquence ordonnée d'événements de la forme $\langle (e_1, t_1), (e_2, t_2), ..., (e_n, t_n) \rangle$ et T_s, T_e sont des entiers naturels tels que $T_s \leq t_1 \leq t_n \leq T_e$.

T_s et T_e représentent respectivement la date de début et la date de fin de la séquence d'événements. On notera que t_1 peut être différent de T_s et que t_n peut être différent de T_e.

Définition 29 ***(épisode)*** Un *épisode* est un tuple α de la forme $\alpha = \langle e_1, e_2, \ldots, e_k \rangle$ avec $e_i \in E$ pour tout $i \in \{1, \ldots, k\}$. Dans ce travail, nous utiliserons la notation $e_1 \rightarrow e_2 \rightarrow ... \rightarrow e_k$ pour décrire l'épisode $\langle e_1, e_2, ..., e_k \rangle$ où '$\rightarrow$' peut être lu comme 'suivi de'. Nous notons l'épisode vide à l'aide du symbole $\emptyset$.

Définition 30 ***(taille, suffixe et préfixe d'un épisode)*** Soit $\alpha = \langle e_1, e_2, ..., e_k \rangle$ un épisode. La *taille* de α, notée $|\alpha|$, est égale au nombre d'éléments du tuple α, i.e., $|\alpha| = k$. Le $suffixe$ de α est défini comme un épisode composé du seul et dernier élément du tuple α, i.e., $suffix(\alpha) = \langle e_k \rangle$. Le *préfixe* de α est l'épisode $\langle e_1, e_2, ..., e_{k-1} \rangle$. Nous le notons $prefix(\alpha)$.

Définition 31 ***(occurrence)*** Soit $gapmax$ l'intervalle maximum autorisé entre 2 événements consécutifs. Un épisode $\alpha = \langle e_1, e_2, ..., e_k \rangle$ apparaît dans une séquence d'événements $S = (s, T_s, T_e)$ s'il existe au moins une *séquence ordonnée d'événements* $s' = \langle (e_1, t_1), (e_2, t_2), ..., (e_k, t_k) \rangle$ telle que $s' \sqsubseteq s$ et si $\forall i \in \{1, ..., k-1\}, 0 < t_{i+1} - t_i \leq gapmax$.

L'intervalle $[t_1, t_k]$ est appelé *occurrence* de α dans S. L'ensemble des occurrences de α dans S est noté $occ(\alpha, S)$.

Les épisodes définis ci-dessus (définition 29) correspondent aux épisodes *série* définis dans [MTV97], ce qui signifie que les différents types d'événements constituant les occurrences d'un épisode α doivent apparaître l'un après l'autre dans la séquence d'événements (dans le même ordre que celui de leurs apparitions dans le tuple α). On notera aussi que pour de tels épisodes série, une occurrence est composée d'événements apparaissant à des dates toutes différentes entre elles, mais que cette contrainte s'applique aux occurrences des motifs extraits et non au jeu de données lui-même (i.e. plusieurs événements peuvent apparaître à la même position dans la séquence d'événements).

Définition 32 ***(occurrence minimale)*** Soit $[t_s, t_e]$ une occurrence d'un épisode α dans la séquence d'événements S. S'il n'existe pas d'autre occurrence $[t'_s, t'_e]$ telle que $[t'_s, t'_e] \subset [t_s, t_e]$ (i.e. $(t_s < t'_s \wedge t'_e \leq t_e) \vee (t_s \leq t'_s \wedge t'_e < t_e)$), alors l'occurrence $[t_s, t_e]$ est appelée *occurrence minimale* de α. L'ensemble des occurrences minimales de α dans S est noté $mo(\alpha, S)$.

Intuitivement, une occurrence minimale est une occurrence d'un épisode qui ne contient pas d'autre occurrence de ce même épisode. Les occurrences minimales utilisées ici correspondent (hormis les aspects liés à la contrainte *gapmax*) aux occurrences minimales définies dans [MTV97]. Elles sont toutefois notées différemment. Une occurrence minimale $[t_s, t_e]$ selon la définition 32 est une occurrence minimale de [MTV97] notée $[t_s, t_e + 1)$, i.e. la borne supérieure de l'intervalle est prise sur l'unité temporelle suivante, mais la borne est exclue. Ce choix a été motivé par le fait qu'il semble faciliter, dans la pratique, la compréhension de la notion par les experts du domaine.

Définition 33 ***(largeur d'une occurrence)*** Soit $o = [t_s, t_e]$ une occurrence. La distance $t_e - t_s$ est appelée *largeur* de o . Nous la notons $width(o)$. L'ensemble de toutes les occurrences (resp. occurrences minimales) d'un épisode α dans une séquence d'événements S ayant une largeur égale à w est noté $occ(\alpha, S, w)$ (resp. $mo(\alpha, S, w)$).

Définition 34 ***(support d'un épisode)*** Le *support* d'un épisode α dans une séquence d'événements S, pour une largeur w, est défini comme suit :
$Support(\alpha, S, w) = \sum_{0 \leq i \leq w} \mid mo(\alpha, S, i) \mid$.
Nous définissons aussi le *support général* de α dans S par :
$GSupport(\alpha, S) = \mid mo(\alpha, S) \mid$

Les notions d'occurrence, d'occurrence minimale et de support d'un épisode tiennent compte de la satisfaction de la contrainte de gap maximum. Cependant, pour des raisons de simplicité, le paramètre *gapmax* n'apparaît pas explicitement dans les conventions de notation choisies.

Afin d'illustrer les définitions précédentes, considérons la séquence d'événements $S = (s, T_s, T_e)$ de la figure 3.1.

Dans cet exemple, $T_s = 9, T_e = 22$, $s = \langle (A, 10), (A, 11), (B, 12), (C, 13), (C, 14), (C, 15), (A, 16), (C, 17), (B, 18), (C, 22) \rangle$. Si l'on considère l'épisode $\alpha = A \rightarrow B$, et la contrainte $gapmax = 3$, alors $occ(\alpha, S) = \{[10, 12], [11, 12], [16, 18]\}$. On remarquera par exemple que $[10, 18]$ n'appartient pas à $occ(\alpha, S)$ puisque la contrainte *gapmax* n'est pas satisfaite. Les occurrences minimales de α sont $mo(\alpha, S) = \{[11, 12], [16, 18]\}$. L'occurrence $[10, 12]$ n'appartient pas à $mo(\alpha, S)$ car elle contient l'occurrence $[11, 12]$. De la même façon, si l'on considère l'épisode $\beta = A \rightarrow B \rightarrow C$, et la contrainte $gapmax = 3$, on a $occ(\beta, S) = \{[10, 13], [10, 14],$

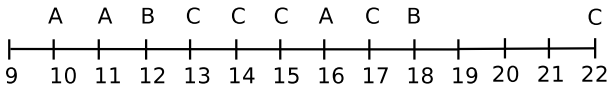


FIG. 3.1 – Exemple d'une séquence d'événements.

$[10, 15], [11, 13], [11, 14], [11, 15]\}$ et $mo(\beta, S) = \{[11, 13]\}$. À partir de ces informations, on peut, par exemple, établir que $GSupport(\alpha, S) = 2$, $Support(\alpha, S, 1) = 1$, $Support(\alpha, S, 2) = 2$ et $Support(\beta, S, 2) = 1$. On notera que la valeur du support est une fonction croissante de la valeur de la largeur des occurrences des épisodes considérés.

3.2.2 Les *FLM-règles*

Définition 35 ***(règle d'épisodes)*** Soient α et β des épisodes tels que $prefix(\beta) = \alpha$. Une telle paire (α, β) définit une *règle d'épisodes*, notée $\alpha \Rightarrow \beta$ dans la section 2.2, que nous noterons ici pour simplifier l'écriture $\alpha \Rightarrow suffix(\beta)$. Cette notation n'entraîne pas d'ambiguïté car nous imposons $prefix(\beta) = \alpha$.

Par exemple, si $\alpha = e_1 \rightarrow e_2$ et $\beta = e_1 \rightarrow e_2 \rightarrow e_3$, la règle d'épisodes correspondante est notée $e_1 \rightarrow e_2 \Rightarrow e_3$. Les règles d'épisodes considérées dans ce travail sont restreintes aux règles d'épisodes ayant un seul type d'événement dans leur partie droite pour réduire les coûts de calcul. Toutefois, les définitions et les algorithmes proposés peuvent être étendus dans le cas des règles dont la partie droite contient plusieurs types d'événements n'apparaissant pas de façon simultanée, e.g. $e_1 \rightarrow e_2 \Rightarrow e_3 \rightarrow e_4$. En effet, les algorithmes proposés ci-après permettent de calculer tous les épisodes fréquents, i.e. les épisodes dont le support général est supérieur à une valeur fixée par l'utilisateur. Or, c'est à partir de ces épisodes que sont construites les règles. On peut donc imaginer une solution à minima qui consisterait en un post-processing des épisodes fréquents. Ainsi, il serait possible, à partir des épisodes fréquents $e_1 \rightarrow e_2$ et $e_1 \rightarrow e_2 \rightarrow e_3 \rightarrow e_4$, de construire la règle $e_1 \rightarrow e_2 \Rightarrow e_3 \rightarrow e_4$. Étant donné le coût de calcul potentiel supplémentaire et les besoins applicatifs rencontrés, nous n'avons pas, pour le moment, développé cette extension.

En ce qui concerne la forme des règles, un autre cas est celui des règles basées sur des épisodes dont les événements peuvent être composés de plusieurs types d'événements apparaissant au même instant, e.g. la règle *si* e_1 *est suivi de* e_2 *alors* e_3 *et* e_4 *suivent* e_2 *et apparaissent en même temps*. Dans notre travail, nous utilisons directement la définition des épisodes séries proposée dans [MTV97] qui n'inclue pas cette possibilité. La prise en compte de telles règles constitue une piste

de réflexion qui est proposée au chapitre 5, et semble nécessiter une modification plus importante des algorithmes et des démonstrations des théorèmes.

Définition 36 ***(support et confiance d'une règle d'épisodes)*** Soient α et β des épisodes tels que $prefix(\beta) = \alpha$. Le *support* et la *confiance* de la règle d'épisodes $\alpha \Rightarrow suffix(\beta)$ sont définis pour une largeur w par :
$Support(\alpha \Rightarrow suffix(\beta), S, w) = Support(\beta, S, w)$ et
$Confiance(\alpha \Rightarrow suffix(\beta), S, w) = \frac{Support(\beta,S,w)}{Support(\alpha,S,w)}$

Soit γ un seuil minimum de confiance fixé par l'utilisateur tel que $0 \leq \gamma \leq 1$, et soit σ un seuil minimum de support fixé par l'utilisateur. Si $Confiance(r, S, w) \geq \gamma$ (resp. $Support(r, S, w) \geq \sigma$) alors la règle est dite $confiante$ (resp. *fréquente*) pour la largeur d'occurrence w.

La définition de la confiance peut être illustrée avec la séquence précédente (figure 3.1). Sachant que $mo(A \rightarrow B, S) = \{[11, 12], [16, 18]\}$ et $mo(A \rightarrow B \rightarrow C, S) = \{[11, 13]\}$, on a alors, avec par exemple $w = 2$:
$Confiance(A \rightarrow B \Rightarrow C, S, 2) = \frac{Support(A \rightarrow B \rightarrow C,S,2)}{Support(A \rightarrow B,S,2)} = 1/2$.

Une interprétation intuitive possible de cette règle est alors : « si un intervalle de taille 2 dans la séquence S débute sur une occurence minimale o_1 de $A \rightarrow B$ et que cet intervalle contient entièrement o_1, alors, dans 50% des cas, cet intervalle débute aussi sur une occurence minimale o_2 de $A \rightarrow B \rightarrow C$ et contient entièrement o_2 ».

Définition 37 ***(LM, FLM, FLM-règle et fenêtre optimale)*** Une règle r présente un *maximum local* de confiance LM pour une largeur donnée i dans une séquence d'événements S si et seulement si les trois propriétés suivantes sont vérifiées :

- $Confiance(r, S, i) \geq \gamma \wedge Support(r, S, i) \geq \sigma$
- $\forall j,\ j < i \wedge Support(r, S, j) \geq \sigma \ \Rightarrow\ Confiance(r, S, i) > Confiance(r, S, j)$
- $\exists j,\ j > i \wedge Confiance(r, S, j) \leq Confiance(r, S, i) - (decRate * Confiance(r, S, i))$ avec $decRate$ un seuil de décroissance entre 0 et 1 fixé par l'utilisateur, et $\forall\ k,\ i < k < j \ \Rightarrow\ Confiance(r, S, k) \leq Confiance(r, S, i)$

La règle r présente un *premier maximum local de confiance FLM (First Local Maximum)* pour une largeur donnée i si et seulement si r a un LM situé à la largeur i, et si r ne présente aucun LM pour des largeurs dont la valeur est strictement inférieure à i. Une règle ayant au moins un LM, et donc un FLM (unique), est appelée *FLM-règle*. La largeur i associée au FLM est appelée *fenêtre optimale*.

Intuitivement, le LM d'une règle r est associé à une largeur i telle que (1) r soit fréquente et confiante pour cette largeur d'épisodes, (2) les valeurs de confiances associées aux largeurs d'épisodes strictement inférieures à i, telles que r soit fréquente, sont strictement inférieures à la valeur de la confiance établie pour i, et (3) les largeurs immédiatement supérieures à i correspondent à des confiances

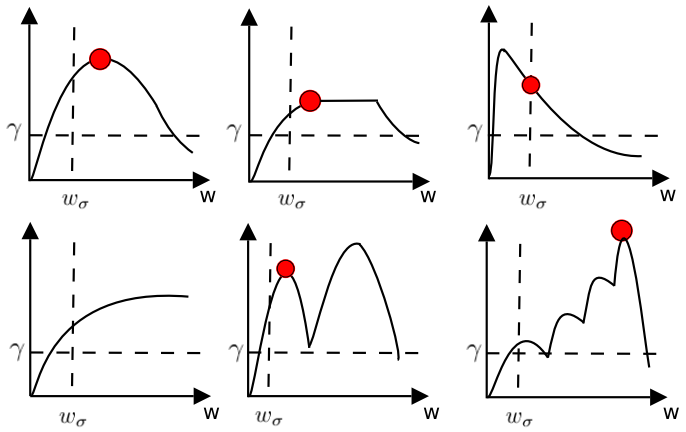


FIG. 3.2 – Confiance vs. largeur des épisodes : différentes situations possibles.

inférieures ou égales à celle obtenue pour i et ce jusqu'à ce que la confiance devienne inférieure à un certain pourcentage, fixé par *decRate*, de la confiance obtenue pour i. Cette dernière condition a pour objectif de permettre la sélection de maximums locaux *prononcés*, en imposant que ceux-ci soient suivis d'une décroissance significative contrôlée à l'aide du paramètre *decRate*. La figure 3.2 illustre, pour une règle donnée, diverses variations possibles de la confiance en fonction des valeurs de largeur. Les FLM correspondants, s'ils existent, sont représentés par un point. Les ordonnées représentent la confiance de la règle et les lignes horizontales en pointillés représentent la limite minimum de confiance γ. Les abscisses donnent les différentes valeurs de largeur et les lignes verticales en pointillés correspondent à une largeur spécifique notée w_σ, à partir de laquelle la règle est fréquente. Dans cette figure, deux situations particulières sont à remarquer. Premièrement, le graphique en bas à gauche représente une situation pour laquelle aucun FLM ne peut être identifié. Deuxièmement, le graphique en bas à droite détaille une configuration où les trois premiers maximums locaux de confiance ne sont pas retenues car ils ne sont pas suivis par une décroissance suffisante de la confiance.

3.2.3 Gestion de la contrainte de gap maximum

Dans [CG03] un algorithme a été proposé pour extraire les règles d'épisodes fréquentes et confiantes, sous une contrainte similaire à la contrainte de gap maximum. Cependant, l'algorithme proposé n'est pas complet. Afin de le montrer, à l'aide d'une contre-exemple, il nous faut rappeler la définition de *fenêtre* donnée dans [MTV97] : « *Une fenêtre sur une séquence d'événements* $S = (s, T_s, T_e)$ *est une séquence d'événements* $W = (w, t_s, t_e)$ *où* $t_s < T_e$ *et* $t_e > T_s$, *et* w *est*

constituée par les paires (A,t) de s telles que $t_s \leq t < t_e$ *».* On notera que cette définition autorise des fenêtres pouvant commencer avant T_s et finir après T_e. On notera aussi qu'une fenêtre (w, t_s, t_e) ne contient pas d'événement (A, t) avec $t = t_e$.

Soit $\mathcal{W}(S, win)$ l'ensemble de toutes les fenêtres (w, t_s, t_e) sur S telles que $t_e - t_s = win$. Alors [CG03] définit le support d'un épisode α sous la contrainte de gap maximum par $sp(\alpha) = \frac{|\{W \in \mathcal{W}(S,win) | \alpha \ apparait \ dans \ W\}|}{|\mathcal{W}(S,win)|}$, où $win = (|\alpha| - 1) \times gapmax$. On remarquera que cette définition fixe la durée maximale entre le 1^{er} et le dernier événement d'une occurrence en fonction de la taille du motif. Cette contrainte s'apparente à celle que nous avons définie, laquelle fixe la durée maximale entre 2 événements consécutifs d'une occurrence, et ce de façon identique à la contrainte de gap maximum utilisée dans les bases de séquences (cf. section 2.1.3).

L'algorithme d'extraction proposé dans [CG03] est basé sur une exploration par niveaux avec comptage sur la séquence, c'est à dire du type de celle utilisée par WINEPI [MTV95, MTV97] présentée dans la section 2.2.2. Il inclue certaines améliorations sans toutefois déroger à ce principe général. Une de ses spécificités est notamment de considérer un épisode candidat de la forme $A \rightarrow B \rightarrow C$ que si les épisodes $A \rightarrow B$ et $B \rightarrow C$ sont fréquents. Si une telle technique de génération a été utilisée avec succès dans des algorithmes tels que PSP [MCP98], dans le contexte de [CG03] elle est la cause d'une incomplétude lors de la détermination des candidats.

En effet, soit une séquence d'événements S de date de début 1 et de date de fin 10, définie par $(\langle (A,5),(B,8),(C,9)\rangle, 1, 10)$. Pour la contrainte $gapmax = 4$, $sp(A \rightarrow B) = 1/12$ et $sp(A \rightarrow B \rightarrow C) = 4/16 = 1/4$. En effet, pour ce qui est de l'épisode $A \rightarrow B$, la largeur de fenêtre utilisée est $(2-1) * 4 = 4$. Dans cette configuration, une seule fenêtre, $(< (A,5),(B,8) >, 5, 9)$ contient cet épisode alors que 12 fenêtres de taille 4 peuvent être construites sur S : $(<>, -2, 2)$, $(<>, -1, 3)$, $(<>, 0, 4)$, $(<>, 1, 5)$, $(< (A,5) >, 2, 6)$, $(< (A,5) >, 3, 7)$, $(< (A,5) >, 4, 8)$, $(< (A,5),(B,8) >, 5, 9)$, $(< (B,8),(C,9) >, 6, 10)$, $(< (B,8),(C,9) >, 7, 11)$, $(< (B,8),(C,9) >, 8, 12)$, et $(< (C,9) >, 9, 13)$. Quant à l'épisode $A \rightarrow B \rightarrow C$, la fenêtre utilisée est $(3-1) * 4 = 8$. 4 fenêtres de largeur 8 contiennent $A \rightarrow B \rightarrow C$: $(< (A,5),(B,8),(C,9) >, 2, 10)$, $(< (A,5),(B,8),(C,9) >, 3, 11)$, $(< (A,5),(B,8),(C,9) >, 4, 12)$, et $(< (A,5),(B,8),(C,9) >, 5, 13)$. Or, 16 fenêtres de taille 4 peuvent être construites sur S : $(<>, -6, 2)$, $(<>, -5, 3)$, $(<>, -4, 4)$, $(<>, -3, 5)$, $(< (A,5) >, -2, 6)$, $(< (A,5) >, -1, 7)$, $(< (A,5) >, 0, 8)$, $(< (A,5),(B,8) >, 1, 9)$, $(< (A,5),(B,8),(C,9) >, 2, 10)$, $(< (A,5),(B,8),(C,9) >, 3, 11)$, $(< (A,5),(B,8),(C,9) >, 4, 12)$, $(< (A,5),(B,8),(C,9) >, 5, 13)$, $(< (B,8),(C,9) >, 6, 14)$, $(< (B,8),(C,9) >, 7, 15)$, $(< (B,8),(C,9) >, 8, 16)$, et $(< (C,9) >, 9, 17)$.

Si la limite de fréquence est fixée à 1/4, et si l'on considère les valeurs de support établies précédemment, alors $A \rightarrow B$ n'est pas fréquent. $A \rightarrow B \rightarrow C$ n'est donc pas considéré par l'algorithme d'extraction proposé dans [CG03] alors

même qu'il est fréquent, entraînant l'incomplétude de l'algorithme. On notera que la justesse et la complétude de l'algorithme ne sont pas étudiées dans [CG03].

On notera aussi que les choix faits dans [CG03] aboutissent à des cas très particuliers. Par exemple, la confiance de la règle $A \rightarrow B \Rightarrow C$ est, dans notre exemple, $\frac{sp(A \rightarrow B \Rightarrow C)}{sp(A \rightarrow B)} = \frac{1/4}{1/12} = 3$, c'est à dire supérieure à 1.

Pour toutes ces raisons, nous n'avons pas réutilisé d'éléments de l'approche décrite dans [CG03] pour la recherche des épisodes fréquents sous la contrainte de gap maximum. Sur le plan algorithmique, on remarquera principalement que notre proposition, décrite ci-après, utilise une exploration en profondeur avec comptage par listes d'occurrences alors que celle avancée dans [CG03] est basée sur une exploration par niveaux avec comptage sur la séquence.

Comme nous le verrons par la suite, notre approche permet d'extraire les épisodes fréquents sous la contrainte de gap maxium, mais aussi de calculer efficacement les FLM-règles, point sur lequel nous ne voyons pas de solution satisfaisante dans le cadre d'une exploration par niveau avec comptage sur la séquence, telle qu'utilisée dans [MTV95, MTV97, CG03].

3.3 Extraction des FLM-règles

Afin d'extraire les FLM-règles, nous proposons un algorithme appelé *WinMiner*. *WinMiner* est basé sur l'approche par *listes d'occurrences*. Cette approche est utilisée aussi bien pour rechercher des règles d'épisodes dans une longue séquence d'événements (e.g., [MTV95, MT96, MTV97]) que pour rechercher des motifs séquentiels dans une base de séquences (e.g., [Zak00, LRBE03a]). Le principe sur lequel repose cette approche réside dans le stockage et l'utilisation des occurrences des motifs pour calculer les occurrences des motifs plus longs. Toutefois, dans notre cas, si nous gardons seulement l'information concernant la localisation des occurrences minimales des épisodes de taille k, il nous est alors impossible de générer toutes les occurrences minimales des épisodes de taille $k+1$.

