Extraction de séquences fréquentes : des données numériques aux valeurs manquantes

Céline Fiot

Université Montpellier II
— Sciences et Techniques du Languedoc —

Thèse

pour obtenir le grade de
Docteur de l'Université Montpellier II

Discipline : Informatique
Spécialité Doctorale : *Informatique*
Ecole Doctorale : *Information, Structure, Systèmes*

présentée et soutenue publiquement par

Céline Fiot
le 28 septembre 2007

Extraction de séquences fréquentes : des données numériques aux valeurs manquantes

Jury

Bernadette Bouchon-Meunier , Directeur de Recherche, Université Paris 6, ... Présidente
Bruno Crémilleux , Professeur, Université de Caen, ... Rapporteur
Eyke Hüllermeier , Professeur, Philipps-Universität Marburg, Allemagne, ... Rapporteur
Pascal Poncelet , Professeur, Ecole des Mines d'Alès, ... Examinateur
Anne Laurent , Maître de Conférence, Université Montpellier II, ... Co-directrice de Thèse
Maguelonne Teisseire , Maître de Conférence, Université Montpellier II, ... Directrice de Thèse

La réalisation d'une thèse de doctorat est un parcours initiatique. Je remercie ma directrice de thèse Maguelonne Teisseire, pour m'avoir donné l'occasion de vivre cette expérience et pour avoir su me guider au fil de ces étapes qui ont conduit à l'accomplissement de mon travail mais également à mon épanouissement professionnel. Avec ma co-directrice, Anne Laurent, elles ont su trouver les mots pour me faire avancer et m'ont donné tous les moyens de devenir une chercheuse passionnée. Pour cela et pour tout ce qu'elles m'ont apporté, je leur adresse ma plus profonde gratitude.

Je souhaite également remercier les rapporteurs du présent manuscrit, le Pr. Bruno Crémilleux ainsi que le Pr. Eyke Hüllermeier, pour leur lecture approfondie de mon travail ainsi que pour leurs questions et remarques des plus enrichissantes. Je remercie également le Pr. Bernadette Bouchon-Meunier et le Pr. Pascal Poncelet pour avoir accepté de prendre part à l'évaluation de ce travail par leur présence lors de ma soutenance.

Pour leurs encouragements et discussions avisées, je remercie chaleureusement le Pr. Jocelyne Nanard et le Pr. Stefano Cerri. Je suis également reconnaissante à ces personnes, qui, un jour, par un commentaire, une petite phrase ou une intervention m'ont permis de découvrir et d'apprécier l'informatique puis l'extraction de connaissances et de rencontrer celle qui devint par la suite ma directrice de thèse. Pour cela, merci à Narendra Jussien, Philippe David et Sébastien Druon.

Je remercie également mes camarades qui ont partagé avec moi certains moments difficiles, Simon, Stéphane, Anselme, Emmanuel et Federico, ainsi que Gilles, Valentine, Jean-Christophe et Patrice, et j'adresse mes encouragements à ceux dont le chemin de la thèse n'est pas encore terminé : José, Greg, Chedy, Marc, Lisa et Cécile. Merci aussi à Sandra et Paola pour leur regard neuf et leurs remarques constructives lors de la préparation de ma soutenance.

Enfin, j'adresse une pensée particulière à mon grand-père, pour n'avoir jamais freiné cette curiosité qui, alors que j'étais enfant conduisait à d'innombrables questions auxquelles j'obtenais toujours des réponses. Cette envie d'apprendre et de découvrir, ainsi que cette rigueur, nécessaire à l'accomplissement d'un tel travail, je les dois également à mes parents. Je les remercie de m'avoir donné les moyens de réussir, de m'avoir soutenue dans mes choix depuis bientôt dix ans et de m'avoir accompagnée à chaque étape.

Mes remerciements vont également à ma soeur, Isabelle, ainsi qu'à Jérôme, pour leur patience, leur soutien et leurs nombreux conseils, en toute circonstance.

Aux personnes qui ont contribué à faire de moi
celle que je suis aujourd'hui.

$\sim$

Le but n'est pas le but, c'est la voie.
Lao Tseu

Sommaire

Les technologies de l'information offrent désormais de nombreux moyens pour rassembler et stocker efficacement de grandes quantités de données, d'origines et de formats très diversifiés. Les masses d'informations ainsi emmagasinées rendent impossibles l'analyse, le résumé ou l'extraction manuelle de ces potentielles connaissances.
C'est pourquoi, au carrefour de thèmes de recherche variés tels que les statistiques, l'intelligence artificielle, l'interaction homme-machine ou encore les bases de données, les chercheurs de ces 20 dernières années, motivés par des problèmes d'aide à la décision, se sont intéressés à la conception et au développement d'outils permettant d'extraire automatiquement de la connaissance de ces données volumineuses. Ces thèmes de recherche, regroupés sous l'appellation "Extraction de Connaissances dans les grandes bases de Données" (ECD) ou "Knowledge Discovery in Databases" (KDD), avaient initialement pour but la gestion de données volumineuses. Ils ont évolué et visent désormais à tenir compte de l'hétérogénéité de ces données, de leur format multiple, souvent complexe et de leur qualité variable.

Ainsi, afin de répondre aux diverses problématiques posées par les applications industrielles de l'extraction de connaissances et plus particulièrement de la fouille de données, de nombreux algorithmes ont été développés, proposant chacun divers types de connaissances. Les travaux que nous présentons dans ce mémoire ont pour but l'analyse de données particulières, semi-structurées. Ces données, regroupées sous la forme de séquences ordonnées, sont appelées séquences de données. Elles peuvent être issues de domaine aussi variés que des corpus de textes, des enregistrements de requêtes sur des sites web ou encore des séquences de protéines ou d'ADN.
Peu de techniques permettent de traiter l'information temporelle contenue dans de telles bases. L'une d'elles, l'extraction de motifs séquentiels, permet de décrire ces données grâce à la mise en évidence de séquences fréquentes maximales, caractéristiques de la base. Néanmoins, l'utilisation des motifs séquentiels n'étaient jusqu'alors possible que sur des bases de données symboliques et parfaites, c'est-à-dire des bases ne contenant que des informations binaires ou pouvant être traitées comme telles et ne contenant aucun enregistrement incomplet.
Par ailleurs, le but de l'analyse des données peut être plus précis que l'extraction de comportements fréquents. Il peut notamment s'agir de caractériser certains schémas faisant intervenir des contraintes ou respectant certaines propriétés. Cependant, la spécification de telles propriétés et l'utilisation des algorithmes actuels requièrent le plus souvent une bonne connaissance préalable des données par l'utilisateur, ce qui n'est pas toujours le cas.

Dans ce chapitre introductif, nous rappelons les différentes étapes du processus d'ECD dans la section 1, en insistant plus particulièrement sur l'étape de fouille de données. Nous verrons qu'il existe déjà un certain nombre d'algorithmes utilisant la théorie des sous-ensembles flous afin de gérer les incertitudes et imprécisions contenues dans les données. Dans la section 2, nous décrivons ensuite les concepts liés à l'extraction de règles d'association et de motifs séquentiels, qui seront ensuite utilisés tout au long de ce mémoire. Enfin, la section 3 développe les objectifs de notre travail et décrit l'organisation du présent manuscrit.

1 Extraction de Connaissances dans les grandes bases de Données

1.1 Le processus d'ECD

Le processus d'extraction de connaissances dans les bases de données (ECD), présenté sur la figure 1, désigne l'ensemble des opérations qui permettent d'exploiter avec facilité et rapidité des données stockées massivement. Il s'agit d'un processus non trivial, consistant à identifier dans les données des schémas nouveaux, valides, potentiellement utiles et surtout compréhensibles et utilisables [FPSS96a].
Le processus d'ECD peut avoir deux objectifs, soit vérifier les hypothèses d'un utilisateur, soit découvrir de nouveaux motifs. Un motif, ou schéma, est une expression dans un langage spécifique qui décrit un sous-ensemble de données ou un modèle applicable à ce sous-ensemble [FPSS96b].

Ce processus comporte quatre étapes principales : (1) la sélection, (2) la préparation des données, (3) la fouille de données et (4) l'interprétation.

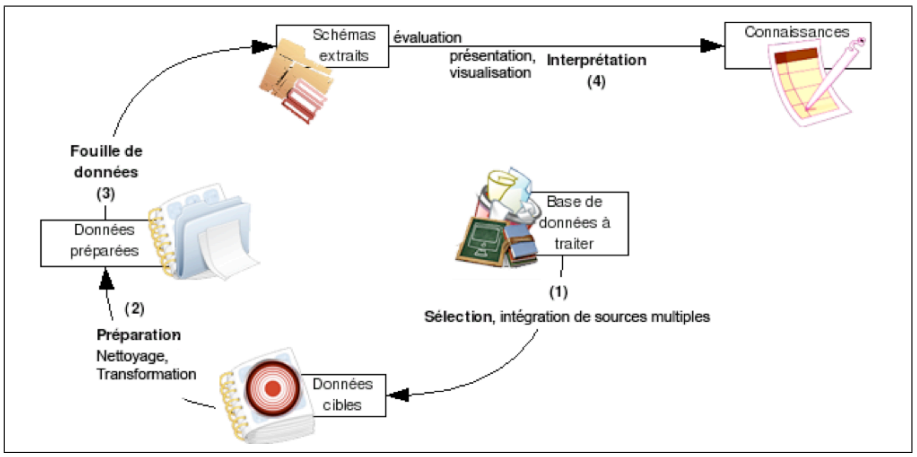


FIG. 1 – Les différentes étapes du processus d'ECD

Les deux premières étapes, ***sélection*** et ***pré-traitement*** (ou ***préparation***), regroupent les différentes transformations que les données doivent subir avant d'être disponibles pour l'analyse : le nettoyage, l'intégration des sources multiples, la transformation des données (agrégation, normalisation, ...) et enfin la réduction des dimensions d'analyse [HK00].
Ces transformations dépendent du type de fouille à entreprendre. Le but de ces étapes est de construire une base de données contenant un ensemble d'enregistrements *(ou lignes, n-uplets, tuples, exemples, transactions)*, décrits en fonction d'attributs *(ou colonnes, items)*. Par exemple, on considère une base de données de grande surface où chaque ligne correspond à un achat, identifié par une date et un client et où les colonnes correspondent aux produits achetés.
C'est lors du nettoyage que les données erronées ou bruitées sont identifiées, corrigées si possible ou supprimées et que les inconsistances sont résolues [BA96]. C'est également lors de cette étape que les valeurs manquantes sont prises en compte si elles ne peuvent être gérées lors de la fouille de données.
Pour réaliser cette préparation des données et choisir la bonne technique de fouille, il est bien entendu nécessaire de bien connaître les données ainsi que les besoins de l'utilisateur.

Une fois mises en forme, les données sont traitées par l'algorithme d'extraction, c'est l'étape de ***fouille de données***. Cette phase consiste à utiliser des méthodes d'analyse de données et des algorithmes de recherche qui permettent d'obtenir, dans un temps acceptable, un certain nombre de schémas et de

motifs caractéristiques des données [FPSS96b]. Il s'agit d'un processus interactif d'analyse d'un grand ensemble de données, destiné à extraire des connaissances exploitables.
L'objectif de l'extraction de connaissances est l'extraction de motifs, c'est-à-dire la recherche d'un modèle, d'une structure ou encore tout autre description de haut niveau, caractérisant les données. Ce modèle peut être prédictif, dans ce cas, les motifs trouvés ont pour but de prédire des comportements futurs, ou bien il peut être descriptif, les motifs extraits ont alors pour but de décrire les données de façon compréhensible et intelligible pour l'utilisateur.
Selon les objectifs que l'on souhaite remplir, les algorithmes de fouilles employés pour analyser les données seront différents. Ces différentes approches sont décrites dans la sous-section 1.2.

Enfin, pour déterminer le niveau d'intérêt des découvertes réalisées lors de la fouille de données, on réalise une ***interprétation*** des résultats obtenus. Une fois cette interprétation terminée, l'utilisateur peut enfin savoir quels schémas intéressants se cachent dans la quantité de données qu'il a stockées. Pour cela, des mesures quantitatives et qualitatives sont définies, afin d'évaluer les motifs extraits et d'aider à leur utilisation.

1.2 L'étape de fouille de données

La fouille de données désigne les algorithmes qui ont pour but d'analyser les données. Il existe différentes approches dont les résultats varient et l'utilisation dépend de l'objectif de la fouille de données. On distingue les approches ***supervisées***, qui nécessitent une phase d'apprentissage et une expertise sur les données, des approches ***non supervisées***, pour lesquelles on ne dispose d'aucune autre information préalable que la description des exemples (enregistrements). Les différentes techniques de fouille de données permettent ainsi de répondre à l'un des deux objectifs principaux :

- résumer ou caractériser les propriétés générales des données (objectif descriptif),
- inférer les données actuelles pour faire des prévisions (objectif prédictif).

La ***classification*** est une approche supervisée qui consiste à définir une fonction qui attribue une ou plusieurs classes à chaque donnée [FPSS96b]. Ensuite, les nouvelles données sont analysées et affectées, en fonction de leurs caractéristiques, à l'une des classes préalablement apprises sur un ensemble d'entraînement. Les techniques de classification sont par exemple utilisées lors d'opérations de mailing pour cibler la bonne population et éviter ainsi un nombre trop important de non-réponses. Cette démarche peut également permettre de déterminer, pour une banque, si un prêt peut être accordé, en fonction de la classe d'appartenance d'un client.

La ***segmentation***, ou ***clustering***, ressemble à la classification, mais diffère dans le sens où il n'existe pas de classes pré-définies. L'objectif des techniques de segmentation est de grouper des enregistrements proches dans une même classe de manière à ce que deux données d'un même cluster *(ou segment, groupe)* soient le plus similaires possible et que deux enregistrements de clusters différents soient le plus dissemblables possible [Har75, NH94]. Les applications concernées incluent, entre autre, la segmentation de marché (ciblage marketing) ou encore la segmentation démographique, par exemple en identifiant des caractéristiques communes entre populations.

Enfin, les techniques de régression cherchent à déterminer une fonction qui convertit une donnée en une prédiction en utilisant les relations fonctionnelles qui peuvent exister entre les variables.

Différentes techniques permettent de décrire les données. La découverte de résumés tout d'abord, consiste à fournir une description compacte d'un sous-ensemble de données. Il existe différentes approches permettant de formuler ces représentations, dont certaines utilisent la théorie des sous-ensembles flous [KYZ00].

La découverte de règles d'association permet de décrire des ensembles d'items fréquemment liés dans une même transaction, ainsi que les règles les combinant [AIS93]. Un exemple d'association pourrait révéler que "75% des gens qui achètent de la bière achètent également des couches". Ce type de règles concerne un grand champ d'applications, telles que la conception de catalogues, la promotion de ventes, le suivi d'une clientèle, etc.

La recherche de motifs séquentiels, qui peut être vue comme une extension de la problématique des règles d'association intégrant diverses contraintes temporelles, consiste à extraire des ensembles d'items, couramment associés sur une période de temps [AS95]. Par exemple, des motifs séquentiels peuvent montrer que "60% des gens achètent une machine à laver puis achètent un sèche-linge dans les deux ans qui suivent". Ce problème, posé à l'origine dans un contexte marketing, intéresse à présent des domaines aussi variés que les télécommunications (détection de fraudes), la finance, ou encore la médecine (identification de symptômes précédant des maladies).

1.3 La théorie des sous-ensembles flous en fouille de données

Le but de l'extraction de connaissances est d'extraire des schémas et des modèles intelligibles pour l'utilisateur humain. Loin d'être binaire, la pensée humaine n'est pas toujours aisément modélisable par un programme informatique et parfois des outils permettant de raisonner avec des termes nuancés sont très utiles [BM99]. Par ailleurs, les bases de données du monde réel contiennent souvent de nombreuses imperfections : des informations non renseignées (incomplétudes), des données erronées (incertitudes) ou encore des données imprécises. Il est important de proposer des techniques permettant de détecter ces données afin de les corriger ou des méthodes de fouille dont les résultats restent fiables malgré les différentes imperfections des données exploitées. C'est pourquoi la théorie des sous-ensembles flous [Zad65] (annexe A) a largement été employée et de nombreux algorithmes de data mining ont désormais leurs extensions floues. Ils permettent ainsi de répondre à des problématiques plus larges et souvent de faciliter l'interprétation des résultats par l'utilisateur final [MPM02] en fournissant des schémas approximatifs robustes aux imperfections et utilisant des termes linguistiques.

Plus généralement, les sous-ensembles flous sont utilisés pour gérer les problèmes liés à :
- l'interprétabilité et la compréhensibilité des schémas extraits,
- l'incomplétude et le bruit dans les données,
- la fusion des sources d'information de types variés,
- les interactions humaines.

L'utilisation de la logique floue permet également dans certains cas de fournir des solutions approximatives plus rapidement. Enfin, elle autorise le traitement simultané de plusieurs types de variables [Yag96, Ped98] ou encore de requêtes flexibles aussi bien dans un contexte de base de données relationnelles que multidimensionnelles [Lau02].

Parmi les techniques de fouille de données utilisant la logique floue, on compte des méthodes de classification comme, par exemple, les arbres de décision flous [MBM99, OW03, HT03], différents algorithmes de clustering dont le plus connu est l'algorithme *fuzzy c-means* [Dun73, Bez81, BHLK06]. Enfin, des techniques ont été proposées pour l'extraction de règles d'association floues [AC98, FWS+98, KFW98, CWK00, Gye00a, DSV02, HLW03].

2 Découverte de motifs séquentiels

Dans le cadre de ce travail, nous nous sommes intéressés aux techniques d'extraction de corrélations fréquentes telles que les règles d'association et plus précisément à la **découverte de motifs séquentiels**.

La recherche de règles d'association [AIS93] répond à la problématique du panier de la ménagère qui peut être résumée ainsi : étant donnée une base de données de transactions (les paniers), chacune composée d'items (les produits achetés), la découverte de règles d'association consiste à chercher des ensembles d'items, fréquemment liés dans une même transaction, ainsi que des règles les combinant.
La recherche de motifs séquentiels, introduite dans [AS95], consiste à extraire des ensembles d'items couramment associés sur une période de temps bien spécifiée. Elle permet de mettre en évidence des associations intertransactions (entre les différents achats des clients), par rapport à celle de règles d'association qui extrait des combinaisons intratransactions (entre les produits achetés au cours d'un même achat). Ainsi, contrairement aux règles d'association, les motifs séquentiels permettent de suivre des comportements au cours du temps.

2.1 Différents types de bases de données

Nous avons vu précédemment que le choix d'une technique de fouille de données dépend des objectifs de l'extraction de connaissances. Mais certaines techniques sont également plus adaptées à certaines données, comportant des informations caractéristiques. Il est donc important d'identifier le type de base de données dont on dispose.

Les ***bases de données relationnelles*** regroupent un ensemble de données stockées dans des tables et décrites par un ensemble d'attributs. Généralement, la fouille dans de telles bases a pour but de découvrir des schémas et des tendances. Les ***bases de données de transactions***, quant à elles, sont une collection d'enregistrements de transactions assimilables à des achats de supermarché. L'analyse de ces données consiste alors à trouver des corrélations entre les items des transactions enregistrées. Dans les ***bases de données temporelles***, enfin, les données relationnelles sont associées à un attribut temporel. Les algorithmes de fouille de données utilisés ont alors pour objectif d'extraire des motifs périodiques, des épisodes ou encore des motifs séquentiels [ZB03, JKK99, LC92].

Les ***bases de séquences de données*** sont des bases de données temporelles particulières. Il s'agit en fait de bases de données relationnelles ou de transactions dans lesquelles les enregistrements peuvent être organisés en séquences d'évènements ordonnés selon une notion de temps, concrète ou non (e.g. achats de clients dans un supermarché, apparition de mots ou de concepts dans un texte, logs de navigation Internet) [HK00]. On peut y rechercher différents types de motifs :

- des schémas d'évolution des attributs au cours du temps, afin d'analyser des tendances,
- des séquences qui ne diffèrent que légèrement les unes des autres, pour déceler des similitudes,
- des motifs séquentiels, afin de trouver des relations entre les occurrences d'évènements séquentiels et l'ordre spécifique qui peut exister entre ces apparitions,
- des motifs périodiques, afin de caractériser des successions d'évènements récurrents et répétés dans les séries temporelles.

2.2 Un détour par les règles d'association

Les motifs séquentiels peuvent être vus comme une extension temporelle des règles d'association. Formellement, le problème des règles d'association est présenté dans [AIS93] de la façon suivante : soit $I = \{i_1, i_2, ..., i_n\}$ un ensemble d'***items***, soit D un ensemble de transactions, tel que chaque transaction est constituée d'un identifiant Tid et d'un ensemble non vide T d'items, vérifiant $T \subseteq I$. Cet ensemble

d'items, ou ***itemset***, est noté $(i_1 i_2 ... i_k)$ où i_j est un item de I. Une transaction ne contient pas de doublon.

Exemple 0.1. Dans une base d'achats, une transaction s'écrit sous la forme d'un couple : (id-ticket, ensembles de produits achetés), par exemple : "Ticket 1, (Farine, Bières, Tomates)".

On dit qu'une transaction T ***supporte*** l'itemset X si elle contient tous les items de X *(i.e. si $X \subseteq T$)*.

Une ***règle d'association*** est une implication de la forme $X \rightarrow Y$, où X est un itemset inclus dans I, et Y un item de I tel que $Y \notin X$. Cette définition est ensuite étendue par [AS94], qui permet à la partie *conséquence* d'une règle d'être composée de plusieurs items, tels que $Y \subset I$ et $Y \cap X = \varnothing$.

Exemple 0.2. Une règle d'association dans la base TAB. 1 pourrait être *"Farine → Tomates"*.

Le ***support*** d'une règle $X \rightarrow Y$ correspond au pourcentage de toutes les transactions de D supportant l'itemset $X \wedge Y$. La ***confiance*** dans la règle $X \rightarrow Y$ établit la probabilité qu'une transaction T supportant l'itemset X, supporte aussi l'item(set) Y. Ainsi, à partir d'une base de transactions D, le problème consiste à extraire toutes les règles d'association dont le support est supérieur au support spécifié par l'utilisateur *(minSupp)* et dont la confiance est supérieure à la confiance spécifiée par l'utilisateur *(minConf)*. Cette tâche est divisée en deux étapes :

1. rechercher tous les itemsets fréquents (i.e. dont le support est supérieur à *minSupp*),
2. générer, à partir de ces itemsets fréquents, des règles dont la confiance est supérieure à *minConf*.

Plusieurs algorithmes basés sur ce principe ont été proposés dans [AS94, AMS+96].

2.3 Découverte de motifs séquentiels

Contrairement à la recherche de règles d'association, la découverte de motifs séquentiels permet d'introduire une notion d'ordre (souvent temporel) entre les enregistrements. Ainsi, les motifs séquentiels ont été initialement proposés dans [AS95] et reposent sur la notion de ***séquences fréquentes maximales***.

Considérant une base d'achats de supermarché au format de la figure 1, le but d'un algorithme d'extraction de motifs séquentiels est de trouver des comportements fréquents dans les achats des clients. Le résultat attendu est donc une ou plusieurs séquences d'achats, représentant un comportement récurrent, par exemple la séquence <(*Farine, Tomate*) (*Pizza surgelée*) (*La cuisine italienne*)>, que l'on peut traduire par "souvent, les clients achètent d'abord de la farine et des tomates, puis reviennent pour acheter une pizza surgelée. Plus tard, ils achètent un livre "La cuisine italienne".

Client	01/09/06	02/09/06	03/09/06	04/09/06
Client 1	Farine Bière Tomate	Savon	Pizza surgelée	La cuisine italienne Soda
Client 2		Farine Tomate	Pizza surgelée Fromage	La cuisine italienne
Client 3	Soda Gateau	Bière Pizza		Farine Tomate
Client 4	Farine Gateau Tomate	Pizza surgelée Mouchoir	La cuisine italienne Lunette	Chocolat Soda Pantoufle

TAB. 1 – Une base de données exemple, limitée à 4 clients

Définitions

Prenons comme exemple une base de données DB d'achats pour un ensemble $\mathcal{C}$ de clients c. Une ***transaction*** t est un triplet <id_client, id_date, itemset> qui caractérise le client qui a réalisé l'achat, la date à laquelle il a eu lieu et les ***items*** dont il est composé.

Soit $\mathcal{I} = \{i_1, i_2, \cdots, i_m\}$ l'ensemble des items de la base. Un ***itemset*** est un ensemble non vide et non ordonné d'items, noté $(i_1, i_2, \cdots, i_k)$.

Définition 0.1. Une *séquence* est une liste ordonnée non vide d'itemsets notée $< s_1 s_2 ... s_p >$ où s_j est un itemset. Une n-***séquence*** est une séquence de taille n, c'est-à-dire composée de n items.

Exemple 0.3. Si un client achète les produits a, b, c, d et e selon la séquence $S =< (a)(b\ c)(d)(e) >$, cela signifie qu'il a d'abord acheté le produit a, puis les produits b et c ensemble, ensuite le produit d et finalement le produit e.

Définition 0.2. Une séquence $S' =< s'_1 s'_2 \cdots s'_m >$ est une ***sous-séquence*** de $S =< s_1 s_2 \cdots s_p >$ s'il existe des entiers $a_1 < a_2 < \cdots < a_j < \cdots < a_m$ tels que $s'_1 \subseteq s_{a_1}, s'_2 \subseteq s_{a_2}, \cdots, s'_m \subseteq s_{a_m}$. On dit que S' est ***incluse*** dans S.

Exemple 0.4. La séquence $< (b)(e) >$ est une sous-séquence de S car $(b) \subseteq (b\ c)$ et $(e) \subseteq (e)$, mais $< (b)(c) >$ n'est pas une sous-séquence de $< (b\ c) >$, ni l'inverse.

Les transactions de la base sont regroupées par client et ordonnées chronologiquement. Elles définissent ainsi des ***séquences de données***. On dit qu'un client c ***supporte*** une séquence S si cette séquence est incluse dans la séquence de données du client c. Le ***support*** d'une séquence est alors défini comme le pourcentage de clients de la base DB qui supporte S. Une séquence est dite ***fréquente*** si son support est au moins égal à une valeur minimale $minSupp$ spécifiée par l'utilisateur.

La recherche de motifs séquentiels dans une base de séquences telle que DB consiste alors à trouver toutes les séquences maximales (non incluses dans d'autres) dont le support est supérieur à $minSupp$. Chacune de ces ***séquences fréquentes maximales*** est un ***motif séquentiel***.

La recherche d'association comme celle de motifs séquentiels se base sur un format de données bien précis, qui relate des évènements liés à différents acteurs. L'exemple du supermarché permet d'illustrer ce concept simplement.
Dans la suite de ce mémoire, ce modèle est généralisé. Nous considérerons des ***objets*** o (les clients), qui correspondent chacun à des ***séquences de données*** $\mathcal{R}_o$, d'***enregistrements*** r (les transactions), identifiés chacun par une date. Ainsi, le tableau 1 présente alors les enregistrements datés et ordonnés de 4 objets.
De plus, pour éviter toute confusion avec le support d'un sous-ensemble flou (cf. Annexe A), nous parlerons désormais de ***fréquence*** d'un item, d'un itemset ou d'une séquence pour évoquer le support de ces différents éléments.

Algorithmes et extensions

La découverte de motifs séquentiels consiste donc en la recherche de motifs liés au temps ou à tout autre type de séquences se produisant de façon fréquente [HK00]. Afin d'analyser des bases de séquences de données dans le but d'en extraire de telles schémas, différents algorithmes ont été proposés, afin de réduire le temps d'exécution ainsi que les besoins en espace disque ou en mémoire vive.

De nombreux algorithmes peuvent être regroupés en trois catégories :

- Les algorithmes de type ***générer-élaguer***, approches par niveau (***level-wise***), qui génèrent l'ensemble des candidats possibles avant de vérifier s'ils sont fréquents ou non. Ces algorithmes sont basés sur l'algorithme original *Apriori-All* proposé dans [AS95] ;
- Les algorithmes de type ***pattern-growth*** qui projettent chaque séquence de données selon une séquence *préfixe* qui correspondent en fait aux motifs séquentiels ; les préfixes sont allongés de façon récursive dans les sous-bases de séquences projetées et les motifs séquentiels sont trouvés par recherche locale. PrefixSPAN est le premier algorithme proposé sur ce principe [PHMA+01].
- Les algorithmes qui utilisent des structures de données spécifiques afin de réduire l'espace mémoire utilisé ainsi que les temps de parcours de la base, parmi eux on peut citer SPAM [AGYF02], SPADE [Zak01] ou encore MEMISP [LL05].

Afin de répondre à certaines problématiques plus précises, des extensions ont été proposées, basées sur ces algorithmes, permettant par exemple la recherche incrémentale de motifs séquentiels [PZOD99, Mas02, ZKCY02, MPT03, CYH04] ou encore la généralisation des motifs séquentiels pour différents paramètres temporels (espacement des différents évènements d'une séquence, rapprochement d'évènements proches en une même date...) [SA96b, PHW02, MPT04]. Enfin, les algorithmes CloSPAN [YHA03] et BIDE [WH04] visent à extraire certains motifs séquentiels particuliers, dits fermés[1].