En effet, si l'on reprend l'exemple donné par la figure 3.1, et en considérant une contrainte de gapmax $gapmax = 3$, on peut observer qu'il existe une occurrence minimale $[11, 18]$ de $A \rightarrow B \rightarrow C \rightarrow B$. Si l'on ne connaît que $mo(A \rightarrow B \rightarrow C, S) = \{[11, 13]\}$ et $mo(B, S) = \{[12, 12], [18, 18]\}$, alors on ne peut pas générer l'occurrence minimale $A \rightarrow B \rightarrow C \rightarrow B$. En effet, par rapport à la date de fin de l'occurrence $A \rightarrow B \rightarrow C$ (date 13) le premier B (date 12) apparaît trop tôt, tandis que le second B (date 18) ne satisfait pas la contrainte de gap maximum fixée à 3 ($18 - 13 > 3$). Pour pallier à cette difficulté, *WinMiner* est basé sur la

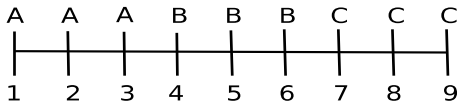


FIG. 3.3 – Séquence d'événements S_{mpo}.

notion d'*occurrence minimale préfixée*, notion introduite dans la section suivante, et dont l'objectif est de fournir l'information nécessaire permettant de garantir la complétude des algorithmes présentés dans la section 3.3.2.

3.3.1 Occurrence minimale préfixée

Définition 38 ***(Occurrence minimale préfixée)*** Soit $o = [t_s, t_e]$ une occurrence d'un épisode α dans une séquence d'événements S. o est une *occurrence minimale préfixée* (*mpo*) de α si et seulement si : $\forall\ [t_1, t_2] \in mo(prefix(\alpha), S)$, si $t_s < t_1$ alors $t_e \leq t_2$. Nous notons $mpo(\alpha, S)$ l'ensemble de tous les *mpo* de α dans S.

Il est important de noter que $mpo(\alpha, S)$ est défini par rapport à $mo(prefix(\alpha), S)$ et non par rapport à $mo(\alpha, S)$. Afin d'illustrer la définition 38 ainsi que cette dernière remarque, considérons la séquence S_{mpo} donnée figure 3.3.

L'ensemble des *mpo* de $A \rightarrow B \rightarrow C$ dans S_{mpo}, sous une contrainte de $gapmax = 8$ (i.e. cette contrainte n'élimine aucune combinaison possible entre les différents événements composant la séquence), est : $[3, 7], [3, 8], [3, 9]$. En revanche tous les autres intervalles possibles ne sont pas des *mpo*. En particulier, $[1, 7]$ et $[2, 7]$ ne sont pas des *mpo* de $A \rightarrow B \rightarrow C$ car il existe une occurrence minimale de $A \rightarrow B$, $[3, 4]$, qui les invalide. En effet, cette occurrence commence après $[1, 7]$ et $[2, 7]$, et finit avant ces mêmes intervalles de temps, ce qui est interdit par la définition 38.

Si l'on reprend l'exemple représenté par la figure 3.1 et si l'on considère une contrainte $gapmax = 3$, alors nous avons $mpo(A \rightarrow B \rightarrow C, S) = \{[11, 13], [11, 14], [11, 15]\}$ et $mpo(B, S) = \{[12, 12], [18, 18]\}$. En utilisant ces ensembles, il devient alors possible, l'information étant disponible, de construire $mpo(A \rightarrow B \rightarrow C \rightarrow B, S) = \{[11, 18]\}$. Les *mpo* nous apportent donc l'information nécessaire à l'obtention d'une occurrence qu'il nous était impossible de calculer à partir des listes des seules *mo*.

Comme cela peut se remarquer de façon intuitive, les occurrences minimales sont des *mpo* particulières et elles peuvent être déterminées à l'aide de l'ensemble des *mpo*. C'est ce qu'expriment les lemmes suivants de façon plus formelle :

Lemme 1 *Si* $[t_s, t_e] \in mo(\alpha, S)$ *alors* $[t_s, t_e] \in mpo(\alpha, S)$ *et il n'existe pas de* $[t_s, t'_e] \in mpo(\alpha, S)$ *telle que* $t'_e < t_e$.

Lemme 2 *Si* $[t_s, t_e] \in mpo(\alpha, S)$ *et s'il n'existe pas* $[t_s, t'_e] \in mpo(\alpha, S)$ *telle que* $t'_e < t_e$, *alors* $[t_s, t_e] \in mo(\alpha, S)$.

Afin de prouver le lemme 1, nous introduisons également le lemme suivant :

Lemme 3 *S'il existe* $[t_s, t_e] \in occ(\alpha, S)$ *et* $[t'_s, t'_e] \in occ(prefix(\alpha), S)$ *telle que* $t_s < t'_s \wedge t'_e < t_e$, *alors* $[t'_s, t_e] \in occ(\alpha, S)$.

Le lemme 3 est démontré par la preuve suivante :

Preuve 1 *Soit* α *l'épisode* $\langle e_1, \ldots, e_n \rangle$ *et* $S = (s, T_s, T_e)$ *une séquence d'événements. Puisque* $[t_s, t_e] \in occ(\alpha, S)$ *et* $[t'_s, t'_e] \in occ(prefix(\alpha), S)$ *alors, par définition d'une occurrence, il existe* $s_1 = \langle (e_1, t_1), \ldots, (e_n, t_n) \rangle$ *avec* $t_1 = t_s$ *et* $t_n = t_e$ *telle que* $s_1 \sqsubseteq s$ *et telle que* $\forall i \in \{1, ..., n-1\}, t_{i+1} - t_i \leq gapmax$, *et il existe aussi* $s_2 = \langle (e_1, t'_1), \ldots, (e_{n-1}, t'_{n-1}) \rangle$ *avec* $t'_1 = t'_s$ *et* $t'_{n-1} = t'_e$ *telle que* $s_2 \sqsubseteq s$ *et telle que* $\forall i \in \{1, ..., n-2\}, t'_{i+1} - t'_i \leq gapmax$.

Soit s_3 *la séquence d'événements* $\langle (e_1, t''_1), \ldots, (e_i, t''_i), \ldots, (e_n, t''_n) \rangle$ *telle que* $\forall i \in \{1, ..., n-1\}, t''_i = max(\{t_i, t'_i\})$ *et telle que* $t''_n = t_n$. *Puisque* s_1 *et* s_2 *sont des séquences ordonnées d'événements et que* $t'_e < t_e$, *alors* s_3 *est aussi une séquence ordonnée d'événements.*

Soit $i \in \{1, ..., n-2\}$. *Sachant que* $t_{i+1} - t_i \leq gapmax$, *que* $t'_{i+1} - t'_i \leq gapmax$ *et que* $t''_i = max(\{t_i, t'_i\})$, *alors* $t_{i+1} - t''_i \leq gapmax$, *et* $t'_{i+1} - t''_i \leq gapmax$. *Par conséquent* $max(\{t_i, t'_i\}) - t''_i \leq gapmax$ *et donc,* $t''_{i+1} - t''_i \leq gapmax$.

Puisque $t_n - t_{n-1} \leq gapmax$, *que* $t''_n = t_n$ *et que* $t''_{n-1} = max(\{t_{n-1}, t'_{n-1}\})$ *alors* $t''_n - t''_{n-1} \leq gapmax$ *(i.e., la propriété est aussi valable pour* $i = n-1$*).*

Ainsi $\forall i \in \{1, ..., n-1\}, t''_{i+1} - t''_i \leq gapmax$. *Comme* $s_1 \sqsubseteq s$ *et* $s_2 \sqsubseteq s$, *on sait alors que* $s_3 \sqsubseteq s$. *Par conséquent,* $[t''_1, t_e] \in occ(\alpha, S)$.

Finalement, puisque $t''_1 = max(\{t_1, t'_1\})$ *avec* $t_1 = t_s$, $t'_1 = t'_s$ *et que* $t_s < t'_s$, *alors* $t''_1 = t'_s$, *et donc* $[t'_s, t_e] \in occ(\alpha, S)$.

$\diamond$

Le lemme 1 peut alors être démontré de la façon suivante :

Preuve 2 *Soit* $[t_s, t_e]$ *une occurrence minimale de* α *dans* S. *Supposons qu'il existe* $[t_1, t_2] \in mo(prefix(\alpha), S)$ *telle que* $t_s < t_1 \wedge t_2 < t_e$. *Selon le lemme 3* $[t_1, t_e] \in occ(\alpha, S)$. *Puisque* $t_s < t_1$ *alors* $[t_s, t_e] \notin mo(\alpha, S)$. *Ceci est contradictoire, et donc* $\nexists [t_1, t_2] \in mo(prefix(\alpha), S)$ *telle que* $t_s < t_1 \wedge t_e > t_2$. *Ainsi par définition des mpo,* $[t_s, t_e] \in mpo(\alpha, S)$.

De plus, puisque $[t_s, t_e] \in mo(\alpha, S)$, *alors* $\nexists [t_s, t'_e] \in occ(\alpha, S)$ *telle que* $t'_e < t_e$. *Par conséquent,* $\nexists [t_s, t'_e] \in mpo(\alpha, S)$ *telle que* $t'_e < t_e$.

$\diamond$

Enfin, le lemme 2 peut être prouvé ainsi :

Preuve 3 *Soit* $[t_s, t_e] \in mpo(\alpha, S)$ *telle qu'il n'existe pas de* $[t_s, t'_e] \in mpo(\alpha, S)$ *telle que* $t'_e < t_e$.

Supposons qu'il existe $[t_1, t_2] \in mo(\alpha, S)$ *telle que* $t_s = t_1$. *Selon le lemme 1,* $[t_1, t_2] \in mpo(\alpha, S)$. *Or, par hypothèse,* $\nexists [t_s, t'_e] \in mpo(\alpha, S)$ *telle que* $t'_e < t_e$. *Donc,* $t_e \leq t_2$.

Supposons maintenant qu'il existe $[t_1, t_2] \in mo(\alpha, S)$ *telle que* $t_s < t_1$. *Alors* $\exists [t'_1, t'_2] \in mo(prefix(\alpha), S)$ *telle que* $t_1 \leq t'_1 \wedge t'_2 < t_2$. *Puisque* $[t_s, t_e] \in mpo(\alpha, S)$ *et* $t_s < t_1 \leq t'_1$, *par définition d'une mpo, on en déduit que* $t_e \leq t'_2 < t_2$.

Ainsi, s'il existe $[t_1, t_2] \in mo(\alpha, S)$ *telle que* $t_s = t_1$ *alors* $t_e \leq t_2$, *et s'il existe* $[t_1, t_2] \in mo(\alpha, S)$ *telle que* $t_s < t_1$ *alors* $t_e < t_2$. *Par conséquent,* $\nexists [t_1, t_2] \in mo(\alpha, S)$ *telle que* $(t_s < t_1 \wedge t_2 \leq t_e) \vee (t_s \leq t_1 \wedge t_2 < t_e)$. *Ainsi, par définition des occurrences minimales,* $[t_s, t_e] \in mo(\alpha, S)$.

$\diamond$

3.3.2 Algorithme *WinMiner*

WinMiner extrait toutes les $FLM - rules$ dans une séquence d'événements S en fonction des paramètres utilisateurs suivants : une seuil de support minimal σ, une seuil de confiance minimale γ, un gap maximum $gapmax$ et un taux de décroissance $decRate$.

Pour cela, *WinMiner* utilise une structure de données particulière, les E/O-pairs. Une E/O-pair permet à l'algorithme *WinMiner* de gérer un épisode α et ses *mpo* à l'aide d'une paire de la forme *(épisode,occurrences)* appelée E/O-pair. Pour une E/O-pair donnée x, nous notons respectivement $x.Pattern$ et $x.Occ$, le premier et le second élément de la paire. La partie $x.Pattern$ représente l'épisode lui-même et $x.Occ$ contient ses *mpo* selon un mode de stockage compact. La partie $x.Occ$ est un ensemble de paires de la forme $(Tbeg, TendSet)$ où $Tbeg$ représente la date de départ d'une *mpo* et où $TendSet$ est l'ensemble des dates de fin des *mpo* de $x.Pattern$ dont la date de départ est $Tbeg$. Intuitivement, l'intérêt de cette représentation réside dans le fait que, selon le lemme 2, si l'on considère une paire $(Tbeg, TendSet)$, alors l'intervalle $[Tbeg, min(TendSet)]$ représente une occurrence minimale de $x.Pattern$.

Le principe de *WinMiner* est présenté à l'aide de l'algorithme 8. Tout d'abord, *WinMiner* calcule les *mpo* de tous les épisodes fréquents de taille 1 en utilisant une fonction nommée *scan*. Cette fonction parcourt S, en recherchant, à la volée, chacune des *mpo* des épisodes de taille 1. Par la suite, l'algorithme appelle la fonction *exploreLevelN* (algorithme 9) afin de déterminer, à l'aide d'une exploration en profondeur, tous les épisodes de taille strictement supérieure à 1

tels que leur *GSupport* soit supérieur ou égal au seuil de support minimal σ. Pour un épisode $x.Pattern$ donné, cette fonction étend le motif sur sa droite en ajoutant un épisode fréquent $y.Pattern$ de taille 1. Cela est réalisé grâce à la fonction *join* (algorithme 10). Cette fonction génère à la fois le nouveau motif $z.Pattern = x.Pattern \rightarrow y.Pattern$, et l'ensemble des *mpo* de $z.Occ$ correspondant en utilisant les *mpo* de $x.Pattern$ et de $y.Pattern$ qui sont respectivement stockées dans $x.Occ$ et $y.Occ$. Le principe de base de cette opération de jointure est exprimée par l'algorithme 10. Une fois l'opération de jointure effectuée, si le *GSupport* de $z.Pattern$ est supérieur ou égal à σ, alors l'algorithme 9 détermine, s'il existe, le FLM de la règle construite à partir du préfixe et du suffixe de $z.Pattern$, et ce en utilisant la fonction $findFLM$ (algorithme 11). Enfin, l'algorithme 9 explore, de façon récursive, les épisodes qui peuvent être générés en ajoutant les épisodes fréquents de taille 1 à $z.Pattern$ lui-même.

Algorithme 8 (*WinMiner*) Entrée : *S une séquence d'événements et E l'ensemble de tous les types d'événement pouvant apparaître dans S.*

```
1.  let L_1 := ∅
2.  for all e ∈ E do
3.      let x.Pattern := e
4.      let x.Occ := scan(S, e)
5.      if | x.Occ | >= σ
6.        let L_1 := L_1 ∪ {x}
7.      fi
8.  od
9.  for all x ∈ L_1 do
10.     exploreLevelN(x, L_1)
11. fi
```

Algorithme 9 (*exploreLevelN*) Entrée : *x une E/O-pair, et L_1 l'ensemble des E/O-pairs des épisodes fréquents de taille 1.*

```
1.  for all y ∈ L_1 do
2.    let z := join(x, y)
3.    if |z.Occ| >= σ
4.      findFLM(x.Pattern ⇒ suffix(z.Pattern), x.Occ, z.Occ)
5.      exploreLevelN(z, L_1)
7.    fi
8.  od
```

Algorithme 10 (*join*) Entrée : *x et y, deux E/O-pairs, contenant respectivement les épisodes $x.Pattern$ et $y.Pattern$ ainsi que l'ensemble de leurs mpo, $x.Occ$*

et y.Occ. De plus, y.Pattern correspond à un épisode de taille 1.
Sortie : *z, une E/O-pair contenant l'épisode* $x.Pattern \rightarrow y.Pattern$ *et l'ensemble de ses mpo.*

1. **let** $z.Pattern := x.Pattern \rightarrow y.Pattern$
2. **let** $z.Occ := \emptyset$
3. **for all** $(t_s, T) \in x.Occ$ **do**
4. **let** $L := \emptyset$
5. **for all** $t \in T$ **do**
6. **let** $EndingTimes := \{t'_s \mid \exists (t'_s, T') \in y.Occ$ *such that* $t'_s > t_s \wedge t'_s - t \leq gapmax \wedge \forall (t_1, T'') \in x.Occ,$ $t_s < t_1 \Rightarrow \forall t_2 \in T'', t'_s \leq t_2\}$
7. **let** $L := L \cup EndingTimes$
8. **od**
9. **if** $L \neq \emptyset$
10. **let** $z.Occ := z.Occ \cup \{(t_s, L)\}$
11. **fi**
12. **od**

Afin d'illustrer le principe général de génération des épisodes candidats et de détermination de leurs E/O-pairs respectives, et sur la base de l'exemple donné par la figure 3.1, considérons que *WinMiner* est exécuté avec pour paramètres $gapmax$=3 et un support minimum fixé à 1. Les paramètres de confiance et de décroissance ne sont pas spécifiés dans cet exemple car ils ne sont utilisés que par l'algorithme $findFLM$, algorithme présenté plus loin dans cette section et qui opère une sélection sur les motifs en ne retenant que les règles d'épisodes ayant un FLM.

Les épisodes A, B, et C sont tous considérés comme fréquents par l'algorithme *WinMiner* (i.e., $L_1 = \{A, B, C\}$) . À partir de là, *WinMiner* tente, pour chacun de ces épisodes, de construire les épisodes fréquents de taille supérieure en faisant appel à la fonction récursive $exploreLevelN$. Pour ce faire, $exploreLevelN$ utilise la fonction *join* pour calculer les E/O-pairs des super-épisodes à partir desquelles leurs supports sont établis. Seuls les épisodes ayant un support suffisant sont utilisés pour poursuivre l'exploration à l'aide d'un nouvel appel à la fonction $exploreLevelN$.

Afin de détailler (cf. figure 3.4) le principe de la fonction de jointure (algorithme 10), considérons l'appel qui est fait à *join* avec x, une E/O-pair contenant les *mpo* de $A \rightarrow B$ et y, une E/O-pair contenant les *mpo* de C. On a $x = (A \rightarrow B, \{(11, \{12\}), (16, \{18\})\})$ et $y = (C, \{(13, \{13\}), (14, \{14\}), (15, \{15\}), (17, \{17\}), (22, \{22\})\})$. La ligne 1 permet de générer le nouveau motif, i.e. $A \rightarrow B \rightarrow C$. Le reste de l'algorithme correspond au calcul des *mpo* de $A \rightarrow B \rightarrow C$. Pour ce faire,

join sélectionne dans x (ligne 3), la *mpo* $[11, 12]$, et parcourt y (ligne 5) à la recherche d'occurrences de C pouvant étendre le motif $A \rightarrow B$ au motif $A \rightarrow B \rightarrow C$ (ligne 6). Seules les occurrences de C aux dates 13, 14 et 15 sont retenues car les occurrences aux dates 17 et 22 sont trop éloignées pour pouvoir satisfaire à la contrainte de *gapmax*. Par la suite, l'algorithme tente de joindre l'autre *mpo* de $x.Pattern$, i.e. $[16, 18]$, et ce avec une *mpo* de C contenue dans y. Cela ne donne rien puisque toutes les dates d'occurrences de C sont inférieures à la date 18, exceptée la date 22, laquelle ne peut être retenue en raison de la contrainte *gapmax*. Ainsi, $z = (A \rightarrow B \rightarrow C, \{(11, \{13, 14, 15\})\})$ et comme le support minimum est de 1, l'exploration se poursuit avec $x = (A \rightarrow B \rightarrow C, \{(11, \{13, 14, 15\})\})$ (cf. figure 3.4).

Admettons maintenant que *WinMiner* tente d'étendre $A \rightarrow B \rightarrow C$ au motif $A \rightarrow B \rightarrow C \rightarrow B$. Pour cela, on utilise les *mpo* de B, c'est-à-dire $y = (B, \{(12, \{12\}), (18, \{18\})\})$. La fonction *join* considère tout d'abord la mpo $[11, 13]$ et tente de trouver une occurrence de B possible permettant d'étendre le motif. L'apparition à la date 12 ne peut être sélectionnée car antérieure à la date de fin de $[11, 13]$, et la date d'apparition $[18, 18]$ n'est pas retenue à cause de la contrainte *gapmax*. La fonction *join* considère alors la mpo $[11, 14]$ et tente de nouveau de trouver une occurrence de B possible permettant d'étendre le motif. L'apparition à la date 12 ne peut être envisagée car antérieure à la date de fin de $[11, 14]$, et la date d'apparition $[18, 18]$ n'est pas retenue à cause de la contrainte *gapmax*. Enfin *join* utilise la mpo $[11, 15]$. L'apparition à la date 12 ne peut être utilisée car antérieure à la date de fin de $[11, 15]$. Par contre la date d'apparition 18 est postérieure et la contrainte de gapmax est respectée. Ainsi, il existe une *mpo* de $A \rightarrow B \rightarrow C \rightarrow B$ (i.e. $[11, 18]$), et comme le support minimal est de 1 l'exploration est relancée à partir de ce motif (cf. figure 3.4).

Si l'on souhaite à nouveau étendre ce motif au motif $A \rightarrow B \rightarrow C \rightarrow B \rightarrow B$, alors *join*, avec $x = (A \rightarrow B \rightarrow C \rightarrow B, \{(11, \{18\})\})$ et de $y = (B, \{(12, \{12\}), (18, \{18\})\})$, tentera d'en trouver les *mpo*. Or, les dates d'apparition de B étant antérieures ou égales à la fin de l'unique *mpo* de $A \rightarrow B \rightarrow C \rightarrow B$, aucune *mpo* ne pourra être formée et aucun sur-motif ne sera envisagé (cf. figure 3.4). On notera que si le support minimal avait été fixé à 2, le motif $A \rightarrow B \rightarrow C$ n'aurait pas été considéré comme fréquent (une seule *mpo*, $[11, 13]$), et les motifs $A \rightarrow B \rightarrow C \rightarrow B$ et $A \rightarrow B \rightarrow C \rightarrow B \rightarrow B$ n'auraient pas été considérés par la fonction *exploreLevelN*.

Intéressons-nous maintenant à la correction de l'algorithme *join*.

Définition 39 Soit S une séquence d'événements, alors :
- une E/O-pair x est *juste* ssi $\forall (t_s, T) \in x.Occ, \forall t \in T$, $[t_s, t] \in mpo(x.Pattern, S)$.
- une E/O-pair x est *complète* ssi $\forall [t_s, t_e] \in mpo(x.Pattern, S)$, $\exists (t_s, T) \in x.Occ$ telle que $t_e \in T$.

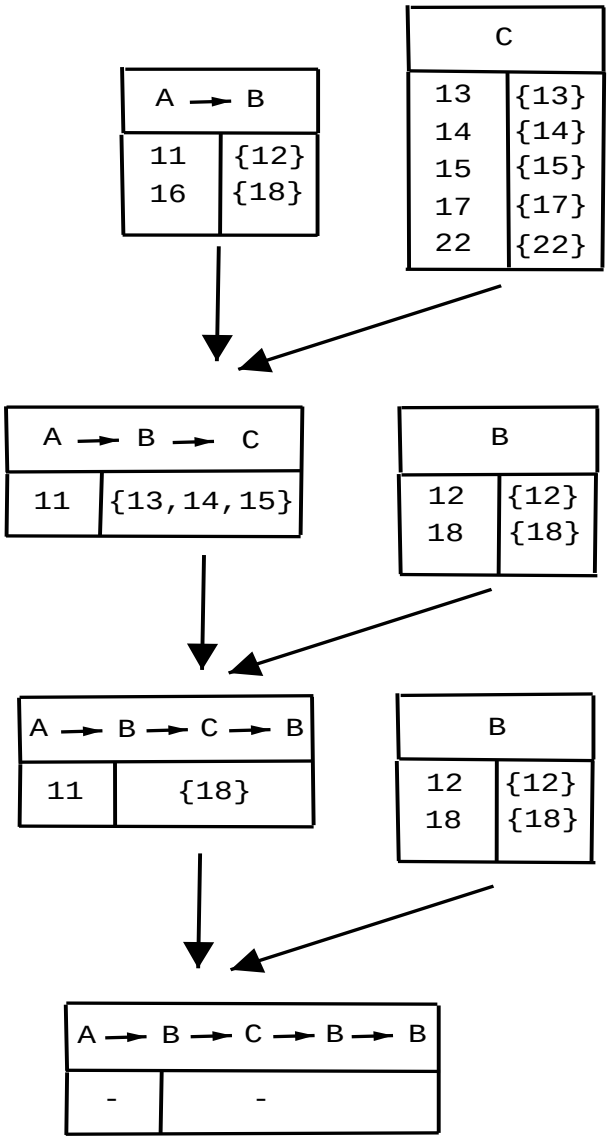


FIG. 3.4 – Exemples de jointures dans *WinMiner*.

– une E/O-pair x est *non-redondante* ssi $\forall (t_s, T) \in x.Occ, \nexists (t_s, T') \in x.Occ$ telle que $T \neq T'$.

Le théorème suivant établit la correction de la fonction *join* (algorithme 10) :

Théorème 1 (correction de *join*) *Si les entrées de join, x et y, sont justes, complètes et non-redondantes, alors la sortie z est juste, complète, et non-redondante.*

Afin de prouver le théorème 1, nous avons besoin du lemme de décomposition suivant :

Lemme 4 $\forall [t_s, t_e] \in mpo(\alpha, S)$, $\exists [t_s, t'_e] \in mpo(prefix(\alpha), S)$ *telle que* $t'_e < t_e \wedge t_e - t'_e \leq gapmax$.

Preuve 4 *Considérons* $[t_s, t_e] \in mpo(\alpha, S)$. *Par définition d'une occurrence,* $\exists [t_s, t'_e] \in occ(prefix(\alpha), S)$ *telle que* $t'_e < t_e$ *et que* $t_e - t'_e \leq gapmax$.

Supposons que $[t_s, t'_e] \notin mpo(prefix(\alpha), S)$.

Cela implique que $\exists [t_1, t_2] \in mo(prefix(prefix(\alpha)), S)$ *telle que* $t_s < t_1 \wedge t_2 < t'_e$. *Or, selon le lemme 3,* $[t_1, t'_e] \in occ(prefix(\alpha), S)$. *Ainsi,* $\exists [t'_1, t'_2] \in mo(prefix(\alpha), S)$ *telle que* $t_1 \leq t'_1 \wedge t'_2 \leq t'_e$. *Puisque* $t_s < t_1 \wedge t'_e < t_e$, *alors* $[t'_1, t'_2] \in mo(prefix(\alpha), S)$ *avec* $t_s < t'_1 \wedge t'_2 < t_e$. *Cela est contradictoire avec l'hypothèse selon laquelle* $[t_s, t_e] \in mpo(\alpha, S)$.

Par conséquent, $[t_s, t'_e] \in mpo(prefix(\alpha), S)$.

$\diamond$

Le théorème 1 peut alors être démontré comme suit :

Preuve 5 *Soit une E/O-pair z, le résultat d'un appel à la fonction join, et soient x et y, les E/O-pairs justes, complètes et non-redondantes à partir desquelles est calculée z.*

Nous montrons tout d'abord que z est complète.

Considérons $[t_s, t_e] \in mpo(z.Pattern, S)$.

Selon le lemme 4, $\exists [t_s, t'_e] \in mpo(prefix(z.Pattern), S)$ *telle que* $t'_e < t_e \wedge t_e - t'_e \leq gapmax$. *Puisque suffix(z.Pattern) est composé d'un seul type d'événement, on sait également que* $[t_e, t_e] \in mpo(suffix(z.Pattern), S)$.

L'épisode $z.Pattern$ *généré par join est tel que* $prefix(z.Pattern) = x.Pattern$ *et que* $suffix(z.Pattern) = y.Pattern$.

Puisque x and y sont complètes, alors :

– $\exists (t_s, T) \in x.Occ$ *telle que* $t'_e \in T$.

– $\exists (t_e, T') \in z.Occ$ *telle que* $T' = \{t_e\}$.

Ainsi, l'algorithme considère la paire $(t_s, T) \in x.Occ$ *(ligne 3) et* $t \in T$ *(ligne 5) tel que* $t = t'_e$*. De plus, il considère également la paire* $(t'_s, T') \in z.Occ$ *telle que* $t'_s = t_e$ *(ligne 6). Puisque* $t_e - t'_e \leq gapmax$ *alors* $t'_s - t \leq gapmax$*. Et comme* $t_e > t_s$ *(*$z.Pattern$ *est un épisode de taille strictement supérieure à 1), on a* $t'_s > t_s$*. Ainsi,* t'_s *satisfait aux conditions imposées par les deux premiers éléments de la conjonction de propriétés nécessaires pour appartenir à* $EndingTimes$*. Considérons maintenant le dernier élément de cette conjunction. Supposons que* $\exists (t_1, T'') \in x.Occ$ *telle que* $t_s < t_1$ *et que* $\exists t_2 \in T''$ *avec* $t'_s > t_2$*. Alors, puisque* x *est juste,* $[t_1, t_2] \in occ(x.Pattern, S)$*. Donc* $\exists [a, b] \in mo(x.Pattern, S)$ *telle que* $t_1 \leq a$ *et* $b \leq t_2$*. Comme* $t_s < t_1$ *and* $t_2 < t'_s = t_e$*, on a également* $t_s < a$ *et* $b < t_e$*. Or,* $x.Pattern = prefix(z.Pattern)$*, ce qui contredit le fait selon lequel* $[t_s, t_e] \in mpo(z.Pattern, S)$*. Par conséquent,* $\forall (t_1, T'') \in x.Occ$*,* $t_s < t_1 \Rightarrow \forall t_2 \in T'', t'_s \leq t_2$*. Ainsi,* $t'_s = t_e$ *est sélectionné afin de participer à la construction de la paire* (t_s, T''') *présente dans* $z.Occ$*, et telle que* $t'_s \in T'''$*.* z *est donc complète.*

Nous allons maintenant montrer que z *est juste. Considérons* $(t_s, L) \in z.Occ$ *et* $t_e \in L$*. Il nous faut montrer que* $[t_s, t_e] \in mpo(z.Pattern, S)$*. L'intervalle* $[t_s, t_e]$ *a été généré en utilisant* $(t_s, T) \in x.Occ$*,* $t \in T$ *et* $(t_e, T') \in y.Occ$ *tels que* $t_e > t_s$*,* $t_e - t \leq gapmax$ *et* $\forall (t_1, T'') \in x.Occ$*,* $t_s < t_1 \Rightarrow \forall t_2 \in T'', t_e \leq t_2$*.*

Puisque $prefix(z.Pattern) = x.Pattern$*, que* $suffix(z.Pattern) = y.Pattern$ *et que* $t_e > t_s$*,* $t_e - t \leq gapmax$ *alors, comme* x *et* y *sont justes, nous avons* $[t_s, t_e] \in occ(z.Pattern)$*.*

D'autre part, nous savons que $\forall (t_1, T'') \in x.Occ$*,* $t_s < t_1 \Rightarrow \forall t_2 \in T'', t_e \leq t_2$*. Comme* x *est complète, alors* $\forall [t_1, t_2] \in mpo(x.Occ, S)$*,* $t_s < t_1 \Rightarrow t_e \leq t_2$*. De plus,* $prefix(z.Pattern) = x.Pattern$ *et selon le lemme 1 une occurrence minimale est aussi une mpo, ce qui implique que* $\forall [t_1, t_2] \in mo(prefix(z.Occ), S)$*,* $t_s < t_1 \Rightarrow t_e \leq t_2$*. Par conséquent* $[t_s, t_e] \in mpo(z.Pattern, S)$*.* z *est donc juste.*

Finalement, la propriété selon laquelle z *est non-redondante provient, de façon triviale, du fait que* x *est non-redondante et qu'un élément* $(t_s, T) \in x.Occ$ *ne peut conduire qu'à la génération d'un seul élément* $(t_s, L) \in z.Occ$*.*

◇

Le théorème suivant établit la correction du comptage du support :

Théorème 2 (correction du comptage du support) *Soit* S *une séquence d'événements et* z *une E/O-pair calculée par l'algorithme 10, pour des entrées* x *et* y *justes, complètes et non-redondantes ; alors :*

- *le nombre d'occurrences minimales de* $z.Pattern$ *de largeur* w *est* $|mo(z.Pattern, S, w)| = |\{(t_s, T) \in z.Occ \mid min(T) - t_s = w\}|$
- $Support(z.Pattern, S, w) = \sum_{0 \leq i \leq w} |\{(t_s, T) \in z.Occ \mid min(T) - t_s = i\}|$
- $GSupport(\alpha, S) = |\{(t_s, T) \in z.Occ\}|$

La preuve de ce théorème est la suivante :

Preuve 6 *Selon les lemmes 1 et 2, et le théorème 1, on obtient la première affirmation. Les deux autres suivent alors par définition du Support et du GSupport.* ◇

Puisque $GSupport(\alpha, S) \geq \sigma \Rightarrow GSupport(prefix(\alpha), S) \geq \sigma$, alors, par les théorèmes 1 et 2, l'énumeration en profondeur effectuée par *WinMiner* est correcte et permet de trouver tous les épisodes tels que leur *GSupport* est supérieur ou égal à σ. De plus, si une règle d'épisodes $prefix(\alpha) \Rightarrow suffix(\alpha)$ est fréquente pour une largeur donnée w, alors nous avons aussi $GSupport(\alpha, S) \geq \sigma$. Ainsi, par les théorèmes 1 et 2, nous pouvons également et correctement trouver le support et la confiance d'une règle pour une largeur donnée w, lorsque la règle est fréquente pour la largeur w.

Nous avons donc toute l'information nécessaire pour déterminer si une règle telle que $GSupport \geq \sigma$ est une FLM-règle, et si oui, son FLM correspondant. Ceci est réalisé par l'algorithme 11. En entrée, une règle de la forme $x.Pattern \Rightarrow suffix(z.Pattern)$, (avec $z.Pattern$ un épisode tel que $prefix(z.Pattern) = x.Pattern$) est fournie ainsi que les ensembles $x.Occ$ et $z.Occ$ contenant respectivement les *mpo* de $x.Pattern$ et de $z.Pattern$. La sortie de l'algorithme contient toutes les règles ayant un FLM ainsi que le FLM lui-même.

Sachant qu'il est possible d'obtenir des millions de règles telles que $GSupport \geq \sigma$ (cf. section 4.2.5), il nous faut donc être capable de calculer le FLM d'une règle, et ce à moindre coût. Pour cela, la première idée clé de l'algorithme est de considérer seulement les valeurs w pour lesquelles la confiance peut changer. Si, pour une règle d'épisodes donnée, le confiance varie, alors, par définition, au moins une des valeurs de support utilisée pour calculer la confiance varie également. Or, si au moins une des valeurs de support varie, cela signifie que le nombre d'occurrences minimales pris en compte pour établir le support varie. Ce nombre ne peut varier qu'en fonction de la largeur w considérée. Aussi, il suffit de connaître les largeurs w telles qu'il existe une occurrence minimale d'un épisode formant la règle pour avoir accès aux largeurs w telles que la confiance est susceptible de varier.

Ainsi, et à cette fin, l'algorithme détermine :

- $wLHS$, l'ensemble de toutes les largeurs w telles qu'il existe une occurrence minimale de largeur w, de la partie gauche (LHS pour Left Hand Side) de la règle $x.Pattern \Rightarrow suffix(z.Pattern)$,
- $wRule$, l'ensemble des largeurs des occurrences minimales de $z.Pattern$ (le plus grand épisode formant la règle, i.e. la partie gauche suivie de la partie droite de la règle).

Ceci est effectué lignes 1 et 2 par la fonction $setOfWidths$, laquelle peut être spécifiée ainsi : $setOfWidths(\mathcal{O}) = \{w \mid \exists (t_s, T) \in \mathcal{O} \wedge min(T) - t_s = w\}$ avec $\mathcal{O}$ une structure contenant, pour un épisode donné, l'ensemble de ses *mpo* stocké de façon identique au mode de stockage défini pour le champ *Occ* des E/O-pairs.

La deuxième idée clé pour optimiser les calculs est d'utiliser des histogrammes qui contiennent, pour une largeur donnée w, le nombre d'occurrences minimales des épisodes formant la règle dont la largeur est w. Un tel histogramme est construit grâce à la fonction $moWidthsHist$. $moWidthsHist(\mathcal{O})$ retourne un tableau h d'entiers défini par $h[w] = |\{t_s | \exists (t_s, T) \in \mathcal{O}, min(T) - t_s = w\}|$. Cette fonction est utilisée lignes 3 et 4 afin d'obtenir respectivement les histogrammes des occurrences minimales de la partie gauche de la règle ($moHistLHS$) et les histogrammes des occurrences minimales de $z.Pattern$ ($moHistRule$).

Le principe de l'algorithme est alors de parcourir les différentes valeurs de w stockées dans $wLHS \bigcup wRule$ selon l'ordre croissant (ligne 8). Cela permet, pour une largeur donnée w, de calculer de façon incrémentale le support d'une règle et de sa partie gauche, et de déterminer également la confiance de cette même règle.

Algorithme 11 ($findFLM$)

Entrée : *$x.Pattern \Rightarrow \langle e \rangle$ la règle à examiner, $x.Occ$ le deuxième élément de la E/O-pair de l'épisode $x.Pattern$, $z.Occ$ le deuxième élément de la E/O-pair de l'épisode $z.Pattern = x.Pattern \rightarrow e$.*

```
1.  let wLHS := setOfWidths(x.Occ)
2.  let wRule := setOfWidths(z.Occ)
3.  let moHistLHS := moWidthsHist(x.Occ)
4.  let moHistRule := moWidthsHist(z.Occ)
5.  let supportLHS := 0, supportRule := 0, confFLM := 0
6.  let w := −1, existsFLM := false, wFLM = −1
7.  while (¬(existsFLM) ∧ w ≠ max(wLHS ⋃ wRule)) do
8.    let w le plus petit élément dans wLHS ⋃ wRule
      strictement plus grand que la valeur courante de w
9.    if w ∈ wLHS
10.     let supportLHS := supportLHS + moHistLHS[w]
11.   fi
12.   if w ∈ wRule
13.     let supportRule := supportRule + moHistRule[w]
14.   fi
15.     if supportRule ≥ σ
16.       let conf := supportRule/supportLHS
17.       if conf ≥ γ ∧ conf > confFLM
18.         let confFLM := conf
19.         let supportFLM := supportRule
20.         let wFLM := w
21.       else
22.         if confFLM − conf ≥ decRate ∗ confFLM
23.           let existsFLM := true
```

24. **fi**
25. **fi**
26. **fi**
27. **od**
27. **if** $existsFLM$
28. **output** $x.Pattern \Rightarrow suffix(z.Pattern)$
29. **output** $< wFLM, confFLM, supportFLM >$
30. **fi**

Par souci de clarté, les algorithmes et les fonctions présentées précédemment ont été exposés sous leur forme abstraite, forme utilisant des structures de données basées sur les ensembles.

En ce qui concerne le prototype implémentant *WinMiner*, tous les ensembles ont été codés sous forme de listes ordonnées. La prise en compte des ordres choisis pour établir les listes permet alors d'optimiser le nombre d'opérations nécessaires. Par exemple, une E/O-pair x contient un ensemble $x.Occ$ d'éléments de la forme (t_s, T), lequel est implémenté sous la forme d'une liste ordonnée selon les valeurs croissantes de t_s. T (ensemble de nombres entiers) est également stocké sous forme de liste ordonnée selon les valeurs croissantes. Lors de la jointure, les E/O-pairs peuvent être alors parcourues selon l'ordre croissant des t_s, puis selon l'ordre des valeurs de T, ce qui permet d'éviter d'essayer de générer des occurrences d'un motif avec une occurrence de conclusion apparaissant avant une occurrence du préfixe. Des optimisations triviales ont aussi été employées, telles que le calcul de toutes les *mpo* des épisodes de taille 1 à l'aide d'une unique passe sur la séquence au lieu d'un appel sans cesse répété à la fonction *scan* pour chaque type d'événement. Enfin, toutes les listes ont été implémentées à l'aide de structures de données spécialement conçues en vue de réduire les ressources de mémoire et de calcul nécessaires à l'exécution de *WinMiner* (utilisation de chaînes de pointeurs). Toutes ces optimisations permettent à *WinMiner*, comme nous pourrons le voir dans la section suivante, d'analyser des jeux de données dont les volumes peuvent être conséquents, et ce en des temps raisonnables.

3.3.3 Expériences sur des jeux de données synthétiques aléatoires

Dans cette section, nous présentons des expériences réalisées sur des jeux de données synthétiques aléatoires en utilisant une implémentation de *WinMiner* en C++ exécutée sur un Intel Pentium IV 2 GHz disposant d'un Go de mémoire vive et dont le système d'exploitation est basé sur un noyau Linux 2.4. D'un point de vue de l'utilisation des ressources mémoire, toutes les expériences décrites dans cette section ont utilisé entre 0.5 et 300 Mo de mémoire vive.

Les jeux de données synthétiques utilisés sont des jeux de données aléatoires. Un jeu de données synthétique (s, T_s, T_e) est généré en utilisant trois paramètres : le nombre de types d'événements (i.e., $|E|$), le nombre d'unités temporelles (i.e., $T_e - T_s + 1$) et $maxP$, la probabilité maximum, pour les épisodes de taille 1, d'apparaître sur une unité temporelle donnée. La probabilité $P(e)$ d'un épisode e de taille 1 d'apparaître à chaque unité temporelle est alors choisie aléatoirement et de façon uniforme dans l'intervalle $[0, maxP]$. Les occurrences de e sont ensuite générées aléatoirement à chaque unité temporelle, en fonction de cette probabilité et de façon indépendante.

Nous avons tout d'abord généré 10 jeux de données avec les paramètres suivants : $|E| = 1000$, $T_e - T_s + 1 = 10000$ et $maxP = 1$. Chaucun des jeux de données obtenus contient environ 50000 événements et à peu près 5 événements par unité temporelle. Nous avons alors exécuté *WinMiner* avec un seuil de support minimum $\sigma = 70$ (ce qui représente environ 300 épisodes de taille 1 fréquents), un seuil de confiance minimale $\gamma = 0.8$, un taux de décroissance $decRate = 10\%$ et un gap maximum $gapmax = 300$. Bien que les valeurs $\gamma = 0.8$ et $decRate = 10\%$ expriment des contraintes plutôt faibles, aucune FLM-règle ne fut trouvée dans aucun des jeux de données.

À l'aide des figures 3.5, 3.6, 3.7, et 3.8, nous présentons les résultats d'autres experiences montrant que l'extraction de FLM-règles peut être réalisée en pratique en un temps raisonnable sur des jeux de données qui peuvent être conséquents, et ce pour différentes valeurs des paramètres.

La courbe représentée dans la figure 3.5 donne les temps d'extraction pour les jeux de données générés avec $maxP = 0.01$, $T_e - T_s + 1 = 100000$ et $|E|$ variant de 1000 à 5000. Les expériences furent réalisées avec les valeurs de paramètres suivantes : $\sigma = 700$, $\gamma = 0.8$, $decRate = 10\%$ et $gapmax = 200$.

La courbe représentée dans la figure 3.6 correspond aux expériences réalisées sur un seul jeu de données généré avec $maxP = 0.01$, $|E| = 1000$ et $T_e - T_s + 1 = 100000$. Ces expériences ont été menées avec $\sigma = 700$, $\gamma = 0.8$, $decRate = 10\%$ et $gapmax$ variant de 100 à 500.

La courbe représentée dans la figure 3.7 présente les expériences effectuées sur le même jeu de données que celui utilisé pour les expériences dont le résultat est donné par la figure 3.6. En revanche, les paramètres choisis sont cette fois-ci : σ variant de 700 à 1000, $\gamma = 0.8$, $decRate = 10\%$ et $gapmax = 500$.

Enfin, la courbe représentée par la figure 3.8 rend compte d'expériences menées sur des jeux de données générés avec $maxP = 0.01$, $|E| = 1000$ et $T_e - T_s + 1$ variant de 10000 à 1000000. Ces expériences ont été menées avec $\sigma = 0.007 \times (T_e - T_s + 1)$ (plus la séquence est courte plus le support minimal est bas), $\gamma = 0.8$, $decRate = 10\%$ et $gapmax = 200$.

Ces quatre séries d'expériences, outre les informations qu'elles apportent à

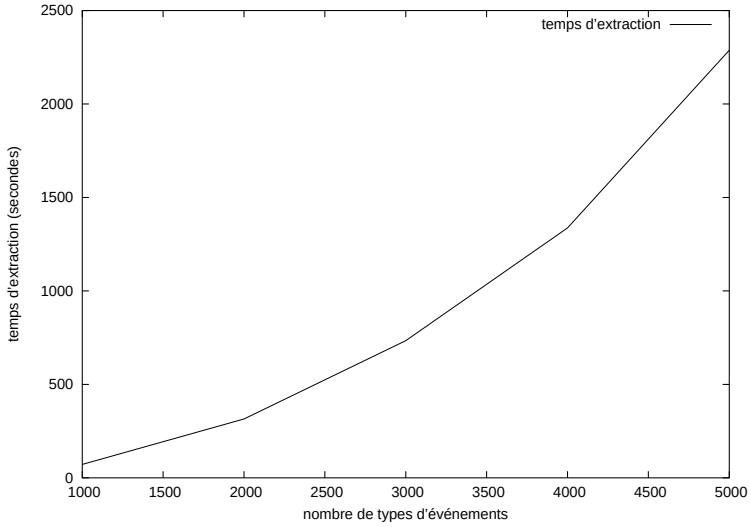


FIG. 3.5 – Nombre d'événements vs. temps d'extraction

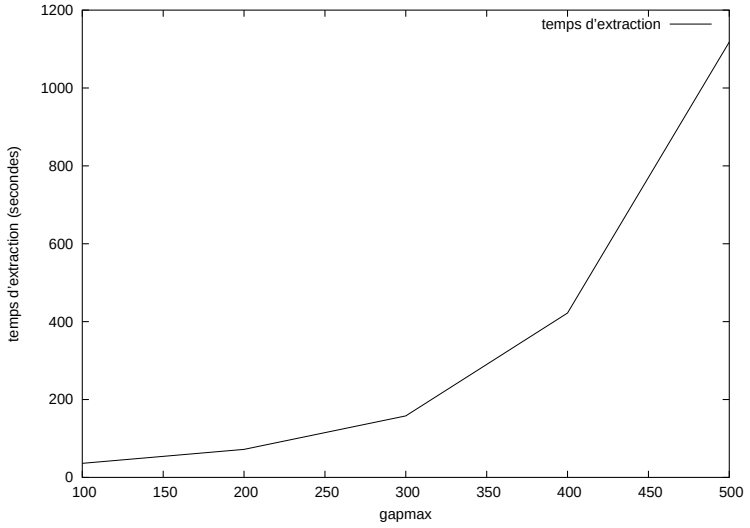


FIG. 3.6 – Gapmax vs. temps d'extraction

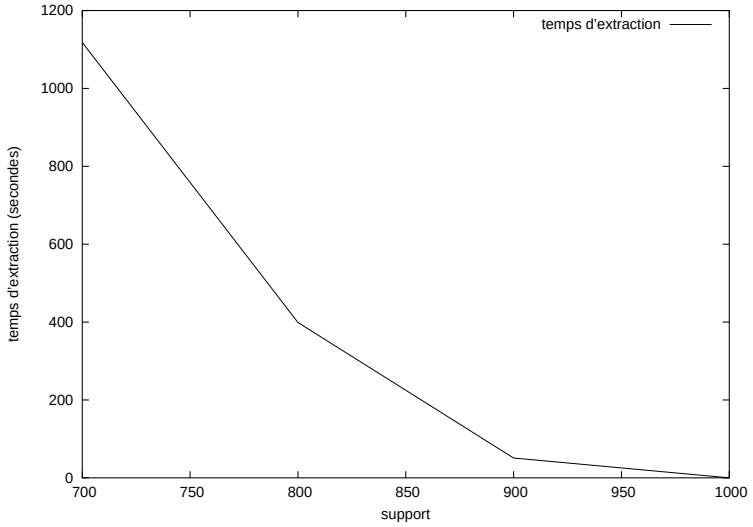


FIG. 3.7 – Support vs. temps d'extraction

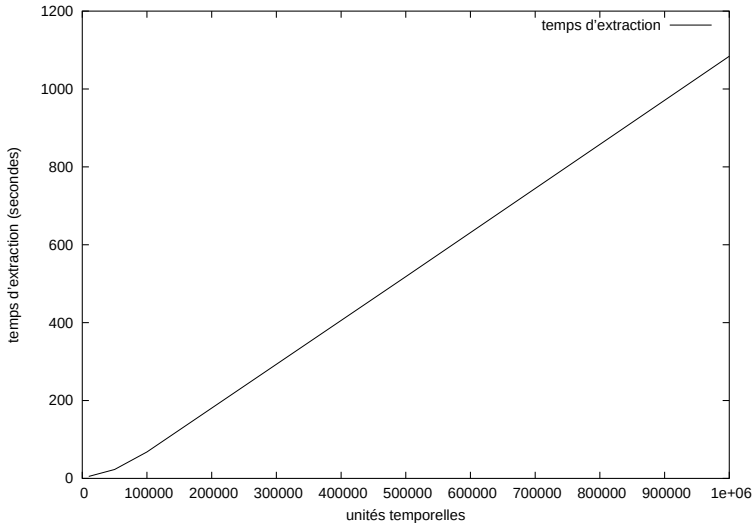


FIG. 3.8 – Nombre d'unités temporelles vs. temps d'extraction

propos des temps d'extraction, ont également permis de vérifier qu'aucune FLM-règle n'était trouvée, et donc de confirmer les résultats obtenus lors des expériences présentées au tout début de cette section.