Les algorithmes présentés dans ce mémoire sont basés sur une approche générer-élaguer et plus précisément sur l'algorithme PSP, présenté dans [MCP98]. Toutefois, ils auraient pu être développés à partir d'une approche pattern-growth.

3 Objectifs et contributions

Les utilisations des motifs séquentiels sont désormais nombreuses [LC92], on peut notamment citer : l'analyse du comportement de consommateurs, de profils d'internautes, ou de performances systèmes et réseaux de télécommunication, le contrôle d'inventaires, l'analyse de symptômes pour du diagnostique médical, ou des schémas de mutation des acides aminés. Les séquences de données alors analysées ne sont plus aussi parfaites que celles contenues dans les bases de données de type supermarché, dans lesquelles les attributs sont binaires et les enregistrements complets.

Les domaines d'utilisation des motifs séquentiels se diversifiant, il apparaît nécessaire d'adapter les algorithmes permettant l'extraction de séquences fréquentes afin de pouvoir prendre en compte des données hétérogènes, incomplètes, incertaines, ou mal connues de leur utilisateur, tout en minimisant les pertes éventuelles d'information. Nous nous sommes donc intéressés à la recherche de motifs séquentiels dans des données imparfaites, en présence de données numériques, tout d'abord, dans des bases de données symboliques et complètes, ensuite, puis, incomplètes.

Ainsi, le travail présenté dans cette thèse porte sur les trois points suivants :

- La mise en œuvre d'un cadre pour l'extraction de motifs séquentiels en présence de données numériques quantitatives ; nous proposons d'utiliser la théorie des sous-ensembles flous afin d'extraire des motifs séquentiels selon trois niveaux d'information, plus ou moins précis, selon les besoins de l'utilisateur.
- La définition de contraintes temporelles relâchées pour l'extraction de motifs séquentiels sur données symboliques. Ces travaux ont pour but d'autoriser à l'utilisateur la spécification de contraintes temporelles approximatives, limitant ainsi le nombre d'essais nécessaires pour obtenir des motifs

[1] Un motif fréquent est dit fermé s'il n'existe pas de sur-motif de ce motif ayant la même fréquence

séquentiels intéressants sous contraintes. Ces contraintes souples nous permettent également de calculer un indice de précision temporelle, utilisé pour trier les résultats obtenus.
– Le développement de deux approches pour l'extraction de motifs séquentiels sur des données symboliques incomplètes au hasard. Le premier algorithme consiste à utiliser l'information partielle contenue dans les enregistrements incomplets pour extraire les séquences fréquentes. Le second algorithme, quant à lui, utilise la distribution des données dans la base afin de proposer une estimation multivaluée des valeurs manquantes lors de l'extraction des motifs. Enfin, nous proposons une première approche de complétion des valeurs manquantes grâce aux motifs séquentiels extraits.

Le présent mémoire est organisé en trois parties qui présentent notre travail, suivies d'un développement de ses perspectives. Chacune de ces parties commence par un chapitre dans lequel nous présentons une synthèse des travaux existants pour le sujet traité. Les chapitres suivants détaillent ensuite nos définitions ainsi que les algorithmes que nous avons développés. Ces algorithmes ont été implémentés et nous présentons les résultats des expérimentations sur données synthétiques et/ou réelles. Chaque partie s'achève sur une discussion des résultats obtenus et sur d'éventuelles améliorations. La Partie I développe notre cadre pour l'extraction de motifs séquentiels à partir de données quantitatives. La Partie II présente ensuite notre proposition de contraintes temporelles souples. Puis dans la Partie III nous décrivons nos propositions d'algorithmes pour l'extraction de séquences fréquentes à partir de séquences de données incomplètes. Enfin, ce mémoire se conclut sur un chapitre dans lequel nous présentons les diverses perspectives ouvertes par notre contribution.

Première partie

Motifs séquentiels et données quantitatives

Savoir, et ne point faire usage de ce qu'on sait, c'est pire qu'ignorer.
E. Chartier (1868-1951) — *Propos sur l'éducation*

Les données traditionnellement exploitées par les règles d'association et les motifs séquentiels ne contiennent pas d'attribut numérique ou cette information n'est pas considérée comme une spécification d'un item dont on cherche à déterminer la fréquence, mais comme une caractéristique sur laquelle il est possible d'imposer une contrainte. Mais ces informations n'apparaîtront pas dans les séquences finalement obtenues. Dans cette partie, nous proposons de conserver l'information numérique disponible afin d'augmenter la connaissance extraite par la découverte de motifs séquentiels.
Le chapitre 1 détaille les précédents travaux permettant de traiter des données numériques quantitatives dans un processus d'extraction de règles d'association ou de motifs séquentiels. Dans le chapitre 2, nous présentons notre cadre pour l'extraction de motifs séquentiels flous sur des données quantitatives. Enfin, le chapitre 3 présente le résultat de plusieurs expérimentations réalisées aussi bien sur données synthétiques que sur données réelles. Nous concluons cette partie par une discussion sur les apports de notre contribution ainsi que sur les améliorations qui pourraient lui être apportées.

Chapitre 1

Règles d'association et données quantitatives

Le présent chapitre détaille les différentes méthodes qui existent afin d'intégrer des données numériques quantitatives dans un processus d'extraction de règles d'association ou de motifs séquentiels. La section 1.1 détaille les approches statistiques ainsi que celles basées sur des intervalles ou intervalles flous, pour la découverte de règles d'association quantitatives. La section 1.2, ensuite, présente les méthodes existantes pour la recherche de motifs séquentiels sur des données quantitatives. Enfin, dans la section 1.3, une discussion de ces travaux existants nous conduit à formuler notre proposition, qui sera développée dans le chapitre suivant.

1.1 Règles d'association pour les valeurs quantitatives

A l'origine, les règles d'association étaient destinées à des applications telles que l'analyse des achats de produits de supermarché et la plupart du temps, les informations quantitatives (prix, quantités) étaient ignorées ou utilisées comme des contraintes de sélection des items. Dans un deuxième temps, les techniques d'extraction se sont élargies afin de pouvoir intégrer ces données lors de l'extraction des fréquents. Un produit A, acheté dans une quantité q ne sera donc plus considéré comme le même item que ce même item acheté dans une quantité q'. Il s'agit alors de réaliser un compromis entre le nombre d'items différents à gérer (chaque couple possible (produit, quantité) devient un item) et le temps d'exécution ainsi que le nombre de fréquents que l'on obtiendra. Différentes propositions ont été formulées afin de pouvoir extraire des règles d'association sur ce type de bases de données, nous les présentons rapidement dans cette section.

1.1.1 Intervalles

La première proposition prenant en compte les données quantitatives lors de la recherche de règles d'association est la méthode présentée dans [SA96a]. Afin de diminuer le nombre de combinaisons possibles entre les items et les valeurs numériques associées, cette technique permet de rechercher des règles d'association dans des bases de données contenant à la fois des données symboliques et numériques.

Pour ce faire, les auteurs utilisent un partitionnement en intervalles de l'univers des quantités. Ces intervalles peuvent être combinés/fusionnés lorsque cela s'avère nécessaire.
Le principal avantage de cette méthode est de pouvoir traiter différents types d'attributs de façon similaire. L'algorithme utilisé est un algorithme de type générer-élaguer, du même type qu'Apriori [AIS93].

Toutefois, il est nécessaire de bien définir a priori le nombre d'intervalles, ainsi que les bornes. Les auteurs proposent ainsi d'utiliser un partitionnement par équi-répartition des enregistrements, dont le nombre d'intervalles dépend du support minimum ainsi que du nombre d'attributs quantitatifs. Dans certains cas, il est possible, en fonction du support et de la confiance obtenue, de fusionner deux règles portant sur les mêmes attributs mais dont les intervalles sont adjacents. Cette possibilité permet d'extraire les règles intéressantes tout en limitant le nombre.

1.1.2 Inférence statistique

[AL99] introduit une nouvelle définition des règles d'association, basée sur une théorie d'inférence statistique. Cette définition est destinée à favoriser l'extraction de règles extraordinaires et correspondant à des phénomènes intéressants. Cette approche part de l'hypothèse que l'objectif d'une règle d'association est de trouver des phénomènes intéressants. Ainsi, afin de déceler de tels phénomènes, cette proposition se base sur l'utilisation de la distribution des valeurs des attributs quantitatifs, ainsi que sur une généralisation statistique de la définition catégorielle.

Exemple 1.1. Une règle extraite selon cette méthode pourrait être : $sexe = femme \Rightarrow salaire : moyenne = 8\$/h$, alors que si l'ensemble des salariés est considéré, la moyenne des salaires horaires est en fait de 9$. Cette règle est donc significative et atypique.

Toutefois, la forme de ces règles et leur construction limitent leurs champs d'application. En effet, l'antécédent des règles extraites donne la description d'un sous-ensemble de la population, le conséquent décrit un comportement intéressant spécifique de la population correspondant à la partie gauche, sous la forme d'une mesure statistique portant sur un attribut : la moyenne, la variance ou encore l'écart-type.

1.1.3 Conversion booléenne

Les travaux présentés dans [ID01] proposent de transformer des items quantitatifs en booléens. Cette proposition se base sur l'hypothèse selon laquelle dans la plupart des cas, les données quantitatives peuvent être réduites à deux valeurs.

Pour cela, des seuils sont spécifiés par des experts ou calculés par des formules statistiques (moyenne, modes, médiane). Chaque univers est alors transformé en deux intervalles, l'un dans lequel les quantités sont inférieures au seuil, le second dans lequel elles sont supérieures. Les règles d'association sont alors extraites avec les algorithmes classiques. Elles peuvent comporter à la fois des attributs quantitatifs binairisés ou des attributs catégoriels.

1.1.4 Intervalles flous

Plusieurs propositions ont été formulées [CA97, AC98, FWS+98, KFW98, CWK00, Gye00a, HKCW00, DSV02, HLW03], présentant les avantages d'utilisation d'intervalles flous pour le traitement de données quantitatives lors de la découverte de règles d'association. Leur principe consiste à découper chaque domaine de quantités en une partition floue.

Une règle d'association floue se présente sous la forme "*Si* X *est* A*, alors* Y *est* B" où la partie "X *est* A" est l'antécédent ou condition de la règle et la partie "Y *est* B" est le conséquent ou conclusion de la règle. Cette règle se note $(X, A) \rightarrow (Y, B)$, X et Y sont deux itemsets disjoints, sous-ensembles de $\mathcal{I}$ et A et B sont les ensembles des sous-ensembles flous des partitions associées aux univers des quantités de X et de Y. Une règle est dite satisfaite si un nombre suffisant de transactions de $\mathcal{T}$ contiennent les items flous [attribut x, sous-ensemble flou a] et [attribut y, sous-ensemble flou b]. Ce nombre est

déterminé par Σ-comptage seuillé des degrés d'appartenance de l'itemset support de la règle pour chaque transaction.

1.2 Motifs séquentiels pour les valeurs quantitatives

1.2.1 Intervalles

L'approche présentée par [KLNS04] utilise des intervalles afin d'extraire des séquences fréquentes dans lesquelles les items sont associés à des quantités.
Dans ces travaux, chaque item est représenté par une paire (i, max_i) qui signifie que la quantité de l'item i est comprise entre 1 et max_i. Pour chaque item, on ne considère qu'un nombre fini et déterminé de quantités. L'algorithme se déroule en deux phases. Tout d'abord, les items non fréquents sont trouvés sans tenir compte des quantités. Ensuite, pour chaque item fréquent i on calculera la valeur max_i.
Les motifs séquentiels sont alors des séquences dont les items sont des couples (item, quantité maximale).

1.2.2 Motifs séquentiels flous

Les principes utilisés lors de la recherche de motifs séquentiels flous sont similaires à ceux définis pour les règles d'association floues. Plusieurs propositions ont été formulées dans ce sens, afin d'extraire des motifs séquentiels à partir de données quantitatives [HLW01, CTCH01, CH02, HCTS03, HTC04, SG05]. Toutes ces approches sont fondées sur le même principe. Afin d'extraire les motifs, les univers de quantités de chaque attribut quantitatif sont partitionnés en plusieurs sous-ensembles flous. Le jeu de données est ensuite converti en une base de degrés d'appartenance. Ces approches diffèrent ensuite dans la manière de calculer le support d'une séquence.

Cette phase est suffisante pour la plupart des algorithmes. Cependant, la première approche proposée pour la découverte de motifs séquentiels flous, [HLW01], opère une seconde étape avant la fouille proprement dite. En effet, afin de minimiser le nombre d'items à traiter et ainsi réduire le temps d'extraction, l'algorithme sélectionne les items flous les plus significatifs : pour chaque item, seul le sous-ensemble flou dont la cardinalité (déterminée par Σ-comptage) est la plus élevée sur toute la base est conservé. Par exemple, si on considère un item X pouvant être associé à trois sous-ensembles flous a, b et c, un seul des trois items flous (X, a), (X, b) ou (X, c) sera conservé.

Une telle sélection nous semble réductrice et peut conduire à la découverte de motifs erronés. Les conséquences de cette stratégie sur les motifs séquentiels découverts sont mises en évidence dans les expérimentations présentées dans le chapitre 3.

La seconde approche [CTCH01, HCTS03, HTC04, CH02], est très formelle. Elle n'introduit qu'un algorithme dont les structures de données sous-jacentes ne sont pas décrites. Aucune expérimentation n'est présentée. Par ailleurs, le formalisme et les notations utilisées pour définir la fréquence d'un itemset flou ou d'une séquence floue sont plutôt ambigus, alors que la notion d'ordre est fondamentale pour la recherche de séquences fréquentes.

Enfin, à l'instar des approches proposées pour les règles d'association floues, aucun de ces travaux ne considère les différents niveaux d'information qui peuvent être obtenus en considérant un cadre global pour la découverte de motifs séquentiels flous, ainsi que les différents niveaux de fuzzification qui peuvent être utilisés pour évaluer la fréquence d'une séquence. C'est également le cas de la proposition la plus récente, [CH06], qui propose un algorithme efficace basé sur les mêmes principes que les travaux précédents.

1.3 Objectifs

Comme nous l'avons vu dans la section précédente, les travaux existant n'utilisent pas toutes les ressources disponibles grâce à l'utilisation des sous-ensembles flous. Notamment, les différents niveaux de fuzzification. En effet, il est possible d'extraire différents types de connaissances à partir d'attributs quantitatifs. Les motifs séquentiels sont caractérisés par leur fréquence, qui indique la proportion de séquences de données dans la base qui contiennent ce motif. Dans le cas de séquence quantitative, il est possible d'extraire différentes informations en utilisant les fonctions d'appartenance des sous-ensembles flous. On peut ainsi rechercher l'existence de séquences fréquentes, rechercher l'existence de séquences fréquentes suffisamment significatives ou encore rechercher des séquences fréquentes dont la fréquence tient compte de leur significativité.

Dans la suite de cette partie, nous présentons donc un cadre global permettant de prendre en compte ces différents types de connaissances lors de l'extraction de motifs séquentiels flous. Ce cadre étend et généralise les approches précédentes pour la découverte de motifs séquentiels flous. Il s'agit d'une approche complète et efficace d'extraction de séquences fréquentes floues qui permet le traitement des attributs quantitatifs tout en considérant les différents niveaux d'information contenus dans les données numériques. Cette approche, comme les précédentes, reposent sur le découpage des univers numériques en intervalles flous. Nous proposons ensuite trois calculs de la fréquence d'une séquence floue, chacune répondant à différents besoins de l'utilisateur.

Dans le chapitre suivant, nous détaillons notre cadre pour l'extraction de motifs séquentiels flous, ainsi que les différentes définitions de la fréquence d'une séquence floue et les algorithmes développés afin d'extraire des motifs séquentiels flous. Les résultats des expérimentations conduites sur données synthétiques et réelles afin de valider notre cadre et nos algorithmes sont présentés dans le chapitre 3.

Chapitre 2

Un cadre pour la recherche de motifs séquentiels flous

Les approches d'extraction de motifs séquentiels tels que définis dans [AS95] ne permettent qu'un traitement partiel des informations contenues dans des données quantitatives. Des propositions ont été formulées afin d'extraire des règles d'association floues dans de telles bases. et quelques travaux se sont intéressés à la découverte de motifs séquentiels flous. Toutefois, comme nous l'avons montré dans le précédent chapitre, ces approches peuvent être généralisées et améliorées.

Nous présentons donc dans le présent chapitre notre cadre pour l'extraction de motifs séquentiels flous. Grâce à trois définitions de la fréquence floue ainsi qu'aux algorithmes associés, ce cadre permet l'extraction de motifs séquentiels selon trois niveaux d'information. L'utilisateur peut alors choisir entre vitesse d'extraction et précision des résultats obtenus.

Nous commencerons par la présentation des trois niveaux d'information ainsi que les définitions associées à la recherche de motifs séquentiels flous. Nous décrivons ensuite, dans la section 2.2, les trois algorithmes de calcul de la fréquence d'une séquence selon les définitions proposées. Enfin, la section 2.3 présente l'algorithme générique SMT-FUZZY, basé sur la structure d'arbre préfixé et permettant la découverte de motifs séquentiels flous.

2.1 Motifs séquentiels flous : définitions

2.1.1 Fuzzification des définitions classiques

Comme lors de la recherche de règles d'association floues, la première étape de la découverte de motifs séquentiels flous consiste à partitionner l'univers des quantités de chaque attribut de la base en plusieurs sous-ensembles flous. Chacun de ses attributs est alors associé à un sous-ensemble flou pour former un ***item flou***.

Définition 2.1. Un *item flou* est l'association d'un item avec l'un des sous-ensembles flous de quantité associés. On note $[x, a]$ un item flou correspondant à l'item x, associé au sous-ensemble flou a, défini sur l'univers des quantités de x.

Exemple 2.1. $[chocolat, beaucoup]$ est l'item flou correspondant à l'item $chocolat$, associé au sous-ensemble flou $beaucoup$ dont la fonction d'appartenance est définie sur l'univers des quantités de l'item $chocolat$.

Définition 2.2. Un *itemset flou* est un ensemble non-ordonné d'items flous. Il peut être noté comme un couple (ensemble d'items, ensemble des sous-ensembles flous associés à ces items), ou comme une

liste d'items flous. Nous utiliserons la première notation : (X, A) où X est un ensemble d'items et A l'ensemble des sous-ensembles flous associés ; l'ordre des sous-ensembles flous $a_1...a_p$ dans A correspond à l'ordre des items $x_1...x_p$ dans X, ainsi le premier item flou de l'itemset (X, A) correspond en fait à l'item flou $[x_1, a_1]$.

Exemple 2.2. $(X, A) = ([chocolat, beaucoup]\ [soda, peu])$ est un itemset flou ; il peut aussi être noté $((chocolat, soda)\ (beaucoup, peu))$.

Un itemset flou contient un seul item flou relatif à un attribut donné. Par exemple, l'itemset flou $([chocolat, beaucoup]\ [chocolat, peu])$ n'est pas un itemset flou valide car il contient deux fois l'item $chocolat$.

Définition 2.3. Une *g-k-séquence floue* $S =< s_1 \dots s_g >$ est une liste ordonnée de g itemsets flous $s_i = (X, A)$ qui regroupent au total k items flous $[x, a]$.

Exemple 2.3. La séquence $S =< ([soda, beaucoup]\ [chocolat, beaucoup])\ ([video, peu]) >$ regroupe 3 items flous, répartis dans 2 itemsets flous. S est une 2-3-séquence.

2.1.2 Préparation de la base de données

Tout d'abord, la base de données quantitative est convertie en une base de données de degrés d'appartenance. Pour cela, chaque attribut est partitionné en plusieurs sous-ensembles flous. La qualité des motifs séquentiels flous extraits est fortement liée à l'adéquation des sous-ensembles flous du partitionnement avec les données analysées. Il s'agit donc de réaliser un partitionnement automatique des données, spécifique à chaque base, et non pas d'utiliser systématiquement un modèle de partition unique, indépendant du jeu de données traité.

Dans certaines de nos expérimentations, nous utiliserons un partitionnement basé sur l'équi-répartition des enregistrements dans chaque sous-ensemble flou, mais plusieurs techniques sont applicables. Afin d'obtenir un découpage de chaque attribut en sous-ensembles flous, il est également possible d'utiliser un algorithme de clustering flou. De plus, l'utilisation de certains algorithmes en particulier permet de gérer l'incertitude liée à la présence de nombreuses données incomplètes dans la base [TK99, TDK03].

Une fois le partitionnement réalisé, la base de degrés d'appartenance est générée. C'est cette base qui sera utilisée afin d'extraire les motifs séquentiels flous.

Exemple 2.4. La base de données TAB. 2.1 va servir d'exemple pour chacune des définitions de la fréquence présentées par la suite.

La figure 2.1 montre le découpage en sous-ensembles flous et les fonctions d'appartenance pour chacun de ces sous-ensembles. L'attribut *dentifrice* par exemple est découpé en trois parties, "peu de dentifrice", "moyen de dentifrice" et "beaucoup de dentifrice". Un achat de 1 dentifrice par exemple est considéré comme appartenant à "peu de dentifrice", de 2 comme "moyen", 3 dentifrices est à la fois une quantité "moyenne" et une "grande" quantité et à partir de 4 dentifrices, l'achat comporte "beaucoup de dentifrice".

Les données du premier client sont alors encodées et la base des degrés d'appartenance obtenue est donnée par la table 2.2.

Dans la suite de cette section, nous utilisons cet exemple pour illustrer les calculs de fréquences pour l'itemset $Ix = ([chocolat, peu][soda, beaucoup])$ et la séquence $Sx = < ([chocolat, peu])([soda, beaucoup]) >$.

Dans nos définitions et algorithmes, nous utiliserons les notations suivantes : $\mathcal{O}$ représente l'ensemble des objets o de la base, $\mathcal{R}_o$ représente la séquence des enregistrements de l'objet o, $\mathcal{I}$ représente l'ensemble des items i de la base, $r[i]$ donne la quantité de l'attribut i dans l'enregistrement r.

Clients	Date	Items chocolat	dentifrice	soda	ballon	video
C1	d1	4		1		
	d2	3				
	d3	5	3	1		
	d4	2		1		
	d5			1	5	
	d6	2		2		2
C2	d1	2		2	1	
	d2			2		
	d3		4	1		
	d4	4				
C3	d1					3
	d2	3	1	2		
	d3				4	5
	d4			2		
	d5		2			
C4	d3	2		2		4
	d4			3		
	d5		2			
	d6			2		

TAB. 2.1 – Transactions regroupées par clients et ordonnées selon leur date

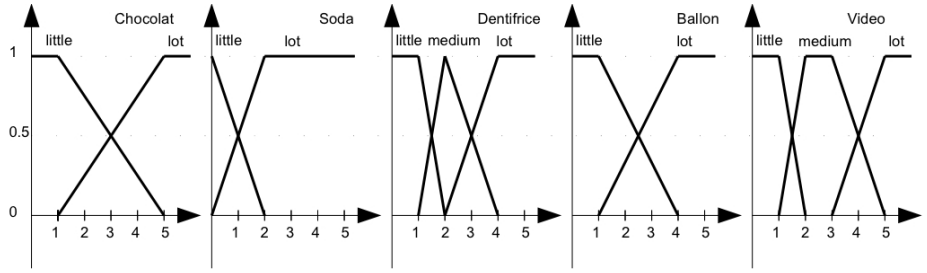


FIG. 2.1 – Fonctions d'appartenance pour chaque attribut quantitatif

D.	Items chocolat li.	chocolat L.	dentifrice li.	dentifrice m	dentifrice L.	soda li.	soda L.	ballon f.	ballon L.	video f.	video m.	video L.
d1	0.25	0.75				0.5	0.5					
d2	0.5	0.5										
d3		1		0.5	0.5	0.5	0.5					
d4	0.75	0.25				0.5	0.5					
d5						0.5	0.5		1			
d6	0.75	0.25					1				1	

TAB. 2.2 – Degrés d'appartenance pour le client 1

2.1.3 Plusieurs comptages...

Le principe du modèle que nous présentons est de proposer trois niveaux de fuzzification, chacun dépendant de la méthode adoptée pour calculer la fréquence de chaque itemset et de chaque séquence. Les méthodes de dénombrement proposées reposent sur les différents comptages existants pour déterminer la cardinalité d'un sous-ensemble flou A :

- compter les éléments de A pour lesquels le degré d'appartenance est non nul ($card(supp(A))$) ;
- compter l'ensemble des éléments de A pour lesquels le degré d'appartenance est supérieur à un seuil donné ($card(supp(A_\alpha))$), ce comptage est appelé ***comptage seuillé*** ;
- utiliser un Σ-***comptage*** en sommant les degrés d'appartenance de l'ensemble des éléments de A ($\sum_{x \in supp(A)} \mu_A(x)$) ;
- sommer les degrés d'appartenance de l'ensemble des éléments de A dont le degré d'appartenance est supérieur à un seuil donné ($\sum_{x \in supp(A)} \mu_{A_\alpha}(x)$), ce comptage est appelé Σ-***comptage seuillé***.

Chacun de ces comptages est à l'origine d'un algorithme d'extraction de motifs séquentiels, dont le schéma général peut être décrit par l'algorithme SMT-FUZZY, présenté dans la section 2.3.

2.1.4 ... plusieurs fréquences

La notion de fréquence d'une séquence est définie, pour la recherche de motifs séquentiels non-flous, comme la proportion de séquences de données dans la base entière qui supportent cette séquence.

Dans le cas classique, il suffit de trouver la séquence ordonnée des items parmi les enregistrements d'un objet pour ajouter celui-ci à la fréquene de la séquence. Dans le cas d'une base de degrés d'appartenance, il s'agit de définir, en fonction des degrés d'appartenance des items de la séquence, si un objet la supporte ou non, mais également de déterminer si tous les objets ont la même importance quelque soit la valeur des degrés d'appartenance des items de la séquence.

Ce problème rejoint la problématique du comptage et de la détermination de la cardinalité d'un sous-ensemble flou. Ainsi, selon le comptage choisi, le *"nombre d'objets supportant S"* se calcule différemment. Plusieurs définitions de la fréquence sont alors possibles.

La **fréquence d'une g-k-séquence floue** est calculée comme le ratio du nombre d'objets supportant cette séquence floue par rapport au nombre total d'objets dans la base :

$$FFreq_{(X,A)} = \frac{\sum_{o \in \mathcal{O}} \Big[F(o, gS) \Big] \Big]}{|\mathcal{O}|} \tag{2.1}$$

où le degré de fréquence $F(o, gS)$ indique si l'objet o supporte la séquence floue gS. Comme nous l'avons précédemment mentionné, la cardinalité d'un sous-ensemble flou dépend de la méthode de comptage adoptée. Nous transposons ici ces différentes techniques dans le cadre des motifs séquentiels et proposons trois définitions pour la fréquence floue. La table 2.3 montre la fréquence trouvée pour l'itemset flou Ix = $([chocolat, peu]\ [soda, beaucoup])$ dans chacun des cas et les différentes fréquences obtenues pour la séquence Sx=< $([chocolat, peu])([soda, beaucoup])$ > sont mises en évidence dans la table 2.4.

	Items											
	chocolat		dentifrice			soda		ballon		video		
D.	li.	L.	li.	m	L.	li.	L.	f.	L.	f.	m.	L.
d1	0.25	0.75				0.5	0.5					
d2	0.5	0.5										
d3		1		0.5	0.5	0.5	0.5					
d4	**0.75**	0.25				0.5	**0.5**					
d5						0.5	0.5		1			
d6	0.75	0.25					1				1	

TAB. 2.3 – Degrés d'appartenance pour le client 1 et fréquences floues pour l'itemset Ix

Prenons la séquence gS, composée de g itemsets (X_i, A_i), afin de décrire trois possibilités de comptage.
• SPEEDYFUZZY s'appuie sur un comptage binaire (supporte / ne supporte pas) (1). Le calcul de la fréquence consiste alors à compter tous les objets pour lesquels l'itemset apparaît au moins une fois dans

	Items											
	chocolat		dentifrice			soda		ballon		video		
D.	li.	L.	li.	m	L.	li.	L.	f.	L.	f.	m.	L.
d1	0.25	0.75				0.5	0.5					
d2	**0.5**	0.5										
d3		1		0.5	0.5	0.5	**0.5**					
d4	0.75	0.25				0.5	0.5					
d5						0.5	0.5		1			
d6	0.75	0.25					1				1	

TAB. 2.4 – Degrés d'appartenance pour le client 1 et fréquences de la séquence Sx

les enregistrements. Quel que soit le degré d'appartenance, l'objet a le même poids :

$$F_{SF}(o, gS) = \begin{cases} 1 & \text{si } \exists r_1 ... r_g \in \mathcal{R}_o | date(r_1) < ... < date(r_g) \text{ et } \forall i \in [1, g], \forall [x, a] \in (X_i, A_i),\ \mu_a(r_i[x]) > 0 \\ 0 & \text{sinon} \end{cases} \tag{2.2}$$

Exemple 2.5. Avec un comptage SPEEDYFUZZY, le premier client supporte l'itemset Ix=([$chocolat, peu$] [$soda, beaucoup$]) car la transaction d1 (TAB. 2.3) contient l'itemset considéré, le degré d'appartenance pour les deux items étant non nul. Les transactions d4 et d6 supportent également l'itemset Ix, mais seule la première apparition de la séquence candidate sera retenue lors du calcul de la fréquence.