Deux enseignements sont à tirer de l'ensemble des expériences menées sur les jeux de données synthétiques aléatoires :

- des jeux de données aléatoires tels que ceux que nous avons générés ne peuvent raisonnablement contenir que peu de phénomènes particuliers (FLM-règles avec une forte décroissance) à exhiber et qui soient fréquents. Dans ce sens, la notion de FLM-règle semble intéressante car dans la pratique l'extraction des FLM-règles s'accorde avec cette intuition. En effet, pour des paramètres exprimant des contraintes plutôt faible (confiance minimum de 0.8, taux de décroissance de 10%), aucune FLM-règle n'a été trouvée sur les jeux de données aléatoires utilisés dans les expériences précédentes. Toutefois, deux types au moins de contextes d'extraction peuvent mener à la découverte de FLM-règles sur de tels jeux. Tout d'abord, si l'on utilise un taux de décroissance très faible, ceci peut conduire à la découverte de FLM-règles (c'est d'ailleurs ce qui arrive sur les jeux précédents lorsque la valeur du taux de décroissance devient inférieure les 5%). Enfin, à très faible support, il est également possible d'extraire des FLM-règles, phénomène que nous interprétons comme étant le reflet d'une répartition non-uniforme des motifs à ces très bas supports.
- toutes les expériences ont pu être réalisées dans des temps raisonnables, ce qui permet à l'utilisateur final de tester rapidement plusieurs configurations possibles et d'orienter au plus tôt ses recherches.

En revanche, la notion de FLM-règle est-elle capable d'exhiber certains des comportements qui sous-tendent des jeux de données non-aléatoires ? Nous répondrons par l'affirmative au chapitre 4. Nous verrons que des FLM-règles, considérées comme intéressantes par les utilisateurs finaux, ont en effet été découvertes par *WinMiner*, que ce soit dans des données médicales ou des données sismiques. De plus, le volume des FLM-règles extraites est significativement inférieur au volume des règles fréquentes et confiantes analysées par *WinMiner* (i.e. les règles pour chacune desquelles il existe au moins une fenêtre telle que la règle est fréquente et confiante). C'est ce que détaille la courbe de la figure 4.9 (cf. chapitre 4). Cette courbe permet également d'observer que le volume des FLM-règles extraites est tel qu'il peut être analysé et appréhendé par l'utilisateur final, ce qui est impossible pour ce qui est du volume des règles fréquentes et confiantes. La notion de FLM-règle présente donc l'avantage de fournir à l'utilisateur final un résultat dont le volume ne dépasse pas son entendement.

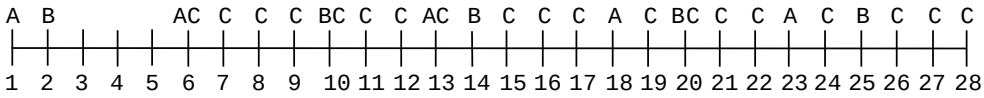


FIG. 3.9 – Séquence d'événements S_1.

3.3.4 Mesure et paramètres additionnels

La recherche des FLM-règles présente trois avantages simultanés :

– (1) : les contraintes minimales de support et de confiance permettent d'extraire des règles d'épisodes apparaissant souvent et étant fréquemment vérifiées.

– (2) : le fait de ne s'intéresser qu'aux règles présentant un FLM permet d'exhiber les règles ayant un comportement singulier en fonction des largeurs de leurs occurrences, c'est à dire un comportement particulier à fonction du temps. De plus, les FLM-règles portent une information supplémentaire, i.e. la fenêtre de temps optimale.

– (3) : les FLM-règles sont un sous-ensembles des règles fréquentes et confiantes. Lorsque l'on sait que des millions de règles fréquentes et confiantes peuvent être extraites de jeux de données considérés comme courants, la définition d'un sous-ensemble semble nécessaire si l'on souhaite que les experts puissent réellement appréhender et utiliser les résultats obtenus.

En revanche, rien ne vient apporter un éclairage sur la signification statistique des règles extraites. Par exemple, si l'on s'intéresse à la règle $A \rightarrow B \Rightarrow C$ et que l'on considère la séquence d'événements représentée par la figure 3.9, alors en choisissant une support minimum égal à 3, une confiance minimum égale à 0.7, une contrainte de gapmax égale à 5, et une décroissance de 20%, cette règle peut être retenue. On a alors un FLM situé à une largeur de 3, pour une confiance de 0.75 et un support de 3.

Mais qu'en-est-il réellement de sa *significativité* ? En effet, de très nombreuses occurrences de l'épisode C jalonnent l'étendue toute entière de la séquence d'événements. D'une certaine façon, cette règle peut n'être qu'un reflet d'une répartition très dense des épisodes C et n'exprimer aucune relation réelle entre les épisodes $A \rightarrow B$ et C.

Inversement, si l'on considère la figure 3.10, l'épisode $A \rightarrow B$ n'a que peu de chances d'être suivi par C de façon relativement proche car beaucoup moins

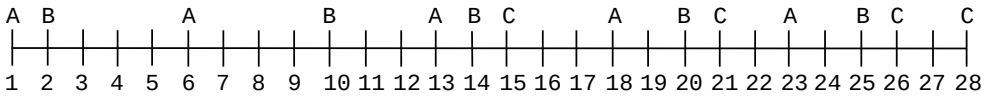


FIG. 3.10 – Séquence d'événements S_2.

d'épisodes C sont présents dans la séquence d'événements. Pourtant, en conservant les paramètres utilisés précédemment, $A \rightarrow B \Rightarrow C$ est sélectionnée, ce qui donne une vrai force à la relation exprimée. On notera que dans ce cas, les valeurs de largeur optimale, de confiance et de support de $A \rightarrow B \Rightarrow C$ sont, tous comme les paramètres d'extraction, identiques aux valeurs données dans le cas de l'exemple de la figure 3.9.

Alors que les situations sont très différentes d'un point de vue de la répartition des événements sur la séquence d'événements, la règle $A \rightarrow B \Rightarrow C$ est sélectionnée selon les mêmes critères. Comment distinguer les deux situations ? Comment sélectionner la règle exprimant une relation forte par rapport à la répartition des occurrences des épisodes ?

Le *lift*

Comme le suggèrent les exemples des figures 3.9 et 3.10, une règle peut être considérée d'autant plus significative que sa conclusion n'apparaît pas souvent dans la séquence d'événements par rapport aux occurrences de l'épisode formant sa partie gauche. Considérons une règle de la forme $\alpha \Rightarrow suffixe(\beta)$. Pour une largeur donnée, il est possible de calculer le support de α et le support de β. On peut donc calculer la confiance de cette règle. Admettons maintenant que nous puissions calculer la *confiance attendue* de $\alpha \Rightarrow suffix(\beta)$ pour la même largeur, dans le cas où $suffix(\beta)$ apparaîtrait avec la même fréquence dans la séquence d'événements, mais serait réparti de façon aléatoire et uniforme, et indépendamment des autres événements de la séquence, y compris des événements de même type. Alors, en calculant le ratio entre la confiance et la *confiance attendue*, ce ratio étant nommé *lift* [2], nous pourrions évaluer si le fait d'observer une confiance forte pour $\alpha \Rightarrow suffix(\beta)$ est simplement liée à une fréquence élevée d'apparition de $suffix(\beta)$ (valeur du ratio proche de 1) ou si au contraire, l'apparition de $suffix(\beta)$ semble réellement lié à celle de α (valeur du ratio supérieure à 1).

[2] Terme choisi en référence à la mesure nommée lift et utilisée dans le domaine des données non-séquentielles [HTF01].

Autrement dit, le *lift* peut nous permettre d'établir si la confiance d'une règle est plus forte que sa confiance attendue, i.e. de distinguer le cas où la règle émerge car issue d'une situation dans laquelle sa conclusion est très fréquente, du cas où la règle exprime une relation possible qui dépasse le hasard des répartitions.

De façon plus formelle, considérons $S =< s, T_s, T_e >$ une séquence d'événements, et $\alpha \Rightarrow suffix(\beta)$ une règle d'épisodes.

Définissons tout d'abord la fréquence d'un type d'événement :

Définition 40 ***(fréquence d'un type d'événement)*** Soit $e \in E$. La fréquence de e est définie par : $Freq(e, S) = \frac{|occ(<e>,S)|}{(T_e - T_s + 1)}$

Cette fréquence peut être interprétée comme la probabilité a priori d'apparition d'un type d'événement e_i à une date de la séquence d'événements S. Dans le cas de l'exemple donné par la figure 3.9, on a $Freq(C, S_1) = \frac{19}{28}$. Pour ce qui est de S_2 (cf. figure 3.10), on a $Freq(C, S_2) = \frac{4}{28}$.

Pour une largeur donnée w, nous allons déterminer le support attendu de $\alpha \Rightarrow suffix(\beta)$ à partir duquel nous définissons la confiance attendue de cette règle ainsi que la valeur de lift correspondante de la façon suivante :

Définition 41 ***(confiance attendue d'une règle d'épisodes)*** La confiance attendue d'une règle d'épisode $\alpha \Rightarrow suffix(\beta)$ est :
$confianceAttendue(\alpha \Rightarrow suffix(\beta), S, w) = \frac{supportAttendu(\alpha \Rightarrow suffix(\beta),S,w)}{Support(\alpha,S,w)}$

La confiance attendue d'une règle épisode $\alpha \Rightarrow suffix(\beta)$ pour une largeur w exprime la probabilité d'observer une occurrence minimale de β de largeur au plus w commençant à t sachant qu'une occurrence minimale de α de largeur au plus $w - 1$ commencent également à la date t.

Définition 42 ***(lift d'une règle d'épisodes)*** Le lift d'une règle d'épisodes $\alpha \Rightarrow suffix(\beta)$ est défini par :
$lift(\alpha \Rightarrow suffix(\beta), S, w) = \frac{Confiance(\alpha \Rightarrow suffix(\beta),S,w)}{confianceAttendue(\alpha \Rightarrow suffix(\beta),S,w)}$

Le lift d'une règle d'épisode $\alpha \Rightarrow suffixe(\beta)$ pour une largeur w exprime la comparaison entre la confiance observée de la règle et la confiance attendue de celle-ci (sous les hypothèses de distribution uniforme et d'indépendance des occurrences de $suffix(\beta)$ entre elles et par rapport aux occurrences de α). Plus le ratio est élevé, plus la règle observée est considérée comme significative.

En fait, nous allons déterminer une borne supérieure du support attendu de $\alpha \Rightarrow suffix(\beta)$, noté $supportAttendu^{+}(\alpha \Rightarrow suffix(\beta), S, w)$, ce qui nous permettra d'obtenir une borne supérieure pour la confiance attendue et ainsi une borne inférieure pour la valeur de lift (notées respectivement $confianceAttendue^{+}$ $(\alpha \Rightarrow suffix(\beta), S, w)$ et $lift^{-}(\alpha \Rightarrow suffix(\beta), S, w)$).

Considérons les occurrences minimales de $suffix(\beta)$. La probabilité qu'une occurrence d'un type d'événement $suffix(\beta)$ n'apparaisse pas à une date donnée est $1 - Freq(suffix(\beta), S)$. La probabilité que ce type n'apparaisse pas sur y dates consécutives est $(1 - Freq(suffix(\beta), S))^{y}$. La probabilité qu'il apparaisse au moins un type d'événement $suffix(\beta)$ sur cette durée y est alors $1 - (1 - Freq(suffix(\beta), S))^{y}$.

Soit n_x le nombre d'occurrences minimales de α de largeur x. Soit $m_{x,y}$ le nombre de ces occurrences qui sont suivies par au moins un $suffix(\beta)$ dans les y dates immédiatement postérieures. Une estimation par excès $k_{x,y}$ de $m_{x,y}$ est alors : $k_{x,y} = n_x * [1 - (1 - Freq(suffix(\beta), S))^{y}]$.

C'est une estimation par excès de $m_{x,y}$ car le type d'événement $suffix(\beta)$ peut apparaître dans α, ce qui fait de $Freq(suffix(\beta), S)$ une estimation par excès de la probabilité d'occurrence de $suffix(\beta)$.

Soit une largeur w. Considérons les $m_{i,w-i}$, i.e. les occurrences de α de largeur i suivies par au moins un $suffix(\beta)$ dans les $w - i$ dates immédiatement postérieures.

Puisque $\alpha = prefix(\beta)$, pour toute occurrence minimale $[t_s, t_e]$ de largeur égale ou inférieure à w, il existe une valeur de i, $i < w$, telle que : $[t_s, t_s + i]$ soit une occurrence minimale de α et il existe une occurrence $[t, t]$ de $suffix(\beta)$ telle que $t - (t_s + i) \leq w - i$.

On remarquera que $[t_s, t_s + i]$ est une occurrence minimale de α de longeur i et que l'occurrence de $suffix(\beta)$ apparaît dans les $w - i$ dates immédiatement postérieures à $t_s + i$. De plus, pour une occurrence minimale de α débutant à la date t_s, il existe au plus une occurrence minimale de β débutant à t_s. D'où :

$\sum_{0 \leq i < w} m_{i,w-i} \geq \sum_{0 \leq i \leq w} \mid mo(\beta, S, i) \mid$

Dans le cas général, cette inégalité ne peut pas être une égalité car les occurrences minimales de α suivies d'une occurrence de $suffix(\beta)$ ne permettent pas toutes de former des occurrences minimales et ce pour deux raisons :

- *gapmax* peut ne pas être satisfait entre la fin de l'occurrence minimale de α et l'occurrence de $suffix(\beta)$,
- l'occurrence minimale de α complétée avec l'occurrence de $suffix(\beta)$ n'est pas forcément une occurrence minimale.

En remplaçant $m_{i,w-i}$ par son estimation par excès $k_{i,w-i}$, on obtient :
$\sum_{0\leq i<w} n_i * [1-(1-Freq(suffix(\beta),S))^{w-i}] \geq \sum_{0\leq i\leq w} \mid mo(\beta,S,i) \mid$
Or, $n_i = \mid mo(\alpha,S,i) \mid$, d'où l'estimation par excès du support de $\alpha \Rightarrow suffix(\beta)$ sur une séquence S et pour une largeur w définie par :

Définition 43 ***(support attendu par excès d'une règle d'épisodes)*** $supportAttendu^{+}(\alpha \Rightarrow suffix(\beta),S,w) = \sum_{0\leq i<w} \mid mo(\alpha,S,i) \mid *[1-(1-Freq(suffix(\beta),S))^{w-i}]$

Dans le cas de la séquence S_1, on a $|mo(A \rightarrow B,S_1,0)| = 0$, $|mo(A \rightarrow B,S_1,1)| = 2$, et $|mo(A \rightarrow B,S_1,2)| = 2$. Ainsi : $supportAttendu^{+}(A \rightarrow B \Rightarrow C,S_1,3) = [2*(1-(1-\frac{19}{28})^2)] + [2*(1-(1-\frac{19}{28})^1)] \approx 3.1506$.
Pour ce qui est de S_2, nous avons les mêmes occurrences minimales de α, et l'on obtient : $supportAttendu^{+}(A \rightarrow B \Rightarrow C,S_2,3) = [2*(1-(1-\frac{4}{28})^2)] + [2*(1-(1-\frac{4}{28})^1)] \approx 0.8163$.

La borne supérieure de la confiance attendue peut alors être calculée. Pour S_1, on a :
$confianceAttendue^{+}(A \rightarrow B \Rightarrow C,S_1,3) = \frac{supportAttendu^{+}(A\rightarrow B\Rightarrow C,S_1,3)}{\sum_{0\leq i\leq 3}|mo(A\rightarrow B,S_1,i)|} \approx \frac{3.1506}{4} \approx 0.7876$.
Tandis que pour S_2, on a :
$confianceAttendue^{+}(A \rightarrow B \Rightarrow C,S_2,3) = \frac{supportAttendu^{+}(A\rightarrow B\Rightarrow C,S_2,3)}{\sum_{0\leq i\leq 3}|mo(A\rightarrow B,S_2,i)|} \approx \frac{0.8163}{4} \approx 0.2041$.

Enfin, en ce qui concerne le calcul de la borne inférieure du lift, on a pour S_1 :
$lift^{-}(A \rightarrow B \Rightarrow C,S_1,3) = \frac{Confiance(A\rightarrow B\Rightarrow C,S_1,3)}{confianceAttendue^{+}(A\rightarrow B\Rightarrow C,S_1,3)} \approx \frac{0.75}{0.7876} \approx 0.9523$
et pour S_2 :
$lift^{-}(A \rightarrow B \Rightarrow C,S_2,3) = \frac{Confiance(A\rightarrow B\Rightarrow C,S_2,3)}{confianceAttendue^{+}(A\rightarrow B\Rightarrow C,S_2,3)} \approx \frac{0.75}{0.2041} \approx 3.6747$

Sur cet exemple, on se rend donc compte de la capacité de cette mesure à différencier la situation représentée par la figure 3.9 de la situation représentée par la figure 3.10.

Il serait possible de proposer d'autres estimations du support attendu mais celle que nous avons utilisée présente l'avantage de pouvoir être calculée de façon incrémentale pour les différentes largeurs, comme nous allons le voir dans le théorème 3. Il est alors possible de calculer de façon incrémentale une borne supérieure de la confiance attendue et une borne inférieure du lift.

Théorème 3 *Soit $S =< s,T_s,T_e >$ une séquence d'événements et soit une règle d'épisodes de la forme $\alpha \Rightarrow suffix(\beta)$ telle que $prefix(\beta) = \alpha$. Soit w et w' tels*

que $w' = w + 1$.

Soient A, A', B *et* B' *tels que :*

$A = \sum_{0 \leq i < w} \mid mo(\alpha, S, i) \mid$
$B = \sum_{0 \leq i < w} \mid mo(\alpha, S, i) \mid * (1 - Freq(suffix(\beta), S))^{w-i}$
$A' = \sum_{0 \leq i < w'} \mid mo(\alpha, S, i) \mid$
$B' = \sum_{0 \leq i < w'} \mid mo(\alpha, S, i) \mid * (1 - Freq(suffix(\beta), S))^{w'-i}$

alors :

$supportAttendu^{+}(\alpha \Rightarrow suffix(\beta), S, w) = A - B$
$supportAttendu^{+}(\alpha \Rightarrow suffix(\beta), S, w') = A' - B'$

et

$A' = A+ \mid mo(\alpha, S, w) \mid$
$B' = (1 - Freq(suffix(\beta), S)) * [B+ \mid mo(\alpha, S, w) \mid]$

Ce théorème indique que A' et B' tels que définis peuvent être obtenus à partir de A et de B. En l'appliquant ensuite pour $w'' = w' + 1$, on peut déterminer des valeurs A'' et B'' à partir de A' et B'. A'' et B'' pourront à leur tour être utilisés pour calculer $supportAttendu^{+}(\alpha \Rightarrow suffix(\beta), S, w'')$.

Le théorème 3 peut être montré de la façon suivante :

Preuve 7 *Soit* $S =< s, T_s, T_e >$ *une séquence d'événements,* w *et* w' *des largeurs d'occurrences telles que* $w' = w + 1$. *Considérons maintenant une règle d'épisodes de la forme* $\alpha \Rightarrow suffix(\beta)$.

Si l'on définit A, A', B *et* B' *tels que :*

$A = \sum_{0 \leq i < w} \mid mo(\alpha, S, i) \mid$
$B = \sum_{0 \leq i < w} \mid mo(\alpha, S, i) \mid * (1 - Freq(suffix(\beta), S))^{w-i}$
$A' = \sum_{0 \leq i < w'} \mid mo(\alpha, S, i) \mid$
$B' = \sum_{0 \leq i < w'} \mid mo(\alpha, S, i) \mid * (1 - Freq(suffix(\beta), S))^{w'-i}$

D'après la définition 43, nous avons :

$supportAttendu^{+}(\alpha \Rightarrow suffix(\beta), S, w) = A - B$
$supportAttendu^{+}(\alpha \Rightarrow suffix(\beta), S, w') = A' - B'$

Montrons que :

$A' = A+ \mid mo(\alpha, S, w) \mid$
$B' = (1 - Freq(suffix(\beta), S)) * [B+ \mid mo(\alpha, S, w) \mid]$

Puisque $w' = w + 1$*,* B' *s'écrit :*

$B' = \sum_{0 \leq i < w'} (\mid mo(\alpha, S, i) \mid * (1 - Freq(suffix(\beta), S)) * (1 - Freq(suffix(\beta), S))^{w-i})$
$B' = (1 - Freq(suffix(\beta), S)) * \sum_{0 \leq i < w'} \mid mo(\alpha, S, i) \mid * (1 - Freq(suffix(\beta), S))^{w-i}$
$B' = (1 - Freq(suffix(\beta), S)) * (\sum_{0 \leq i < w} \mid mo(\alpha, S, i) \mid * (1 - Freq(suffix(\beta), S))^{w-i}$
$+ \mid mo(\alpha, S, w) \mid * (1 - Freq(suffix(\beta, S)))^{w-w})$ *car* $w' = w + 1$

D'où :

$B' = (1 - Freq(suffix(\beta), S)) * [B+ \mid mo(\alpha, S, w) \mid]$

Quant à A' *:*

$A' = \sum_{0 \leq i < w'} \mid mo(\alpha, S, i) \mid$
$A' = \sum_{0 \leq i < w} \mid mo(\alpha, S, i) \mid + \mid mo(\alpha, S, w) \mid$ *car* $w' = w + 1$*, d'où :*
$A' = A+ \mid mo(\alpha, S, w) \mid$

◇

$supportAttendu^{+}(\alpha \Rightarrow \beta, S, w)$ et $lift^{-}(\alpha \Rightarrow \beta, S, w)$ peuvent être calculés de façon incrémentale pour toutes les largeurs w en utilisant les occurrences minimales de α et de β sur la séquence S quand on considère les largeurs w par valeurs croissantes. C'est en fait ce que fait l'algorithme 11 pour déterminer les valeurs de confiance et les FLM. Le calcul de $lift^{-}(\alpha \Rightarrow suffix(\beta), S, w)$ peut ainsi être intégré directement dans l'agorithme 11 et ainsi permettre pour chaque FLM-règle de fournir, pour la largeur correspondante, la borne inférieure calculée pour le lift. Ce calcul a été intégré dans le prototype implémentant *WinMiner*.

Paramètres additionnels

Outre les paramètres de support minimum, de confiance minimum, de gap maximum, de taux de décroissance, le prototype de *WinMiner* permet la prise en compte d'autres paramètres (cf. figure 3.11) dont voici le détail :

– *ml* : ce paramètre permet de limiter la taille des épisodes extraits, et donc le nombre d'événements composant les règles extraites. Cela peut se révéler utile pour deux raisons. Tout d'abord, certaines extractions sont parfois

```
-h this help
-i input file
-o output file, optional, default stdout
-s minimal support, compulsory and higher than 0
-c minimum confidence, default 0.9
-l minimum lift, default 1.5
-g maximum gap between two events, default 1
-d decrease rate, default 5 (in percentage)
-ml maximum number of events per rule (maximum level), default
20
-u (unfair) counting methods do not handle border effect (this
option is set by default)
-f (fair) counting methods handle border effect
Parameters u and f are exclusives!
```

FIG. 3.11 – Message d'aide de *WinMiner*.

trop coûteuses en termes de temps et/ou de mémoire car les espaces de motifs candidats à explorer sont trop grands. Une limitation de la taille des motifs peut alors permettre de réduire ces espaces (contrainte anti-monotone). La deuxième raison concerne l'expert et l'appréhension qu'il peut avoir des règles extraites. En effet, la lecture de règles constituées de dizaines d'événements est parfois considérée comme rédhibitoire.

– l : paramètre permettant à l'utilisateur de sélectionner uniquement les FLM-règles pour lesquelles la borne inférieure de lift est supérieure ou égale à ce paramètre.

– f : ce paramètre permet de ne pas prendre en compte, lors des calculs de confiance et de lift, les occurrences des parties gauches des règles dont les occurrences la conclusion pourraient se situer à une date ultérieure à la date à laquelle se finit la séquence d'événements. En effet, un arrêt des mesures ne signifie pas pour autant un arrêt des phénomènes dont est censée rendre compte la séquence d'événements. Cette correction nous permet d'atténuer cet effet de bord, effet de bord inévitable de par la nature même des séquences d'événements. Cependant, expérimentalement, on observe que les résultats ne diffèrent que très peu lorsque cette correction est appliquée, les mêmes règles étant généralement retrouvées à des valeurs de lift et de confiance sensiblement identiques à celles initialement fournies sans correction.

3.4 Conclusion

Dans ce chapitre, nous avons défini un nouveau motif, les FLM-règles. Les FLM-règles sont des règles d'épisodes dont le comportement dans le temps est singulier, i.e. ce sont des règles pour lesquelles il existe une fenêtre temporelle optimale. Ces règles constituent un sous-ensemble des règles fréquentes et confiantes, ce qui permet de fournir, à l'issue d'une fouille de données, un collection de règles dont le volume reste généralement à la portée des capacités d'appréhension de l'expert. Ces motifs sont intéressants en pratique car ils permettent à l'expert d'obtenir des tailles de fenêtres optimales (propres à chaque règle) dans des contextes où ces tailles ne sont pas déjà connues dans le domaine.

Nous avons proposé un algorithme, *WinMiner*, pour extraire les FLM-règles. Les preuves de justesse et de complètude de cette algorithme ont également été apportées. L'implémentation de *WinMiner* nous a permis de constater que, sur des jeux de données synthétiques aléatoires, aucune FLM-règle n'était extraite, ce qui est encourageant. Cela l'est d'autant plus que, comme le montre le chapitre 4, des FLM-règles jugées comme intéressantes sont extraites de jeux de données réels et ce pour des domaines d'application différents (données médicales et données sismiques). Enfin, nous avons présenté une mesure d'interêt dédiée, le lift, qui permet de comparer la confiance observée d'une règle d'épisodes à la confiance attendue, i.e. la confiance que l'on peut estimer lorsque l'on suppose l'indépendance entre le corps et la tête d'une règle.

Chapitre 4

Applications

Alors qu'aucune FLM-règle n'est trouvée dans des jeux de données synthétiques aléatoires (cf. section 3.3.3), qu'en est-il des jeux de données réels ? La notion de FLM-règle permet-elle de capturer des règles d'épisodes dont le comportement dans le temps (ou l'espace) est singulier ? Est-ce que le volume des FLM-règles extraites ne dépasse pas l'entendement de l'expert du domaine ? Afin de répondre à ces questions, et donc de tester l'algorithme *WinMiner* ainsi que la notion de FLM-règle qu'il met en jeu, nous avons utilisé le prototype implémentant *WinMiner* dans deux contextes d'application différents : l'étude de données médicales traitant de l'athérosclérose (section 4.1) et l'analyse de catalogues de tremblements de terre (section 4.2).

4.1 Application aux données médicales

4.1.1 Contexte

L'athérosclérose est à l'origine de la majorité des maladies cardio-vasculaires et constitue la première cause de mortalité dans le monde développé. De plus, il est prévu qu'à l'aube du siècle prochain, elle soit également la principale cause de mortalité dans les pays dits émergents. Cette maladie est un processus qui tend à la création d'amas, nommés *plaques*, qui sont constitués, entre autres, de débris cellulaires, de cholésterol, de calcium et qui sont localisés dans les artères musculaires. Ces plaques, lorsqu'elles sont assez importantes, peuvent réduire le débit sanguin dans les artères, ce qui peut provoquer, chez le sujet atteint, l'apparition d'une claudication, voire l'apparition d'une gangrène au niveau des membres inférieurs. Le scénario le plus grave se déroule lorsque ces plaques se brisent. Dans ce cas, les débris des plaques peuvent boucher les veines, engendrant des attaques cardiaques et/ou cérébrales tout en augmentant le risque d'apparition d'une gangrène au ni-

veau des membres inférieurs.

Divers facteurs tels que de hauts niveaux de cholésterol et de triglycérides, ou une pression sanguine trop élevée, ont été identifiés comme facteurs à risque dans l'apparition de l'athérosclérose. Malheureusement, aucun traitement curatif n'est disponible. Il est aujourd'hui communément admis que les maladies cardio-vasculaires résultent d'un ensemble de facteurs, et qu'il faudrait considérer un risque global [AOWK90] plutôt que de concentrer, comme cela est actuellement le cas, les efforts de prévention sur des facteurs isolés. Cela soulève le problème de la méconnaissance des relations entre les différents facteurs de risque. Si l'action préventive disponible aujourd'hui n'est pas aussi efficace qu'elle pourrait l'être, elle devrait, en revanche, être significativement améliorée par une meilleure compréhension de telles relations.

Différents programmes de recherche médicale ont été mis en place pour collecter le plus d'information possible sur l'athérosclérose, afin de découvrir les facteurs à risque et d'exhiber les relations qui les lient les uns aux autres. Les données STULONG, proposées dans le cadre du Discovery Challenge ECML/PKDD depuis 2002, sont le résultat d'une telle démarche. Plus précisément, ces données concernent une étude longitudinale de vingt ans sur les facteurs à risque de l'athérosclérose. La population étudiée était constituée de 1417 hommes d'âge mûr. Cette étude a été mise en place dans les années 70, en (ex) Tchéquoslovaquie. Les résultats d'analyse de ces données lors des Discovery Challenges 2002 et 2003 étaient encourageants, mais la plupart n'exploitaient pas les informations sur l'observation des patients sur le long terme. Or, la recherche des dépendances entre les facteurs de risque et les manifestations cliniques de l'athérosclérose ainsi que l'analyse de leurs relations au temps semblent pouvoir être à la source d'une information de premier plan pour le traitement de l'athérosclérose. En effet, l'information sur un risque sans renseignement aucun sur ses conséquences dans le temps ne suffit pas pour émettre un jugement médical de qualité. Or, si les médecins savent à la fois quand le risque apparaît et quand son impact est le plus fort, alors ceux-ci détiennent une information précieuse. Si l'on considère le cancer du poumon [HDSG97], qui est un exemple relativement simple en médecine, on sait que fumer augmente dangereusement le risque d'apparition du cancer après 10 ans environ. Si le patient arrête de fumer, le risque relatif d'apparition du cancer du poumon décroît progressivement sur 15 ans. Cette information est utile car les médecins connaissent ainsi les risques de leurs patients. Ainsi, ils savent quels examens ou prescriptions sont recommandés ou non. Mais, en ce qui concerne les maladies cardio-vasculaires, le risque dépend de beaucoup de facteurs et les médecins ne savent pas exactement comment prendre en compte ces facteurs en fonction du temps.

Nous avons donc concentré nos efforts sur les données STULONG afin d'ex-

traire, à l'aide de *WinMiner*, les relations, entre les différents attributs disponibles, dont le comportement par rapport au temps apparaît comme particulier. Plus précisément, nous avons essayé d'extraire les dépendances entre facteurs à risque et/ou manifestations cliniques de l'athérosclérose ainsi que leurs fenêtres temporelles optimales respectives (cf. chapitre 3).

La section suivante présente les données STULONG tout en décrivant le pré-traitement appliqué à ce jeu pour pouvoir effectuer des extractions à l'aide de *WinMiner*. Ce pré-traitement, ainsi que l'ensemble du processus ECD déroulé à partir de ces données, ont été mis en oeuvre en collaboration avec Noël Lucas, médecin et membre du LRI (Laboratoire de Recherche en Informatique, CNRS UMR 8623) de l'Université d'Orsay. Enfin, la section 4.1.3 détaille les résultats obtenus.

4.1.2 Objectifs et préparation des données

Les données STULONG ainsi que leur description détaillée sont disponibles à l'adresse http ://lisp.vse.cz/challenge/ecmlpkdd2004. Les données se répartissent en quatre tables : ENTRY, CONTROL, LETTER et DEATH.

Description des tables

La table ENTRY contient les données relatives aux examens pratiqués sur 1417 hommes d'âge mûr lors de leur entrée dans l'étude. Chaque enregistrement contient 64 attributs, catégoriels ou numériques. Ces attributs peuvent se scinder en sous-groupes en fonction de leur signification : les données d'identification, les caractéristiques sociales, l'activité physique, la consommation de cigarettes, la consommation d'alcool, les résultats des examens physiques, et la présence/absence de facteurs de risque.

La table CONTROL rassemble essentiellement les données liées aux facteurs de risque et aux démonstrations cliniques de l'athérosclérose. Ces données ont été accumulées sur le long terme, lors du suivi des patients. Les valeurs de 66 attributs différents (attributs détaillés ci-après) ont ainsi été enregistrées à chaque contrôle. Tout comme pour la table précédente, les attributs sont catégoriels ou numériques, et sont répartis en sous-groupes : les données d'identification, les changements depuis le dernier examen, les données concernant d'éventuels congés maladie, le questionnaire A_2 (données relatives aux maladies découvertes chez les patients), les examens physiques, les examens biochimiques. Seulement 1226 patients issus de la population initiale sont concernés par la table CONTROL. Pour chaque patient, il existe 21 examens de contrôle au plus. L'ensemble de la table est ainsi constituée de 10572 examens de contrôle pratiqués entre 1976 et 1999.

Les données contenues dans la table LETTER sont dérivées d'un questionnaire envoyé par la poste à 403 patients. Elles prennent la forme de 62 attributs catégoriels donnant une information supplémentaire à propos de la condition de santé des individus concernés. Enfin, la table DEATH donne l'information sur la mort de 389 patients impliqués dans l'étude longitudinale. Les 5 attributs contenus dans cette table sont relatifs à l'identification des patients, à la date du décès, et à la cause du décès.

Comme cela sera détaillé ci-après, nos efforts se sont principalement concentrés sur la table CONTROL. Sur la base d'une expertise médicale, certains attributs ont été importés, par opération de jointure, de la table ENTRY, pour être soit utilisés pendant la fouille de données, soit employés afin de calculer de nouveaux attributs qui sont à leur tour intégrés dans cette fouille de données. Le tout a été stocké dans une table résultat à partir de laquelle nous avons construit une longue séquence d'événements. Quant aux tables LETTER et DEATH, celles-ci n'ont pas été utilisées lors de nos expériences.

Objectif des expériences

Selon les experts médicaux, les facteurs de risque les plus importants concernant l'athérosclérose sont le cholésterol, l'hypertension, le tabagisme et l'activité physique. D'autres facteurs aggravants sont l'âge, le diabète, la consommation d'alcool, le BMI [1] et les antécédents familiaux. Enfin, l'information concernant le niveau d'études des patients semble être également à prendre en compte. Comme cela est précisé dans la prochaine section, nous avons sélectionné, pour nos expériences, un sous-ensemble de ces facteurs de risque, en ne retenant que ceux qui pouvaient faire sens dans une séquence d'événements. Notre objectif a été, en utilisant *WinMiner*, de donner aux experts médicaux un moyen de suivre à la fois l'incidence et l'évolution des facteurs à risque et (1) l'impact des interventions médicales telles que la prescription de médicaments ou la mise en place d'un régime alimentaire, (2) les modifications dans le comportement des patients tels que le changement dans l'activité physique ou le niveau de tabagisme. Nous souhaitions également fournir aux experts une idée des périodes de temps sur lesquelles les dépendances trouvées sont significativement observées ainsi que la fréquence et la probabilité de ces dépendances dans les données.

Sélection et discrétisation des attributs

[1] Le BMI (Body Mass Index) établit un rapport entre le poids et la taille. Il est calculé en divisant le poids (en kilos) par le carré de la taille (en mètres). Cet indice est généralement utilisé pour quantifier l'obésité.

Dans un premier temps, nous avons construit une nouvelle table en effectuant une opération de jointure entre les tables ENTRY et CONTROL afin d'importer, de la table ENTRY, les attributs suivants :

- ROKNAR (année de naissance du patient),
- VYSKA (taille du patient),
- VZDELANI (niveau d'études du patient),
- RARISK (antécédents familiaux).

Les deux premiers attributs ont été respectivement utilisés pour calculer, à chaque contrôle, l'attribut Age (âge du patient) et BMI (Body Mass Index du patient). Les deux derniers attributs ont été utilisés sur recommandation des experts médicaux. Nous n'avons pas importé les nombreux attributs décrivant la consommation d'alcool des patients, car ceux-ci n'indiquent la consommation d'alcool que pour le premier examen d'entrée dans l'étude, aucune information relative à la consommation d'alcool n'étant disponible pour les examens de contrôle suivants.

Nous avons ensuite sélectionné les attributs considérés comme intéressants dans le cadre d'une extraction de règles d'épisodes avec *WinMiner*. Ci-dessous, nous détaillons cette sélection en considérant les différents sous-groupes des attributs constituant les tables ENTRY et CONTROL. Par souci de clarté, nous ne détaillerons systématiquement ni l'ensemble des attributs disponibles dans ces deux tables, ni les différentes valeurs pouvant être prises par les attributs sélectionnés.

Données d'identification et caractéristiques sociales

L'attribut catégoriel VZDELANI (niveau d'études) a été utilisé tel quel. Les attributs numériques ICO (numéro d'identification du patient), ROKVYS (année de l'examen de contrôle), MESVYS (mois de l'examen de contrôle), et PORADK (numéro de l'examen de contrôle) ont été combinés afin de construire une longue séquence d'événements (cf. paragraphe suivant *Construction de la séquence d'événements*). De même, les attributs ROKNAR et ROKVYS ont été utilisés pour calculer l'attribut Age (Age = ROKVYS - ROKNAR). Puis, à partir des valeurs de l'attribut Age, nous avons créé un nouvel attribut CATAge qui vaut 1 si Age < 50, et qui vaut 2 sinon.

Changements depuis le dernier examen de contrôle

L'attribut ZMCHARZA (changement de travail) n'a pas été sélectionné car il ne semblait pas être pertinent dans le cadre de notre étude. L'attribut ZMKOUR (changement de la consommation journalière de cigarettes) n'a pas été utilisé non plus car il n'apparaît pas comme étant suffisamment fiable. En revanche, afin de pouvoir prendre en compte le tabagisme, l'attribut POCCIG (nombre de cigarettes par jour) a été remplacé par deux attributs, SMOKE_bin et CATCig, qui indiquent respectivement si le patient fumait à un examen de contrôle ou non,

et si oui, à quelle catégorie de fumeur il appartient. Ces deux attributs sont définis plus précisément dans le tableau 4.1.

À cause du trop grand nombre de catégories définies pour l'attribut LEKTLAK (prise de médicaments pour réduire la pression sanguine), celui-ci a été remplacé par l'attribut CATMED dont les valeurs sont définies en fonction de LEKTLAK (cf. tableau 4.2).

Les attributs catégoriels ZMTELAKT (changement dans l'activité physique), AKTPOZAM (activité physique pendant les loisirs), ZMDIET (changement de régime alimentaire), LEKCHOL (prise de médicaments anti-cholestérol) ont quant à eux été sélectionnés et utilisés tels quel.

Congés maladie

Les attributs catégoriels SRDCE, HYPERT, CEVMOZ et DIAB (attributs spécifiant différentes causes de congés maladie pris depuis la dernière visite) ont tous été utilisés tels quel.

Questionnaire A_2

Les attributs catégoriels HODN0, HODNO1, HODN02, HODN03, HODN04, HODN011, HODN012, HODN013, HODN014, HODN015, HODN021, HODN023 (série d'attributs encodant diverses maladies cardio-vasculaires) ont tous été sélectionnés. De même, les attributs BOLHR (douleur à la poitrine), BOLDK (douleur dans les membres inférieurs) et DUSN (dyspnée) ont été utilisés et catégorisés comme cela est indiqué dans le tableau 4.1 (respectivement CATBohlr, CATBoldk et CATDusn).

Examens physiques et biochimiques

Les attributs TRIC (plis de la peau au niveau des triceps), SUBSC (plis de la peau au niveau des muscles sous-capillaires), GLYCEMIE (glycémie) et KYSMOC (acide urique) n'ont pas été sélectionnés.

Valeurs	**0**	**1**	**2**	**3**
CATBMI	-	$BMI < 25$	$25 \leq BMI < 30$	$BMI \geq 30$
SMOKE_bin	$POCCIG = 0$	$POCCIG > 0$	$POCCIG$ n.r.	-
CATCig	$POCCIG \leq 10$	$10 < POCCIG \leq 20$	$POCCIG > 20$	-
CATBohlr	-	$BOHLR$ =1 (pas de douleur) ou 2 (douleur non ischémique) ou 4 (autre type de douleur)	$BOHLR$ =3 (angine de poitrine) ou 5 (infarctus du myocarde)	-
CATBoldk	-	$BOLDK$ =1 (pas de douleur) ou 2 (douleur non ischémique)	$BOLDK$ =3 (claudication)	$BOLDK$ n.r.
CATDusn	-	$DUSN$ = 1 (pas de dyspnée)	$DUSN$ = 2 (niveau I) ou 5 (niveau IV)	$DUSN$ n.r.

TAB. 4.1 – Discrétisation de certains attributs ('-' = non utilisé, 'n.r.' = non renseigné).

LEKTLAK	CATMED
ne prend pas de médicament pour la pression artérielle	0
prend 1 médicament pour la pression artérielle	1
prend plus de 1 médicament pour la pression artérielle	2
donnée absente	3

TAB. 4.2 – Construction de l'attribut CATMED en fonction de LEKTLAK.

L'attribut HMOT (poids du patient en kilogrammes) a été utilisé conjointement avec l'attribut VYSKA (taille du patient en centimètres, importé de la table ENTRY) pour calculer le BMI (Body Mass Index) d'un patient à chaque contrôle, en utilisant la formule : $BMI = (HMOT * 1000)/(VYSKA)^2$. Le BMI est l'unique variable numérique sélectionnée et utilisé dans les expériences. Nous avons décidé de coder sa tendance entre deux examens de contrôle consécutifs plutôt que de coder directement sa valeur. Pour cela, nous avons considéré un paramètre x fixé par l'utilisateur et C_i et C_{i+1} 2 contrôles consécutifs d'un même patient. Si $|BMI(C_{i+1}) - BMI(C_i)| \leq x\%$, alors le BMI est codé BMI_stab (stabilité) ; sinon si $BMI(C_{i+1}) - BMI(C_i) > 0$, alors le BMI est codé BMI_pos (augmentation) ; sinon si $BMI(C_{i+1}) - BMI(C_i) < 0$, alors le BMI est codé BMI_neg (diminution). Ensuite, BMI_stab a été associé à la valeur 0, BMI_pos à la valeur 1 et BMI_neg à la valeur 2. De plus, le BMI des patients a été utilisé pour évaluer un autre nouvel attribut, CATBMI (voir tableau 4.1), lui aussi utilisé lors de nos expériences. En ce qui concerne la pression sanguine, nous n'avons pas utilisé les attributs numériques SYST (pression systolique) et DIAST (pression diastolique). Nous avons plutôt utilisé l'attribut HYPERSD qui exprime directement si le patient a de l'hypertension systolique-diastolique ou non. Pour ce qui est des attributs HYPERS (hypertension systolique) et HYPERD (hypertension diastolique), ils n'ont pas été utilisés car jugés comme redondants par rapport à HYPERSD. Quant au cholestérol, nous avons seulement choisi l'attribut HYPCHL, qui indique si un patient présente ou non de l'hypercholesterolémie. Les attributs numériques CHLST, CHLSTMG, HDL, HDLMG, LDL étant redondants par rapport à HYPCHL, ceux-ci n'ont pas été sélectionnés. En ce qui concerne les triglycérides, nous avons également retenu un seul attribut, HYPTGL. Enfin, MOC a été utilisé tel quel. Cette attribut indique, pour un examen, si l'urine contient ou non du sucre et/ou de l'albumine.

Construction de la séquence d'événements

La table construite à partir de la jointure entre les tables CONTROL et ENTRY a alors été exportée sous forme de fichier texte et a été traitée de façon à construire une longue séquence d'événements qui puisse être analysée par *WinMiner*. Les événements de la séquence ont été construits comme suit : pour chaque patient, nous avons construit une sous-séquence contenant tous les examens de contrôle le concernant. Puis, nous avons mis bout à bout toutes ces sous-séquences, une sous-séquence correspondant à un patient. Pour un patient,

```
//Premier examen du premier patient
1000                                    602
1000                                    701
1000                                    902
...
1000                                    6701
//Second examen du premier patient
1016                                    501
...
1016                                    7100
...
//Dernier examen (18ème) du premier patient
1230                                    502
1230                                    602
...
1230                                    6703
1230                                    7100
//Premier examen du second patient (Gap > 500)
2000                                    502
...
2000                                    6701
.......
```

FIG. 4.1 – Extrait de la séquence d'événements construite à partir des données STULONG.

un événement est un couple *(date de l'examen de contrôle, valeur de l'attribut sélectionné)*. La date de l'événement a été calculée selon la formule : $date = n * 10^3 + nombre\ de\ mois\ depuis\ le\ premier\ examen$, avec n le numéro du patient ($1 \leq n \leq 1226$). C'est ainsi que ICO et PORADSK ont été indirectement utilisés pour respectivement différencier les patients, puis, pour un patient, différencier les examens de contrôle (un patient ayant au maximum 21 examens répartis entre 1976 et 1999). L'information portée par les attributs ROKVYS et MESVYS a été utilisée pour calculer le nombre de mois écoulés depuis le premier contrôle (associé au mois 0) pour un patient donné. Ce codage permet, lors d'une recherche avec un gap maximum de 500 mois, de ne pas former de règles d'épisodes à partir d'événements correspondant à deux patients différents. En effet, les patients sont suivis sur 23 ans au maximum, soit 276 mois. Pour chaque variable catégorielle sélectionnée, les types d'événement associés sont construits en concaténant le numéro d'ordre associé à la variable catégorielle choisie et les différents valeurs prises par la variable considérée. Enfin, la séquence d'événements totale a été obtenue par concaténation de toutes les sous-séquences construites pour chaque patient ayant subi au moins 3 examens de contrôle. Un extrait de cette séquence est donné par la Figure 4.1.