• MINIFUZZY s'appuie sur un comptage seuillé. Cette méthode n'incrémente le nombre d'objets supportant l'itemset que si chaque item de la séquence candidate est trouvé dans la même transaction, avec un degré supérieur a un seuil minimum :

$$F_{MF}(o, gS) = \begin{cases} 1 & \text{si } \exists r_1 ... r_g \in \mathcal{R}_o | date(r_1) < ... < date(r_g) \text{ et } \forall i \in [1, g], \forall [x, a] \in (X_i, A_i),\ \mu_a(r_i[x]) > \omega \\ 0 & \text{sinon} \end{cases} \tag{2.3}$$

Exemple 2.6. Avec un comptage MINIFUZZY, le client 1 supporte l'itemset Ix=([$chocolat, peu$] [$soda$, $beaucoup$]) car la transaction d4 (TAB. 2.3) contient les deux items de Ix avec un degré d'appartenance supérieur au seuil minimum (dans notre cas, ω=0.49). d6 supporte également Ix, mais seule d4 sera retenue, puisque seule une apparition par client est nécessaire pour incrémenter la fréquence.

• TOTALLYFUZZY s'appuie sur un Σ-comptage seuillé. Avec cette définition, l'importance de la présence de chaque item flou est prise en compte lors du calcul de la fréquence. Pour ce faire, on utilise la fonction d'appartenance seuillée α définie par :

$$\alpha_a(r[x]) = \begin{cases} \mu_a(r[x]) & \text{si } \mu_a(r[x]) > \omega \\ 0 & \text{sinon} \end{cases} \tag{2.4}$$

Le degré de fréquence d'une séquence, pour un objet, est alors calculé par agrégation des degrés de fréquence des itemsets, trouvés dans l'ordre pour cet objet. Pour chaque itemset, ce degré de fréquence correspond à l'intersection des degrés d'appartenance de chacun des items qui le composent (opérateur t-norme, $\top$, en logique floue). Ensuite, pour chaque objet, on conservera la meilleure représentation de

la séquence (opérateur t-conorme, $\perp$, en logique floue).
Le calcul de la fréquence devient alors :

$$F_{TF}(o,gS) = \underline{\perp}_{\varsigma\subseteq\zeta_o|\varsigma=gS\forall i\in[1,g]\varsigma_i\subseteq r_i} Agr_{\forall i\in[1,g]\varsigma_i=(X_i,A_i)} \overline{\top}_{[x,a]\in(X_i,A_i)} \Big[\alpha_a\big(r_i[x]\big)\Big] \tag{2.5}$$

où $\overline{\top}$ et $\underline{\perp}$ sont les opérateurs t-norme et t-conorme généralisés au cas n-aire. Nous détaillons le choix des différents opérateurs dans le chapitre suivant, section 3.1.3.

On notera qu'un Σ-comptage est en fait un Σ-comptage seuillé à zéro, c'est pourquoi nous n'avons pas fait la distinction avec une quatrième formulation.

Exemple 2.7. Avec un comptage TOTALLYFUZZY, le client 1 supporte l'itemset Ix=([$chocolat, peu$] [$soda$, $beaucoup$]) si une transaction est trouvée, contenant l'itemset flou avec un degré d'appartenance supérieur à ω, le seuil minimum. La meilleure valeur pour l'itemset est ensuite conservée. Dans notre exemple, cet itemset Ix est supporté par l'enregistrement d6 (TAB. 2.3).

L'ordre d'apparition des items n'intervient pas lors du calcul de la fréquence d'un itemset, mais celui-ci est important lorsqu'il s'agit de calculer la fréquence d'une séquence. Ainsi, pour valider l'inclusion d'une séquence candidate dans une séquence de données, chaque itemset aura dû être trouvé à sa position d'apparition dans la séquence candidate. C'est pourquoi, le degré de fréquence est calculé en utilisant l'algorithme *CalcFuzzyFreq*, ALG. 1, qui utilise la fonction *FindFuzzySeq*, qui peut être $FindSpeedySeq$, $FindMiniSeq$ ou $FindTotallySeq$ selon le niveau d'information souhaité. Ces fonctions sont détaillées dans l'annexe B.

CalcFuzzyFreq - **Input** : gS, g-k-sequence candidate
Ouput : $FFreq$ fréquence floue de la séquence gS

$FFreq, nbFreq, m \leftarrow 0$;
Pour chaque objet $o \in \mathcal{O}$ **faire**
  $m \leftarrow$ FindFuzzySeq(g-S, $\mathcal{R}_o$) ;
  $nbFreq$ += m ;
Fin Pour
$FFreq \leftarrow nbFreq/|\mathcal{O}|$;
Retourner $FFreq$;

ALG. 1 : *CalcFuzzyFreq*

2.2 Calcul du support d'une séquence

2.2.1 SPEEDYFUZZY et MINIFUZZY

Ces algorithmes sont basés sur le même principe que les algorithmes d'extraction de motifs séquentiels classiques. Pour être compté comme supportant un itemset, il suffit qu'un objet contienne tous les items de l'itemset dans un même enregistrement.

Le comptage SPEEDYFUZZY consiste à comptabiliser, pour la fréquence d'une séquence, chaque objet ayant enregistré au moins une fois cette séquence. Quel que soit le degré d'appartenance, non nul, pour chaque item flou, chaque objet aura le même poids. Le principe de MINIFUZZY est le même, mais on considère qu'un objet ne contient un item que si le degré d'appartenance de cet item pour un enregistrement de l'objet dépasse le seuil minimal d'appartenance ω.

Le fonctionnement des algorithmes SpeedyFuzzy et MiniFuzzy est similaire, il est décrit en détail en annexe B. Pour chaque objet, il s'agit de scanner l'ensemble des enregistrements pour trouver la séquence candidate. Pour chaque itemset de la séquence, il est nécessaire de vérifier si le degré d'appartenance dans l'enregistrement est non nul pour SpeedyFuzzy ou supérieur au seuil ω pour MiniFuzzy. Dès que la séquence candidate est validée (la liste ordonnée des itemsets est supportée par l'objet), le parcours de la séquence de données est stoppé et la fréquence de la séquence candidate est incrémentée. Le parcours se poursuit avec l'objet suivant.
Dans le tableau 2.4, les transactions qui valident la séquence Sx=$< ([chocolat, peu])([soda, beaucoup]) >$ avec un comptage SpeedyFuzzy sont d1 suivie de d3. Ces items sont soulignés sur la figure. Les transactions qui valident la séquence Sx avec un comptage MiniFuzzy sont d2 suivie de d3. Ces items sont en gras sur la figure.

2.2.2 TotallyFuzzy

Cet algorithme repose sur un Σ-comptage seuillé. Il s'agit de tenir compte, pour chaque objet, du poids des items dans le calcul de la fréquence d'une séquence.

Pour calculer la fréquence d'une séquence, on doit tenir compte de l'ordre des itemsets qui la compose. Ce calcul se présente donc directement sous la forme d'un algorithme. En effet, il s'agit d'agréger le degré de fréquence de chaque itemset de la séquence si celui-ci se trouve parmi les enregistrements de l'objet et que l'ordre de la séquence est respecté.
Ainsi, l'algorithme $FindTotallySeq$ réalise un parcours ordonné des enregistrements d'un objet et stocke le degré des itemsets. L'algorithme $CalcTotallySupp$ agrège ensuite le degré de fréquence de la séquence et trouve la valeur optimale pour chaque objet avant de calculer la fréquence floue $FFreq$.

Complexité spatiale vs. complexité temporelle

L'algorithme TotallyFuzzy implémente le calcul de la fréquence floue par Σ-comptage seuillé des objets supportant les séquences candidates.

[Yan04] montre que l'énumération des motifs séquentiels contenus dans une base de données est un problème NP-complet. Plus précisément, pour les algorithmes SpeedyFuzzy et MiniFuzzy, la recherche d'une séquence parmi les enregistrements de chaque objet se fait au maximum en un parcours des enregistrements, donc pour chaque séquence, le calcul de la fréquence se fait en $O(\sum_{o \in \mathcal{O}} \mathcal{R}_o) = O(\mathcal{R})$. Lors de l'utilisation de TotallyFuzzy, par contre, dans le pire des cas, la recherche de la séquence se fait pour chaque objet en $O(\mathcal{R}_o^2)$. La complexité maximale de ce comptage est donc en $O(\mathcal{R}^2)$.

En effet, l'utilisation du Σ-comptage et la recherche de la meilleure occurrence requièrent un parcours exhaustif des séquences de données afin de trouver les meilleures occurrences de chaque séquence candidate. Considérons par exemple le tableau 2.4, la première occurrence de la séquence Sx est soulignée, supportée par les transactions d1 puis d3, alors que la meilleure occurrence est la séquence supportée par les transactions d4 suivie de d6, soulignée deux fois.

Une approche naïve consisterait pour chaque *k-séquence candidate* (séquence potentiellement fréquente), en une recherche exhaustive qui implique n^k parcours de la base, où n est le nombre d'items fréquents. La seule structure conservée en mémoire est alors la liste des séquences candidates. Toutefois, cette solution est inefficace en terme de temps d'exécution.

Afin de réduire le nombre de parcours et ainsi le temps d'exécution, nous avons donc imaginé une structure de données permettant de trouver toutes les représentations de toutes les k-séquences en seulement k parcours de la base. Le temps d'exécution est alors réduit mais l'espace mémoire nécessaire augmente. Quelques optimisations ont toutefois été ajoutées afin de diminuer l'espace mémoire requis.

La base de notre implémentation est donc la structure de *chemin*. Un chemin correspond à une instanciation d'une séquence candidate dans une séquence de données.

Définition 2.4. Un *chemin* est un triplet contenant la séquence déjà trouvée, seq, l'itemset actuellement recherché $curIS$ (celui arrivant juste après seq dans la séquence candidate) et le degré de fréquence de la séquence déjà trouvée, $curDeg$.

Plusieurs chemins peuvent être initialisés pour un même objet. Pour le calcul de la fréquence, seul le chemin représentant la meilleure instanciation est conservée.

Illustration

Prenons comme exemple la séquence $Sx = < ([chocolat, peu])([soda, beaucoup]) >$ et recherchons sa meilleure occurrence dans la séquence de données du client 1. Pour cela, nous utilisons l'algorithme $FindTotallySeq$ décrit en annexe B avec un seuil ω=0.2. Ce parcours est résumé dans le tableau 2.5.

	$pth1 : (\varnothing, ([chocolat, peu]), 0)$
Après d1	$pth1 : (< ([chocolat, peu]) >, ([soda, beaucoup]), 0.25)$
Après d2	$pth1 : (< ([chocolat, peu]) >, ([soda, beaucoup]), 0.25)$ $pth2 : (< ([chocolat, peu]) >, ([soda, beaucoup]), 0.5)$
Opt.	$pth1$ est supprimé
Après d3	$pth2 : (< ([chocolat, peu])\ ([soda, beaucoup]) >, \varnothing, 0.5)$, clos $pth3 : (< ([chocolat, peu]) >, ([soda, beaucoup]), 0.5)$
Après d4	$pth2 : (< ([chocolat, peu])\ ([soda, beaucoup]) >, \varnothing, 0.5)$, clos $pth3 : (< ([chocolat, peu])\ ([soda, beaucoup]) >, \varnothing, 0.5)$, clos $pth4 : (< ([chocolat, peu]) >, ([soda, beaucoup]), 0.5)$ $pth5 : (< ([chocolat, peu]) >, ([soda, beaucoup]), 0.75)$
Opt.	$pth3$ et $pth4$ sont supprimés
Après d5	$pth2 : (< ([chocolat, peu])\ ([soda, beaucoup]) >, \varnothing, 0.5)$, clos $pth5 : (< ([chocolat, peu]) >\ ([soda, beaucoup]) >, \varnothing, 0.63)$, clos $pth6 : (< ([chocolat, peu]) >, ([soda, beaucoup]), 0.75)$
Opt.	$pth2$ est supprimé
Après d6	$pth5 : (< ([chocolat, peu]) >\ ([soda, beaucoup]) >, \varnothing, 0.63)$, clos $pth6 : (< ([chocolat, peu]) >\ ([soda, beaucoup]) >, \varnothing, 0.87)$, clos
Opt.	$pth5$ est supprimé
Deg. pour C1	0.87

TAB. 2.5 – Recherche de Sx par TOTALLYFUZZY dans la séquence de données C1

Tout d'abord, le processus est initialisé en créant un chemin vide $pth1 = (\varnothing, ([chocolat, peu]), 0)$ (TAB. 2.5, ligne 1). Ensuite, le parcours de la séquence de données commence par la transaction d1 du client 1. L'item recherché, $curIS$, est trouvé avec un degré *d1[chocolat, peu]=0.25* $\geq \omega$. Le chemin $pth1$ est donc mis à jour par $pth1.seq \leftarrow < ([chocolat, peu]) >$, $pth1.curIS \leftarrow ([soda, beaucoup])$ et $pth1.seq \leftarrow 0.25$ (TAB. 2.5, ligne 2).

Ensuite, la transaction d2 contient le premier itemset de la séquence candidate g-S, $([chocolat, peu])$. Un nouveau chemin est créé, $pth2 \leftarrow (< ([chocolat, peu]) >, ([soda, beaucoup]), 0.5)$. Comme cette transaction ne contient pas l'itemset recherché par $pth1$, le parcours de la base continue. Toutefois, avant de passer à la transaction d3, une optimisation est réalisée afin de diminuer l'espace mémoire utilisé : pour deux chemins à la même étape de la séquence candidate, seul celui ayant la meilleure valeur de $curDeg$ est conservé. Ici, $pth1.curDeg < pth2.curDeg$, $pth1$ est donc supprimé et le parcours de la base se poursuit avec $pth2$ (TAB. 2.5, ligne 3).

La transaction d3 est ensuite vérifiée. Elle contient $pth2.curIS = ([soda, beaucoup])$. Ce chemin est donc mis à jour $pth2 \leftarrow (< ([chocolat, peu])\ ([soda, beaucoup]) >, \varnothing, 0.5)$. Ce chemin est ensuite clos puisqu'il contient tous les itemsets de g-S. Cependant, cette solution peut être améliorée. C'est pourquoi, avant de modifier $pth2$, ce chemin est copié dans $pth3$. A ce stade, nous avons deux chemins : $pth2$, clos avec un degré de fréquence de 0.5 et $pth3 = (< ([chocolat, peu]) >, ([soda, beaucoup]), 0.5)$ (TAB. 2.5, ligne 4).

On vérifie ensuite la transaction d4. $pth3.curIS$ est trouvé, ce chemin est donc dupliqué, modifié et clos, avec un degré de 0.5. Comme d4 contient également le premier itemset de g-S, un nouveau chemin $pth5$ est créé. A ce stade, la liste des chemins contient 4 éléments (TAB. 2.5, ligne 5). Puis la fonction $Optimize$ supprime, pour chaque état d'avancement des chemins ceux dont le degré est le plus bas, c'est-à-dire $pth3$ et $pth4$.

Le parcours de la séquence de données se poursuit. A la date d5, $pth5$ est dupliqué en $pth6$, modifié et clos avec un degré de 0.63, le chemin $pth2$ est donc supprimé. Enfin, à d6, $pth6$ est mis à jour et clos avec un degré de 0.87. C'est ce chemin qui est conservé pour le client 1.

2.3 SMT-FUZZY : Extraction de motifs séquentiels flous

2.3.1 Algorithme

L'algorithme général permettant d'extraire les motifs séquentiels flous, quelque soit le comptage, SMT-Fuzzy, est une approche par niveau utilisant la structure d'arbre préfixé présentée dans [MCP98].

Cet algorithme est détaillé ALG. 2. Quelque soit la fonction de comptage utilisée, le fonctionnement est le suivant. Tout d'abord, la fréquence de chaque item flou est calculée afin d'éliminer les non fréquents. Tant que la génération de candidats est possible, la fonction $generate$ construit les séquences candidates de taille k en combinant les séquences fréquentes de taille k-*1*. Ensuite, la base est parcourue et les fréquences sont calculées en utilisant $CalcFuzzyFreq$. Enfin, toute les séquences de taille k non fréquentes sont élaguées. A la fin du processus, nous obtenons un arbre préfixé contenant toutes les séquences fréquentes de la base de données avec leur fréquence sur chaque feuille de l'arbre.

Optimize - **Input** : DB, une base de données, $minFreq$
Ouput : PT, arbre préfixé contenant les séquences fréquentes

find-1-Frequent() ;
Tant que (#candidate > 0) **faire**
  $generate(PT.\text{depth}+1)$;
  Pour chaque sequence s in PT **faire**
    $CalcFuzzyFreq(s)$;
  Fin Pour
  $prune(PT)$;
Fin Tant que
Retourner (PT) ;

ALG. 2 : *SMT-Fuzzy*

Comme pour PSP, la complexité globale de SMT-Fuzzy dépend seulement de la longueur des séquences candidates et donc du calcul de la fréquence. En effet, l'exécution des fonctions $generate$ et

$prune$ est négligeable en comparaison de celle de $CalcFuzzyFreq$.

2.3.2 Trois niveaux d'information

Selon le niveau de fuzzification paramétré par l'utilisateur, la fonction $CalcFuzzFreq$ appelle :
- $FindSpeedySeq$; SMT-Fuzzy est alors exécuté avec un comptage SpeedyFuzzy. Il retourne alors les motifs séquentiels correspondant à une vue globale de la base de données, indiquant pour chacun la fréquence des différents attributs quantitatifs.
- $FindMiniSeq$; SMT-Fuzzy est alors exécuté pour réaliser un comptage seuillé MiniFuzzy. Il retourne alors les motifs séquentiels correspondant aux attributs quantitatifs vérifiant un seuil minimum de pertinence.
- $FindTotallySeq$; SMT-Fuzzy est alors exécuté pour réaliser un comptage TotallyFuzzy. Il retourne alors des motifs séquentiels plus informatifs puisqu'ils dépendent du niveau de pertinence de chaque item flou ainsi que de leur fréquence d'apparition dans le jeu de données.

A partir de la base de séquences Tab. 2.1, les motifs séquentiels flous obtenus, après transformation de la base en base de degrés d'appartenance, avec $minFreq = 52\%$ et un seuil de pertinence minimum $\omega = 0.49$ pour MiniFuzzy et TotallyFuzzy sont donnés par le tableau 2.6.

Motifs séquentiels avec SpeedyFuzzy	<([video, moyen])>	75%
	<([chocolat, peu])([dentifrice, moyen])>	75%
	<([chocolat, beaucoup])([dentifrice, moyen])>	75%
	<([chocolat, beaucoup][soda, beaucoup])([soda, beaucoup])([soda, beaucoup])>	75%
	<([chocolat, peu][soda, beaucoup])([soda, beaucoup])([soda, beaucoup])>	75%
Motifs séquentiels avec MiniFuzzy	<([video, moyen])>	75%
	<([chocolat, beaucoup])>	75%
	<([chocolat, peu])([dentifrice, moyen])>	75%
	<([chocolat, peu][soda, beaucoup])([soda, beaucoup])([soda, beaucoup])>	75%
Motifs séquentiels avec TotallyFuzzy	<([video, moyen])>	62.5%
	<([chocolat, beaucoup])>	56.5%
	<([chocolat, peu])([dentifrice, moyen])>	53.5%
	<([chocolat, peu][soda, beaucoup])([soda, beaucoup])([soda, beaucoup])>	57%

Tab. 2.6 – Motifs séquentiels découverts

Les items flous fréquents sont les mêmes pour les trois méthodes de comptage. La différence se situe dans le nombre et la longueur des séquences. Pour une même fréquence minimale, les motifs extraits sont plus longs avec MiniFuzzy ou SpeedyFuzzy qu'avec TotallyFuzzy, en raison du Σ-comptage.

Cette réduction du nombre de motifs peut être utilisée pour des bases de données contenant un grand nombre de motifs fréquents afin d'en extraire les plus significatifs. L'avantage de TotallyFuzzy est, en effet, d'être une méthode plus sélective et donc d'extraire les motifs séquentiels les plus pertinents. L'utilisateur disposera donc d'une sélection de motifs et non plus d'une liste importante qu'il devrait trier.

Chapitre 3

Expérimentations

Dans ce chapitre, nous présentons une comparaison des performances des algorithmes, SPEEDYFUZZY, MINIFUZZY et TOTALLYFUZZY, introduits dans le chapitre précédent avec les performances de l'algorithme PSP [MCP98], qui utilise les mêmes principes d'implémentation pour extraire des motifs séquentiels sur données binaires. Nous comparons également l'algorithme TOTALLYFUZZY avec l'approche [HLW01] et étudions l'influence des différents paramètres (choix de l'opérateur d'agrégation et de la valeur du seuil ω) sur le comportement de TOTALLYFUZZY. Dans la section 3.2 de ce chapitre, nous présentons les résultats obtenus lors de l'analyse de deux jeux de données réelles.

3.1 Données synthétiques

Les jeux de données utilisés dans cette section ont été créés en plusieurs étapes. Tout d'abord des bases de données quantitatives ont été générées en utilisant une version modifiée[1] du générateur DatGen [Mel]. Ensuite, nous avons réalisé un partitionnement automatique de type équi-répartition[2] grâce à une implémentation modifiée du module *DiscretizeFilter* de *Weka* [WF00]. Cette nouvelle version propose un découpage en sous-ensembles flous plutôt qu'en intervalles stricts. Une fois le partitionnement réalisé, la base de degrés d'appartenance est générée. Le tableau 3.1 recense les différents paramètres utilisés pour la génération des données ainsi que les caractéristiques des jeux de données utilisés pour les expérimentations présentées ci-après.

Nom	Nb d'objets dans la base O	Nb d'enregistrements en base R	Nb moyen d'items par enregistrement I	Nb d'items en base X	Valeur maximale des quantités Q	Nb de sous-ensembles flous par item P
ob5rec100I10	5 000	100 000	10	100	10	3
ob5rec200I10	5 000	200 000	10	100	10	3
ob5rec250I10	5 000	250 000	10	100	10	3
ob5rec350I10	5 000	350 000	10	100	10	3
ob5rec450I10	5 000	450 000	10	100	10	3
ob5rec500I10	5 000	500 000	10	100	10	3
ob10rec250I10	10 000	250 000	10	100	10	3
ob10rec500I10	10 000	500 000	10	100	10	3
ob5rec100I10Q100	5 000	100 000	10	100	100	3

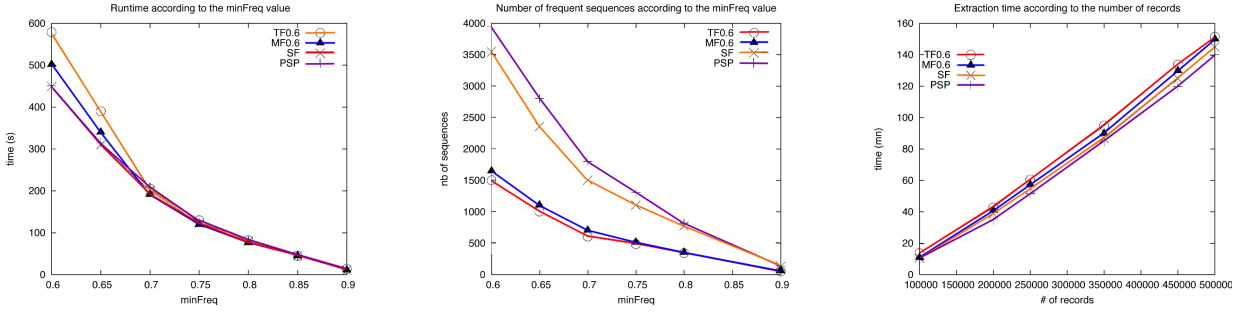
TAB. 3.1 – Valeurs des différents paramètres pour les jeux de données utilisés dans les expérimentations

[1] Notre implémentation de ce générateur de données permet en effet de générer des séquences de données et non plus seulement des transactions.

[2] Chaque intervalle contient autant d'éléments.

3.1.1 Algorithmes flous vs. PSP

Le but de ces expérimentations est de comparer les performances des algorithmes flous avec celles de PSP. La figure 3.1(a) montre l'évolution du temps d'extraction en fonction de $minFreq$. On remarque que ce temps d'extraction augmente à mesure que la valeur de $minFreq$ diminue. En effet, le nombre de motifs fréquents et de séquences candidates est plus élevé lorsque la valeur de $minFreq$ est plus basse. Les algorithmes parcourent donc plus de fois la base de données.
On peut également noter que SpeedyFuzzy est presque aussi rapide que PSP bienqu'il parcourt environ trois fois plus d'items. En effet, chaque item quantitatif a été transformé en un item binaire pour la base fouillée par PSP, et en trois sous-ensembles flous pour les algorithmes flous. TotallyFuzzy et MiniFuzzy sont légèrement plus lents.



(a) (b) (c)

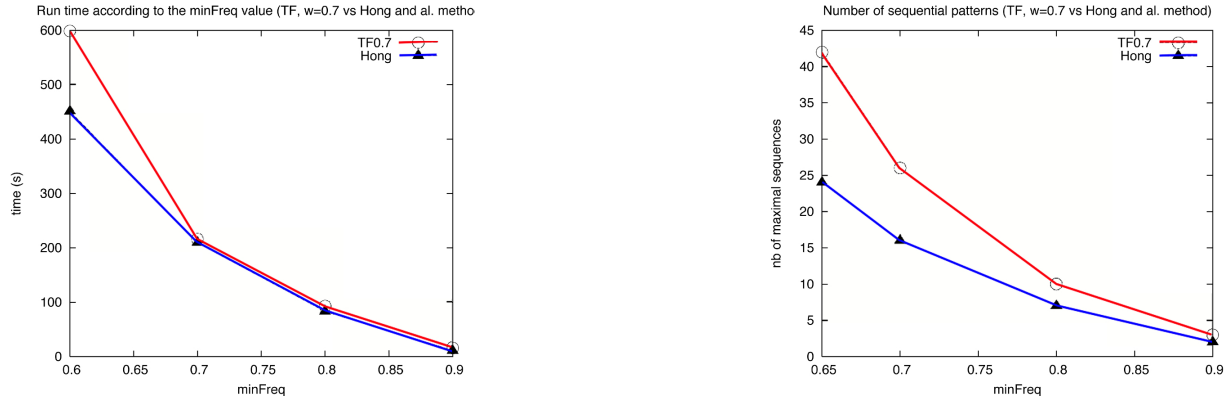
Fig. 3.1 – (a) : Temps d'extraction en fonction de $minFreq$ pour ob5rec100l10 ; (b) : Nombre de séquences fréquentes en fonction de $minFreq$ pour ob5rec100l10 ; (c) : Temps d'extraction en fonction du nombre d'enregistrements dans la base de données, $minFreq$=0.6.

Si on compare le nombre de séquences fréquentes extraites pour une même valeur de $minFreq$, on peut constater sur la figure 3.1(b) que MiniFuzzy et TotallyFuzzy extraient nettement moins de séquences que SpeedyFuzzy et PSP. Cela est dû au mode d'évaluation de la fréquence. Comme les deux algorithmes MiniFuzzy et TotallyFuzzy, ne conservent que les items dont le degré est supérieur au seuil ω, items pertinents pour l'utilisateur, le nombre de séquences fréquentes est nécessairement réduit par rapport aux motifs extraits avec SpeedyFuzzy ou PSP.

Enfin, la figure 3.1(c) montre le temps d'extraction en fonction du nombre d'enregistrements dans la base de données, pour 5000 objets et $minFreq = 0.6$. On remarque que le comportement général des algorithmes flous est le même que celui de PSP.

3.1.2 TotallyFuzzy vs. Hong et al.

Le but de ce paragraphe est de montrer les avantages de notre méthode par rapport à l'approche de Hong et al. [HLW01]. En particulier, nous étudions la perte d'information résultant de leur pré-sélection des items flous. Dans nos expérimentations, nous comparons l'algorithme TotallyFuzzy, solution se rapprochant le plus de cette approche, à une implémentation de l'approche Hong et al. utilisant les mêmes structures de données.



(a) (b)

FIG. 3.2 – (a) : Temps d'extraction en fonction de $minFreq$ pour ob5rec100l10 avec TOTALLYFUZZY (ω = 0.7) et [HLW01] ; (b) : Nombre de séquences fréquentes extraites en fonction de $minFreq$ pour ob5rec100l10, avec TOTALLYFUZZY (ω = 0.7) et [HLW01].

Comme le montre la figure 3.2(a), la sélection des items flous opérée par [HLW01] réduit le temps d'exécution de façon significative. Cette solution apparaît nettement plus rapide que TOTALLYFUZZY. Cependant, nous pouvons constater sur le tableau 3.2 que certains items flous (e.g. *[1000, 3]* et *[990, 3]*), de fréquence très significative (resp. *88.44%* et *91.46%*), sont trouvés par TOTALLYFUZZY mais pas par la méthode Hong et al. Ces items sont en effet supprimés durant la phase de sélection.

Items fréquents	[960, 1]	[970, 1]	[980, 1]	[980, 3]	[990, 1]	[990, 2]	[990, 3]	[1000, 1]	[1000, 2]	[1000, 3]	Nb de fréquents
TOTALLYFUZZY (ω=0.7)	65.12%	74.32%	85.36%	73.18%	97.24%	70.28%	91.46%	95.94%	66.76%	88.44%	10
Hong et al.	65.12%	74.32%	85.36%	abs.	97.24%	abs.	abs.	95.94%	abs.	abs.	5

TAB. 3.2 – Items fréquents extraits par TOTALLYFUZZY (ω=0.7) et l'approche Hong et al., pour $minFreq = 60\%$ sur la base ob5rec100l10.

En conséquence, le nombre de séquences fréquentes maximales trouvées par TOTALLYFUZZY est plus élevé que le nombre de séquences extraites par la méthode de Hong et al.. En effet, les séquences fréquentes résultant des cinq items fréquents écartés ne sont jamais générées et ne sont donc pas découvertes. La figure 3.2(b) montre cette perte d'information. Pour $minFreq$=70%, l'approche de Hong et al. ne trouve qu'environ deux tiers des motifs extraits par TOTALLYFUZZY.

3.1.3 Opérateurs

L'implémentation de la troisième définition de la fréquence d'une séquence floue, correspondant à TOTALLYFUZZY, nécessite la spécification d'un opérateur de t-norme afin de déterminer le degré d'un itemset, d'un opérateur d'agrégation pour calculer le degré d'une séquence et d'un opérateur de t-conorme afin de déterminer, parmi plusieurs inclusions d'une même séquence candidate dans une séquence de données, laquelle est la plus représentative de la séquence de données.

Concernant les opérateurs de t-norme et t-conorme, nous avons choisi d'utiliser respectivement les opérateurs min et max. En effet, nous souhaitons que lorsque tous les items d'un itemset sont trouvés

dans un enregistrement, chacun avec un degré suffisamment significatif (i.e. supérieur au seuil ω), cet itemset soit considéré comme également significatif et que son degré soit au moins égal au degré de l'item qu'il contient ayant le degré minimal. Or, la seule t-norme permettant de satisfaire cette condition est la t-norme min, grâce à sa propriété d'idempotence.

Pour l'opérateur d'agrégation, deux points de vue peuvent être envisagés. L'inclusion d'une séquence candidate dans une séquence de données peut être vue comme l'intersection des sous-ensembles flous correspondant à chacun des itemsets et l'agrégation se fait par une t-norme. Pour les mêmes raisons que précédemment, nous privilégierons dans ce cas l'opérateur min.

Toutefois, on peut considérer que l'inclusion de la séquence est déjà prise en compte lors du parcours de la base à la recherche d'enregistrements incluant les itemsets avec un degré suffisamment élevé. Dans ce cas, le niveau d'inclusion des différentes items dans la séquence de données est déjà intégré dans le calcul du degré de chaque itemset par l'utilisation d'une t-norme. Il n'y a alors aucune raison de ne pas considérer tous les itemsets d'une séquence de façon équivalente et dans ce cas, on peut envisager l'utilisation d'une agrégation par la moyenne. Afin d'assurer la complétude des résultats, il sera alors nécessaire d'imposer que le seuil minimal ω soit supérieur ou égal à la valeur de $minFreq$, afin d'utiliser la technique proposée dans [PHW02] pour le traitement des contraintes d'agrégation forte.

Nous avons donc implémenté chacune des deux approches en utilisant le min et le max respectivement comme t-norme et t-conorme mais des opérateurs d'agrégation différents. Les résultats décrits sur la figure 3.3 montrent que l'agrégation par le *min* est très sélective et qu'elle conduit à l'élagage de très nombreux motifs.

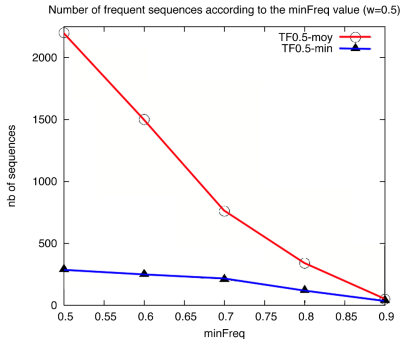
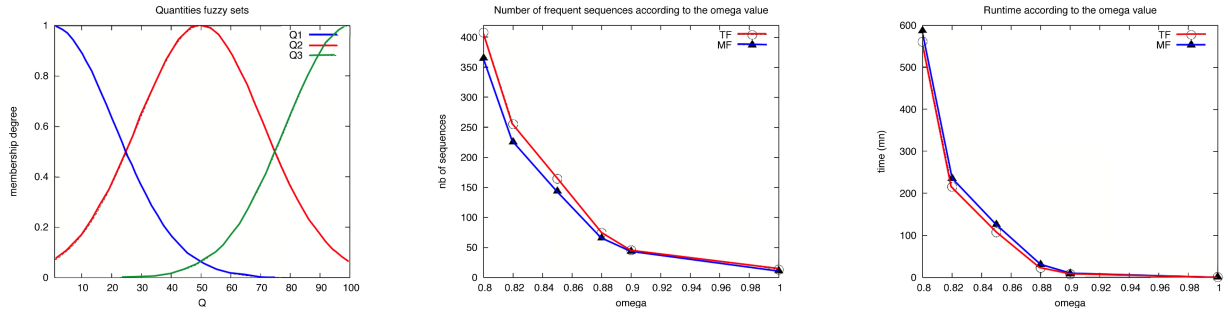


FIG. 3.3 – Nombre de séquences fréquentes en fonction de $minFreq$ selon l'opérateur d'agrégation pour ob5rec100l10

3.1.4 Réponse aux variations du paramètre ω

Le paramètre ω introduit pour les algorithmes MINIFUZZY et TOTALLYFUZZY permet de réguler le comptage. Il peut en effet être utilisé comme un filtre de la pertinence des enregistrements selon le degré d'appartenance des items flous qui les composent. Une valeur élevée de ce paramètre permet à l'utilisateur de ne sélectionner que les motifs séquentiels bien représentés dans chacune les séquences de données qui les supportent. Au contraire, une valeur basse du paramètre ω permet l'extraction de motifs séquentiels supportés par plus d'objets, mais ces motifs n'apportent que peu d'information sur la manière dont ils sont supportés par chacune des séquences de données.

Afin de montrer la réponse des algorithmes TotallyFuzzy et MiniFuzzy aux variations du paramètre ω, nous utilisons le jeu de données synthétiques (ob5rec100l10Q100) dont les quantités de chaque attribut peuvent varier entre 1 et 100. Les fonctions d'appartenance des sous-ensembles flous obtenus, décrites par la figure 3.4(a), permettent ainsi une variation plus précise du degré d'appartenance.



(a) (b) (c)

Fig. 3.4 – (a) : Sous-ensembles flous pour des quantités allant de 1 à 100 pour ob5rec100l10Q100 ; (b) : Nombre de motifs séquentiels extraits par TotallyFuzzy et MiniFuzzy pour ob5rec100l10Q100 ; (c) : Temps d'extraction de TotallyFuzzy et MiniFuzzy pour ob5rec100l10.

La figure 3.4(b) montre que le nombre de séquences extraites augmentent à mesure que la valeur d'ω diminue. En effet, les algorithmes sont moins sélectifs, ils conservent donc plus de séquences, moins significatives, découvrant ainsi plus de séquences fréquentes. En conséquence, le temps d'extraction augmente également alors que la valeur d'ω diminue, comme le montre la figure 3.4(c), car l'algorithme réalise plus de parcours sur la base.

3.2 Données réelles

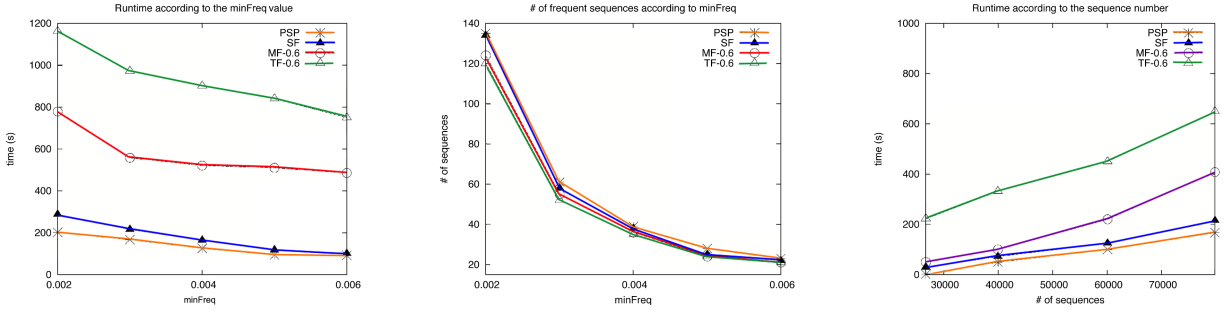
Différentes expérimentations ont été menées sur des bases de données issues d'applications concrètes. Les premières présentées dans ce rapport ont été réalisées sur des fichiers de logs d'accès à un site internet. Le but de ces expérimentations était de montrer les connaissances additionnelles apportées par l'utilisation des motifs séquentiels flous par rapport à l'utilisation de motifs séquentiels sur données binaires, notamment en ce qui concerne le nombre de connexions répétées sur une même page au cours d'une même session. La seconde partie de ces expérimentations concernent la base de données des noms de marque de l'INPI et l'analyse de la composition des noms déposés.

3.2.1 Web Access Log Mining

La quantité de données provenant du web augmente de façon exponentielle : les URLs accédées, le nombre de requêtes ou les durées de connexions sont stockées automatiquement par les serveurs web et conservées dans des fichiers de logs. Analyser ces données peut fournir nombre d'informations intéressantes en vue de l'amélioration de la qualité de service ou pour cibler certains internautes ou clients de commerces en ligne. Dans ce contexte, de nombreux travaux ont été proposés afin d'extraire des schémas d'utilisation et des profils d'utilisateurs [YJGMD96, ZXH98, SF99, DOQT01]. En particulier, [MPC99] propose d'extraire des connaissances à partir de séquences de visites de pages web, en utilisant les motifs

séquentiels et l'algorithme PSP. Nous proposons donc de conduire quelques expérimentations afin de montrer l'apport d'information permis par l'utilisation des algorithmes flous par rapport aux méthodes classiques d'extraction de motifs séquentiels. Dans notre cas, les données utilisées sont les logs d'accès au site web d'un laboratoire. Ces données ont été préparées afin d'identifier d'une part les pages fréquemment visitées, aussi bien grâce aux algorithmes flous qu'avec PSP, mais également pour découvrir les pages visitées de façon répétées au cours d'une même session.

Le jeu de données utilisé regroupe le nombre d'accès à une page, durant la même demi-journée par un utilisateur, identifié par son IP. Il contient 27209 pages web, visitées par 79756 IPs différentes, au cours d'une période de 16 jours (32 demi-journées). Comme nous l'avons expliqué précédemment, les quantités sont converties en degrés d'appartenance afin d'être fouillées par les algorithmes flous. Le partitionnement est réalisé de la même façon que pour les données synthétiques, en construisant 3 sous-ensembles flous par URL.
Nous avons commencé par extraire les motifs séquentiels flous grâce à SpeedyFuzzy, afin d'extraire l'information générale concernant les URL souvent visitées de façon répétée au cours d'une même session. Ensuite, nous avons extrait des informations plus détaillées en utilisant TotallyFuzzy. Ces expérimentations confirment le comportement global observé sur données synthétiques.



(a) (b) (c)

FIG. 3.5 – (a) : Temps d'extraction en fonction de $minFreq$ pour 79756 séquences ; (b) : Nombre de séquences fréquentes extraites en fonction de $minFreq$ pour 79756 séquences ; (c) : Temps d'extraction en fonction du nombre de séquences en base, pour $minFreq$=0.2%.

La figure 3.5(a) montre l'évolution du temps d'extraction en fonction de $minFreq$. On remarque que le temps d'extraction augmente à mesure que $minFreq$ diminue. Si on compare le nombre de fréquences extraites pour un même support minimum, la figure 3.5(b) montre que MiniFuzzy et TotallyFuzzy extraient moins de séquences fréquentes que SpeedyFuzzy et PSP, comme observé pour les données synthétiques. La figure 3.5(c), enfin, montre le temps d'extraction en fonction du nombre d'enregistrements dans la base pour $minFreq = 0.2\%$. On constate que le comportement des algorithmes flous reste similaire à celui de PSP.

En ce qui concerne l'analyse qualitative des motifs séquentiels obtenus, les mêmes URL sont découvertes avec les algorithmes flous ou avec PSP. L'avantage des algorithmes flous est de fournir une information complémentaire. Ainsi, alors que l'algorithme PSP indique qu'une URL, par exemple, celle numérotée 139, est fréquemment accédée, l'algorithme TotallyFuzzy nous indique que cette URL est accédée de façon fréquente, 2 à 5 fois par session, et que ce comportement est significatif pour la plupart des objets le supportant. Les motifs flous montrent également que la page d'accueil du site est

seulement une étape dans la navigation et qu'elle n'est en général visitée qu'une fois par session.

3.2.2 Composition des noms de marques déposées

Nous proposons dans cette section de présenter quelques résultats obtenus lors de la formulation de résumés linguistiques [Yag82], sous la forme de résumés flous [Kac88, KYZ00] (cf. annexe A), décrivant la composition de noms de marques déposés au près de l'INPI[3]. La base de noms de marques dont nous disposons contient plus de 2 millions de noms de marques déposés de 1961 à 2004. Ces noms de marques sont enregistrés par champ industriel comme par exemple "jeux et jouets" ou "vêtements, chapeaux, chaussures". Chaque enregistrement est composé de plusieurs attributs tels que le nom de marque, la date de dépôt, la société ou personne déposante et la (les) catégorie(s) industrielle(s) du dépôt.

L'analyse d'une telle base par un linguiste a pour but de comprendre comment les noms de marques agissent sur les consommateurs. Le but de nos analyses a donc été, à partir d'une première description statistique, de proposer des descriptions de la composition des noms de marques déposés dans certains champs d'application, en utilisant des motifs séquentiels flous obtenus par TotallyFuzzy.
Ainsi, une analyse statistique a montré que, comparé à d'autres domaines, un nombre appréciable de marques déposées dans le champ des télécommunications contiennent des chiffres et des symboles. Ces résultats étant atypiques, nous avons cherché à décrire de façon plus précise la composition de ces marques.

Une fois isolées, les marques déposées dans la catégorie "télécommunications" représentent une base de 480 000 enregistrements de noms composés de 1 ou plusieurs mots. Le but est d'extraire des motifs exprimant, par exemple, que "environ 1/3 des marques sont composées d'un mot contenant beaucoup de lettres, puis d'un mot contenant peu de chiffres et enfin d'un mot contenant peu de ponctuations et peu de lettres". Cette proposition est l'interprétation du motif séquentiel *<([lettres, beaucoup])([chiffres, peu])([ponctuation, peu],[lettres, peu])>*.
Dans ce cas, nous pouvons considérer les noms de marques composés de un ou plusieurs mots : un nom composé d'un seul mot apparaîtra comme un itemset alors qu'un nom composé de plusieurs mots sera modélisé comme une séquence. Afin d'extraire de tels motifs, les données de la base INPI sont converties au format [Id_TM, #mots, Item Flou, Degré], comme décrit par le tableau 3.3. Les items flous sont décrits par leur fonction d'appartenance, figure 3.6. Ces sous-ensembles flous ont été construits à partir de la connaissance du linguiste et de la composition des mots du vocabulaire commun.

Formalisme de la fouille		trademark database
Objet	↔	une marque
étiquette temporelle	↔	numéro d'ordre du mot dans le nom de marque
item flou	↔	# de signes, de lettres, de chiffres, de ponctuations ou de symboles dans le mot

Tab. 3.3 – Conversion de la base de données pour une recherche de motifs intranom

Certains motifs montrent que "presque la moitié des noms de marque dans le domaine des télécommunications contiennent 3 mots avec beaucoup de caractères. Le premier et le dernier sont composés de

[3]L'Institut National de la Propriété Industrielle (INPI) est un établissement public français, placé sous la tutelle du ministère chargé de l'industrie. Il a pour missions de recevoir les dépôts et délivrer les titres de propriété industrielle (brevets, marques, dessins et modèles), de participer à l'élaboration du droit de la propriété industrielle, de mettre à la disposition du public toute information nécessaire pour la protection des titres de propriété industrielle et enfin, de centraliser les registres du commerce et des sociétés des différents tribunaux dans le Registre National du Commerce et des Sociétés, ainsi que le Répertoire Central des Métiers

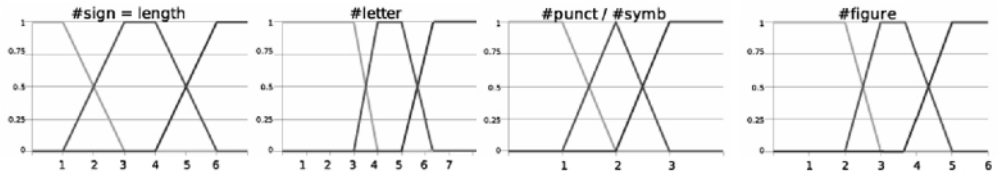


FIG. 3.6 – Sous-ensembles flous des items pour la fouille intranom

beaucoup de caractères et de lettres. Le second mot contient peu de lettres ou une quantité moyenne de caractères." (<([#signes, bcp.][#lettres, bcp.]) ([#lettres, peu]) ([#signes, bcp.][#lettres, bcp.])> (42%), <([#signes, bcp.][#lettres, bcp.]) ([#signes, moyen]) ([#signes, bcp.][#lettres, bcp.])> (42%)).

En diminuant progressivement les valeurs de $minFreq$ et ω, nous n'avons pas trouvé de motifs contenant des chiffres ou des symboles avec TOTALLYFUZZY. Ces résultats ne coïncidant pas avec les analyses statistiques, nous avons réalisé d'autres analyses avec SPEEDYFUZZY et MINIFUZZY. Les mesures statistiques ont alors été vérifiées.

Notre interprétation de ces résultats est que les sous-ensembles flous construits à partir des connaissances du linguiste sur le vocabulaire du langage courant ne sont pas adaptés à la composition des noms de marque. Il serait donc intéressant de conduire de nouvelles analyses en construisant les sous-ensembles flous de façon automatique grâce à une segmentation floue, comme présenté dans [FWS+98] et [Gye00b].

Dans un second temps, nos algorithmes pourraient également permettre d'extraire les tendances de l'évolution de la composition des noms de marque afin de préciser les résultats obtenus avec la méthode [LAS97] destinée à découvrir des tendances dans des bases de données textuelles. Ces analyses devraient ensuite être approfondies en utilisant des contraintes temporelles du type de celles présentées dans la partie suivante de ce mémoire.

Discussion

Dans cette partie, nous avons présenté une approche complète et efficace pour l'extraction de motifs séquentiels flous, permettant le traitement de séquences de données numériques telles que les données démographiques ou des relevés de capteurs, alors que les algorithmes existants ne permettaient d'extraire qu'une partie de l'information disponible dans les bases de données quantitatives.

Notre approche définit clairement et de façon complète les différents concepts et principes associés aux motifs séquentiels flous, à travers un cadre générique pour l'extraction de tels motifs. Ce cadre permet de choisir entre trois niveaux de fuzzification et ainsi entre trois niveaux d'information : présence et fréquence de certains attributs quantitatifs ou de séquences, fréquence des séquences composées d'items flous suffisamment significatifs, et enfin, fréquence pondérée par la pertinence des attributs quantitatifs composant les séquences. Ce cadre permet également de considérer les attributs quantitatifs de façon binaire et d'extraire ainsi des motifs séquentiels symboliques. Pour cela, chaque attribut quantitatif est assimilé à un sous-ensemble flou pour lequel la fonction d'appartenance vaut 1 lorsque la quantité est différente de 0, et 0 sinon. Les motifs séquentiels sont alors extraits sur cette base binaire en utilisant l'algorithme SpeedyFuzzy. Enfin, l'utilisation d'une partition floue forte ainsi qu'un seuil minimum d'appartenance ω=0.5 avec l'algorithme MiniFuzzy permet de revenir à des intervalles discrets.

Les différents niveaux de fuzzification sont pris en charge par trois algorithmes de dénombrement des séquences potentiellement fréquentes. Chacun d'entre eux, SpeedyFuzzy, MiniFuzzy ou TotallyFuzzy, permet d'extraire certaines séquences selon les besoins de l'utilisateur. Toutefois, en terme de performances, ces algorithmes ne sont pas équivalents. L'information extraite par TotallyFuzzy est plus détaillée et plus précise, mais elle requiert un parcours exhaustif de la base de données, tandis que SpeedyFuzzy permet d'identifier les séquences fréquentes, sans détail supplémentaire, de façon plus efficace. Il est alors nécessaire de réaliser un compromis entre efficacité de traitement et pertinence des résultats. En effet, MiniFuzzy et TotallyFuzzy permettent de filtrer les items flous retenus pour le calcul de la fréquence en fonction de leur degré d'appartenance. L'utilisateur dispose ainsi d'un moyen de diminuer le nombre de motifs séquentiels obtenus à l'issu de la fouille et peut retenir les séquences fréquentes ne contenant que les items flous les mieux supportés dans la base.

Par ailleurs, nous avons pu constater au cours de nos expérimentations, sur données réelles, que les résultats obtenus ne sont pas toujours conformes à la connaissance experte que l'on peut avoir des données. La définition de mesures de qualité des motifs extraits, du même genre que celles formulées pour les règles d'association floues [DHP03, CCK04, DHP06] serait d'un grand intérêt. Notamment afin d'évaluer la qualité des motifs extraits selon le partitionnement réalisé sur les attributs quantitatifs.

En effet, il existe de nombreuses façons de construire une partition d'un univers numérique, que ce soit une discrétisation en intervalles ou une partition en sous-ensembles flous. Certains travaux, ce sont intéressés à développer des techniques souvent basées sur une segmentation floue [Gye00b, HK01, VCB06]

afin de construire une partition optimale dépendant uniquement du jeu de données d'entrée.
S'il peut être intéressant de comparer les motifs extraits à partir de telles partitions, par rapport à ceux découverts au cours de nos différents tests, on peut s'interroger sur l'inconvénient de négliger la connaissance experte. On pourrait donc envisager l'utilisation de méthodes de segmentation semi-supervisées afin de constituer une partition floue tenant à la fois compte de la distribution des données, mais également de la connaissance du domaine.

Deuxième partie

Contraintes de temps étendues

Choisir son temps, c'est gagné du temps.
Francis Bacon (1561-1626) — *Essays, Civil and Moral*

La découverte d'épisodes récurrents à partir d'une longue séquence [MTV97, RPT05] ou de bases de séquences [AS95, MCP98] peut aboutir à l'extraction de très nombreux schémas. Afin de mieux cibler les connaissances extraites et éventuellement réduire le nombre de motifs obtenus, des techniques ont été proposées afin de prendre en compte un certain nombre de contraintes entre les évènements comme par exemple la durée minimale ou maximale séparant deux évènements, [SA96b, Zak00, MPT04, MR04], ou encore des contraintes d'expressions régulières ou de répétitions, [GRS02, CMB02, LRBE03, ALB03]. C'est dans ce cadre qu'a été introduite la recherche de motifs séquentiels généralisés dans [SA96b]. Différents algorithmes ont été proposés afin de gérer ces contraintes.

Dans cette partie, nous proposons plus de flexibilité à l'utilisateur lors de la spécification de contraintes temporelles entre les éléments d'une séquence, ainsi qu'un indice d'évaluation des contraintes temporelles utilisées pour générer les motifs séquentiels extraits. Dans le chapitre 1, nous présentons les différents travaux proposés afin d'extraire des motifs séquentiels sous contraintes (aussi bien sur les items que sur les séquences). Le chapitre 2 détaille ensuite notre proposition de contraintes de temps étendues pour les motifs séquentiels généralisés. Enfin, le chapitre 3 présente l'algorithme que nous avons développé afin de gérer ces contraintes et d'extraire la précision temporelle de chaque motif séquentiel. Nous concluons cette partie par une brève discussion sur les apports de la précision temporelle.

Chapitre 1

Motifs séquentiels et contraintes de temps

Dans ce chapitre, nous présentons dans une première section la recherche de motifs séquentiels sous contraintes. La section 1.2 est ensuite consacrée aux motifs séquentiels généralisés proposés dans [SA96b]. La section 1.3 est destinée à détailler plus précisément les contraintes de temps ainsi que leur utilisation. Enfin, la section 1.4 expose les différents algorithmes qui ont été proposés afin de prendre en compte les contraintes de temps lors de la découverte de motifs séquentiels.

1.1 Contraintes pour la recherche de séquences

Le nombre de séquences extraites est de plus en plus élevé, dû à la croissance des données stockées. Il apparaît donc nécessaire de trier les motifs séquentiels proposés à l'utilisateur en fonction de ses besoins. C'est pourquoi de nombreux travaux ont visé l'utilisation de contraintes lors de l'extraction de motifs, la première proposition s'appliquant aux itemsets fréquents et à la recherche de règles d'association avec des contraintes sur la présence ou l'absence de certains items [SVA97].

Dans [MI96, WMZ02], les contraintes sont vues comme des filtres, qui sont utilisés sur la base de données en entrée et non en post-traitement ou sur les candidats afin de réduire la quantité à fouiller. En utilisant cette idée, [Woj01, Woj03] développe un langage de requêtes pour les bases de séquences afin de rechercher des motifs séquentiels spécifiques, répondant à certaines contraintes.

[Zak00] propose un algorithme efficace, cSPADE, permettant de rechercher des motifs séquentiels en prenant en compte plusieurs contraintes syntaxiques sur les séquences : la longueur des séquences et des itemsets, la présence ou l'absence de certains items ou encore des contraintes temporelles entre les évènements consécutifs de la séquence.

[PHW02] présente également un algorithme de recherche de motifs séquentiels sous contraintes, basé cette fois sur le principe de pattern-growth. Outre les contraintes citées précédemment, cet algorithme permet également d'appliquer des opérations d'agrégation sur certains items ou de spécifier des expressions régulières sur un ensemble d'items.

[GRS02] utilise également des expressions régulières comme outil de spécification de contraintes qui permet à l'utilisateur de cibler les motifs séquentiels qui seront extraits, via l'algorithme SPIRIT. Cet algorithme a ensuite été amélioré grâce à l'utilisation d'une structure plus efficace par [ALB03].

Les travaux [CMB02] quant à eux ne proposent pas d'utiliser d'expressions régulières mais un motif de référence, donné par l'utilisateur. L'algorithme de recherche des fréquents utilise alors ce motif afin

d'extraire les motifs séquentiels qui lui sont similaires.

D'autres propositions ont eu pour but d'apporter plus de souplesse dans la spécification des contraintes. Ainsi, les propositions [BB05, AO04] définissent un formalisme pour des contraintes assouplies portant sur les items de motifs fréquents (séquentiels ou non), tandis que [MB06] s'intéresse à l'extraction de motifs approchés grâce à des critères de recherche flexibles. [BB05] utilise ainsi la logique floue afin de définir un cadre pour la recherche d'itemsets fréquents sous contraintes, dont les seuils sont flous. Les contraintes peuvent alors être respectées partiellement. Ce formalisme permet également de pondérer chacune des contraintes par rapport aux autres.
L'objectif de [AO04] est de ne pas supprimer d'éventuels motifs qui pourraient être intéressants pour l'utilisateur mais qui sont inattendus et ne respectent pas les contraintes spécifiées initialement. Pour ce faire, des contraintes d'expression régulière sur la forme des motifs sont appliquées en post-traitement. Les motifs sont alors ordonnés selon qu'ils respectent les contraintes, qu'ils ne les respectent que partiellement ou qu'ils ne les respectent pas du tout.

La difficulté de l'extraction de motifs sous contraintes posent des problèmes algorithmiques. Afin d'améliorer les performances et de réduire le temps de l'extraction, deux approches peuvent être envisagées : l'application des contraintes en pré-traitement sur les données avant la fouille, ou bien l'intégration des contraintes lors de la fouille. Le choix de l'une ou l'autre des approches dépend du type des contraintes à prendre en compte (cf. annexe D). Certaines sont en effet plus simple à intégrer aux algorithmes de fouille alors que d'autres ne peuvent être prises en compte facilement. Il s'agit alors de faire un compromis afin d'associer les deux. Ce compromis dépend également souvent du jeu de données [BGMP03a, BJ00]. Ainsi, [BGMP03b] propose une approche à deux phases, les contraintes sur le contenu et les caractéristiques des itemsets étant utilisées afin de réduire l'espace de recherche. Les contraintes monotones, plus avantageuses, sont utilisées sur la base d'entrée (ou en pré-traitement), les contraintes anti-monotones quant à elles, sont appliquées en post-traitement, sur les itemsets fréquents.

Enfin, quelques travaux se sont intéressés aux contraintes sur les dates et le temps dans les séquences ou encore aux répétitions. [LRBE03] traite ainsi de la recherche de motifs séquentiels dans des bases de données à fortes répétitions d'items, développant l'algorithme GoSPADE afin de pallier les faiblesses des algorithmes classiques sur ce type de bases. [MR04] propose de déterminer la fenêtre optimale à utiliser lors de l'extraction d'épisodes dans une séquence de données. Toutefois ce travail est difficilement adaptable à l'extraction de motifs séquentiels.