4.1.3 Résultats

Nous avons réalisé une série d'expériences [2] en utilisant la séquence d'événements

[2] Toutes les expériences ont été menées à l'aide d'une implémentation de *WinMiner* en C/C++, sur un Intel Pentium IV 2 GHz géré par un système d'exploitation reposant sur le noyau Linux 2.4, et disposant d'un 1 Go de mémoire vive (toutes les expériences ont utilisé entre 0.5 et 300 Mo de mémoire vive).

issue de la préparation des données décrite dans la section 4.1.2.

Nous présentons dans cette section les résultats obtenus lors d'une extraction pour laquelle les valeurs des paramètres ont été fixées comme suit : σ= 100 (support minimal), γ= 0.8 (confiance minimale), $gapmax$= 500 (gap maximum) et $decRate$= 10 (taux de décroissance minimum). Lors de cette extraction, nous avons limité la taille des épisodes extraits à 3 (i.e. les épisodes extraits ont une taille qui varie de 1 à 3). L'expérience a été réalisée en 316.87 secondes. 42262 règles d'épisodes satisfaisant au seuil de support minimal ont été trouvées, parmi lesquelles 9248 satisfaisaient également le seuil de confiance minimale (pour au moins une taille de fenêtre), Finalement, seulement 6 FLM-règles ont été trouvées par *WinMiner*. Cela montre clairement que la définition de FLM-règle permet d'aboutir à des résultats dont le volume n'est pas trop important (cf. section 4.2.5). Ceci est important car un expert ne peut pas parcourir et analyser des milliers de règles. Les 6 règles d'épisodes trouvées sont :

$HYPCHL\_2 \rightarrow chDiet\_ZMDIET\_3 \Rightarrow HYPCHL\_2 : w = 40 : cw = 0.800797 : sw = 201$

$HYPTGL\_2 \rightarrow bmi\_BMI\_neg \Rightarrow HYPTGL\_2 : w = 43 : cw = 0.807595 : sw = 319$

$CATAge\_1 \rightarrow chDiet\_ZMDIET\_3 \Rightarrow bmi\_BMI\_stab : w = 94 : cw = 0.87234 : sw = 123$

$CATAge\_1 \rightarrow chDiet\_ZMDIET\_3 \Rightarrow dyspnea\_CATDusn\_1 : w = 86 : cw = 0.93617 : sw = 132$

$CATAge\_1 \rightarrow CATMED\_1 \Rightarrow urine\_MOC\_1 : w = 116 : cw = 0.898305 : sw = 106$

$CATAge\_1 \rightarrow CATMED\_1 \Rightarrow limbpain\_CATBoldk\_1 : w = 116 : cw = 0.915254 : sw = 108$

Afin d'interpréter ce résultat, les précisions suivantes sont nécessaires :

- le caractère séparateur de champs est ' :',
- le premier champ donne la règle d'épisode découverte,
- le champ w donne sa fenêtre optimale (FLM),
- le champ cw donne sa confiance pour la fenêtre optimale w,
- le champ sw donne son support pour la fenêtre w,
- le type d'événement $HYPCHL\_2$ signifie « pas d'hypercholesterolémie »,
- le type d'événement $HYPTGL\_2$ signifie « pas d'hypertriglycéridémie »,
- le type d'événement $chDiet\_ZMDIET\_3$ signifie « le patient suit parfois le régime recommandé »,
- le type d'événement bmi_BMI_neg signifie « le BMI du patient a diminué depuis le dernier examen »,
- le type d'événement bmi_BMI_stab signifie « le BMI du patient est resté stable depuis le dernier examen »,
- le type d'événement $dyspnea\_CATDusn\_1$ signifie « pas de dyspnée/de problème respiratoire »,
- le type d'événement $urine\_MOC\_1$ signifie « l'urine est normale »,
- le type d'événement $limbpain\_CATBoldk\_1$ signifie « pas de douleur dans

les membres inférieurs »,
– le type d'événement $CATMED\_1$ signifie « le patient prend un seul médicament pour la pression sanguine ».

La première ligne du résultat peut alors être lue ainsi : *si le patient n'a pas d'hypercholesterolémie et s'il suit parfois son régime, alors ce patient n'a pas d'hypercholesterolémie, avec une probabilité de 0.8, dans les 40 mois suivants (intervalle considéré comme la fenêtre optimale pour cette règle). Enfin, cette règle se retrouve 201 fois dans la séquence d'événements.* Chacune des règles d'épisodes découverte lors de cette expérience exprime une connaissance validée dans le domaine concerné. L'information additionnelle est alors portée par le champ w, i.e. la fenêtre de temps optimale. Le fait que nous exhibions des phénomènes connus nous donne une indication sur la correction de l'ensemble du processus, de la préparation des données à l'utilisation de *WinMiner*. On peut par ailleurs noter qu'aucune règle d'épisodes ne conclut sur un type d'événement exprimant un problème de santé. En effet, ce type d'événement n'est pas fréquent dans les données. Nous avons donc réalisé une nouvelle expérience en employant les valeurs de paramètres utilisées lors de la première expérience, à l'exception de la valeur de support minimal qui a été cette fois-ci fixée à 20. 80604 règles satisfaisant au seuil de support minimal ont été trouvées. 11466 règles satisfaisaient également au seuil de confiance minimale (pour au moins une taille de fenêtre). Parmi celles-ci, 217 règles sont des FLM-règles. La règle suivante est l'une de ces FLM-règles :

$chDiet\_ZMDIET\_6 \rightarrow claudication\_HODN12\_12 \Rightarrow claudication\_HODN12\_12 : w = 31 : cw = 0.913043 : sw = 21$

Cette règle d'épisodes affirme que *si un patient mange moins de graisses et de carbohydrates et si ce patient est atteint de claudication, alors le patient est encore atteint de claudication dans les 30 mois suivants (fenêtre optimale) avec une probabilité de 0.9.*

Une fois de plus, la règle d'épisode exprime un phénomène attendu tout en apportant une nouvelle information, la fenêtre optimale.

L'ensemble de ces résultats est encourageant. En effet, les maladies cardiovasculaires dont rend compte le jeu de données STULONG sont plutôt bien connues et la connaissance que nous avons *re*découverte à partir de ces données est une connaissance entièrement validée. Toutefois, nous raffinons cette connaissance en apportant une nouvelle information relative à des aspects temporels. On peut alors raisonnablement envisager que l'accès à de nouvelles données permettra la prise en compte d'autres facteurs de risque découverts ces 10 dernières années, et rendra possible la découverte de phénomènes nouveaux ainsi que leurs fenêtres temporelles optimales respectives.

Enfin, on notera que, dans ce contexte d'application, la mesure de lift ne peut pas être utilisée. En effet, afin d'empêcher la mise en relation des examens de deux patients différents, la construction de la séquence d'événements impose l'ajout

d'un certain nombre d'unités temporelles *artificielles* entre les différentes sous-séquences construites pour chaque patient. Les fréquences des différents types d'événements sont alors sous-évaluées, ce qui de fait fausserait la mesure de lift en sur-évaluant celle-ci. Il eut été possible de prendre en compte ce phénomène, et de rectifier les fréquences des types d'événements en fonction du nombre d'unités temporelles artificielles en modifiant le prototype utilisé. Ceci n'a pu être fait par manque de temps pour cette application spécifique.

4.2 Application aux données sismiques

4.2.1 Contexte

Ce travail de thèse a été financé par le projet européen AEGIS (Ability Enlargement for Geophysicists and Information technology Specialists, IST-2000-26450). L'objectif de ce projet est de créer des synergies entre la communauté des géophysiciens et la communauté des informaticiens au travers de formations mutuelles et de travaux de recherche communs. Le projet AEGIS implique des partenaires aussi bien privés (le Centre d'Études et Productions Schlumberger - Clamart, le Centre de Recherche Schlumberger - Cambridge) que publics (École Normale Supérieure de Paris, Institut National des Sciences Appliquées de Lyon, Institut de Physique du Globe de Paris, Université de PATRAS, Conseil Supérieur de la Recherche Scientifique Espagnole).

Dans le cadre de ce projet, nous avons été amenés à collaborer avec Hélène Lyon-Caen et Francesco Pacchiani, membres du laboratoire de géologie de l'ENS Paris (UMR 8538 du Centre National de la Recherche Scientifique). Leurs travaux concernent la description et la compréhension des tremblements de terre et des systèmes géologiques concernés. Pour ce faire, de nombreux jeux de données sont à leur disposition. Ces jeux de données peuvent se répartir en deux catégories. La première rassemble tous les jeux de données accessibles en ligne, gratuitement, tels que ceux fournis par le système américain ANSS (Advanced National Seismic System). La deuxième est relative aux données directement obtenues par l'ENS Paris, ou par l'un de leurs partenaires, et qui sont issues de campagnes de mesures ou de réseaux de surveillance. La nature des données disponibles est très variée. Cela va de simples catalogues recensant des tremblements de terre (heure, position, magnitude) à des données plus détaillées indiquant, par exemple, pour chaque tremblement de terre, les déplacements du sol (horizontal, vertical), les forces auxquelles est soumis le sol (extension, compression), ou bien encore les variations du champ magnétique. Plus généralement, on distingue les données brutes, essentiellement des séries temporelles comme les sismogrammes, des données traitées, comme la valeur de la magnitude d'une tremblement de terre obtenue suite au

traitement de ces mêmes sismogrammes.

Bien que disposant de techniques classiques d'analyse de données robustes et éprouvées, telles que les transformées de Fourier, les transformées en ondelettes, ou bien encore la classification, les géophysiciens n'en restent pas moins ouverts à toute autre technique leur permettant de mieux appréhender les jeux de données dont ils disposent. En effet, si l'on s'intéresse, dans un but de description, voire de prédiction, aux dépendances qui sous-tendent ces données, celles-ci sont potentiellement tellement nombreuses qu'il est difficilement envisageable de procéder selon une démarche de test [3], et ce d'autant plus que la connaissance disponible pour analyser et expliquer ces jeux de données est généralement parcellaire au vu de l'étendue et de la complexité des systèmes physiques mis en jeu.

Les catalogues de tremblements de terre en sont une parfaite illustration. Ces catalogues contiennent essentiellement la date, la position et la magnitude (i.e. la puissance) des tremblements de terre ayant affecté une zone géographique donnée durant une période donnée. Très peu de connaissance a pu être dérivée de ces catalogues. Pour l'instant, celle-ci se résume à une loi qui pose que le nombre de séismes observés est exponentiellement décroissant avec la magnitude. En revanche, pour ce qui est des dépendances éventuelles entre les tremblements de terre, qu'elles soient spatiales ou temporelles, les diverses études statistiques menées jusqu'ici n'ont dégagé aucun comportement jugé d'intérêt. Deux raisons peuvent être avancées pour expliquer cette difficulté. Premièrement, il est impossible pour les géophysiciens de tester toutes les dépendances potentielles. Or, les tests statistiques pratiqués sont élaborés en fonction d'une connaissance préalable, ce qui peut constituer un frein à la découverte de nouvelles connaissances. Deuxièmement, les données disponibles concernent des périodes temporelles infiniment étroites au regard des temps géologiques.

4.2.2 Données et objectifs

Nous avons concentré nos efforts sur les catalogues de tremblements de terre fournis par l'ANSS (http ://quake.geo.berkeley.edu/cnss/). Ces catalogues présentent plusieurs avantages. Tout d'abord, comme cela a été précédemment évoqué, très peu de connaissance a pu être dérivée de ces jeux de données. De plus, ces jeux de données sont mis en ligne gratuitement. Enfin, ils couvrent la totalité de la surface de la terre et ce pour une période allant de 1898 (pour certaines zones) à nos jours.

De façon plus détaillée, les catalogues que nous avons sélectionnés donnent, pour chaque tremblement de terre :

[3] Démarche selon laquelle le scientifique émet une hypothèse qu'il tente par la suite de confirmer ou d'infirmer à l'aide des données et connaissances disponibles.

– l'identifiant du tremblement de terre,
– la date du tremblement de terre (année, mois, jour, heure, seconde),
– les coordonnées géographiques du tremblement de terre (latitude, longitude),
– la magnitude du tremblement de terre (échelle de Richter),
– la profondeur du tremblement de terre (en kilomètres),
– le numéro de la station dont sont issues les mesures.

Si l'on souhaite découvrir des relations temporelles entre les différents tremblements de terre, il apparaît comme nécessaire d'obtenir, de façon automatique, les dépendances entre tremblements de terre ayant une fenêtre optimale ainsi que la valeur de cette fenêtre. En effet, il est aujourd'hui difficile pour un géophysicien de privilégier, au vu de la connaissance disponible, une taille de fenêtre plutôt qu'une autre, et ce d'autant plus que cette taille peut varier d'une dépendance à une autre. Autrement dit, les géophysiciens, dans le cadre d'une recherche de dépendances, considèrent les fenêtres optimales comme une information à obtenir et non comme un paramètre à définir pour mener à bien la dite recherche. C'est d'ailleurs l'expression de ce besoin qui a en partie suggéré l'ensemble des travaux présentés dans ce mémoire. Notre objectif est donc d'exhiber les relations potentiellement intéressantes pour les géophysiciens tout en fournissant les fenêtres optimales temporelles ou spatiales d'observation de ces mêmes relations.

4.2.3 Préparation des données

Les données retenues sont définies à partir des critère d'espace et de temps. Le critère d'espace permet de définir un zone géographique que l'on considère comme couvrant un système géologique/géophysique dont les propriétés sont plutôt bien établies. En effet, si tel n'est pas le cas, les géophysiciens ne peuvent pas se prononcer de façon suffisamment réactive sur les dépendances mises à jour. Le critère de temps permet de sélectionner les périodes de temps lors desquelles apparaissent les tremblements de terre. Par exemple, on peut choisir de n'étudier que certaines crises sismiques, lesquelles sont le plus souvent localisées à la fois dans le temps et dans l'espace.

Dans notre cas, nous avons choisi de nous intéresser à la zone de subduction située le long de la côte chilienne. Cette zone couvre en effet un système géologique/géophysique considéré comme connu. Ce système peut être expliqué à l'aide de la théorie des plaques tectoniques. Selon cette théorie, la couche externe de la terre, la litoshpère, est composée de plaques en mouvement les unes par rapport aux autres. Ce mouvement est provoqué par des cellules de convection localisées dans l'asthénosphère, entre le noyau de la terre et sa couche externe. Plus précisément, l'asthénosphère est une couche de matériaux fondus sur laquelle *flottent* les plaques composant la lithosphère. Les limites entre les plaques sont alors les zones qui compensent et réajustent le mouvement de celles-ci.

De façon simplifiée, quatre types de limites, ou de zones, sont présentes à la surface de la terre :

- zones de collision : ces zones apparaissent lorsque deux plaques se rencontrent et rentrent l'une dans l'autre. Ceci est typiquement le cas de l'Himalaya où la plaque du sous-continent indien vient poinçonner la plaque eurasienne. La séismicité de ces zones est très élevée. En effet, alors qu'elles totalisent 10% des tremblements de terre, elles produisent 20% de l'énergie dégagée par les tremblements de terre.

- zones de subduction (figure 4.2) : il s'agit de zones où les plaques se chevauchent l'une l'autre, l'une des plaques amorçant alors une plongée en dessous de l'autre plaque, en direction des couches profondes de la terre. Le Chili est un très bon exemple de ce système, exemple dans lequel la plaque dite de *Nazca* vient s'enfoncer sous la plaque de l'Amérique du Sud. Environ 80% de l'énergie produite par les tremblements de terre est issue de telles zones.

- zones d'expansion (figure 4.2) : ces zones peuvent être vues comme d'immenses fentes de la couche externe de la terre, au travers desquelles remontent les différents matériaux fondus issus des couches profondes de la terre. Ces matériaux s'épanchent de chaque côté de la fente, repoussant de part et d'autre les plaques tectoniques présentes, lesquelles se voient augmentées par un flux quasi-continu de matières issues des profondeurs de la terre. La plupart de ces zones, appelées aussi dorsales, sont situées au fond des océans, à l'exemple de la dorsale atlantique dont l'Islande constitue la partie émergée.

- les failles transformantes : ce sont des zones où les plaques *glissent* l'une contre l'autre sans pour autant entrer en collision frontale ou se chevaucher l'une l'autre. Un exemple bien connu est la faille de San Andreas située le long de la côte californienne et nord mexicaine.

La figure 4.2, représente, au niveau de la zone de subduction, la configuration géophysique rencontrée le long de la côte chilienne. Après de multiples essais et extractions, nous avons finalement sélectionné une zone géographique assez restreinte au regard de la taille du Chili et située le long de sa côte. Cette zone, située entre 72° et 75° de longitude ouest et entre 32° et 34° de latitude sud, correspond à une zone assez perturbée du point du vue sismique, dont la taille est suffisamment petite pour pouvoir relier entre eux les tremblements de terre qui s'y produisent.

Pour ce qui est de la période temporelle choisie, nous avons sélectionné tous les tremblements de terre entre le 1^{er} janvier 1960 et le 1^{er} janvier 2002, car ce n'est

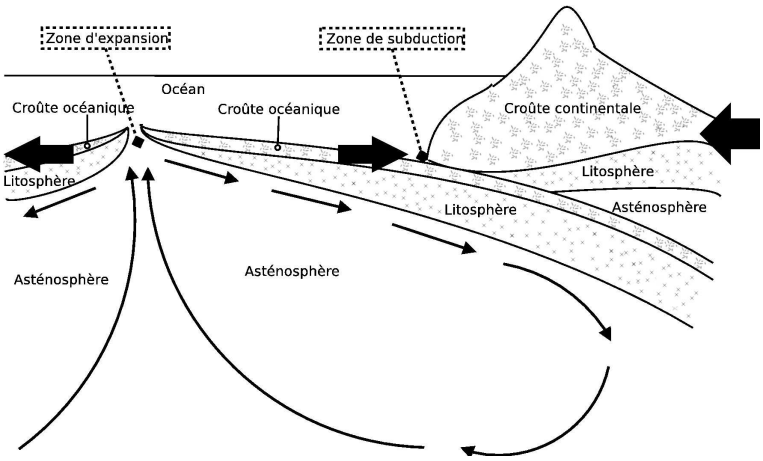


FIG. 4.2 – Les zones de subduction et d'expansion.

qu'à partir des années 60 (date d'apparition des premiers réseaux de surveillance sismique) que le catalogue peut être considéré comme suffisamment complet.

À partir de là, plusieurs types de discrétisation s'offrent à nous. Par exemple, on peut choisir de créer un alphabet de types d'événements, dans lequel chaque type correspond à une sous-zone géographique de la zone sélectionnée. Ensuite, à chaque tremblement de terre est associée une sous-zone géographique en fonction de ses coordonnées. Chaque tremblement de terre donne alors lieu à la création d'une paire *(date du tremblement de terre, sous-zone géographique associée)*. Pour finir, l'ensemble de ces paires, ordonnées selon la *date du tremblement de terre*, définit une séquence d'événements qui peut être utilisée en entrée par *WinMiner*. Une autre discrétisation possible intervient au niveau du type d'événement, lequel est alors défini comme la combinaison entre la zone géographique d'un tremblement de terre et sa magnitude, laquelle peut être catégorisée afin d'éviter une trop forte dispersion des événements (i.e. une situation dans laquelle chaque type d'événements n'est associé qu'avec un nombre relativement faible d'événements). Enfin, on peut, lorsque la zone définie est assez petite, créer une séquence d'événements constituée de paires *(date du tremblement de terre, CAT-Mag)* où *CAT-Mag* correspond à la catégorie à laquelle appartient la magnitude du tremblement de terre.

C'est ce dernier de type de discrétisation que nous avons finalement retenu pour notre série d'expériences, car les dépendances extraites à partir des jeux de données ainsi préparés peuvent être assez souvent re-situées dans le cadre d'une connaissance disponible.

Pour ce qui est de la discrétisation des magnitudes, nous avons procédé comme suit : soit une magnitude exprimée sous la forme $e.d$, avec e sa partie entière et d sa partie décimale à deux chiffres. Si $25 \leq d < 75$ alors $CAT\text{-}Mag = e.5 * 10$. Si

//date (en heures)	CAT-Mag
58515434	40
58516316	45
58516672	50
58516969	40
58518121	45
58519744	40
58519766	40
58520294	40
58520658	40
58524360	45
58524721	40
58525664	45
58525864	40
58526634	45
58526853	45
58528064	45
58528265	40
58528421	45
58530008	50
58530514	45
58532236	45

FIG. 4.3 – Extrait de la séquence d'événements construite à partir des catalogues sismiques de l'ANSS.

$d \geq 75$, alors *CAT-Mag*$= (e+1)*10$. Enfin, si $d < 25$ alors *CAT-Mag*$= e*10$. Il est à noter qu'une discrétisation plus fine, outre la dispersion provoquée au niveau des événements, n'est pas envisageable car les intervalles qui seraient définis seraient plus étroits que les niveaux d'erreur associés aux mesures de la magnitude.

Quant à la date du tremblement de terre, celle-ci a été encodée en heures juliennes (nombre d'heures écoulées depuis le 1^{er} janvier 4713 Avant J-C). Ce codage permet d'avoir une résolution suffisamment fine, et ce en particulier pour prendre en compte de façon correcte les crises sismiques lors desquelles la fréquence des tremblements de terre est élevée.

Ainsi, nous avons constitué une séquence d'événements, basée sur 11 types d'événements, et contenant 7524 événements répartis sur 339284 unités temporelles (une unité temporelle = une heure). Un extrait de cette séquence est donné dans la figure 4.3.

4.2.4 Résultats

La séquence dont nous venons de décrire la construction dans la section 4.2.3 constitue le jeu de données sur lequel ont été réalisées les différentes extractions présentées dans cette section.

Le but de ces expériences est de trouver des relations impliquant des tremblements de terre à très forte magnitude (supérieure ou égale à 6) qui peuvent être considérés comme des chocs initiaux et non comme des répliques à des chocs initiaux. Or, le jeu de données dont nous disposons ne contient que 26 tremblements de terre de magnitude supérieure ou égale à 6. Par conséquent, et afin d'obtenir des règles d'épisodes contenant des types d'événements correspondant à des séismes de magnitude supérieure ou égale à 6, le seuil de support utilisé devra être très

faible. Celui-ci a été fixé à 5.

Lors de la première extraction, les autres paramètres ont été fixés comme suit : la confiance minimale à 0.9, la taux de décroissance à 15%, le gap maximum à 240 heures, et la taille maximum des épisodes extraits à 4. Nous avons également utilisé la mesure du lift et imposé que celui-ci soit supérieur ou égal à 20. 3915 règles se sont révélées fréquentes, dont 21 étaient également confiantes (pour au moins une taille de fenêtre). Finalement 16 FLM-règles pour lesquelles le lift est supérieur ou égal à 20 ont été sélectionnées. Parmi celles-ci, nous avons relevé la règle suivante :

$$60 \rightarrow 50 \Rightarrow 45 : w = 33 : cw = 1 : sw = 5$$

La lecture de cette règle s'effectue comme cela est précisé dans la section 4.1.3. Pour ce qui est des attributs, ceux-ci sont directement lisibles comme étant la magnitude du tremblement de terre multipliée par 10.