Très récemment, des travaux ont été proposés afin de fournir une indication supplémentaire à l'utilisateur sur les séquences fréquentes extraites. [GNP06] annote ainsi temporellement les séquences fréquentes. Il ne s'agit pas pour l'utilisateur de spécifier des contraintes de temps mais d'obtenir la durée la plus typique, séparant dans la base les évènements des séquences trouvées fréquentes, présentée sous la forme d'un intervalle.

On peut noter que tous ces travaux reposent sur la bonne connaissance de l'utilisateur des données dont il dispose, d'une part, et de ce qu'il souhaite extraire – des contraintes à spécifier – d'autre part. Nous nous sommes concentrés sur les contraintes temporelles que peut choisir de spécifier un utilisateur, avec pour objectif d'offrir une certaine flexibilité dans le choix de ces contraintes. La section suivante présente plus en détail ce que sont les motifs séquentiels généralisés qui sont des motifs extraient en considérant, entre autres, des contraintes de temps.

1.2 Motifs séquentiels généralisés

Les motifs séquentiels généralisés ont été proposés par [SA96b]. Ils permettent de prendre en compte différents niveaux de granularité dans la spécification des items ainsi que dans le découpage des transactions. L'utilisateur peut alors rechercher des motifs qui correspondent plus précisément à ses besoins en terme de composition ou d'organisation temporelle.

1.2.1 Généralisation des items

Afin d'utiliser le plus d'informations possibles et notamment une éventuelle taxonomie des items de la base de séquences. [SA96b] propose un algorithme prenant en compte simultanément des items situés à différents niveaux de la hiérarchie.

Exemple 1.1. Une taxonomie (hiérarchie *est-un*) peut être représentée par un ou plusieurs arbres, dans lesquels les fils d'un nœud sont des spécialisations de ce nœuds. Par exemple, la taxonomie présentée figure 1.1 signifie que le jus d'oranges est un jus de fruits, que la limonade est un soda et que ces deux types sont des boissons. Par contre il n'existe aucun lien avec les yaourts qui sont des laitages.

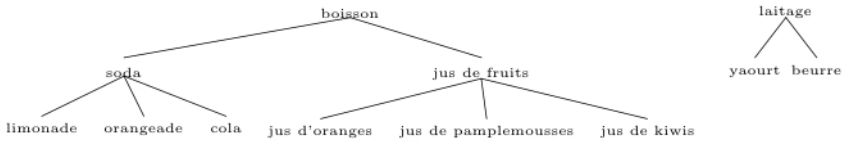


FIG. 1.1 – Exemple de taxonomie à deux racines.

La définition de l'inclusion de séquence présentée dans [AS95] est alors modifiée afin de prendre en compte les différents niveaux dans la hiérarchie. Une transaction T contient un item $x \in \mathcal{I}$ si x appartient à T ou si x est un ancêtre (i.e. un nœud plus général) d'un item de T. Une transaction T contient un itemset $y \subseteq \mathcal{I}$ si T contient tous les items de y. Une séquence de données $d =< d_1 \dots d_m >$ contient une séquence $s =< s_1 \dots s_n >$ s'il existe des entiers $i_1 < \dots < i_n$ tels que $s_1 \subseteq d_{i_1}, \dots, s_n \subseteq d_{i_n}$.

S'il n'y a pas de taxonomie, on retombe dans le cas standard de la définition [AS95].

Exemple 1.2. On considère la base de données décrite par la table 1.1, dont les produits apparaissent dans la taxonomie de la figure 1.1. En fixant le support minimum à 100%, les motifs séquentiels de cette base sont :

- <(jus d'oranges) (jus de pamplemousses)>
- <(limonade) (jus de pamplemousses)>

	Date 1	Date 2	Date 15	Date 20	Date 50
$C1$	jus d'oranges	limonade	jus de pamplemousses orangeade	- -	- -
$C2$	jus d'oranges limonade	- -	- -	cola	jus de pamplemousses

TAB. 1.1 – Base de données exemple

Si on utilise la taxonomie, on trouve un motif plus long et plus générique : <(jus d'oranges) (soda) (jus de pamplemousses)>. Ce motif est potentiellement plus utile ou plus intéressant pour l'utilisateur.

Des travaux ont par ailleurs été réalisés afin d'utiliser ce type de taxonomies dans le cadre des bases de données multidimensionnelles dans lesquelles les liens hiérarchiques existants entre les items sont précisés dans la structure de la base [PLT06]

1.2.2 Généralisation de la notion de transaction

La notion de transaction a également été généralisée afin de pouvoir rassembler et considérer simultanément des évènements proches dans le temps. [SA96b] définit ainsi une fenêtre glissante dont la taille est spécifiée par l'utilisateur. On peut alors considérer deux évènements comme simultanés si l'écart de temps qui les sépare est inférieur à la fenêtre spécifiée.

Exemple 1.3. Reprenons la base de données précédente (TAB. 1.1), avec le même support minimum, ainsi qu'une fenêtre glissante de 2 jours. Dans ce cas, les transactions des dates 1 et 2 du client 1 peuvent être regroupées, le seul motif séquentiel de la base est alors <(jus d'oranges, limonade) (jus de pamplemousses)>.

1.2.3 Prise en compte des contraintes de temps

Afin de pouvoir extraire des motifs sélectifs par rapport aux besoins de l'utilisateur, [SA96b] propose de spécifier certaines contraintes sur le temps et la durée séparant les itemsets d'une séquence.
L'utilisateur peut alors choisir d'imposer :

- une durée minimale à respecter entre deux itemsets consécutifs d'une séquence
- une durée maximale à respecter entre deux itemsets consécutifs d'une séquence

Exemple 1.4. Reprenons la base proposée dans l'exemple précédent, TAB. 1.1. Si l'on impose une durée minimale de 14 jours entre les itemsets d'une séquence, le 2ème motif séquentiel n'est plus supporté par le client 1, le seul motif séquentiel extrait de la base sera alors <(jus d'oranges) (jus de pamplemousses)>. Par contre, si l'utilisateur choisi de spécifier un écart maximal de 30 jours entre les itemsets d'une séquence, le client 2 ne supporte plus les deux séquences de l'exemple 1.2, car la durée séparant le premier itemset du deuxième est pour les deux séquences, de 49 jours.

Dans la suite de cette section, nous nous intéresserons exclusivement à la gestion des contraintes de temps ainsi qu'à la fenêtre glissante que nous considérerons désormais également comme une contrainte temporelle.

1.3 Contraintes temporelles pour les motifs séquentiels

La notion de séquence présentée dans l'introduction de ce rapport présente une certaine rigidité pour de nombreuses applications. En effet, si l'intervalle de temps entre deux transactions est suffisamment court, on pourrait envisager de les considérer simultanées. A l'inverse, deux évènements trop éloignés peuvent ne pas avoir de lien entre eux. C'est pourquoi la notion de séquence généralisée a été proposée par [SA96b] qui introduit la prise en compte de ***contraintes temporelles***.

Ces contraintes sont au nombre de trois. Tout d'abord, ***mingap*** est la durée minimale qui doit séparer deux itemsets consécutifs d'une même séquence. Ensuite, ***maxgap*** est l'écart maximal dans lequel doivent se trouver deux itemsets consécutifs d'une même séquence. Enfin, ***windowSize*** est la taille de la fenêtre dans laquelle les items de deux transactions différentes peuvent être regroupés dans un même itemset. On modifie alors la notion d'inclusion décrite précédemment (page 19) pour tenir compte de ces contraintes.

Définition 1.1. Une séquence de données $d =< d_1 ... d_m >$ supporte une séquence s, $< s_1 \cdots s_n >$ s'il existe des entiers $l_1 \leq u_1 < l_2 \leq u_2 < ... < l_n \leq u_n$ tels que :

i. $s_i \subset \cup_{k=l_i}^{u_i} d_k$, $1 \leq i \leq n$;
ii. date(d_{u_i}) - date(d_{l_i}) $\leq windowSize$, $1 \leq i \leq n$;
iii. date(d_{l_i}) - date($d_{u_{i-1}}$) $> mingap$, $2 \leq i \leq n$;
iv. date(d_{u_i}) - date($d_{l_{i-1}}$) $\leq maxgap$, $2 \leq i \leq n$;

Cette formalisation de la notion de ***séquence incluse*** permet alors une manipulation plus souple des transactions en autorisant :

- de regrouper des itemsets de dates proches via $windowSize$,
- de considérer des itemsets comme trop proches pour appartenir à la même séquence avec la contrainte $mingap$,
- de considérer des itemsets comme trop éloignés pour appartenir à la même séquence avec la contrainte $maxgap$.

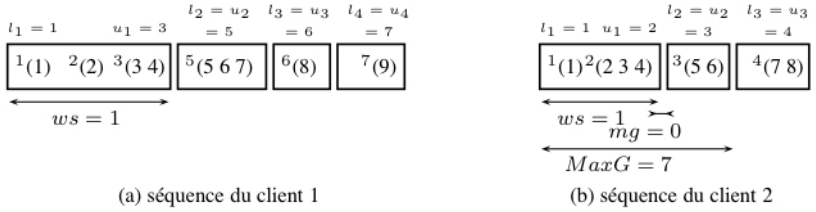
Exemple 1.5. On dispose des achats réalisés par les clients 1 et 2, TAB. 1.2.

Client	jour 1	jour 2	jour 3	jour 4	jour 5	jour 6	jour 7
Client 1	1	2	3 4		5 6 7	8	9
Client 2	1	2 3 4	5 6	7 8			

TAB. 1.2 – Achats réalisés par deux clients sur une période de 7 jours

Soit la séquence candidate $s =<(1\ 2\ 3\ 4)>$ et les contraintes de temps :
- $mingap$=0, des itemsets consécutifs sont séparés d'au moins un jour,
- $maxgap$=7, des itemsets consécutifs sont au maximum distants de 7 jours,
- $windowSize$=1, les achats peuvent être groupés sur deux jours consécutifs au maximum.

La FIG. 1.2 illustre l'utilisation des contraintes de temps afin de déterminer si les séquences C1 et C2 supportent ou non la séquence $s =< (1\ 2\ 3\ 4) >$.



(a) séquence du client 1 (b) séquence du client 2

$^i(ab)$: à la date i, le client a acheté les items a et b

FIG. 1.2 – Représentation des contraintes de temps $windowSize$ (ws), $mingap$ (mg) et $maxgap$ ($MaxG$) sur les séquences des clients 1 et 2.

Pour que la séquence s apparaisse dans la séquence du client 1, on doit regrouper les achats réalisés lors des jours 1, 2 et 3. Toutefois, ce groupement ne respecte pas la contrainte de temps (ii), en effet date(d_{u_i}) - date(d_{l_i}) = $jour$ 3 - $jour$ 1 = 2 > $windowSize$. Il n'y a pas d'autre façon de trouver s, donc la séquence $C1$ ne supporte pas la séquence s.
Pour que la séquence s apparaisse dans la séquence $C2$, on regroupe les achats réalisés lors des jours 1 et 2. Ce groupement respecte bien la contrainte sur $windowSize$ puisque les deux jours sont consécutifs. L'écart minimum séparant alors ce premier itemset de l'itemset suivant est $jour$ 3 - $jour$ 2 = 1 > 0 = $mingap$. La contrainte sur $mingap$ (iii) est donc bien respectée, ainsi que la contrainte sur $maxgap$ (iv) : $jour$ 3 - $jour$ 1 = 2 ≤ $maxGap$. La séquence $C2$ supporte donc la séquence s.

1.4 Algorithmes pour les contraintes de temps

Plusieurs algorithmes ont été proposés afin de gérer les contraintes de temps. Certains les introduisent directement dans le processus d'extraction, comme l'algorithme original GSP [SA96b] et l'algorithme DE-

LISP [LLW02]. D'autres appliquent les contraintes lors d'un prétraitement de la base de séquences qui peut ensuite être analysée par n'importe quel outil d'extraction de motifs séquentiels, comme GTC, proposé dans [MPT04].

L'algorithme GSP (Generalized Sequential Patterns), proposé dans [SA96b] a pour objectif d'extraire des motifs séquentiels généralisés. Il étend la proposition initiale d'algorithme pour la recherche de motifs séquentiels [AS95] en prenant en compte les contraintes de temps et la gestion des taxonomies. Cet algorithme utilise une approche générer-élaguer.
Lors du parcours de la base pour le calcul de la fréquence des k-candidats, les contraintes sont prises en compte. Afin de gérer les trois contraintes (fenêtre glissante, durées minimale et maximale entre deux itemsets), des phases de *backward-forward* sont nécessaires. Les phases vers l'avant (forward) ont pour but de considérer progressivement chaque transaction et d'utiliser la contrainte $windowSize$ afin de redimensionner le découpage temporel des transactions. Les phases de retour arrière (backward) sont nécessaires dès lors que la contrainte $maxgap$ est violée, le parcours de la séquence de données est réitéré depuis le premier item satisfaisant la condition $maxgap$.

Dans [MCP98], l'approche PSP est proposée afin d'améliorer les performances de GSP. Elle reprend l'intégralité des fondements de GSP mais s'appuie sur une structure de données différente. Celle-ci permet une meilleure organisation du stockage des candidats et améliore ainsi les performances de GSP.

Plus récemment, l'algorithme DELISP [LLW02] a été proposé afin de permettre la gestion des contraintes de temps lors de l'extraction de motifs séquentiels par un algorithme basé sur les préfixes (tels que PrefixSPAN [PHW02]). Le principe de ces algorithmes repose sur le découpage de la base originale en plusieurs sous-ensembles pour chaque préfix d'un potentiel motif séquentiel. Lors de la formation de ces sous-ensembles, DELISP réduit la taille de la base projetée. Les expérimentations montrent que les performances de DELISP dépassent largement celles de GSP. Toutefois, cette technique est restreinte aux algorithmes de type pattern-growth.

L'algorithme proposé dans [OPS04] permet également de traiter les contraintes de temps *mingap* et *maxgap*. Là encore les performances sont améliorées par rapport à GSP, mais cette technique est restreinte aux approches par niveau.

C'est pourquoi l'algorithme GTC (Graph for Time Constraints), a été développé [MPT04]. Il améliore les performances de PSP par une gestion efficace des contraintes de temps. Pour une séquence de données, l'algorithme GTC précalcule l'ensemble pertinent de toutes les séquences à vérifier. Le temps nécessaire à la vérification des candidats est alors réduit, puisque seules les séquences vérifiant les contraintes de temps sont conservées.

1.5 Objectifs de notre contribution

Différents algorithmes ont été proposés afin de gérer les contraintes de temps pour l'extraction de motifs séquentiels. Toutefois, si ces méthodes sont robustes et efficaces, elles nécessitent de l'utilisateur qu'il connaisse précisément les valeurs des contraintes à spécifier, sous peine d'obtenir des connaissances erronées ou inutiles. Pourtant, dans certains cas, ces valeurs ne sont pas connues avec certitude. Ainsi, les contraintes temporelles telles qu'elles sont spécifiées permettent de mettre en évidence de nouveaux motifs séquentiels, mais elles sont encore trop rigides et il peut être nécessaire de faire plusieurs tentatives avec différentes combinaisons de ces paramètres avant d'obtenir des résultats satisfaisants.

Par ailleurs, pour certaines applications, ils pourraient également être intéressant d'assouplir les contraintes spécifiées par les experts du domaine afin d'affiner leurs connaissances. La connaissance de l'expert est utilisée comme un point de départ et les résultats obtenus la complètent.

Enfin, le nombre de motifs séquentiels extraits, selon les contraintes de temps utilisées, peut rapidement devenir trop important pour que leur analyse soit efficace. Une mesure permettant l'exploitation des motifs séquentiels généralisés serait donc d'une grande utilité.

C'est pourquoi nous proposons les contributions suivantes. Premièrement, nous définissons des contraintes de temps étendues pour la découverte de motifs séquentiels généralisés. Deuxièmement, nous proposons une mesure de précision temporelle pour l'analyse des séquences fréquentes. Enfin, nous présentons l'algorithme que nous avons développé afin de gérer ces contraintes de temps étendues et calculer la précision temporelle.

Chapitre 2

Contraintes de temps étendues

Après avoir présenté les concepts fondamentaux associés aux motifs séquentiels généralisés dans le chapitre précédent, dans ce chapitre, nous présentons notre extension des contraintes de temps et la définition de la précision temporelle d'une séquence. La section 2.1 présente ainsi le principe général de notre approche ainsi que les notations utilisées. La section 2.2 développe ensuite plus précisément l'extension de chacune des trois contraintes $windowSize$, $mingap$ et $maxgap$. La section 2.3, enfin, termine ce chapitre par la définition de la précision temporelle d'une séquence.

2.1 Principes et notations

Notre proposition d'extension des contraintes de temps pour les motifs séquentiels est fondée sur une analogie avec la théorie des sous-ensembles flous. Ainsi, on ne souhaite plus simplement qu'une séquence respecte ou non les contraintes spécifiées mais permettre à l'utilisateur de relâcher ces contraintes. Chacune des contraintes de temps peut alors être vue comme un sous-ensemble flou dont la fonction d'appartenance nous donnera, pour chaque valeur attribuée au paramètre de contrainte, la précision avec laquelle la contrainte initialement spécifiée par l'utilisateur est respectée. Afin de répondre au mieux aux besoins de l'utilisateur, nous lui donnons la possibilité de spécifier la valeur ρ_x de respect minimum de la contrainte $\mathcal{X}$, de valeur initiale x_{init}.

2.1.1 Principe général

Le degré de respect des contraintes reposant sur la fonction d'appartenance de chacune des contraintes, les coefficients spécifiés sont compris dans l'intervalle [0,1]. Il est également possible de fixer une contrainte avec certitude :

- Si $\rho_x = 1$, l'utilisateur ne souhaite pas faire varier la valeur de la contrainte, qui restera donc fixée à sa valeur spécifiée et toutes les séquences générées auront une précision de 1.
- Si $\rho_x = 0$, l'utilisateur souhaite parcourir l'ensemble des valeurs possibles pour la contrainte, le poids d'une séquence dépendra alors de la valeur de la contrainte qui permet de la générer.
- Sinon, $\rho_x \in]0,1[$, la valeur x de la contrainte $\mathcal{X}$ va varier entre sa valeur fixée x_{init} et une valeur limite x_ρ pour laquelle $\rho(x) = \rho_x$.

Il existe des valeurs limites utiles pour chacune des contraintes de temps. Celles-ci correspondent au parcours de la totalité de l'espace de recherche, c'est-à-dire lorsque $\rho_x = 0$.

2.1.2 Valeurs limites utiles

Les ***valeurs limites utiles*** des contraintes de temps sont calculées à partir des contraintes de temps strictes (ii), (iii) et (iv) (chapitre 1, section 1.3). Elles correspondent aux valeurs limites autorisées par ces définitions et au-delà desquelles plus aucune séquence de données ne peut satisfaire les contraintes. Les contraintes $maxgap$ et $windowSize$ correspondent à l'écart maximal qui sépare deux itemsets, la valeur maximale qu'elles pourront prendre pour un objet correspond donc à la durée qui sépare la premier enregistrement de cet objet du dernier. Pour toute la base, cette valeur correspondra donc à l'écart maximum pour tous les objets entre les dates minimales et maximales des enregistrements, c'est-à-dire que la valeur limite utile de $maxgap$ et $windowSize$ sera $M = \max_{o \in \mathcal{O}}(D_{o_{max}} - D_{o_{min}})$.
La contrainte $mingap$ correspond à l'écart minimal qui sépare deux itemsets. Il s'agit donc de définir la valeur limite de cette contrainte, en tenant compte de l'inégalité stricte qu'elle impose. L'écart minimum qui sépare deux enregistrements d'un objet o est donné par $\min_{r \in \mathcal{R}_o}(D_{r+1} - D_r)$, et pour l'ensemble de la base, cet écart correspond à la valeur minimale pour l'ensemble des objets, soit $\min_{o \in \mathcal{O}}(\min_{r \in \mathcal{R}_o}(D_{r+1} - D_r))$. Dans le cas limite, la contrainte sur $mingap$ s'exprime par l'inégalité :

$$\begin{array}{lcl} \min_{o \in \mathcal{O}}(\min_{r \in \mathcal{R}_o}(D_{r+1} - D_r)) & > & mingap \\ \min_{o \in \mathcal{O}}(\min_{r \in \mathcal{R}_o}(D_{r+1} - D_r)) & \geq & mingap + 1 \\ \min_{o \in \mathcal{O}}(\min_{r \in \mathcal{R}_o}(D_{r+1} - D_r)) - 1 & \geq & mingap \end{array}$$

La valeur limite de $mingap$ est donc $\min_{o \in \mathcal{O}}(\min_{r \in \mathcal{R}_o}(D_{r+1} - D_r)) - 1$. Or dans le cas où un objet n'aurait qu'un enregistrement, cette valeur serait -1, ce qui reviendrait à dire que l'on pourrait avoir un écart nul entre deux itemsets consécutifs. Autrement dit, on pourrait transformer un itemset $(a\ b\ c)$ en une séquence $< (a)\ (b)\ (c) >$. On impose donc dans ce cas que la valeur de $mingap$ est nulle. La valeur limite utile de $mingap$ s'exprime alors par $m = \max(\min_{o \in \mathcal{O}}(\min_{r \in \mathcal{R}_o}(D_{r+1} - D_r)) - 1, 0)$.

2.1.3 Notations

Dans la suite de ce chapitre, nous utiliserons les notations suivantes, résumées par le tableau 2.1, afin de distinguer les trois contraintes de temps :

- Soit ws_{init}, mg_{init} et MG_{init} les valeurs initiales spécifiées pour les contraintes de temps $windowSize$, $mingap$ et $maxgap$ et ρ_{ws}, ρ_{mg} et ρ_{MG} les niveaux minimum de précision associés à chacune d'elles. Ces coefficients vont permettre de définir les limites de variation des contraintes de temps correspondant aux besoins de l'utilisateur.
- On notera ws, mg et MG les valeurs variables des contraintes et $\rho(ws)$ (resp. $\rho(mg)$ et $\rho(MG)$) les précisions des contraintes selon la valeur de ws (resp. mg et MG).

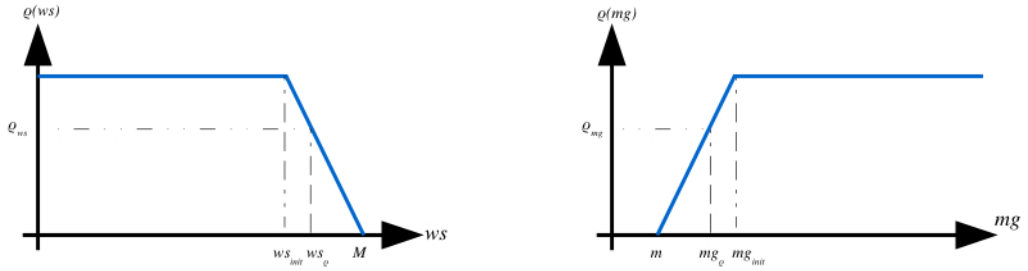
2.2 Extension des contraintes de temps

Nous allons maintenant présenter et illustrer l'extension de chacune des trois contraintes.

Soit la contrainte $\mathcal{X}$. Sa valeur x peut varier entre x_{init} et sa valeur limite utile x_{limite}. Dans le cas d'une contrainte supérieure (telle que $\forall a, a \leq x$) cette variation se représente par la figure 2.1(b), dans le cas d'une contrainte inférieure, figure 2.1(a).
L'utilisateur peut alors choisir de contraindre la valeur x de la contrainte $\mathcal{X}$ à respecter une précision minimale ρ_x. Cette précision correspond à une valeur limite x_ρ de x.

M	valeur maximale possible de $windowSize$ et $maxgap$ $M = \max_{o \in \mathcal{O}}(D_{o_{max}} - D_{o_{min}})$
ws_{init}	valeur initiale du paramètre $windowSize$, fixé par l'utilisateur
ws	variable correspondant au paramètre $windowSize$, peut varier entre ws_{init} et M
ρ_{ws}	précision minimale souhaitée par l'utilisateur $windowSize$
$\rho(ws)$	précision de $windowSize$ pour la valeur ws
MG_{init}	valeur initiale du paramètre $maxgap$, fixé par l'utilisateur
MG	variable correspondant au paramètre $maxgap$, peut varier entre MG_{init} et M
ρ_{MG}	précision minimale souhaitée par l'utilisateur $maxgap$
$\rho(MG)$	précision de $maxgap$ pour la valeur MG
m	valeur minimale possible de $mingap$ $m = \max(\min_{o \in \mathcal{O}}(\min_{r \in \mathcal{R}_o}(D_{r+1} - D_r)) - 1, 0)$
mg_{init}	valeur initiale du paramètre $mingap$, fixé par l'utilisateur
mg	variable correspondant au paramètre $mingap$, peut varier entre m et mg_{init}
ρ_{mg}	précision minimale souhaitée par l'utilisateur pour $mingap$
$\rho(mg)$	précision de $mingap$ pour la valeur mg

TAB. 2.1 – Notations



(a) (b)

FIG. 2.1 – (a) Exemple de contrainte supérieure ; (b) Exemple de contrainte inférieure.

2.2.1 Extension de $windowSize$

La valeur ws de la contrainte $windowSize$ peut varier entre sa valeur spécifiée ws_{init} et sa valeur maximum utile M, comme décrit FIG. 2.1(a). L'utilisateur peut alors choisir de contraindre cette valeur à respecter une précision minimale ρ_{ws}. Cette précision implique une valeur limite ws_ρ de ws.

L'extension de la contrainte s'exprime sous la forme du sous-ensemble flou décrit par la fonction d'appartenance (2.1) qui donne la précision correspondant à une valeur donnée de ws :

$$\rho(ws) = \begin{cases} 1 & \text{si} \quad ws \leq ws_{init} \\ \frac{1}{ws_{init} - M} ws - \frac{M}{ws_{init} - M} & \text{si} \quad ws_{init} < ws \leq ws_\rho \\ 0 & \text{sinon} \end{cases} \tag{2.1}$$

La valeur limite ws_ρ correspond à la valeur de $windowSize$ pour laquelle la précision des séquences générées vaut ρ_{ws}. Elle est donnée par l'équation :

$$ws_\rho = \lfloor (ws_{init} - M)\rho_{ws} + M \rfloor \tag{2.2}$$

Exemple 2.1. Reprenons la base d'achats présentée page 59, et reprise TAB. 2.2. Soit la séquence s=<(1 2 3 4)> et les paramètres de contraintes ws_{init}=1, mg_{init}=0 et MG_{init}=7, avec ρ_{ws}=0.7, ρ_{mg}=1 et ρ_{MG}=1.

Seul ρ_{ws} est différent de 1, donc seul $windowSize$ va prendre plusieurs valeurs. En appliquant l'équation (2.2) avec $M = \max((7-1),(4-1))$=6, on obtient $ws_\rho = \lfloor(1-6)*0.7+6\rfloor$= 2. La valeur ws de la contrainte $windowSize$ prendra donc successivement les valeurs 1 et 2.

Client	jour 1	jour 2	jour 3	jour 4	jour 5	jour 6	jour 7
Client 1	1	2	3 4		5 6 7	8	9
Client 2	1	2 3 4	5 6	7 8			

TAB. 2.2 – Achats réalisés par deux clients sur une période de 7 jours

Pour ws=1, en regroupant les jours 1, 2 et 3 de la séquence $C1$, on ne respecte pas la contrainte (ii). Par contre, lorsque ws=2, cette contrainte est respectée, avec une précision $\rho(ws) = \rho(2) = 0.8$ (2.1). Les autres contraintes étant également respectées, $C1$ supporte la séquence s.
Pour le client 2, on regroupe les achats des jours 1 et 2, la contrainte (ii) est donc respectée pour ws=1 avec une précision maximale, $\rho(ws) = 1$ (valeur initiale). Les autres contraintes étant aussi respectées, $C2$ supporte la séquence s.

2.2.2 Extension de $maxgap$

La valeur MG de la contrainte $maxgap$ peut varier entre sa valeur spécifiée MG_{init} et sa valeur maximale utile M. L'extension de la contrainte s'exprime sous la forme du sous-ensemble flou décrit par la fonction d'appartenance (2.3) qui donne :

$$\rho(MG) = \begin{cases} 1 & \text{si } MG \leq MG_{init} \\ \frac{1}{MG_{init}-M}MG - \frac{M}{MG_{init}-M} & \text{si } MG_{init} < MG \leq MG_\rho \\ 0 & \text{sinon} \end{cases} \quad (2.3)$$

La valeur limite MG_ρ correspond à la valeur de $maxgap$ pour laquelle le poids des séquences générées vaut ρ_{MG}. Elle est donnée par l'équation :

$$MG_\rho = \lfloor(MG_{init} - M)\rho_{MG} + M\rfloor \quad (2.4)$$

Exemple 2.2. Soit les séquences $C1$ et $C2$ de l'exemple 2.1, la séquence s=<(1 2 3 4) (7 8)> et les paramètres de contraintes ws_{init}=2, mg_{init}=0 et MG_{init}=3, avec ρ_{ws}=1, ρ_{mg}=1 et ρ_{MG}=0.3.
Seul ρ_{MG} est différent de 1, donc seul $maxgap$ va prendre plusieurs valeurs. En appliquant l'équation (2.4) avec M=6, on obtient MG_ρ= 5. La contrainte $maxgap$ prend donc successivement les valeurs 3, 4 et 5.
Pour le client 1, on regroupe les achats des dates 1, 2 et 3. Or, ce regroupement ne respecte pas la contrainte (iv) pour $MG = 3$, ni pour $MG = 4$. Pour $MG = 5$, la contrainte $maxgap$ est respectée avec une précision $\rho(MG) = \rho(5)$=0.3 (2.3). Les autres contraintes étant également respectées, $C1$ supporte la séquence s.
Pour le client 2, on regroupe les achats des dates 1 et 2, la contrainte (iv) est respectée pour $MG = 3$, la précision pour la contrainte $maxgap$ est donc $\rho(MG) = 1$ (valeur initiale). Les autres contraintes étant respectées, $C2$ supporte s.

2.2.3 Extension de $mingap$

La valeur mg de la contrainte $mingap$ peut varier entre sa valeur minimale utile m et sa valeur spécifiée mg_{init}. L'extension de la contrainte s'exprime sous la forme du sous-ensemble flou décrit par la

fonction d'appartenance (2.5) qui donne :

$$\rho(mg) = \begin{cases} 1 & \text{si} \quad mg \geq mg_{init} \\ \frac{1}{mg_{init}-m} mg - \frac{m}{mg_{init}-m} & \text{si} \quad mg_{init} > mg \geq mg_{\rho} \\ 0 & \text{sinon} \end{cases} \tag{2.5}$$

La valeur limite mg_ρ correspond à la valeur de $mingap$ pour laquelle le poids des séquences générées vaut ρ_{mg}. Elle est donnée par l'équation :

$$mg_\rho = \lceil (mg_{init} - m)\rho_{mg} + m \rceil \tag{2.6}$$

Exemple 2.3. Considérons les séquences $C1$ et $C2$ de l'exemple 2.1, la séquence s=<(1 2 3 4) (5 6)> et les paramètres de contraintes ws_{init}=2, mg_{init}=2 et MG_{init}=7, avec ρ_{ws}=1, ρ_{mg}=0 et ρ_{MG}=1. Seul ρ_{mg} est différent de 1, donc seul $mingap$ va prendre plusieurs valeurs. En appliquant l'équation (2.6) avec $m = 0$, on obtient mg_ρ=0. La contrainte $mingap$ prendra donc successivement les valeurs 2, 1 et 0.
Pour le client 1, on regroupe les achats des dates 1, 2 et 3. La contrainte (iii) n'est alors pas respectée pour mg=2, elle l'est par contre pour mg=1. Dans ce cas, la précision pour $mingap$ vaut $\rho(mg) = \rho(1)$=0.5 (équation (2.5)). Les autres contraintes étant respectées, le client 1 supporte la séquence s.
Pour le client 2, on regroupe les achats des dates 1 et 2. La contrainte (iii) n'est alors pas respectée pour mg=2, ni pour mg=1. Elle l'est par contre pour mg=0. Dans ce cas, la précision pour $mingap$ vaut $\rho(mg) = \rho(0)$= 0 (2.5). Les autres contraintes étant respectées, le $C2$ supporte s.

On remarque que les définitions des sous-ensembles flous des trois contraintes correspondent à une variation linéaire de la fonction d'appartenance entre la valeur initiale d'une contrainte et sa valeur limite, mais cette fonction pourrait également être définie par palliers ou en proportion du nombre de clients de la base respectant chacune des valeurs de la contrainte.

2.3 Précision temporelle d'une séquence

Nous allons maintenant définir le degré de respect des contraintes d'une séquence en considérant les trois contraintes simultanément. Pour chacune des séquences fréquentes trouvées à la fin du processus d'extraction, il s'agit de combiner les valeurs des contraintes de temps qui permettent de les générer afin de déterminer la précision avec laquelle chacune d'elles respecte les valeurs initiales.

On définit la ***précision temporelle*** d'une séquence s pour un objet o comme le niveau de respect simultané des trois contraintes de temps $windowSize$, $mingap$ et $maxgap$, calculé à l'aide d'une t-norme ($\top$). Pour chaque objet o, on cherche, parmi l'ensemble ς_o de ses séquences respectant les contraintes étendues, l'occurrence de s qui respecte au mieux les contraintes de temps initiales, en utilisant une t-conorme ($\perp$). La précision temporelle d'une séquence $s =< s_1 \cdots s_n >$ pour l'objet o est donnée par :

$$\begin{aligned} \varrho(s,o) = \perp_{s\in\varsigma_o} \Big(& \overline{\top}_{i\in[1,n]} \big(\rho_{ws}(date(s_{u_i}) - date(s_{l_i}))\big), \\ & \overline{\top}_{i\in[2,n]} \big(\rho_{mg}(date(s_{l_i}) - date(s_{u_{i-1}})), \\ & \rho_{MG}(date(s_{u_i}) - date(s_{l_{i-1}}))\big)\Big) \end{aligned} \tag{2.4}$$

Pour l'ensemble de la base, la précision temporelle d'une séquence s est donnée par l'agrégation par la moyenne des précisions de chacun des objets, c'est-à-dire :

$$\Upsilon(s) = \frac{1}{|\mathcal{O}|} \sum_{o\in\mathcal{O}} \varrho(s,o) \tag{2.5}$$

Par définition, la précision temporelle d'une séquence varie entre 0 et 1. Lorsque celle-ci vaut 1, cela signifie que toutes les séquences de données supportent la séquence en satisfaisant les valeurs initiales spécifiées pour toutes les contraintes de temps. Par contre, si cette valeur est 0, cela signifie qu'au moins une des contraintes de temps étendues jusqu'à sa limite n'est jamais vérifiée.

Exemple 2.4. Considérons la base de données de l'exemple 2.1 et la séquence $s =< (1\ 2\ 3\ 4)\ (5\ 6) >$ avec les spécifications de contraintes de temps étendues ws_{init}=1, mg_{init}=2, MG_{init}=4 et ρ_{ws}=0.6, ρ_{mg}=0.4 et ρ_{MG}=0.5. On a toujours M=6 et m=0. Dans cette exemple, on considère les opérateurs min pour la t-norme ($\overline{\top}$) et max pour la t-conorme ($\underline{\perp}$).
Pour le client 1, la séquence s apparaît en regroupant les achats des dates 1, 2 et 3 d'une part et des dates 5 et 6 d'autre part. On a donc $date(s_{u_1})$=1, $date(s_{l_1})$= 3, $date(s_{u_2})$=5 et $date(s_{l_2})$= 6. Il n'y a qu'une représentation possible, la précision de la séquence s s'exprime donc pour le client 1 par :

$$\begin{aligned} \varrho(s, C1) &= min(\rho_{ws}(2), \rho_{ws}(1), min\big(\rho_{mg}(1), \rho_{MG}(5)\big)) \\ \varrho(s, C1) &= min(0.8, 1, min\big(0.5, 0.5\big)) \\ \varrho(s, C1) &= 0.5 \end{aligned}$$

On réalise le même calcul pour la séquence du client 2 et on trouve également $\varrho(s, C2) = 0.5$. La précision temporelle de la séquence s pour l'ensemble de la base est alors donnée par :

$$\Upsilon(s) = \frac{\varrho(s, C1) + \varrho(s, C2)}{2} = 0.5$$

Le chapitre suivant présente les algorithmes permettant d'une part la gestion des contraintes de temps étendues, d'autre part le calcul de la précision temporelle des motifs séquentiels généralisés obtenus. Cette solution a été implémentée et les résultats des expérimentations sont présentés.

Chapitre 3

GETC : Graph for Extended Time Constraints

Notre proposition d'implémentation des contraintes de temps étendues repose sur les graphes de séquences pour les contraintes de temps (GTC) proposées dans [MPT04]. Il s'agit de transformer une séquence d'un client en un graphe de séquences respectant les contraintes de temps. Les graphes de séquences des différents clients sont ensuite utilisés pour déterminer les séquences fréquentes par un algorithme d'extraction de motifs séquentiels. L'efficacité de cette approche ayant été démontrée, nous avons choisi de nous en inspirer pour développer notre solution. Nous proposons donc un algorithme correct et complet permettant de construire des graphes de séquences pour les contraintes de temps étendues qui nous permettra également, dans un deuxième temps, de calculer la précision des motifs séquentiels généralisés extraits.

3.1 Construction du graphe de séquences

Nous utilisons GETC comme pré-traitement pour la prise en compte des contraintes de temps étendues. Une fois les séquences des clients transformées en graphes de séquences respectant les contraintes de temps étendues, les motifs séquentiels sont extraits avec PSP [MCP98]. En utilisant ainsi les graphes de séquences obtenus par GETC, la vérification des contraintes de temps est rendue inutile pendant la recherche des candidats, seule l'inclusion devant être vérifiée. Cette méthode est similaire à celle proposée dans [MPT04] qui permet d'optimiser l'extraction de motifs séquentiels généralisés par un parcours indépendant et sans retour arrière de l'arbre des séquences candidates pour la vérification des contraintes de temps. Nous ajoutons à ce processus une étape supplémentaire. Une fois les motifs séquentiels extraits, les graphes de séquences sont valués puis reparcourus une dernière fois, afin de calculer la précision temporelle de chacun des motifs séquentiels généralisés extraits.
L'algorithme 1 présente le processus global de l'extraction de motifs séquentiels sous contraintes de temps étendues.

A partir d'une séquence d'entrée d, l'algorithme GETC (Alg. 2) construit son graphe de séquences $G_d(S, A)$. Tout d'abord, GETC commence par créer les sommets correspondants aux itemsets de la séquence. Chaque sommet x du graphe de séquences est caractérisé par sa date de début x.begin() et par sa date de fin x.end(). On peut également accéder à sa liste de prédécesseurs par x.prev() et à la liste de ses successeurs par x.succ(), ainsi qu'à l'itemset x.itemset(). La fonction $addWindowSize$ ajoute ensuite à l'ensemble des sommets, l'ensemble des combinaisons d'itemsets permises selon les différentes valeurs de la contrainte $windowSize$.

Main - **Input** : $minFreq$, DB
Ouput : F, frequent sequences on DB, k longest length of frequent sequences

$F_0 \leftarrow \varnothing$; $k \leftarrow 1$;
$F_1 \leftarrow \{\{< i >\}/i \in \mathcal{I} \& freq(i) > minFreq\}\}$;
addWindowSize(S) ;
Tant que ($Candidate(k) \neq \varnothing$) **faire**
  Pour chaque $d \in DB$ **faire**
    $(G) \leftarrow$ GETC(d) ;
    countFrequency($Candidate(k), minFreq, (G)$) ;
  Fin Pour
  $F_k \leftarrow \{s \in Candidate(k)/freq(s) > minFreq\}$;
  $Candidate(k+1) \leftarrow$ generate(F_k) ;
  k++ ;
Fin Tant que
ComputeAccuracy(F_k) ;
Retourner $F \leftarrow \bigcup_{j=0}^{k} F_k$

ALG. 1 : *Main algorithm*

L'étape suivante consiste en l'ajout des arcs respectant les contraintes $mingap$ et $maxgap$. Ainsi, pour chaque sommet, on cherche le premier niveau accessible pour la contrainte $mingap$ (i.e. l.begin() - x.end() $> mg_\rho$) et pour chaque sommet z de ce niveau, on construit les arcs (x,z), pour chaque sommet z tel que z.end() - x.begin() $\leq MG_\rho$. La fonction $addEdge$ permet d'éviter les inclusions de chemins, grâce à la construction d'arcs temporaires dans des cas d'inclusions possibles. Dans le cas où pour un sommet x, on ne peut atteindre le niveau l à cause du non respect de la contrainte $mingap$, on utilise la fonction $propagate$ pour "propager ce saut". Ensuite la fonction $pruneMarked$ élimine les sommets de sous-séquences incluses. Enfin, $convertEdges$ transforme les arcs temporaires indispensables en arcs définitifs et supprime les arcs de sous-séquences incluses.

GETC - **Input** : d, une séquence de données
Ouput : $G_d(S, A)$, le graphe de séquences de d, S l'ensemble des sommets de G_d, A l'ensemble des arêtes

S $\leftarrow$ buildVertices(d) ;
addWindowSize(S) ;
Tant que ($x \neq$ S.first()) **faire**
  $l \leftarrow x$.level().prec() ; $mg \leftarrow mg_{init}$;
  Tant que (x.begin() - l.end() $\leq mg$) **faire**
    $contmg \leftarrow$ FALSE ;
    Si (x.begin() $> l$.end()) **Alors**
      Tant que ($mg \geq mg_\rho$) **faire**
        Si (constming(x,l)) **Alors**
          $contmg \leftarrow$ TRUE ;
          $mg \leftarrow mg_\rho$-1 ;
        Sinon
          mg - - ;
        Fin Si
      Fin Tant que
    Fin Si
    Si ($contmg ==$ FALSE) **Alors**
      propagate(x,l) ; $l \leftarrow l$.prec() ;
    Fin Si
  Fin Tant que
  Pour chaque $w \in l$ **faire**
    $included \leftarrow$ TRUE ;
    $MG \leftarrow MG_{init}$;
    Tant que ($MG \leq MG\rho$) **faire**
      Si (constMaxG(x,w)) **Alors**
        addEdge(w,x) ;
        $MG \leftarrow MG_\rho + 1$;
      Sinon
        MG++ ;
      Fin Si
    Fin Tant que
  Fin Pour
  $x \leftarrow$ S.next(x) ;
Fin Tant que
pruneMarked($G_d(S, A)$) ;
convertEdges($G_d(S, A)$) ;
Retourner $G_d(S, A)$;

ALG. 2 : *GETC*

3.1.1 Construction des sommets

Tout d'abord, GETC commence par créer les sommets x correspondants aux itemsets de la séquence. Ensuite, l'algorithme $addWindowSize$ (ALG. 3) parcourt chaque sommet x et détermine pour chacun d'entre eux quels sommets y (différents de x) peuvent être "fusionnés" avec x (si y.date() - x.date() $\leq ws$). Chaque sommet i correspond alors à un itemset i.itemset() qui possède une date de début i.begin() et une date de fin i.end(). Les sommets ainsi construits sont regroupés en niveau par date de fin des itemsets, l.end(). Cela permet par la suite de tester le respect des contraintes pour un niveau et non plus pour chaque sommet.

---

addWindowSize - **Input** : S, ensemble S des sommets à traiter (ordonnés en ordre croissant, d'abord par date de début, puis par date de fin)

$copyS \leftarrow$ S ; $x \leftarrow$ S.first() ;
Tant que ($x \neq$ S.last()) **faire**
  $xnext \leftarrow$ S.next(x) ; $y \leftarrow$ S.next(x) ;
  $ws \leftarrow ws_{init}$;
  Tant que ($ws \leq ws_\rho$) **faire**
    Tant que (constWS(x,y)) **faire**
      $i \leftarrow$ group(x, y) ;
      $copyS$.addOrReplace(i) ;
      $x \leftarrow i$; $y \leftarrow$ S.next(y) ;
    Fin Tant que
    ws ++ ;
  Fin Tant que
  $x \leftarrow xnext$;
Fin Tant que
S $\leftarrow copyS$;

ALG. 3 : *addWindowSize*

Exemple 3.1. Soit la base exemple TAB. 3.1 et les paramètres suivants pour les contraintes de temps : pour $windowSize$, ws_{init}=2 et $\rho_{ws} = 0.86$, donc ws_ρ=4 ; pour $maxgap$, MG_{init}=4 et $\rho_{MG} = 0.84$ donc MG_ρ=6 ; pour $mingap$, mg_{init}=2 et $\rho_{mg} = 0.5$, donc mg_ρ=1. D'après TAB. 3.1, M = 17 et m = 0.

Date	1	3	4	5	6	8	9	10	12	17	18
Client 1	1	-	2 3	3 4	4	4	-	5	6	7	8
Client 2	2 3	4	-	-	5	6	-	-	-	-	-
Client 3	1 2	-	3	3 4	4	-	5 6	-	-	-	-

TAB. 3.1 – Base exemple

Nous présentons ici la construction des sommets du graphe de séquences pour la séquence de données du client 1. La première étape consiste en la création de l'ensemble des sommets initiaux, correspondant aux itemsets des transactions présentés dans la base TAB. 3.1. Ensuite, la contrainte $windowSize$ est prise en compte grâce à la fonction $addWindowSize$ (FIG. 3.1).

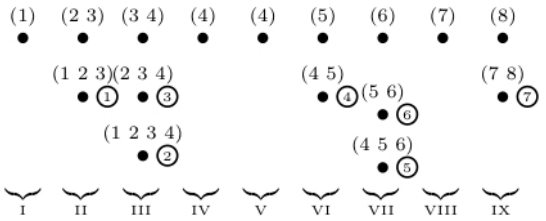


FIG. 3.1 – Graphe de séquences après prise en compte de $windowSize$ étendue ; ordre de construction des sommets par $addWindowSize$.

3.1.2 Construction des arcs

L'algorithme $addEdge$ (ALG. 4) permet de construire les arcs entre des sommets qui respectent les contraintes de temps $mingap$ et $maxgap$. On créera un arc définitif si les sommets ne sont pas déjà liés par une séquence ou une inclusion de leurs successeurs ou prédécesseurs. Dans ce cas, l'arc construit sera temporaire et ne deviendra définitif que si la séquence qu'il forme est maximale. C'est également lors de l'exécution de cette fonction que les sommets inclus sont marqués pour être ensuite supprimés s'ils sont inutiles.

addEdge - **Input** : deux sommets r et s, A l'ensemble des arcs du graphe

```
Si (r.succ() == ∅) Alors
    Si (s.prev() == ∅) Alors
        A ← A ∪{(r, s)} ;
        unmark(r) ; unmark(s) ;
    Sinon
        Pour chaque p ∈ s.prev() faire
            Si ((r ⊂ p) &&
            (r.succ() ⊂ p.succ())) Alors
                included ← TRUE ;
            Fin Si
        Fin Pour
        Si (included == TRUE) Alors
            arcTmp(r,s) ; mark(r) ;
            [si aucun arc définitif de ou vers r]
        Sinon
            A ← A ∪{(r, s)} ;
            unmark(r) ; unmark(s) ;
        Fin Si
    Fin Si
Fin Si
```

```
Sinon
    Pour chaque t ∈ r.succ() faire
        Si ((s ⊂ t) &&
        (s.succ() ⊂ t.succ())) Alors
            included ← TRUE ;
        Fin Si
    Fin Pour
    Si ((s.prev() ≠ ∅) &&
    (included == FALSE)) Alors
        Pour chaque p ∈ s.prev() faire
            Si ((r ⊂ p) &&
            (r.succ() ⊂ p.succ())) Alors
                included ← TRUE ;
            Fin Si
        Fin Pour
    Fin Si
    Si (included == TRUE) Alors
        arcTmp(r,s) ;
        mark(r) ;
        [si aucun arc définitif de ou vers r]
    Sinon
        A ← A ∪{(r, s)} ;
        unmark(r) ; unmark(s) ;
    Fin Si
Fin Si
```

ALG. 4 : *addEdge*

La fonction $propagate$ (ALG. 5) est utilisée lors du saut d'un niveau l à cause du non respect de la contrainte $mingap$, pour construire les séquences entre les sommets de ce niveau l et les successeurs du sommet x qui ne peuvent atteindre ce niveau. Pour chacun des sommets y d'un niveau inaccessible par x, la fonction $propagate$ ajoute, si nécessaire et si on respecte les contraintes $mingap$ et $maxgap$, un arc entre chacun des successeurs de x et ce sommet y. Comme dans $addEdge$, on construit des arcs temporaires ou définitifs selon que la séquence construite peut être incluse ou non.

Exemple 3.2. L'étape suivante est celle de la construction des arcs respectant les contraintes $mingap$ et $maxgap$, grâce à l'algorithme principal, ainsi qu'aux fonctions $propagate$ et $addEdge$ (FIG. 3.2).

3.1.3 Suppression des inclusions

L'avant-dernière étape utilise la fonction $pruneMarked$ (Alg. 6) qui élimine les sommets des sous-séquences incluses. Enfin, $convertEdges$ (Alg. 7) transforme les arcs temporaires indispensables en arcs définitifs et supprime les arcs des sous-séquences incluses.

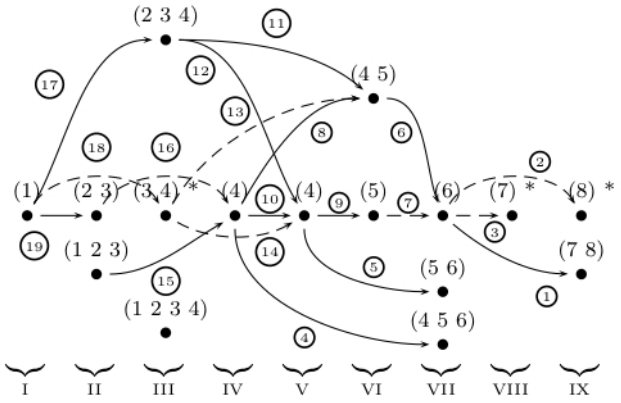


FIG. 3.2 – Graphe de séquences pour le client 1 après la prise en compte de $mingap$ et $maxgap$ étendus ; ordre de construction des arcs par GETC.

---

propagate - **Input** : x, un sommet du graphe de séquence, l, un niveau du graphe de séquence

```
Si (x.begin() > l.end()) Alors
    [on ne respecte pas minGap]
    Pour chaque u ∈ l faire
        Pour chaque v ∈ x.succ() faire
            Si (v.begin() - u.end() > mg)) Alors
                MG ← G ;
                Tant que (MG ≤ G_ρ) faire
                    Si (v.end() - u.begin() ≤ MG) Alors
                        Si (u.succ() ∩v.prev()) Alors
                            Pour chaque t ∈ v.prev() faire
                                Si (u ⊊ t) Alors included ← TRUE ; Fin Si
                            Fin Pour
                            Si (included == FALSE) Alors
                                Pour chaque w ∈ u.succ() faire
                                    Si ((w ⊂ u) && (w.succ() ⊂ u.succ()) ) Alors included ← TRUE ; Fin Si
                                Fin Pour
                            Fin Si
                            Si (included == TRUE) Alors
                                arcTmp(u,v) ; mark(u) ;  [si aucun arc définitif de ou vers r]
                            Sinon
                                A ← A ∪{(u, v)} ; unmark(u) ; unmark(v) ;
                            Fin Si
                        Fin Si
                    Sinon
                        MG++ ;
                    Fin Si
                Fin Tant que
            Fin Si
        Fin Pour
    Fin Pour
Fin Si
```

ALG. 5 : *propagate*

L'algorithme $convertEdges$ permet de transformer les arcs temporaires indispensables en arcs définitifs et de supprimer les arcs de sous-séquences incluses. Pour chaque arc temporaire entre x et y, si y

pruneMarked - **Input** : $G(S, A)$, graphe de séquence

Pour chaque $s \in S$ **faire**
  Si (s.marked == TRUE) **Alors**
    prune(s) ;
  Fin Si
Fin Pour

ALG. 6 : *pruneMarked*

convertEdges - **Input** : $G(S, A)$, un graphe de séquence

Pour chaque arcTmp(x,y) **faire**
  Pour chaque $z \in x$.succ()$\backslash y$ **faire**
    Si (($y \subset z$) && (y.succ() $\subset z$.succ())) **Alors**
      $included \leftarrow$ TRUE
    Fin Si
  Fin Pour
  Si ($included$ == TRUE) **Alors**
    prune(arcTmp(x,y)) ;
  Sinon
    A $\leftarrow$ A $\cup\{(x, y)\}$;
    prune(arcTmp(x,y)) ;
  Fin Si
Fin Pour

ALG. 7 : *convertEdges*

est inclus dans un successeur z de x et si les successeurs de y sont également tous des successeurs de z, alors il existe une sous-séquence incluse, l'arc est inutile, il est donc supprimé. Dans les autres cas, l'arc est indispensable pour obtenir toutes les séquences maximales, il est donc converti en arc définitif.

Exemple 3.3. La fonction $pruneMarked$ supprime ensuite les sommets inclus, puis les arcs temporaires sont convertis en arcs définitifs ou supprimés par la fonction $convertEdges$. Le graphe de séquences obtenu est présenté FIG. 3.3.

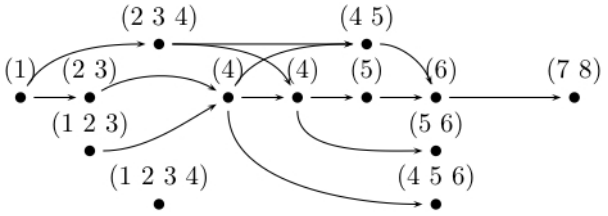


FIG. 3.3 – Graphe de séquences final, pour le client 1

Les séquences maximales incluses dans la séquence du client 1 sont :
- <(1 2 3)(4)(4 5 6)> - <(1 2 3)(4)(4)(5)(6)(7 8)> - <(1)(2 3)(4)(4)(5)(6)(7 8)>
- <(1 2 3)(4)(4 5)(6)> - <(1)(2 3)(4)(4 5 6)> - <(1)(2 3 4)(4)(5 6)>
- <(1 2 3)(4)(4)(5 6)> - <(1)(2 3)(4)(4 5)(6)> - <(1)(2 3 4)(4)(5)(6)(7 8)>
- <(1 2 3 4)> - <(1)(2 3)(4)(4)(5 6)> - <(1)(2 3 4)(4 5)(6)>

GETC construit exactement toutes les séquences maximales supportées par la séquence d'entrée, il peut donc être utilisé comme prétraitement pour la prise en compte des contraintes de temps étendues en vue de l'extraction de motifs séquentiels. La correction et la complétude de l'algorithme GETC sont démontrées en annexe C.

3.2 Calcul de la précision temporelle

Une fois le graphe de séquences construit, on sait quelles sont les séquences autorisées par les contraintes de temps et celles qui sont interdites. Cependant, certaines séquences respectent les contraintes fortes de l'utilisateur alors que d'autres ont été construites en appliquant les contraintes étendues : elles ne sont pas équivalentes. On va donc calculer le niveau de précision temporelle de chacun des chemins (séquences maximales) et l'affecter à chacune des sous-séquences qui le composent.

3.2.1 Valuation des graphes de séquences

Afin de déterminer le respect des contraintes de temps par le chemin, on value chaque arc (x,y) par $\top(\mu_{mg}(y.\text{begin}()\text{-}x.\text{end}()),\mu_{MG}(y.\text{end}()\text{-}x.\text{begin}()))$ selon les valeurs de mg et MG qui permettent de le construire. Chaque sommet est valué sur le même principe, par μ_{ws}. Ces valuations sont réalisées par la fonction $valueGraph$.

---

valueGraph - **Input** : $G_d(S, A)$, graphe de séquences non valué d'une séquence de données d
$\mu_{ws}, \mu_{mg}, \mu_{MG}$, les fonctions d'appartenance des contraintes de temps.
Output : $G_d(S, A)$, graphe de séquences valué de la séquence de données d

Pour chaque $s \in S$ **faire**
  s.valuate($\mu_{ws}(s.\text{end}()\text{-}s.\text{begin}())$;
  Pour chaque $t \in s.\text{succ}()$ **faire**
    arc(s,t).valuate($\top$($\mu_{mg}(t.\text{begin}()\text{-}s.\text{end}())$, $\mu_{MG}(t.\text{end}()\text{-}s.\text{begin}())$)) ;
  Fin Pour
Fin Pour

ALG. 8 : *valueGraph*

La précision temporelle d'une séquence est alors donnée par la formule 2.5, détaillée dans la section 2.3 du chapitre précédent. Ce calcul nécessite une passe supplémentaire, après l'extraction des motifs séquentiels, pour retourner, en plus de la fréquence de chaque motif, sa précision temporelle.

3.2.2 Calcul de la précision temporelle des séquences fréquentes

Proposé dans [MCP98], l'approche PSP est une approche de type générer-élaguer qui utilise une structure d'arbre préfixé pour organiser les séquences candidates et permettre de trouver plus efficacement l'ensemble des candidats inclus dans une séquence de données. Ainsi, à la k-ème étape, l'arbre est de profondeur k. Il stocke toutes les k-séquences candidates de la manière suivante : une séquence candidate est stockée sur une branche, de la racine à une feuille. Chaque nœud de profondeur p correspond au p-ème item de la séquence. A chaque nœud terminal est associé la fréquence de la séquence. On distingue deux types de fils pour un nœud, les fils $same$, dans le même itemset que leur père, et les fils $other$, dans un itemset ultérieur.

A la fin de la phase d'extraction des motifs séquentiels, nous obtenons donc un arbre préfixé des séquences fréquentes. Nous réalisons alors un parcours de la base afin de valuer chaque séquence de l'arbre des fréquents. Pour cela, l'algorithme $calcGenPrec$ compare chaque séquence incluse dans l'arbre préfixé avec les plus longs chemins dans les graphes de séquences valués.