Cette règle est doublement intéressante. Tout d'abord, elle exhibe une dépendance qui peut se replacer dans la cadre d'une connaissance disponible, laquelle peut se ramener à la relation dite d'*aftershocks* ou de *répliques*. Celle-ci dit qu'un fort tremblement de terre, considéré comme un choc initial, peut provoquer, dans des espaces et des temps relativement proches, d'autres tremblements de terre dont la magnitude va généralement en décroissant avec le temps. C'est ce qu'instancie la règle trouvée. L'information nouvelle apportée par cette extraction réside dans le champ w qui établit que la fenêtre optimale est de 33 heures. Pour cette fenêtre, la confiance est de 1, ce qui est très fort, mais doit être relativisé par rapport au support qui lui est faible. En revanche, le fait qu'un maximum local de confiance soit trouvé est très intéressant, et ce d'autant plus que le support de la conclusion de la règle apparaît 703 fois dans le jeu de données. En effet, ce nombre d'apparitions peut laisser supposer que, plus la fenêtre considérée est large, plus la probabilité d'apparition de la conclusion de la règle est forte ; ce qui n'est clairement pas le cas au vu des résultats obtenus avec *WinMiner*.

Une autre règle intéressante trouvée parmi par les 16 FLM-règles renvoyées par *WinMiner* est la suivante :

$$50 \rightarrow 55 \rightarrow 55 \Rightarrow 55 : w = 33 : cw = 1 : sw = 5$$

Cette règle ne traduit pas une connaissance établie pour la zone considérée, mais en suggère une nouvelle qui signifie que des tremblement de terre du même ordre de magnitude peuvent être reliés entre eux. Il est d'ailleurs intéressant de noter que la fenêtre optimale est également de 33 heures dans ce cas et que la confiance est également maximale. Même si le support est faible, cette relation peut être considérée comme forte car sa confiance est de 1 alors que la conclusion de la règle n'apparaît que 68 fois dans la séquence d'événements, ce qui laisse une probabilité très faible à celle-ci d'apparaître à une unité temporelle donnée. D'ailleurs, la valeur de lift pour cette règle est très élevée puisque celui-ci vaut

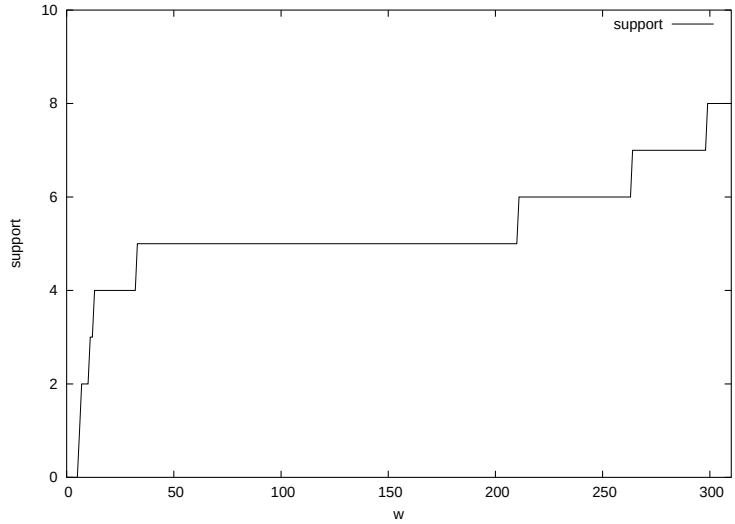


FIG. 4.4 – Support de la règle $50 \rightarrow 55 \rightarrow 55 \Rightarrow 55$ en fonction de w.

215. Afin de vérifier la force de cette règle, nous avons procédé à une deuxième extraction en gardant toutes les valeurs des paramètres précédemment fixées, exceptée celle du taux de décroissance que nous avons faite évoluer de 15% à 50%. Cette fois-ci, une seule règle est sélectionnée, celle dont nous voulons justement tester la force. Autrement dit, le maximum local de confiance est, dans le cas de cette règle, très prononcé.

Cette règle s'étant révélée potentiellement porteuse d'une nouvelle connaissance, nous l'avons étudiée de plus près en produisant, après une légère modification de l'algorithme *WinMiner*, les courbes de support (figure 4.4), de confiance (figure 4.5) et de lift (figure 4.6) en fonction des tailles de fenêtre. Pour chacune de ces courbes, nous avons borné l'axe des abscisses de façon à pouvoir observer l'essentiel des particularités portées par ces courbes. Ces courbes offrent ainsi la possibilité à l'expert d'observer les phénomènes de variation des diverses propriétés de la règle sélectionnée (support, confiance et lift) en fonction des différentes tailles de fenêtre.

Ces courbes sont à comparer avec les courbes obtenues dans le cas de règles fréquentes, confiantes et n'ayant pas de fenêtre optimale. Par exemple, si l'on considère la règle $25 \rightarrow 25 \Rightarrow 25$, on obtient les courbes de support (figure 4.7), et de confiance (figure 4.8) ;Ces courbes montrent clairement que la règle $25 \rightarrow 25 \Rightarrow 25$ (qui est fréquente et confiante pour certaines tailles de fenêtre) ne présente aucun comportement particulier dans le temps si ce n'est celui d'être plus prononcée lorsque la fenêtre de temps considérée augmente (i.e. le support et la confiance de la règle sont deux des fonctions croissantes des tailles de fenêtre), ce qui ne

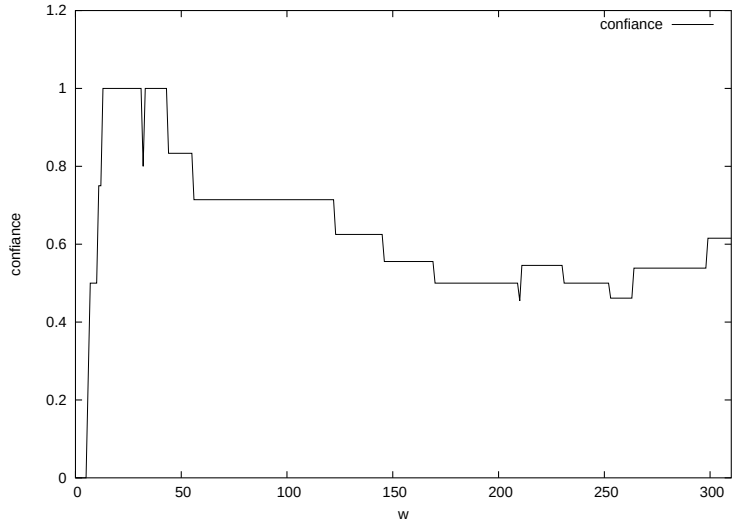


FIG. 4.5 – Confiance de la règle $50 \rightarrow 55 \rightarrow 55 \Rightarrow 55$ en fonction de w.

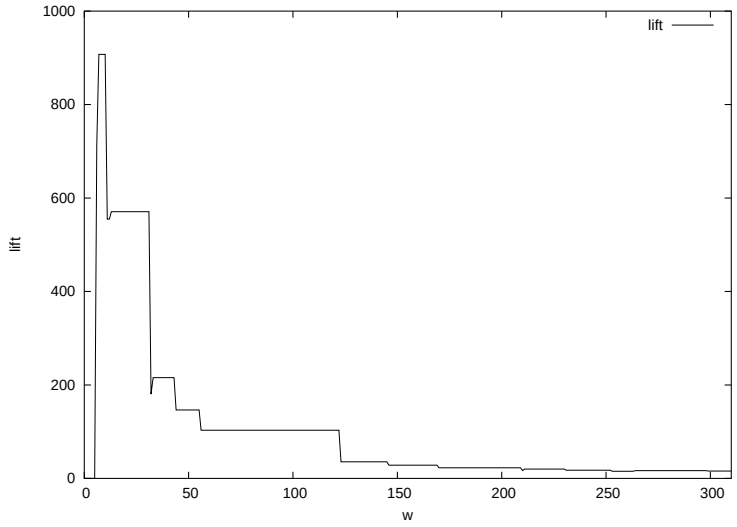


FIG. 4.6 – Lift de la règle $50 \rightarrow 55 \rightarrow 55 \Rightarrow 55$ en fonction de w.

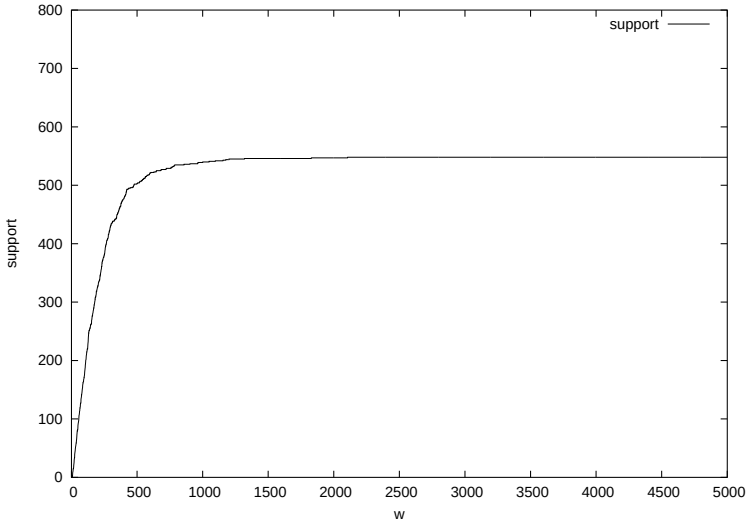


FIG. 4.7 – Support de la règle $25 \rightarrow 25 \Rightarrow 25$ en fonction de w.

présente en soit qu'un intérêt somme toute très faible. En effet, plus l'on considère des intervalles de temps importants et plus il est possible de mettre en relation des événements qui peuvent n'avoir aucun rapport réel entre eux. Or, on constate que, quelles que soient les expériences menées, ce type de comportement est largement plus répandu que celui exhibé par les FLM-règles trouvées, ce qui semble indiquer qu'en pratique, sur des données issues de catalogues sismiques, cette notion de FLM-règle est un outil *intéressant*.

4.2.5 Volume des résultats

Afin de mettre en perspective le volume des résultats obtenus par rapport aux nombres de règles considérées par *WinMiner*, et par rapport aux nombres de règles fréquentes et confiantes qu'il serait possible de fournir à l'utilisateur, nous avons procédé à de nouvelles expériences pour lesquelles une nouvelle discrétisation a été appliquée au jeu de données précédemment présenté. Cette fois-ci, la zone concernée est située entre 0° et 45° de latitude sud et 60° et 90° de longitude ouest. La période temporelle couverte est identique à celle utilisée auparavant (du 1^{er} janvier 1960 au 1^{er} janvier 2002). La discrétisation employée pour ces expériences revient à définir une grille géographique dont chaque cellule est associée à un type d'événement. Dans ce cas précis, le pas de la grille utilisé est de 2 degrés, à la fois en latitude et en longitude. Puis, chaque tremblement de terre est associé à un type d'événement e, lequel correspond à la cellule géographique

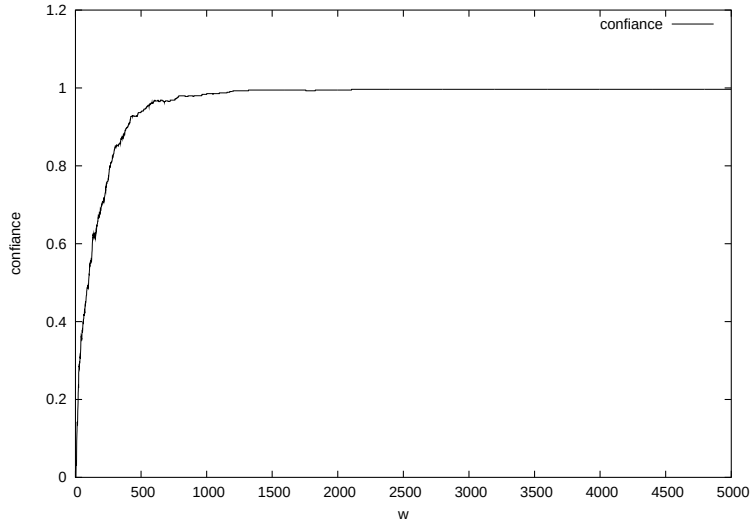


FIG. 4.8 – Confiance de la règle $25 \rightarrow 25 \Rightarrow 25$ en fonction de w.

concernée par le tremblement de terre (dans ce cas l'information sur la magnitude n'est pas conservée). La séquence d'événements est alors construite à partir des paires (*date du tremblement de terre*, e), où la date du tremblement de terre est codée en jours juliens. La séquence issue de ce pré-traitement est construite sur une base de 368 types d'événements. Elle contient 3509 événements répartis sur 14504 unités temporelles. Chaque unité temporelle correspond à 24 heures et l'ensemble de la séquence couvre une période d'une quarantaine d'années. Pour les expériences menées sur cette séquence d'événements, la valeur du support minimal a été fixée à 10, la valeur de confiance minimale à 0.9 et la valeur du taux de décroissance à 30%. Puis les différentes expériences ont été réalisées en faisant varier la valeur de contrainte de gap maximum entre 100 et 140 unités temporelles.

Pour chaque valeur de la contrainte de gap maximum, la courbe de la figure 4.9 présente :

- le nombre de règles fréquentes considérées par *WinMiner* (une règle d'épisodes r est fréquente s'il existe une largeur de fenêtre w telle que r soit fréquente pour w),
- le nombre de règles fréquentes et confiantes considérées par *WinMiner* (une règle d'épisodes r est fréquente et confiante s'il existe une largeur de fenêtre w telle que r soit fréquente et confiante pour w),
- le nombre de FLM-règles.

Les courbes de la figure 4.9 permettent de constater que le nombre de règles

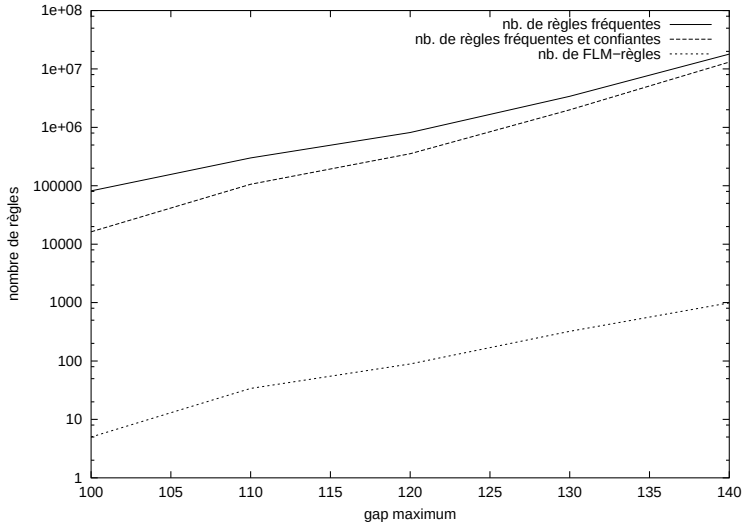


FIG. 4.9 – Volume des résultats.

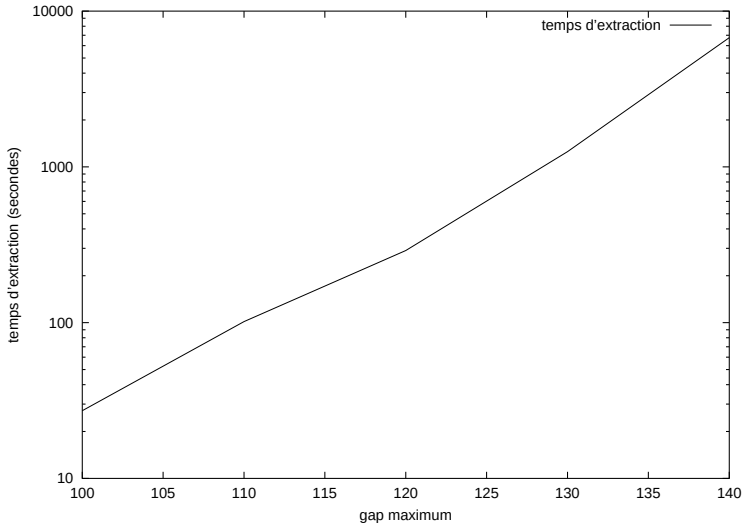


FIG. 4.10 – Temps d'extraction.

fréquentes et confiantes est, dans tous les cas, trop important pour que celles-ci puissent être analysées par un expert du domaine, tandis que le nombre de FLM-règles, bien plus petit, reste à la portée des capacités d'analyse de l'expert.

Un autre aspect de ces dernières expériences est détaillé au travers de la courbe de la figure 4.10. Celle-ci permet de constater que, pour les différentes valeurs de gap max comprises entre 100 et 140 unités temporelles, les temps d'extraction sont raisonnables au vu de l'étendue des espaces de recherche considérés par *WinMiner*. Ainsi, pour une contrainte $gapmax = 140$, et alors que plus de 17000000 de règles fréquentes et confiantes doivent être considérées par *WinMiner*, le temps d'extraction ne dépasse pas les 7000 secondes.

4.3 Bilan

Que ce soit sur un jeu de données médicales ou sur un catalogue de tremblements de terre, la recherche de FLM-règles, mise en oeuvre dans l'algorithme *WinMiner*, nous a permis de trouver des dépendances exprimant une connaissance généralement acceptée tout en apportant une information nouvelle et importante, à savoir les fenêtres de temps optimales de ces dépendances. De plus, pour ce qui est des catalogues de tremblements de terre, nous avons mis à jour des relations jugées comme nouvelles et intéressantes par les sismologues. Afin d'être validées, ces relations doivent être maintenant vérifiées par les experts du domaine, à l'aide de modèles statistiques et/ou géologiques. En l'occurrence, il s'agit d'un travail de longue haleine, mais qui néanmoins se base sur une suggestion de dépendances potentiellement intéressantes entre les données. L'ensemble de ces résultats est d'autant plus encourageant qu'aucune FLM-règle n'est extraite de jeux de données synthétiques aléatoires (cf. section 3.3.3). Enfin, on relèvera que le temps d'extraction et le volume des résultats fournis à l'utilisateur restent raisonnables.

Chapitre 5

Conclusion et Perspectives

Les travaux de recherche présentés dans ce mémoire concernent le processus d'Extraction de Connaissances à partir de Données (ECD), et portent plus précisément sur l'extraction de règles d'épisodes dans de longues séquences d'événements. Au sein du processus ECD (cf. chapitre 1), la phase de fouilles de données est la phase qui est essentiellement concernée par les propositions avancées dans ce mémoire et synthétisées dans [MR04]. La phase d'interprétation des résultats n'a pas été pour autant négligée. En effet, la définition même des motifs recherchés ainsi que la nouvelle mesure d'intérêt proposée ont été envisagées de sorte à faciliter l'analyse, par l'expert du domaine, des résultats issus de la phase de fouille de données. Les domaines d'application peuvent être très variés ainsi qu'en témoigne le chapitre 4, lequel présente une application à des données médicales (dans le contexte du Discovery Challenge ECML/PKDD'04 [MLLR04]) ainsi qu'une application à des données sismiques. C'est à partir des besoins en fouille de données exprimés par les géophysiciens dans le cadre du projet européen AEGIS (IST-2000-26450), auquel nous participons ([Még02, Még03]), que s'est organisé notre travail et qu'ont pris forme les propositions présentées dans ce mémoire. Par ailleurs, notre réflexion s'est également nourrie d'autres collaborations portant notamment sur l'extraction de motifs dans les bases de séquences ([LMR04]) et sur la représentation synthétique de grandes collections de règles d'association ([BDMR02, BDMR03, BDMR04]).

En ce qui concerne les données sismiques, un des aspects particuliers était que les experts ne pouvaient pas en général fixer à priori les durées caractéristiques des dépendances pouvant exister. L'étude des différentes techniques d'extraction de règles d'épisodes nous a alors permis de constater que celles-ci ne pouvaient pas répondre directement à ce besoin à partir du moment où elles imposent, que ce soit dans le cas de la méthode MINEPI ([MT96, MTV97]) ou dans le cas de la méthode WINEPI ([MTV95, MTV97]), une taille de fenêtre maximale pour l'analyse des jeux de données. Afin d'imaginer une solution, nous nous sommes intéressés à un domaine de recherche considéré comme proche du notre, i.e. le

domaine des bases de séquences, domaine pour lequel un nombre plus important de contributions a été produit ces dix dernières années. Bien que le contexte soit différent, l'analyse des solutions proposées a permis de constater qu'un grand nombre de contraintes pouvaient être prises en compte. Que ce soit dans ce domaine ou dans celui de l'analyse de longues séquences d'événements, la gestion active de contraintes est quasi-obligatoire si l'on veut pouvoir mener, dans des conditions réalistes de ressources de calcul, des tâches de fouille de données sur des jeux de données non triviaux. Ainsi, parmi les contraintes gérées de façon active par les algorithmes permettant l'analyse de bases de séquences, avons-nous observé que la contrainte de gap maximum était parmi les contraintes les plus répandues (e.g., [SA96, Zak00, PHW02]) alors que celle-ci est absente des solutions proposées pour l'analyse de longues séquences d'événements. Or, l'intérêt de cette contrainte est de ne pas imposer une fenêtre maximale pour tous les motifs extraits, mais plutôt de faire varier la contrainte de fenêtre maximale en fonction du nombre d'éléments composant les motifs recherchés en imposant l'écart maximum entre deux événements consécutifs du motif. Une tentative de gestion active d'une contrainte similaire à la contrainte de gap maximum pour l'analyse de longues séquences d'événements a bien été proposée [CG03], mais celle-ci est incomplète comme nous l'avons montré dans la section 3.2.3.

Nous avons donc proposé (cf. chapitre 3) un algorithme, *WinMiner* (inspiré par les algorithmes proposés dans [Zak00, Zak01] et [MT96, MTV97]) permettant d'extraire des règles d'épisodes sous la contrainte de gap maximum, et montré sa justesse et sa complétude. En outre, *WinMiner* permet également la recherche des fenêtres optimales des phénomènes exhibés à partir des données. Plus particulièrement, *WinMiner* extrait les FLM-règles (cf. chapitre 3), c'est à dire les règles d'épisodes pour lesquelles il existe une fenêtre temporelle optimale. Cela permet, après un premier filtrage sur la fréquence et la confiance des règles, de ne sélectionner qu'un sous-ensemble de l'ensemble des règles fréquentes et confiantes, ce qui facilite le travail de l'expert du domaine lors de la phase d'interprétation des résultats issus de la phase de fouille de données. À ce niveau, on rejoint un autre des besoins exprimés par les géophysiciens, à savoir d'obtenir, en sortie de processus, une information qui ne dépasse pas la capacité d'analyse et d'interprétation de l'expert (cf. section 4.2.5). Enfin, nous avons également proposé une mesure d'intérêt, le lift (cf. chapitre 3), qui elle aussi cadre avec ce dernier besoin, en ceci qu'elle permet de sélectionner les règles dont on peut considérer que leur confiance, compte tenu de la fréquence d'apparition des différents événements les composant, est sensiblement plus élevée que la confiance à laquelle nous aurions pu nous attendre en supposant une répartition uniforme et indépendante des événements.