Par construction du graphe de séquences, l'existence d'un arc entre deux points garantit le respect des contraintes de temps entre les deux sommets et l'existence d'un chemin garantit qu'il existe une séquence respectant les contraintes entre la source et le puits du chemin. On utilise alors $calcPrec$ et

donc $recCalcPrec$ pour parcourir l'arbre des fréquents et calculer la précision grâce aux graphes de séquences, par application de la formule 2.5.

---

calcGenPrec - **Input** : $\mathcal{O}$, ensemble des objets et de leur graphe de séquences, T l'arbre des séquences fréquentes, $\mu_{ws}, \mu_{mg}, \mu_{MG}$, les fonctions d'appartenance des contraintes de temps.
Output : T l'arbre des séquences fréquentes, munies de leur degré de respect des contraintes.

Pour chaque $o \in \mathcal{O}$ **faire**
  valueGraph(g_o,μ_{ws},μ_{mg},μ_{MG}) ;
  calcPrec(g_o,T) ;
Fin Pour
Pour chaque $L \in T$.leaves() **faire**
  L.degPrec $\leftarrow$ L.degPrec/$\mathcal{O}$;
Fin Pour

ALG. 9 : *calcGenPrec*

---

calcPrec - **Input** : $G_d(S_d, A)$, graphe de séquences valué d'une séquence de données d, T l'arbre des fréquents.
Output : T l'arbre des séquences fréquentes, munies du degré de respect des contraintes pour la séquence d.

Pour chaque $i \in root$.children() **faire**
  Pour chaque $S \in S_d$ **faire**
    Pour chaque $j \in$ S.items **faire**
      Si $(i == j)$ **Alors** recCalcPrec(T,$root$.children[i],S,i,G_d,1) ; **Fin Si**
    Fin Pour
  Fin Pour
Fin Pour

ALG. 10 : *calcPrec*

---

recCalcPrec - **Input** : $G_d(S_d, A)$, graphe de séquences valué d'une séquence de données d,
T l'arbre des séquences fréquentes, N nœud de l'arbre pour la validation en cours,
$sPrec$ sommet du graphe en cours de validation,
$iPrec$ item précédemment vérifié dans le sommet $sPrec$,
p le degré de respect temporaire pour la séquence.
Output : T l'arbre des séquences fréquentes, munies du degré de respect pour la séquence d.

Si (N.isLeaf()) **Alors**
  N.cpt $\leftarrow \perp$(N.degPrec(),p) ;
Sinon
  Pour chaque $i \in sPrec$ **faire**
    [différent de $iPrec$...]
    Si $(i \in N$.same()) **Alors** recCalcPrec(T,N.same[i],$sPrec$,i, G_d,$\top$(p,$sPrec$.valuation())) ; **Fin Si**
  Fin Pour
  Pour chaque $s \in sPrec$.next() **faire**
    Pour chaque $i \in s$.items() **faire**
      Si $(i \in N$.other()) **Alors** recCalcPrec(T,N.other[i],s,i,G_d, $\overline{\top}$(p,arc($sPrec$,s).valuation())) ; **Fin Si**
    Fin Pour
  Fin Pour
Fin Si

ALG. 11 : *recCalcPrec*

3.2.3 Un exemple

A partir de la base de données TAB. 3.1 et des contraintes de temps spécifiées dans le chapitre précédent, on construit les trois graphes de séquences correspondants. Puis on procède à l'extraction des motifs séquentiels. Les motifs séquentiels généralisés extraits avec $minFreq = 70\%$, sont les séquences : $< (2\ 3\ 4) >$, $< (2\ 3)(4)(5\ 6) >$, $< (2)(4\ 5) >$, $< (3\ 4)(5) >$, $< (3\ 4)(6) >$ et $< (3)(4\ 5) >$. Tous ont une fréquence de 100%. Afin de pouvoir distinguer leur pertinence par rapport aux besoins de l'utilisateur, nous calculons la précision temporelle de chacun. Pour cela, les graphes de séquences sont valués comme précisé dans la section 3.2.1. Les sommets construits avec ws=0,1,2 ont un degré de 1, avec ws=3, un degré de 0.93 et avec ws=4, un degré de 0.87. De même les arcs construits avec mg=1 ont, pour $mingap$, un degré de 0.5 et de 1 pour mg=2. En ce qui concerne $maxgap$, le degré est de 1 pour $MG \leq$4, de 0.92 pour MG=5 et 0.84 pour MG=6. On utilise ces valuations pour calculer le degré de respect des contraintes de temps par les motifs extraits. Les résultats sont présentés sur la figure 3.5.

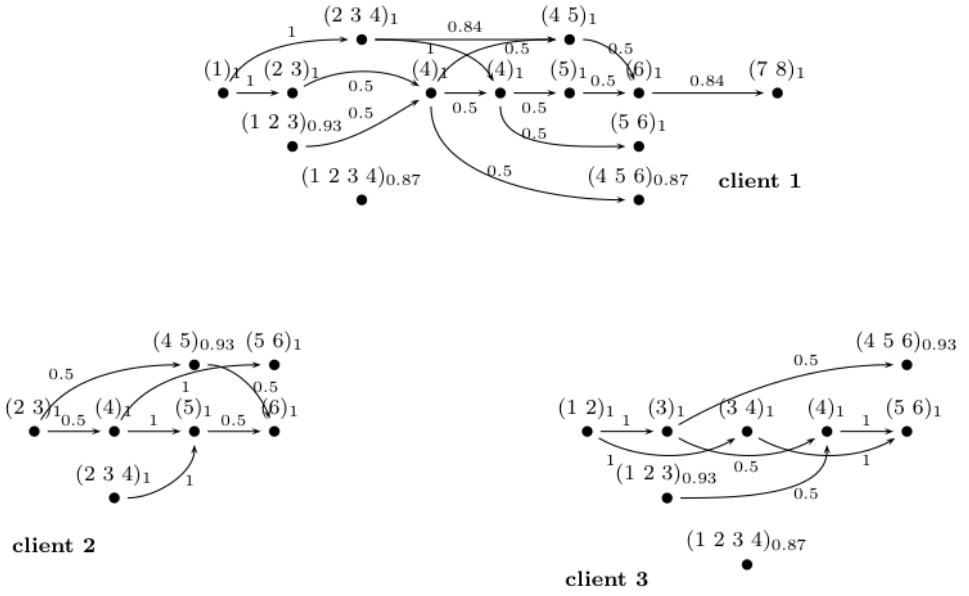


FIG. 3.4 – Graphes de séquences valués pour les clients 1, 2 et 3

Motifs séquentiels	ϱ_{Cl1}	ϱ_{Cl2}	ϱ_{Cl3}	Υ
$< (2\ 3\ 4) >$	1	1	0.87	0.96
$< (3\ 4)(5) >$	0.84	1	1	0.95
$< (2)(4\ 5) >$	0.84	0.5	1	0.78
$< (3\ 4)(6) >$	0.5	0.5	1	0.67
$< (3)(4\ 5) >$	0.84	0.5	0.5	0.61
$< (2\ 3)(4)(5\ 6) >$	0.5	0.5	0.5	0.5

FIG. 3.5 – Calcul de la précision pour les motifs séquentiels extraits

Une fois les motifs obtenus avec leur précision temporelle, on peut analyser plus précisément les contraintes qui ont permis de les générer. Plus la précision est proche de 1, plus les valeurs initiales spécifiées par l'utilisateur correspondent aux dates dans la base de données. A l'inverse, une précision faible indique que les contraintes sont peu appropriées à ce jeu de données.

3.3 Expérimentations

3.3.1 Données synthétiques

Ces expérimentations (implémentation C++) ont été réalisées sur un PC équipé d'un processeur 2,8GHz et de 2Go de mémoire DDR, sous système Linux, noyau 2.6. Le but de ces mesures est de comparer le comportement des algorithmes GETC (contraintes de temps étendues) et GTC (contraintes de temps simples). Pour certaines comparaisons, nous utilisons également une implémentation de PSP, intégrant ou non la gestion des contraintes de temps. Les résultats présentés ici ont été obtenus à partir du traitement de plusieurs jeux de données synthétiques comportant environ 1000 séquences de 20 transactions en moyenne. Chacune de ces transactions comporte environ 15 items choisis parmi 1000.

La première étape a consisté à comparer les temps d'exécution en l'absence de contrainte de temps, c'est-à-dire en prenant $windowSize = 0$, $mingap = 0$ et $maxgap = \infty$ pour GTC ainsi que pour GETC, avec une précision de 1 pour les trois paramètres. Ainsi, nous avons pu comparer le temps d'exécution de notre algorithme avec ceux de PSP et GTC et montrer que le comportement de GETC est similaire à celui de GTC et que les temps d'exécution sont quasiment identiques, pour extraire les mêmes motifs.

Nous avons ensuite répété ces mesures en introduisant le traitement de contraintes de temps, toujours avec une précision de 1 afin de comparer le comportement de GETC et GTC pour la gestion de contrainte de temps non-étendues. La FIG. 3.6(a) montre l'évolution du temps d'extraction en fonction de la valeur de $windowSize$. GETC a un comportement linéaire proche de celui de GTC. La différence provient de la phase de traitement de la précision, dont le temps augmente légèrement avec $windowSize$, le nombre de sommets dans le graphe de séquences augmentant avec ce paramètre.
Enfin, la FIG. 3.6(b) montre l'évolution du temps d'extraction en fonction de la précision pour une fréquence minimum de 0.37. On constate que le temps d'extraction atteint une valeur limite qui correspond à la valeur maximale utile des trois paramètres de contraintes de temps.

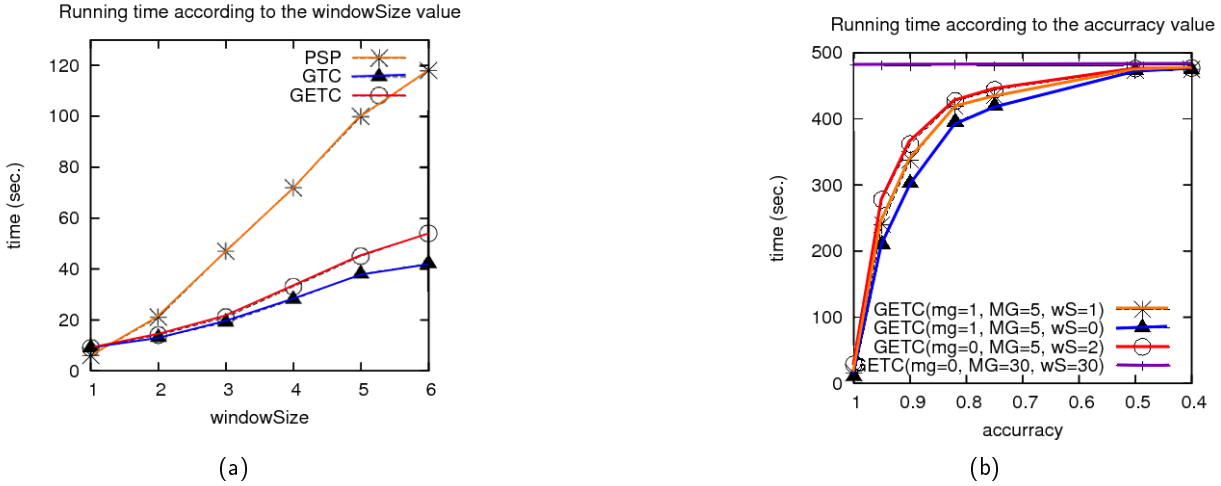


FIG. 3.6 – Temps d'extraction : (a) en fonction de windowSize avec $mingap$=2, $maxgap$=∞ and $minFreq$=0.35 (pour GETC, ρ_{ws}=ρ_{MG}=ρ_{mg}=1) ; (b) en fonction de la précision selon différentes valeurs des contraintes de temps ($minFreq$=0.37).

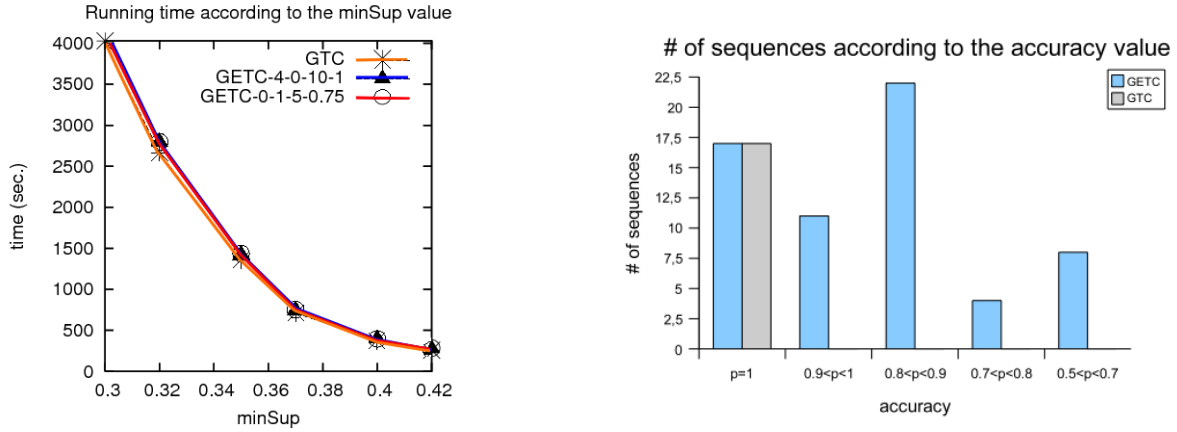
La deuxième partie de nos expérimentations a porté sur l'analyse des motifs séquentiels extraits par GETC, par rapport aux motifs extraits par GTC, en fonction de la précision des différentes contraintes. La figure FIG. 3.7(a) présente les temps d'extraction comparés de GTC et GETC en fonction de la fréquence

minimum selon des valeurs choisies des différents paramètres. Ces valeurs ont été calculées afin que les contraintes de temps utilisées pour GTC et GETC avec une précision de 1 correspondent aux valeurs limites de GETC avec une précision différente de 1. Les paramètres retenus sont :

- GETC avec ws_{init}=0, mg_{init}=1 et MG_{init}=5 avec une précision de 0.75, qui nous donne $ws_\rho = 4$, $mg_\rho = 0$ et $MG_\rho = 10$,
- GETC avec ws_{init}=4, mg_{init}=0 et MG_{init}=10 avec $\rho = 1$,
- GTC avec ws_{init}=4, mg_{init}=0 et MG_{init}=10,

On peut constater que l'utilisation de GETC avec des contraintes de temps étendues n'est pas plus coûteuse que celle de GTC avec les valeurs limites des contraintes, tout en permettant d'obtenir les mêmes motifs séquentiels, accompagnés de leur précision temporelle. Nous montrons ainsi qu'il peut être intéressant de choisir des valeurs initiales et des valeurs de précision permettant d'aboutir à la même liste de motifs qu'avec des contraintes simples, en sachant qu'on obtiendra dans un temps équivalent ces motifs détaillés de leur précision temporelle.

Par ailleurs, il est intéressant, dans le cas où on ignore la valeur optimale d'une ou plusieurs contraintes de temps, d'utiliser GETC avec une précision différente de 1, afin de balayer un ensemble de possibilités plus large. L'analyse des motifs obtenus et de leur précision pourra renseigner sur une valeur plus adéquate des contraintes de temps. Ainsi, nous avons comparé les motifs extraits par GTC avec les motifs extraits par GETC, avec les mêmes contraintes de temps initiales. Le nombre de motifs trouvés est alors supérieur, comme le montre la FIG. 3.7(b). En les ordonnant par ordre décroissant de précision, on retrouve tout d'abord les motifs extraits par GTC (qui ont une précision égale à 1) puis une liste de motifs de précision inférieure correspondant aux valeurs étendues des contraintes. Cet histogramme montre également que pour cette base, les contraintes permettant d'extraire le plus de motifs correspondent à une précision dans $[0.8, 0.9]$.



(a) (b)

FIG. 3.7 – (a) : Temps d'extraction en fonction de $minFreq$ en considérant des contraintes de temps avec une précision de 1 ou non ; (a) : Répartition des motifs trouvés par GETC selon leur précision temporelle (ws_{init}=0, mg_{init}=1, MG_{init}=5 et ρ_{ws}=ρ_{mg}=ρ_{MG}=0.5).

3.3.2 Contraintes étendues pour la description d'accès Web atypique

Le but des expérimentations présentées dans cette section est de montrer le gain informationnel obtenu par l'utilisation de contraintes étendues lors de la recherche de motifs séquentiels généralisés.

Le jeu de données utilisé pour ces expérimentations regroupe les logs d'accès à un site web d'usage privé, dans laquelle nous avons cherché à caractériser des comportements atypiques. Ce site web est hébergé sur un serveur Apache et utilise une base de données mySQL accédée via PHP. Il est divisé en deux parties, l'une accessible de façon anonyme, l'autre nécessitant une identification. Nous avons procédé à une analyse des logs d'erreurs et isolé les logs d'accès correspondant à ces erreurs ainsi que ceux correspondant à des accès inhabituels (i.e. les accès ne correspondant pas à des affichages de pages ou ceux n'étant pas identifiés par un navigateur web). Ces logs d'accès seront qualifiés dans leur globalité comme "*atypiques*".

Ce jeu de données est pré-traité de la façon suivante : chaque accès est transformé en un enregistrement dans lequel

- l'identifiant de l'objet est l'IP de provenance de la requête,
- les items sont les URLs demandées, le type de requête, l'erreur retournée ainsi que le logiciel utilisé pour se connecter,
- l'étiquette temporelle est donnée par la date de la connexion mesurée en secondes depuis le 1er Janvier 1970, 00 :00 :00.

Exemple 3.4. Le tableau 3.4 représente les tentatives de connexion le 1er Jan. 1970 à $00:00:05$ A.M. à partir de l'IP encodées par 253, vers l'URL "/PictureGallery/home.htm" par une requête de type "GET" avec le navigateur "Mozilla/4.0". L'erreur renvoyée est l'erreur 404 (la page n'existe pas). Cet accès est suivi 6 secondes plus tard par le même type de requête à l'URL "/PictureGallery/index.htm". Le logiciel n'est alors pas identifié et aucune erreur n'est retournée.

IP id	time	request	URL	error	software
253	5	GET	"/PictureGallery/home.htm"	404	"Mozilla/4.0"
253	11	GET	"/PictureGallery/index.htm"	–	–

TAB. 3.2 – Exemples de logs d'accès

Les accès atypiques représentent environ 22,000 tentatives de connexions sur une période d'environ un an, à partir de 510 IP différentes, vers 1191 URLs différentes.

Tout d'abord, nous avons comparé les performances de GETC à celles de GTC. Nous avons pu constater que le comportement des deux algorithmes en terme de temps d'exécution est globalement le même bien que GETC soit légèrement plus lent à cause du temps nécessaire au calcul de la précision temporelle.

L'intérêt de GETC réside dans l'information additionnelle donnée par la précision temporelle. En effet, comme dans la section précédente dans les expérimentations sur données synthétiques, nous avons exécuté GTC en prenant comme contraintes les valeurs des contraintes initiales correspondant aux paramètres d'entrée de GETC, le but étant de comparer les motifs extraits.

Tout d'abord, nous avons déterminé quelles valeurs spécifiées comme valeurs des contraintes initiales pour GETC et pour les contraintes de GTC. Le but de la fouille consistant à caractériser les comportements atypiques, nous avons choisi de décrire les profils de comportements non-humains, i.e. les profils résultant d'accès par des systèmes automatiques. Ce type de profils peut être décrit, par exemple, pour des requêtes répétées sur une courte période. Autrement dit, une telle séquence devrait être composée d'itemsets enregistrés sur une seconde (ws_{init}=0 et $\rho_{ws} = 1$, aucun regroupement d'enregistrements n'est autorisé). De même, la durée minimum entre deux itemsets prend sa plus petite valeur possible, sans varier (mg_{init}=0 et ρ_{mg}=1).
La contrainte correspondant aux comportements atypiques que nous souhaitons souligner est en fait la contrainte $maxgap$. En effet, des requêtes automatisées peuvent être vues comme des requêtes trop

proches pour être exécutées par une action humaine. La durée $maxgap$ séparant deux itemsets devra donc être la plus courte possible : MG_{init}=1 seconde. Cependant, afin de ne pas ignorer d'autres profils, cette contrainte est assouplie, en spécifiant un degré de précision temporelle minimale pour $maxgap$ inférieur à 1, ici ρ_{MG}=0.75. Nous utilisons ainsi la flexibilité de la contrainte étendue $maxgap$ afin de décrire plus précisément les profils atypiques.

Ensuite, nous avons comparé les différents résultats obtenus en faisant varié la précision temporelle de $maxgap$ de 1 (tous les motifs extraits sont les mêmes pour GETC et GTC) à 0.75 (plus de motifs sont découverts par GETC). Comme pour les données synthétiques, nous obtenons plus de motifs en utilisant les contraintes de temps étendues, une partie d'entre eux couvrant l'ensemble extrait sous contraintes strictes.

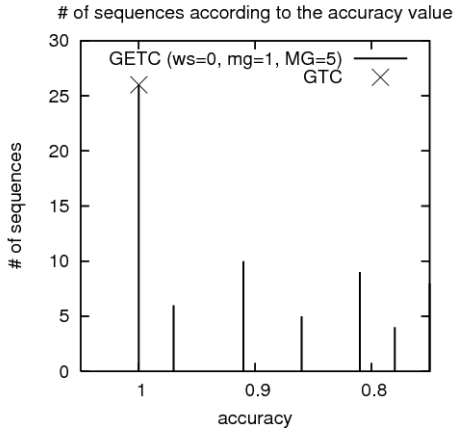


FIG. 3.8 – Répartition des motifs extraits selon leur précision temporelle (ws_{init}=0, mg_{init}=0, MG_{init}=0 et ρ_{ws}=ρ_{mg}=1, ρ_{MG}=0.75, $minFreq$=0.2.

L'histogramme 3.8 montre le nombre de motifs découverts par GETC, classés par valeur de la précision temporelle, ainsi que ceux extraits par GTC. On peut noter qu'environ 25 des motifs découverts par GETC ont une précision de 1. L'analyse qualitative montre que ces motifs sont exactement ceux extraits par GTC. De plus, GETC extrait des motifs supplémentaires dont la précision temporelle est inférieure à 1. Par exemple, une dizaine de motifs avec une précision d'environ 0.9.

En ce qui concerne les motifs extraits à la fois par GETC et GTC, en voici deux extraits :

- Le bot pxyscand envoie 5 requêtes CONNECT à l'URL 1185 du site Web. Chaque fois, il reçoit l'erreur 405 (Unauthorized method) (<("CONNECT", URL 1185, 405 Error, "pxyscand/2.1")...(3 times)...("CONNECT", URL 1185, 405 Error, "pxyscand/2.1")>).
- Un autre motif observé à la fois par GTC et GETC, est l'accès insistant à l'URL "/mambo..." qui n'existe pas. Il semblerait que cette requête soit une tentative d'attaque (<("/cvs/mambo/index2.php?_REQUEST...", 404 Error)("/cvs/mambo/index2.php?_REQUEST...", 404 Error)>).

En ce qui concerne les motifs additionnels, découverts par GETC, mais non trouvés par GTC, voici deux exemples :

- Le logiciel Pompos tente d'accéder régulièrement à des pages inexistantes, <("cvs/index2.php?request...", 404 Error,Pompos)...once...("cvs/index2.php?request...", 404 Error,Pompos)>, ce motif séquentiel a une précision temporelle de 0.95 et une fréquence de 35%.
- Un autre comportement typique d'un accès automatique est la navigation des robots des moteurs de recherche, en particulier Google's Image Bot. Ce moteur n'apparaît pas dans les analyses sous contraintes strictes. L'analyse des logs d'accès nous a permis de constater que la durée moyenne

entre deux requêtes est de 2 secondes <("/PictureGallery/thumbnail.php3?...",Googlebot-Image/1.0)...once... ("/PictureGallery/thumbnail.php3?...',Googlebot-Image/1.0)...>.

Discussion

Les motifs séquentiels généralisés étendent les motifs séquentiels en intégrant la spécification de contraintes temporelles entre les évènements des séquences. Ces contraintes, qui permettent de regrouper des transactions ou de les distinguer en séquences différentes, permettent d'extraire une connaissance moins évidente et plus proche des besoins de l'utilisateur. Toutefois, cette définition reste encore trop rigide, notamment dans le cas où l'utilisateur n'a qu'une vague idée des contraintes temporelles qui lient ses données.

C'est pourquoi nous avons proposé une extension des contraintes de temps pour les motifs séquentiels généralisés, basée sur la théorie des sous-ensembles flous, qui permet plus de souplesse dans la spécification des paramètres de contraintes temporelles, tout en autorisant aussi la spécification de contraintes rigides, selon les besoins de l'application et de l'utilisateur.

Par ailleurs, une fois extraits, les motifs séquentiels généralisés ne comportent aucune indication sur les données correspondant aux contraintes de temps effectivement présentes dans la base.

Notre proposition de contraintes de temps étendues permet de répondre partiellement à cette problématique, puisque la spécification de contraintes relâchées, grâce à la notion de précision temporelle, permet de calculer un niveau de respect moyen des valeurs initiales spécifiées pour chacune des contraintes $windowSize$, $mingap$ et $maxgap$. La mise en œuvre de notre approche via la génération de graphes de séquences permet par ailleurs de conserver l'efficacité de traitement et des performances d'exécution, tout en calculant l'information supplémentaire contenue dans la précision temporelle.

Néanmoins, si les motifs séquentiels généralisés peuvent être analysés par cet indicateur complémentaire, on peut s'interroger sur la facilité de lecture de cette mesure de précision temporelle pour un utilisateur peu expérimenté. En effet, la lecture de la précision temporelle sous la forme d'un nombre entre 0 et 1 donne une indication sur le choix initial des contraintes de temps et sur leur validité par rapport aux séquences de la base de données, mais elle ne renseigne pas précisément sur les durées fréquemment observées dans la base entre les évènements de chaque motif séquentiel. De plus, comme pour la fréquence des motifs séquentiels flous, la valeur de la précision temporelle dépend également du choix des opérateurs de t-normes et de t-conormes utilisés.

Il serait donc intéressant d'utiliser les graphes de séquences valués des degrés de respect des contraintes afin de proposer à l'utilisateur la liste des motifs séquentiels généralisés annotés, pour chaque itemset, du degré moyen de précision de $windowSize$, et pour chaque intervalle entre itemsets, du degré de précision moyen de $mingap$ et $maxgap$, sur le principe des séquences annotées par intervalles proposées dans [YISI00, GNPP06]. Au-delà de l'indication concernant la validité des contraintes spécifiées initialement, l'utilisateur disposerait alors d'un outil d'analyse plus fin.

Troisième partie

Motifs séquentiels et données incomplètes

Il n'est pas certain que tout soit incertain.
Blaise Pascal (1623-1662) — *Pensées*

Les bases de données issues d'applications pratiques (industrie, médecine, biologie, ...) contiennent de nombreuses informations non-renseignées, pour différentes raisons. Durant le processus d'ECD, une phase d'élimination des données incomplètes est souvent nécessaire car de nombreux outils de fouille de données n'analysent que les données complètes, sans tenir compte des enregistrements incomplets, ce qui constitue une grande perte d'information.

Lors de cette phase de pré-traitement, les données non complètement renseignées peuvent être supprimées ou complétées. Ces deux solutions sont lourdes de conséquences pour la connaissance qui sera ensuite extraite. En effet, la suppression peut parfois conduire à l'élimination de plus de la moitié de la base, aussi, l'information extraite n'est plus représentative. Par ailleurs, la complétion peut introduire un biais dans les données. Certaines méthodes permettent ainsi de choisir des valeurs afin de compléter au mieux les données incomplètes, mais elles ne conduisent pas toujours à des résultats satisfaisants [HK00].

Dans cette partie, nous nous intéresserons donc au traitement des valeurs manquantes et données incomplètes lors de l'extraction de motifs séquentiels. Nous proposons ensuite d'utiliser l'information temporelle contenue dans les séquences fréquentes afin d'élaborer une approche de complétion basée sur les motifs séquentiels. Le premier chapitre est consacré aux différentes techniques de fouille de données robustes aux valeurs manquantes. Les chapitres 2 et 3 sont ensuite consacrés à nos deux solutions pour l'intégration des données incomplètes lors de l'extraction de motifs séquentiels. Le chapitre 4, ensuite, propose une méthode de complétion des valeurs manquantes basée sur l'utilisation des séquences fréquentes. Enfin, cette partie s'achève sur une discussion des résultats obtenus, notamment concernant la complétion.

Chapitre 1

Traitement des valeurs manquantes et fouille de données

Le présent chapitre présente un état de l'art des travaux permettant la prise en compte des données incomplètes lors d'une fouille de données. Nous introduisons ainsi rapidement les approches dont nous nous sommes inspirés pour concevoir les deux algorithmes détaillés dans les chapitres suivants. La dernière section détaille plus précisément nos motivations.