Les différentes expériences menées à l'aide d'une implémentation prototype de *WinMiner* nous ont permis de tester l'approche proposée. Tout d'abord, des expériences sur des jeux de données aléatoires (cf. chapitre 3) nous ont permis

de constater qu'aucune FLM-règle n'était extraite (pour des seuils raisonnables), ce qui est intéressant car, a priori, un jeu de données aléatoires contient peu de phénomènes particuliers à exhiber. Puis, nous nous sommes intéressés à des jeux de données réels (cf. chapitre 4). Les résultats sont encourageants, car en qui concerne l'application aux données médicales, nous avons retrouvé des phénomènes connus pour lesquels nous avons apporté une nouvelle information : les fenêtres optimales. Pour ce qui est de l'application aux données sismiques, celle-ci est également encourageante car nous avons pu isoler des phénomènes jugés comme potentiellement intéressants par les géophysiciens. L'application aux données sismiques nous a également permis de vérifier l'utilité du lift, lequel nous a particulièrement aidé à nous concentrer sur les règles pouvant être intéressantes.

Bien évidemment, les propositions avancées dans ce mémoire ne sont pas figées et diverses évolutions pourraient être considérées.

En ce qui concerne les motifs extraits, il serait par exemple possible d'envisager la gestion de règles d'épisodes comportant des événements simultanés. En effet, bien que les séquences d'événements traitées par *WinMiner* puissent comporter des événements dont les dates d'apparition sont identiques, les règles d'épisodes extraites ne peuvent pas comporter des événements considérés comme apparaissant simultanément. Or, cela permettrait de capturer tout une nouvelle catégorie de phénomènes. Plus précisément, cela permettrait de mieux analyser les données pour lesquelles beaucoup d'événements apparaissent à une même date. De même, il serait intéressant de considérer la définition et l'utilisation d'occurrences moins restrictives que les occurrences minimales, ce qui permettrait également de capturer d'autres phénomènes.

Un autre aspect des évolutions possibles a trait aux résultats actuellement fournis par *WinMiner*. Il serait par exemple possible de les préciser en introduisant une mesure statistique. Une mesure statistique telle que le χ^2 permettrait ainsi de compléter l'information apportée par le lift en proposant une mesure prenant notamment en compte la taille des échantillons. Cette possible utilisation reste néanmoins conditionnée aux propriétés calculatoires de la mesure choisie, i.e. est-il possible de calculer cette mesure à moindre coût ? Toujours dans l'idée d'augmenter la précision des résultats obtenus, l'extraction d'autres descripteurs permettant d'analyser les variations de confiance en fonction des tailles de fenêtre semble être intéressante. En effet, la notion de FLM-règle permet de se concentrer sur des règles d'épisodes pour chacune desquelles il existe une fenêtre temporelle telle que la mesure de confiance soit maximisée. Or, bien que cela soit l'un des premiers aspects auquel notre collaborations avec les géophysiciens nous a mené, cela ne constitue pas pour autant la seule information intéressante qu'il soit possible d'extraire à partir d'une courbe représentant la variation de confiance en fonction de temps. Ainsi, les FLM-règles peuvent, par exemple, être complétées

par leurs courbes de confiance. D'ailleurs, celles-ci ont pu être fournies (c.f. chapitre 4) lorsque les géophysiciens en ont fait la demande. De même, un minimum de confiance ou maximum global de confiance peuvent également être considérés comme intéressants. Inversement, il serait parfois utile d'obtenir une description plus synthétique des résultats. Par exemple, le calcul de la moyenne et de l'écart-type des fenêtres optimales semble être un outil pertinent, à la condition que les phénomènes contenus dans les données soient assez homogènes dans leur relation au temps, e.g., l'écart-type obtenu est faible.

En ce qui concerne les applications, de nombreuses autres applications pourraient être envisagées, telles que l'analyse de séquences ADN ou l'analyse de données boursières. De plus, le champ des applications possibles pourrait être élargi grâce à la combinaison de *WinMiner* avec d'autres techniques d'analyse des données. Par exemple, on pourrait utiliser les résultats d'une extraction pour une classification. Dans [KLT03], il est montré qu'une classification basée sur des motifs fréquents donne des résultats intéressants. Qu'en-est-il alors d'une classification basée sur des épisodes extraits sous une contrainte de gap maximum et basés sur la notion d'occurrence minimale ? Enfin, une adaptation des notions et des algorithmes présentés au contexte de l'extraction incrémentale semble une extension prometteuse. L'extraction incrémentale a pour but le calcul des résultats d'une fouille de données à coût réduit en utilisant les résultats obtenus lors de précédentes fouilles de données. Plusieurs contributions dans ce sens ont vu le jour, que ce soit dans le domaine de la recherche de règles d'association (e.g., [CHW96, TBAR97, DGLS02]) ou dans le domaine de la recherche de motifs séquentiels (e.g., [MPT00, PZOD99, LL98]). Ces approches présentent un potentiel de gain pour la plupart des processus de fouilles de données (dont ceux dans les domaines abordés au chapitre 4) qui sont très souvent des processus itératifs. De façon plus spécifique, leur adaptation pour le calcul de FLM-règles nous semble particulièrement intéressante pour les catalogues sismiques, car un même catalogue est en général utilisé par de nombreux géophysiciens et de plus, il peut être sujet à des mises à jour quotidiennes.

Bibliographie

[AAP01] Agarwal R., Aggarwal C., Prasad V. *A tree projection algorithm for finding frequent itemsets*. **In** : J. Parallel Distrib. Comput., vol. 61, n°3, 2001, pp. 350–371.

[AFGY02] Ayres J., Flannick J., Gehrke J., et al. *Sequential pattern mining using bitmap representation*. **In** : *Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Edmonton, Alberta, Canada, July 2002, pp. 429–435.

[AHKV97] Ahonen H., Heinonen O., Klemettinen M., et al. *Applying data mining techniques in text analysis*. Tech. Rep. C-1997-23, University of Helsinki, Department of Computer Science, March 1997. 12 pages.

[AMS+96] Agrawal R., Mannila H., Srikant R., et al. *Fast discovery of association rules*. **In** : Fayyad U., al., Eds., *Advances in Knowledge Discovery in Data Mining*. Menlo Park : AAAI Press, 1996, pp. 307–328.

[AOWK90] Anderson K., Odell P., Wilson P., et al. *Cardiovascular disease risk profiles*. **In** : Am Heart J, vol. 121 , 1990, pp. p. 312–318.

[AS94] Agrawal R., Srikant R. *Fast algorithms for mining association rules in large databases*. **In** : *Proc. of the 20th International Conference on Very Large Data Bases (VLDB'94)*. Santiago de Chile, Chile, 1994, pp. 487–499.

[AS95] Agrawal R., Srikant R. *Mining sequential patterns*. **In** : Yu P.S., Chen A.S.P., Eds., *Proc. of the 11th International Conference on Data Engineering (ICDE'95)*. IEEE Computer Society Press, Taipei, Taiwan, 1995, pp. 3–14.

[BA99] Bayardo R., Agrawal R. *Mining the most interesting rules*. **In** : *Proc. of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM Press, San Diego, USA, June 1999, pp. 145–154.

[BC96] Berndt D.J., Clifford J. *Finding patterns in time series : A dynamic programming approach*. **In** : Fayyad U., Piatetsky-Shapiro G., Smyth P., et al., Eds., *Advances in Knowledge Discovery and Data Mining*. AAAI Press, 1996, pp. 229–248.

[BDMR02] Bykowski A., Daurel T., Méger N., et al. *Association maps as integrity constraints in inductive databases*. **In** : *In Proc. of the First International Workshop on Knowledge Discovery in Inductive Databases (KDID'02)*. Helsinki, Finland, August 2002, pp. 61–75.

[BDMR03] Bykowski A., Daurel T., Méger N., et al. *Finding interesting association rules using confidence variations*. **In** : *Proc. of the International Conference on Computer, Communication and Control Technologies (CCCT03)*. Orlando, USA, August 2003. 12 pages.

[BDMR04] Bykowski A., Daurel T., Méger N., et al. *Integrity constraints over association rules*. **In** : Meo R., Lanzi P., Klemettinen M., Eds., *Database Support for Data Mining Applications*. Springer-Verlag, LNCS 2682, 2004, pp. 306–324.

[BSWJL98] Bettini C., Sean Wang X., Jajodia S., et al. *Discovering frequent event patterns with multiple granularities in time sequences*. **In** : Knowledge and Data Engineering, vol. 10, 2, 1998, pp. 222–237.

[BT01] Buhler J., Tompa M. *Finding motifs using random projections*. **In** : *Proc. of the 5th Annual Int. Conf. on Computational Biology (RECOMB'01)*. ACM Press, Montreal, Quebec, Canada, April 2001, pp. 69–76.

[CG03] Casas-Garriga G. *Discovering unbounded episodes in sequential data*. **In** : *Proc. of the Int. Conf. on Principles of Data Mining and Knowledge Discovery in Databases (PKDD'03)*. Springer-Verlag LNCS 2838, Cavtat-Dubrovnik, Croatia, September 2003, pp. 83–94.

[CHW96] Cheung D., Han J., Wong C. *Maintenance of discovered association rules in large databases : An incremental updating technique*. **In** : *Proc. of the International Conference on Data Engineering (ICDE'96)*. IEEE Computer Society, New Orleans, USA, March 1996, pp. 106–114.

[CMS97] Cooley R., Mobasher B., Srivastava J. *Web mining : Information and pattern discovery on the world wide web*. **In** : *Proc. of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97)*. Newport Beach, CA, USA, 1997, pp. 558–567.

[Coo00] Cooley R. *Web Usage Mining : Discovery and Application of Interesting Patterns from Web Data*. Ph.D. thesis, University of Minnesota, 2000. 170 pages.

[DGLS02] Diop C., Giacometti A., Laurent D., et al. *Composition of mining contexts for efficient extraction of association rules*. **In** : *Proc. of the 8th International Conference on Extending Database Technology (EDBT'02)*. Springer-Verlag LNCS 2287, Pragues, Czech Republic, March 2002, pp. 106–123.

[Fay96] Fayyad U. *Advances in Knowledge Discovery and Data Mining*. Menlo Park : AAAI Press, 1996. 611 pages.

[FGLS98] Fayet A., Giacometti A., Laurent D., et al. *Mining significant rules from databases*. **In** : Networking and Information Systems Journal, vol. 1, 1, 1998, pp. 653–682.

[FPSM91] Frawley W., Piatetsky-Shapiro G., Matheus C. *Knowledge discovery in databases : an overview*. **In** : Piatetsky-Shapiro G., Frawley W., Eds., *Knowledge in Discovery in Databases*. Menlo Park : AAAI Press, 1991, pp. 1–27.

[FPSS96a] Fayyad U., Piatetsky-Shapiro G., Smyth P. *The KDD process for extracting useful knowledge from volumes of data.*. **In** : Communications of the ACM, vol. 39, 11, nov. 1996, pp. 27 – 34.

[FPSS96b] Fayyad U., Piatetsky-Shapiro G., Smyth P. *Knowledge discovery and data mining : Towards a unifying framework*. **In** : *Proc. of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96)*. Portland, Oregon, August 1996, pp. 82–88.

[GGK01] Guralnik V., Garg N., Karypis G. *Parallel tree projection algorithm for sequences mining*. **In** : *Proc. of the European Conference on Parallel Processing*. 2001, pp. 310–320.

[GGP98] Guillaume S., Guillet F., Philippe J. *Improving the discovery of assocotion rules with intensity of implication*. **In** : *Proc. of the 2nd Europena Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'98)*. Springer-Verlag, LNAI, Nantes, France, Septembre 1998, pp. 318–327.

[GNPS03] Giannotti F., Nanni M., Pedreschi D., et al. *Webcat : Automatic categorization of web search results*. **In** : *(Proceedings of the 11th Italian Symposium on Advanced Database Systems SEBD'03)*. Rubettino Editore, Cetraro, Italy, June 2003, pp. 507–518.

[GRK99] Garofalakis M., Rastogi R., K. S. *Spirit : Sequential pattern mining with regular expression constraints*. **In** : *Proc. of the 25th International Conference on Very Large Databases (VLDB'99)*. Edinburgh, United Kingdom, September 1999, pp. 223–234.

[GRSS99] Garofalakis M., Rastogi R., Seshadri S., et al. *Data mining and the web : Past, present and future*. **In** : *Workshop on Web Information and Data Management*. 1999, pp. 43–47.

[HDSG97] Hill C., Doyon F., Sancho-Garnier H. *Epidémiologie des cancers*. Paris : Médecine-Sciences Flammarion, 1997. 111 pages.

[HDY99] Han J., Dong G., Yin Y. *Efficient mining of partial periodic patterns in time series database*. **In** : *Proc. of the 15th Int. Conf. on Data Engineering*. IEEE computer Society, Sydney, Australia, 1999, pp. 106–115.

[HHMAC+00] Han J. J. Pei, Han Mortazavi-Asl B., Chen Q., et al. *Freespan : Frequent pattern-projected sequential pattern mining*. **In** : *Proc. 2000 Int. Conf. Knowledge Discovery and Data Mining (KDD'00)*. Boston, MA, USA, August 2000, pp. 355–359.

[HKM+96] Hätönen K., Klemettinen M., Mannila H., et al. *Tasa : Telecomunications alarm sequence analyzer or : How to enjoy faults in your network*. **In** : *1996 IEEE Network Operations and Management Symposium (NOMS'96)*. Kyoto, Japan, April 1996, pp. 520–529.

[HPY00] Han J., Pei J., Yin Y. *Mining frequent patterns without candidate generation*. **In** : *Proceedings ACM SIGMOD 2000*. Dallas, USA, mai 2000, pp. 1–12.

[HTF01] Hastie T., Tibshirani T., Friedman J. *The Elements of Statistical Learning : Data Mining, Inference, and Prediction*. New York : Springer-Verlag, August 2001. 533 pages.

[IM96] Imielinski T., Mannila H. *A database perspective on knowledge discovery*. **In** : Communications of the ACM, vol. 39, 11, nov. 1996, pp. 58–64.

[JKK99] Joshi M.V., Karypis G., Kumar V. *A universal formulation of sequential patterns*. Tech. rep., Department of Computer Science, University of Minnesota, 4-192 EECS Building, 200 Union Street SE, Minneapolis, MN 55455-0159 USA, May 1999. 17 pages.

[KB00] Kosala R., Blockeel H. *Web mining research : A survey*. **In** : SIGKDD : SIGKDD Explorations : Newsletter of the Special Interest Group (SIG) on Knowledge Discovery and Data Mining, ACM, vol. 2, 2000. 15 pages.

[Kle99] Klemettinen M. *A Knowledge Discovery Methodology for Telecommunication Network Alarm Databases*. Ph.D. thesis, University of Helsinki, Finland, 1999. 146 pages.

[KLT03] Keogh E., Lin J., Truppel W. *Clustering of time series subsequences is meaningless : Implications for previous and future research*. **In** : *Proc. of the 3rd IEEE International Conference on*

Data Mining (ICDM 2003). IEEE Computer Society, Melbourne, Florida, USA, December 2003, pp. 115–122.

[LAS97] Lent B., Agrawal R., Srikant R. *Discovering trends in text databases*. **In** : *Proc. of the 3rd International Conference on Knowledge Discovery in Databases and Data Mining (KDD'97)*. AAAI Press, Newport beach, CA, USA, August 1997, pp. 227–230.

[LB02a] Leleu M., Boulicaut J.F. *Signature de situations boursières représentées par des séquences d'événements*. **In** : *Actes des Journées Francophones d'Extraction et de Gestion des Connaissances (EGC'02)*. Hermes, Montpellier, France, Januray 2002, pp. 89–100.

[LB02b] Leleu M., Boulicaut J.F. *Signing stock market situations by means of characteristic sequential patterns*. **In** : *Proc. of the 3rd International Conference on Data Mining (DM'02)*. WIT Press, Bologna, Italy, September 2002, pp. 655–664.

[Lel04] Leleu M. *Extraction de Motifs Séquentiels Sous Contraintes dans les Données Contenant des Répétitions Consécutives*. Ph.D. thesis, Institut National des Sciences Appliquées de Lyon, 2004. 136 pages.

[LL98] Lin M., Lee S. *Incremental update on sequential patterns in large databases*. **In** : *Proc. of the International Conference on Tools for Artificial Intelligence Conference (ICTAI'98)*. May 1998, pp. 24–31.

[LMR04] Leleu M., Méger N., Rigotti C. *Extraction de motifs séquentiels fréquents sous contraintes dans des données contenant des répétitions*. **In** : Revue Ingénierie des Systèmes d'Information (ISI), vol. 9, 1, 2004, pp. 133–159.

[LRBE03a] Leleu M., Rigotti C., Boulicaut J.F., et al. *Constrained-based mining of sequential patterns over datasets with consecutive repetitions*. **In** : *Proc.2003 Int. Conf. on Principles of Data Mining and Knowledge Discovery in Databases (PKDD'03)*. Springer-Verlag LNCS 2838, Cavtat-Dubrovnik, Croatia, September 2003, pp. 303–314.

[LRBE03b] Leleu M., Rigotti C., Boulicaut J.F., et al. *Go-spade : Mining sequential patterns over datasets with consecutive repetitions*. **In** : *Proc.2003 Int. Conf. Machine Learning and Data Mining (MLDM'03)*. Springer-Verlag LNCS 2734, Leipsig, Germany, July 2003, pp. 293–306.

[MA01] Mortazavi-Asl B. *Discovering and Mining User Web-Page Traversal Patterns*. Master's thesis, Simon Fraser University, 2001. 93 pages.

[Mas02] Masseglia F. *Algorithmes et applications pour l'extraction de motifs séquentiels dans le domaine de la fouille de données : de l'incrémental au temps réel*. Ph.D. thesis, Université de Versailles St Quentin, France, 2002.

[MCP98] Masseglia F., Cathala F., Poncelet P. *The PSP approach for mining sequential patterns*. **In** : *Proc. of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery in Databases (PKDD'98)*, vol. 1510. LNAI, Springer Verlag, Nantes, France, September 1998, pp. 176–184.

[Még02] Méger N. *Deliverable d4.1 : State of the art on seismic data management systems*. Tech. rep., Laboratoire LIRIS, INSA de Lyon, 20, avenue Albert Einstein, 69621 Villeurbanne cedex, France, January 2002. 27 pages.

[Még03] Méger N. *Présentation invité : Data mining applied to seismic data*. **In** : *AEGIS workshop on Information Science Science and Technology Earth Sciences*. Fréjus, France, February 2003. (exposé).

[MLLR04] Méger N., Leschi C., Lucas N., et al. *Mining episode rules in stulong dataset*. **In** : *Proc. of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'04) Discovery Challenge*. Springer-Verlag, LNAI, Pise, Italie, Septembre 2004. 12 pages.

[MPC99] Masseglia F., Poncelet P., Cicchetti R. *An efficient algorithm for web usage mining*. **In** : Networking and Information Systems Journal, vol. 2, n°5-6, 1999, pp. 571–603.

[MPT00] Masseglia F., Poncelet P., Teisseire M. *Incremental mining of sequential patterns in large databases*. **In** : *Proc. of 16èmes Journées Bases de Données Avancées (BDA'00)*. Blois, France, October 2000, pp. 345–366.

[MR04] Méger N., Rigotti C. *Constraint-based mining of episode rules and optimal window sizes*. **In** : *Proc. of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'04)*. Springer-Verlag, LNAI, Pise, Italie, Septembre 2004, pp. 313–324.

[MT96] Mannila H., Toivonen H. *Discovery generalized episodes using minimal occurrences*. **In** : *Proc. of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96)*. Portland, Oregon, August 1996, pp. 146–151.

[MT97] Mannila H., Toivonen H. *Levelwise search and borders of theories in knowledge discovery*. **In** : Data Mining and Knowledge Discovery, vol. 1, 3, 1997, pp. 241–258.

[MTV95] Mannila H., Toivonen H., Verkamo I. *Discovering frequent episodes in sequences*. **In** : *Proc. of the 1st International Conference on Knowledge Discovery and Data Mining (KDD'95)*. AAAI Press, Montreal, Canada, August 1995, pp. 210–215.

[MTV97] Mannila H., Toivonen H., Verkamo A. *Discovery of frequent episodes in event sequences*. **In** : Data Mining and Knowledge Discovery, vol. 1, 3, November 1997, pp. 259–298.

[PCGS02] Pisanti N., Crochemore M., Grossi R., et al. *A basis for repeated motifs in pattern discovery and text mining*. Internal report, University of Marne-la-Vallée, July 2002. 11 pages.

[PHMAP01] Pei J., Han B., Mortazavi-Asl B., et al. *Prefixspan : Mining sequential patterns efficiently by prefix-projected pattern growth*. **In** : *Proc. of the 17th International Conference on Data Engineering (ICDE'01)*. 2001, pp. 215–226.

[PHW02] Pei J., Han B., Wang W. *Mining sequential patterns with constraints in large databases*. **In** : *Proc. of the 11th International Conference on Information and Knowledge Management (CIKM'02)*. McLean, VA, USA, November 2002, pp. 18–25.

[Pov99] Povinelli R. *Time Series Data Mining : Identifying Temporal Patterns for Characterization and Prediction of Time Series Events*. Ph.D. thesis, Marquette University, 1999. 193 pages.

[PS00] Pevzner P., Sze S.H. *Combinatorial approaches to finding subtle signals in dna sequences*. **In** : *Proc. of the 8th Int. Conf. on Intelligent Systems for Molecular Biology*. AAAI Press, San Diego, CA, USA, August 2000, pp. 269–278.

[PZOD99] Parthasarathy S., Zaki M., Ogihara M., et al. *Incremental and interactive sequence mining*. **In** : *Proc. of the 8th International Conference on Information and Knowledge Management (CIKM'99)*. Kansas City, MO, USA, November 1999, pp. 251–258.

[SA96] Srikant R., Agrawal R. *Mining sequential patterns : Generalizations and performance improvements*. **In** : *Proc. of the 5th International Conference on Extending Database Technology (EDBT'96)*. Avignon, France, September 1996, pp. 3–17.

[SBM98] Silverstein C., Brin S., Motwani R. *Beyond market baskets : Generalizing association rules to dependence rules*. **In** : Data Mining and Knowledge Discovery, vol. 2, 1, January 1998, pp. 39–68.

[SK02] Seno M., Karypis G. *Slpminer : An algorithm for finding frequent sequential patterns using length-decreasing support constraint*. **In** : *Proc. of the 2002 IEEE International Conference on Data Mining (ICDM'02)*. IEEE Computer Society, Maebashi City, Japan, December 2002, pp. 418–425.

[TBAR97] Thomas S., Bodagala S., Alsabti K., et al. *An efficient algorithm for the incremental updation of association rules in large databases*. **In** : *Proc. of the International Conference on Data Mining and Knowledge Discovery (KDD'97)*. AAAI Press, Newport Beach, USA, August 1997, pp. 263–266.

[Zak98] Zaki M. *Efficient enumeration of frequent sequences*. **In** : *Proc. of the 7th International Conference on Information and Knowledge Management (CIKM'98)*. November 1998, pp. 68–75.

[Zak00] Zaki M. *Sequence mining in categorical domains : incorporating constraints*. **In** : *Proc. of the 9th International Conference on Information and Knowledge Management (CIKM'00)*. Washington, DC, USA, November 2000, pp. 422–429.

[Zak01] Zaki M. *Spade : an efficient algorithm for mining frequent sequences*. **In** : Machine Learning, Special issue on Unsupervised Learning, vol. 42, 1/2, Jan/Feb 2001, pp. 31–60.

[ZLO98] Zaki M., Lesh N., Ogihara M. *Planmine : Sequence mining for plan failures*. **In** : Agrawal R., Stolorz P., Piatetsky-Shapiro G., Eds., *Proc. of the 4th Int. Conf. on Knowledge Discovery and Data Mining (KDD'98)*. ACM Press, New York, NY, 1998, pp. 369–373.