1.1 Données incomplètes et valeurs manquantes

De nombreux travaux de recherche se sont focalisés sur l'étape de fouille de données et sur les différentes approches possibles. Or, si les résultats obtenus à l'issu du processus d'ECD dépendent en grande partie de la technique de fouille utilisée, ils dépendent également des données en entrée. En particulier, se pose le problème non trivial des données incomplètes [LL99a, LL99b]. En effet, dans les bases de données, les valeurs manquantes sont inévitables et peuvent avoir différentes significations.

1.1.1 Définitions

Une ***donnée incomplète*** est une donnée pour laquelle la valeur de certains attributs est inconnue, ces valeurs sont dites manquantes. Une ***valeur manquante*** cache donc une valeur d'un attribut. Par exemple, dans la figure 1.1, la valeur de l'attribut X4 pour l'enregistrement de d2 est manquante, l'attribut n'est pas renseigné et la donnée d2 est dite incomplète.

D	X1	X2	X3	X4
d1	a	b	c	d
d2	e	a	d	?

FIG. 1.1 – Valeur manquante et donnée incomplète

Il existe différentes sources d'incomplétude. L'absence de valeur peut être due à des erreurs dans la procédure d'acquisition : codes invalides, attributs inapplicables, refus de réponse... Il peut également s'agir d'attributs dépendants d'autres attributs, dont certains sont non renseignés. Enfin, des informations peuvent être manquantes car inaccessibles *e.g.* défaillance de capteur.

1.1.2 Différents types de valeurs manquantes

La plupart du temps, les valeurs manquent de manière aléatoire. On parle de ***valeurs manquantes au hasard***. La probabilité de rencontrer ces valeurs est alors totalement indépendante de l'échantillon observé [TK99, TDK03]. Dans ce cas, les valeurs manquantes représentent une partie non-disponible de l'information. La question est donc d'utiliser le plus possible l'information disponible pour l'analyse, sans diminuer la fiabilité des résultats obtenus.
Il arrive cependant que les valeurs manquantes soient structurées et informatives, on parle alors de ***valeurs manquantes catégorielles*** ou ***spécifiques de classe***. Dans ce cas, l'absence de valeur renseigne sur les résultats qui peuvent être obtenus. L'idée est alors d'utiliser cette information complémentaire pour la fouille de données. Par exemple, si les algorithmes permettent de découvrir que toutes les informations manquantes concernent des clients de Montpellier, il est alors aisé de classer toute nouvelle ligne de la base de données qui arriverait incomplète dans la catégorie des clients montpelliérains.

Selon le traitement et l'analyse réalisés, il pourra s'avérer nécessaire d'identifier l'origine des valeurs manquantes, car certaines techniques, bien adaptées pour des valeurs manquantes au hasard, sont inappropriées pour des valeurs manquantes catégorielles et inversement.

1.1.3 Les données manquantes dans le processus d'ECD

Lorsque le problème des données incomplètes est pris en charge lors du processus d'ECD (figure 1.2), ceci peut se faire à deux niveaux :

- soit lors de l'étape (2) de ***préparation*** comme c'est le cas le plus souvent ;
- soit lors de l'étape (3) de ***fouille de données***. Si la méthode d'analyse tolère les données incomplètes, l'algorithme de fouille de données ne requiert pas de traitement spécifique pour les valeurs manquantes et les données incomplètes seront gérées directement au moment de la fouille de données.

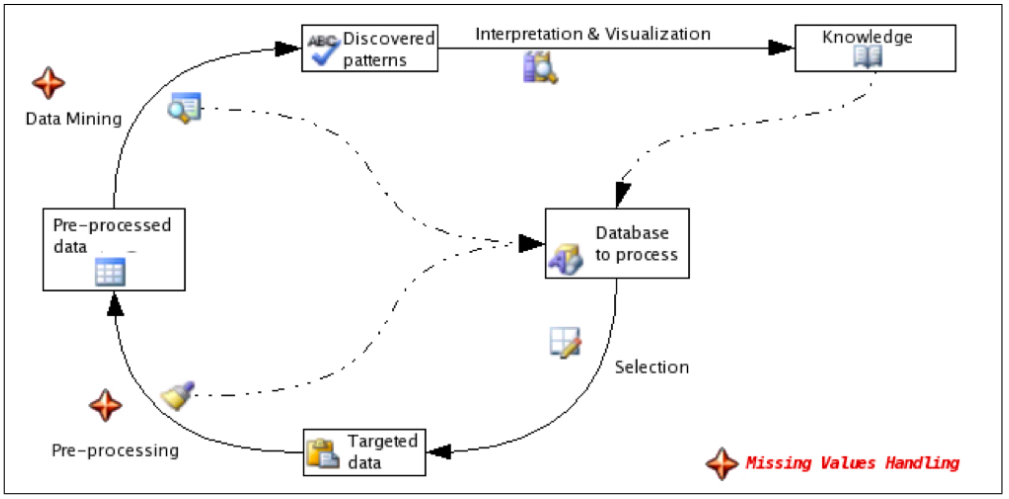


FIG. 1.2 – Les valeurs manquantes dans les différentes étapes du processus d'ECD

Beaucoup de travaux évoquent l'importance des problèmes liés à la présence de valeurs manquantes [FPSS96a, HK00, Pea06] et présentent les différentes approches pour tenir compte de l'incertitude sur ces données lors d'un processus d'ECD et plus précisément lors du pré-traitement des données.

[Pea06] souligne notamment les problèmes liés à la détection des valeurs manquantes qui ne doivent pas être traitées de la même façon que des attributs volontairement non renseignés. A l'inverse, dans certains cas, les valeurs inconnues, inapplicables ou encore non spécifiées sont encodées comme des valeurs valides.

Selon le type de valeurs manquantes que l'on rencontre, leur traitement devra être adapté [Pea06, Wri98]. De plus, de nombreux algorithmes de fouille nécessitent ce pré-traitement spécifique des données incomplètes. Il apparaît donc nécessaire, dans un premier temps, de détecter s'il y a une raison pour que la valeur soit inconnue et si l'ignorer peut détruire une information potentiellement utile. Il est intéressant également de détecter si l'enregistrement est utile ou non, et on ne traitera les valeurs manquantes que lorsque ce sera nécessaire.

Plusieurs possibilités sont alors envisageables afin de gérer les valeurs manquantes lors du pré-traitement :

- Soit on procède à la ***suppression*** des données comportant des valeurs manquantes ou des données incomplètes et/ou à la suppression d'un attribut du jeu de données si celui-ci est souvent non renseigné. Cette méthode n'est appropriée que si les valeurs manquantes sont rares, car si le pourcentage de valeurs manquantes est élevé, la perte d'information résultant de la suppression des données incomplètes n'est pas acceptable. De plus, la représentativité statistique de l'échantillon n'est plus garantie, étant donnée la réduction du nombre d'enregistrements conservés pour l'analyse [Hui00].
- Soit on effectue un ***remplissage*** (ou ***complétion***) des valeurs manquantes. Diverses manières sont alors possibles. Le remplissage par valeur statistique *(moyenne, médiane...)*, difficilement applicable aux gros volumes de données, permet d'obtenir des résultats qui varient de manière importante selon l'estimation réalisée. Par ailleurs, le remplissage doit être le plus proche possible de la réalité pour éviter d'introduire un biais trop important dans les données. En effet, les résultats ultérieurs varient de manière importante selon l'estimation réalisée [HK00]. C'est pourquoi plusieurs travaux se sont intéressés à une complétion par valeurs multiples [HL01, LR87], dans le but d'amoindrir les inconsistances pouvant résulter d'une complétion simple.

Différentes techniques permettent de traiter des données incomplètes lors de l'étape de fouille de données. Celles-ci sont souvent spécifiques à une application ou à un jeu de données. Il existe également quelques algorithmes qui permettent la classification par arbres de décision [Qui89, FKY96, LWTB97, WI00], règles [GJ94, BH97, BH98] ou clustering de jeux de données incomplets [TDK03, TK99]. Dans le cadre de la recherche de règles d'association, [RC98, CGM07] présentent des algorithmes afin de traiter les valeurs manquantes. Celui-ci ne fait pas intervenir la logique floue, mais divise la base de données en sous-ensembles complets. Les travaux [NC01, NL98] sont quant à eux basés sur l'utilisation d'une distribution de probabilités afin d'estimer les différentes valeurs possibles que peut prendre une valeur manquante dans la base de données. La section suivante présente tous ces travaux plus en détails.

1.2 Valeurs manquantes en classification et segmentation

1.2.1 Des valeurs manquantes à chaque étape de l'élaboration d'un classifieur

Le but de la classification est de modéliser un classifieur, c'est-à-dire un ensemble de règles permettant d'affecter des données à des classes prédéterminées. Cet ensemble de règles est obtenu à partir d'un ensemble d'apprentissage. Il s'agit donc, lors d'un processus de classification, de prendre en compte les enregistrements incomplets pendant l'apprentissage, soit par complétion [Qui89, BH98], soit en utilisant

uniquement l'information disponible [Whi86, LWTB97] ou encore en adoptant l'approche proposée par [WI00], en ne considérant pas les cas incomplets durant la phase d'apprentissage et en calculant l'erreur engendrée par cette approximation. Ensuite, il faut également pouvoir classer les nouveaux enregistrements, qui peuvent être incomplets, soit en utilisant la classe la plus probable en fonction des informations disponibles [Qui89], soit en estimant la valeur de l'attribut non renseigné [BFOS84].

[BH97, BH98] présente un classifieur capable de traiter des enregistrements comportant des attributs non-renseignés et utilisable dans trois scénarii différents : la classification avec des valeurs manquantes, le remplissage des vecteurs d'entrée et l'apprentissage sur des données incomplètes. Après la phase d'apprentissage, on dispose d'un classifieur basé sur des règles floues pour chacune des classes possibles. Afin de classer un enregistrement comportant des valeurs manquantes, l'ensemble de règles définies sur l'ensemble des attributs est projeté vers un ensemble de règles n'utilisant que des attributs dont les valeurs sont connues. La classification se fait donc uniquement à partir des attributs renseignés. Ce principe peut également être utilisé pour la complétion des données incomplètes. Pour cela, on détermine à quelle classe l'enregistrement appartient le plus probablement, puis on utilise les éléments de la classe pour compléter les valeurs manquantes.

Parmi les méthodes de classification supervisée, on trouve également les arbres de décision. Un arbre de décision est une structure arborescente, dont les feuilles représentent les classes et les nœuds les caractéristiques. Les chemins de l'arbre constituent les règles de classification. Lors d'une classification, on démarre de la racine de l'arbre et on le parcourt, selon les valeurs des attributs, jusqu'à avoir atteint la feuille de destination de l'objet à classer.

Un arbre de décision se construit par induction, à partir d'un ensemble d'apprentissage. Cet ensemble est généralement constitué d'un ensemble d'exemples de décisions prises par le passé, chacun comportant un certain nombre d'attributs, ainsi qu'un indicateur d'appartenance à une classe. Une fois l'arbre construit, il est utilisé pour prédire l'appartenance de classe d'enregistrements-tests. Il se pose donc deux problématiques dans le cas d'ensembles contenant des valeurs manquantes : l'existence de données incomplètes dans l'ensemble d'apprentissage et la présence d'un attribut non renseigné dans des cas-tests.

Il existe différentes manières de traiter les valeurs manquantes pendant la phase de construction de l'arbre. Selon la technique utilisée, les effets sur la construction seront différents. On distingue trois grands principes : le traitement d'une valeur *"inconnu"*, le remplissage des valeurs manquantes et l'omission des cas incomplets.

[Qui86] propose de traiter la valeur manquante comme une nouvelle valeur pour chaque attribut et donc comme toute autre valeur que peut prendre l'attribut. L'inconvénient de cette méthode vient du fait qu'elle se prête bien à l'analyse de valeurs manquantes catégorielles, mais plus difficilement à celle de valeurs manquantes au hasard. [Qui86] présente donc également une méthode, plus appropriée dans ce deuxième cas. Celle-ci est basée sur l'idée que les cas contenant des valeurs manquantes sont distribués de manière homogène dans l'ensemble d'apprentissage et attribue un statut différent à la valeur "inconnu". Cependant, cette méthode traite spécifiquement chacune des valeurs manquantes et ne tient pas compte de la structure de l'ensemble de données, elle n'utilise donc pas l'intégralité de l'information disponible.

[LWTB97] utilise les informations disponibles (valeurs de l'attribut pour la classe, valeurs des autres attributs pour les cas de la même classe...) afin de déterminer les valeurs manquantes. Toutefois, il apparaît que cette technique n'est appropriée que pour une faible concentration de données incomplètes et un nombre limité d'attributs non-renseignés (explosion combinatoire); [WI00] ne considère pas les cas incomplets durant la phase d'apprentissage et calcule l'erreur engendrée par cette approximation en utilisant le nombre de valeurs manquantes sur l'échantillon et introduit une pondération pour les données incomplètes. Enfin, la génération de chemin dynamique [Whi86] permet de construire l'arbre, en com-

mençant par l'ensemble des attributs pour lesquels les valeurs sont disponibles.

1.2.2 Clustering sur des données incomplètes

Selon [TK99], l'analyse de données incomplètes peut se faire grâce au clustering flou. Cette méthode nécessite toutefois de traiter les données incomplètes différemment selon l'origine des valeurs manquantes. La première étape consiste donc à analyser pour un ensemble de données les raisons de la présence de valeurs manquantes. Dans un deuxième temps, on recherche les corrélations dans la base.

Comme pour le clustering, l'objectif du clustering flou est de diviser un ensemble de données en un ensemble de clusters tels que la similarité intra-classes est nettement supérieure à la similarité inter-classes. Cependant, le but est de pouvoir traiter les données qui pourraient appartenir à plusieurs groupes en même temps. On introduit un degré d'appartenance aux différents clusters, calculé en fonction de la distance entre cette donnée et le cluster. Chaque cluster peut donc être considéré comme un sous-ensemble flou.

L'approche proposée dans [TK99, TDK03] consiste à adapter la formule de calcul des distances de manière à ce qu'elle puisse tenir compte des données incomplètes, ce qui permet d'omettre les valeurs manquantes, le fonctionnement général de l'algorithme restant le même. Ceci revient à considérer qu'un attribut non renseigné n'a pas d'influence sur l'affectation de l'enregistrement à un cluster et à prendre en compte la différence entre données complètes et incomplètes à l'aide d'une pondération. De plus, cette méthode permet de compléter des enregistrements incomplets en fonction du ou des clusters auxquels ils appartiennent.

Cette méthode de traitement des données incomplètes par clustering flou permet de compléter des valeurs manquantes à chaque itération de l'algorithme de clustering, sur le même principe que la complétion proposée par l'algorithme Expectation-Maximization [DLR77]. De plus, pour tenir compte de la différence entre données complètes et incomplètes, on réduit le degré d'appartenance des données incomplètes.

1.3 Règles d'association, valeurs manquantes et complétion

Dans le cadre des techniques de description, des travaux ont été proposés pour la recherche de règles d'association. [RC98, CGM07] présentent un algorithme afin de prendre en compte les données incomplètes lors de l'extraction des règles, par omission partielle et temporaire de ces enregistrements. Ces règles peuvent ensuite être utilisées afin de compléter les valeurs manquantes. [NL98, NC01], mettent en œuvre un système d'approximation probabiliste dans lequel une valeur manquante peut prendre plusieurs valeurs lors de la découverte des règles. Ces méthodes approximatives permettent d'extraire des règles proches de celles qui devraient être obtenues sur la base complète, tandis que [RC04] extrait des représentations condensées[1] exactes. Enfin d'autres méthodes utilisent les règles d'association et certains indices de confiance afin de compléter les valeurs manquantes, [WWC04, JJL+05]. Dans la section suivante, nous détaillons les concepts liés à la découverte de règles d'association, ainsi que le principe des méthodes de complétion qui les utilisent.

[1] Une *représentation condensée* est une collection, exacte ou approximative, de motifs d'une base de données permettant de répondre à une requête en un minimum de temps et en utilisant un minimum d'espace mémoire. Une définition plus précise est donnée dans [MT96]

1.3.1 ∼AR

Les travaux présentés dans [NC01, NL98] reposent sur une technique couramment utilisée dans les domaines de statistique et d'apprentissage. Le principe consiste à utiliser l'information disponible (i.e. les attributs renseignés) et à estimer grâce à elle les informations manquantes, avec un certain niveau de probabilité. Ainsi, ces méthodes mettent en œuvre un système d'approximation probabiliste dans lequel une valeur manquante prend plusieurs valeurs lors de la découverte des règles. Afin de prendre en compte ces estimations, les concepts de support (pourcentage des enregistrements de la base qui contiennent tous les items de la règle) et par extension, celui de confiance (la probabilité qu'un enregistrement qui contient la partie gauche de la règle contienne également la partie droite) ont été redéfinis.

La première étape de l'algorithme ∼AR consiste donc à remplacer chaque valeur manquante par une distribution de probabilité qui représente la probabilité pour la valeur manquante d'être chacune des valeurs possibles de l'attribut considéré. Cette distribution de probabilité est calculée par rapport à l'ensemble des données complètes pour l'attribut considéré.

Exemple 1.1. Soit la base de données donnée TAB. 1.1, dans laquelle " ?" représente une valeur manquante.

Id	X1	X2	X3	X4
1	A	B	C	G
2	E	F	E	?
3	?	B	E	A
4	A	B	F	F

TAB. 1.1 – Base de données exemple

Dans ce cas, la probabilité que la valeur manquante soit A est P(X1 = A) = 2/3, tandis que la probabilité qu'elle soit E est P(X1 = E) = 1/3.

La seconde étape de l'algorithme ∼AR consiste alors à découvrir les itemsets fréquents puis les règles d'association. Pour cela, le principe est le même que pour l'algorithme Apriori, seule la définition du support est modifiée.

Ainsi, au lieu d'incrémenter le nombre d'enregistrements supportant un itemset candidat dès lors que celui-ci est totalement inclus dans un enregistrement, les auteurs considèrent une appartenance partielle. Soit un candidat de longueur k. Chaque item dans un enregistrement va contribuer au support de l'itemset candidat jusqu'à un maximum de $1/k$. Si l'enregistrement est complet pour l'itemset candidat considéré, alors le support de ce candidat sera incrémenté de k * $1/k$ = 1. Par contre si l'enregistrement est incomplet, on va tenir compte de la distribution de probabilités précédemment calculée. Pour chaque item présent le poids de cet item sera $1/k$ et pour chaque item qui pourrait remplacé une valeur manquante, le poids sera $probabilite/k$. L'exemple suivant détaille ce calcul.

Exemple 1.2. Soit l'itemset candidat (A B E D) et l'enregistrement R = (A B ? D). On suppose que la distribution de probabilité pour le troisième attribut de la base est la suivante : P(X3 = C) = 1/4, P(X3 = E) = 1/2 et P(X3 = F) = 1/4.
Dans ce cas, le support du 4-itemset candidat (A B E D) pour l'enregistrement R sera incrémenté de 1/4 + 1/4 + 0.5/4 + 1/4 = 7/8.

1.3.2 Robust Association Rules

Des travaux ont été proposés pour la recherche de règles d'association dans des bases de données relationnelles incomplètes. Nous nous intéressons dans cette section aux travaux présentés dans

[RC98, RC99]. Ces travaux présentent l'algorithme RAR (Robust Association Rules).

Cette méthode, complètement compatible avec la méthode originelle [AIS93], permet la prise en compte des données incomplètes lors de l'extraction des règles dans des bases de données relationnelles incomplètes, par omission partielle et temporaire de ces enregistrements. Pour cela, la base est divisée en trois parties pour chaque règle, comme également présenté par [Kry99] : une partie regroupe les enregistrements contenant la règle de façon certaine, la seconde les enregistrements ne contenant pas la règle de façon certaine et la troisième contient les enregistrements pour lesquels on ne sait pas.

Le principe consiste à ne prendre en compte que les attributs renseignés pour les enregistrements incomplets. La base de données entière n'est pas utilisée pour chaque règle mais pour générer l'ensemble des règles. Cette technique repose sur la définition de bases de données valides, complètes pour un ensemble d'items donnés, le reste de la base étant momentanément ignoré.

Afin de prendre en compte ce partitionnement de la base, les concepts de support et de confiance ont été redéfinis. Par ailleurs, une nouvelle notion est introduite afin de tenir compte de la taille de l'échantillon complet considéré pour déterminer le support de la règle. Cette mesure de représentativité permet ainsi d'éliminer de la liste des règles celles trouvées sur une base peu significative par rapport à la base initiale. Un seuil de représentativité minimale, $minRep$, a donc été défini [RC98].

Exemple 1.3. On cherche les règles contenant X1 et X4 dans la base TAB. 1.1 : les enregistrements 2 et 3 sont désactivés pour cette règle, seuls les enregistrements 1 et 4 sont utilisés pour calculer le support et la confiance.

1.3.3 Règles d'association et complétion des enregistrements

Les méthodes de complétion des données incomplètes basées sur les règles d'association fonctionnent toutes sur le même principe : les règles d'association correspondant à l'enregistrement incomplet sont retenues, puis on utilise différents indices de pertinence afin de pouvoir conclure sur une valeur de remplacement.

Exemple 1.4. Prenons un enregistrement incomplet $(a\ b\ ?)$ ainsi que les règles $a\ b \rightarrow c$ et $a \rightarrow d$ respectivement de confiance 100% et 80%. Dans ce cas, si on prend la confiance comme indice de pertinence, la valeur de remplacement proposée sera c.

La méthode MVC (Missing Value Completion) présentée dans [RC99] utilise les règles de grande confiance dont le conséquent pourrait être une solution pour la valeur manquante et dont l'antécédent est proche de l'itemset à compléter. Deux situations sont alors possibles : toutes les règles correspondantes indiquent la même conclusion, soit plusieurs règles concluent sur différentes valeurs. Dans ce cas, le nombre de règles concluant sur la même valeur est utilisé pour résoudre le conflit automatiquement.

[WWC04], quant à eux, proposent d'utiliser une combinaison de plusieurs mesures pour noter les règles. Cette approche repose sur l'idée que les règles d'association décrivent les relations de dépendances qui existent dans les enregistrements d'une base de données, y compris dans les enregistrements incomplets. Les règles d'association peuvent alors être utilisées pour estimer les valeurs manquantes. Pour cela, les auteurs proposent un score pour chaque règle, défini en fonction du support, de la confiance et du lift de ces règles d'association.

Enfin, [JJL+05] utilisent des règles d'association dont le conséquent est un intervalle de valeurs. Ces règles ont une confiance égale à 1. Elles sont également triées selon une seconde mesure, qui permet d'attribuer un ensemble de valeurs très probables à une valeur manquante.

1.4 Motifs séquentiels et valeurs manquantes

1.4.1 Motivations

Nous souhaitons extraire les motifs séquentiels contenus dans la base de données TAB. 1.2. Avec une fréquence minimale de 50%, les motifs obtenus sur la base complète 1.2 sont : <(a b)(b c)(b c)>, <(a b)(b c d)> et <(a)(b c)(b d)>.

Obj.	Séquence
O1	$(a\ b)\ (b\ c\ d)\ (b\ c\ e)$
O2	$(a)\ (b\ c)\ (b\ d)$
O3	$(a\ b)\ (b\ c)\ (b\ c\ d)$

TAB. 1.2 – Base de données complète.

Obj.	Séquence
O1	$(a\ b)\ (?\ ?\ c)\ (?\ b\ c)$
O2	$(a)\ (?\ c)\ (b\ d)$
O3	$(a\ b)\ (?\ c)\ (?\ ?\ c)$

TAB. 1.3 – Base de données incomplète.

Obj.	Séquence
O1	$(a\ b)\ (c)\ (b\ c)$
O2	$(a)\ (c)\ (b\ d)$
O3	$(a\ b)\ (c)\ (c)$

TAB. 1.4 – Après suppression des valeurs manquantes.

Considérons maintenant la même base, mais incomplète, TAB. 1.3. Pour certaines séquences de données, les informations n'ont pas été transmises et des valeurs sont manquantes. Afin de pouvoir extraire les motifs, les méthodes classiques requièrent une élimination des valeurs manquantes. La base sur laquelle s'effectue la fouille de données est alors la base TAB. 1.4 et les motifs obtenus pour *minFreq* = 50% sont : <(a b)(c)(c)> et <(a)(c)(b)>.

On constate que seule une petite partie de la base est utilisée pour extraire l'information et on ne retrouve qu'une partie des schémas fréquents extraits de la base complète : des sous-séquences des motifs que l'on devrait extraire. De plus, certains items ne sont plus trouvés fréquents. C'est pourquoi il paraît nécessaire d'utiliser l'intégralité de la base lors de la fouille et non pas d'en supprimer une partie.

1.4.2 Objectifs

Dans le contexte de la recherche de motifs séquentiels, les valeurs manquantes n'ont pas été considérées jusqu'ici, l'application principale, les bases de données de supermarchés n'en comportant quasiment jamais. Désormais, les motifs séquentiels sont utilisées afin d'extraire des connaissances d'applications industrielles (analyse de processus, web access logs, ...) qui contiennent inévitablement des données incomplètes. Dans les chapitres suivants, nous présenterons donc deux propositions d'algorithmes pour l'extraction de motifs séquentiels dans des bases de données de séquences incomplètes.

Tout d'abord, dans le chapitre suivant, nous présenterons une première méthode d'extraction de motifs séquentiels pour des bases de données incomplètes, SPoID, basée sur la proposition [RC98]. Cette technique utilise un principe de désactivation partielle et temporaire des séquences de données incomplètes. La seconde approche que nous proposons, ApSPoID, est décrite dans le chapitre 3. Elle est fondée sur une estimation des différentes valeurs possibles d'une valeur manquante, en fonction du contenu de la base. Dans les deux cas, la notion de fréquence est modifiée.

Dans le chapitre 4, nous considérons les données incomplètes d'une autre façon. Il s'agit d'utiliser les motifs séquentiels extraits par les méthodes SPoID et ApSPoID afin de compléter au mieux des séquences de données incomplètes.

Chapitre 2

Traitement par désactivation

L'élimination des enregistrements incomplets conduisant à une perte d'information, nous avons donc envisagé d'adapter une méthode d'extraction de règles d'association robuste aux valeurs manquantes pour extraire des motifs séquentiels. Nous présentons ici la méthode SPoID (Sequential Patterns for Incomplete Database), inspirée de l'algorithme RAR, présenté dans [RC98].

2.1 SPoID : désactivation des séquences de données incomplètes

Le principe général de notre méthode, comme de la méthode RAR, repose sur la désactivation des éléments incomplets, dans notre cas, des séquences. Alors que pour les règles d'association, l'algorithme RAR ne considère que les enregistrements complets, nous proposons ici de ne prendre en compte que les séquences de données complètes pour la séquence candidate recherchée. Autrement dit, on ne considérera que les dates et attributs renseignés pour les séquences incomplètes. Ainsi pour chaque séquence on déterminera si elle est fréquente sur une base partielle, mais la totalité de la base sera utilisée pour l'ensemble des séquences fréquentes.

2.1.1 Principe

Considérant une séquence candidate S, l'ensemble $\mathcal{O}$ des objets de la base peut être divisé en trois sous-ensembles disjoints (Fig. 2.1) :

- l'ensemble des séquences de données qui supportent S, noté $\mathcal{O}_S$,
- l'ensemble des séquences de données qui ne supportent pas S, noté $\mathcal{O}_{\overline{S}}$,
- l'ensemble des séquences de données pour lesquelles on ne sait pas si elles supportent S ou non, noté $\mathcal{O}^*_S$.

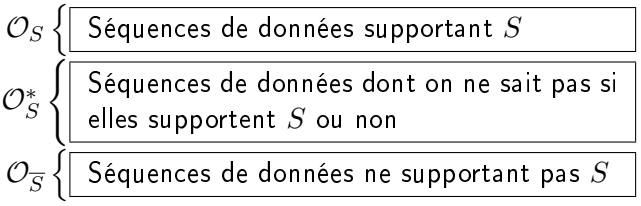


Fig. 2.1 – Découpage de la base de données selon l'inclusion de S

Pour chaque séquence candidate S, on ne va donc considérer que le sous-ensembles $\mathcal{O}_{\overline{S}} \cup \mathcal{O}_S$ afin de déterminer si la séquence S est fréquente ou non. Ce sous-ensemble de séquences de données forme la ***base de données valide*** pour S.

Définition 2.1. Une *base de données valide* est une base de séquences complètes pour une séquence candidate donnée.

La construction d'une base de données valide repose sur la ***désactivation*** temporaire des séquences de données qui contiennent des valeurs manquantes pour les items présents dans la séquence candidate. Cette désactivation implique une redéfinition de la notion de fréquence, pour prendre en compte la désactivation d'une partie de la base.

Définition 2.2. Une séquence de données est *désactivée* pour une séquence candidate S si elle est incomplète pour S (i.e. on ne sait pas si elle supporte S ou non). On note $Dis(S)$ l'ensemble des séquences désactivées pour la séquence candidate S, c'est-à-dire l'ensemble $\mathcal{O}_S^*$.

2.1.2 Fréquence et représentativité

La notion de fréquence doit donc être modifiée afin de prendre en compte la notion de base de données valide et pour considérer le fait que seule une partie de la base est utilisée.

Définition 2.3. La ***fréquence*** d'une séquence S est la proportion d'apparition de cette séquence parmi les séquences de données qui peuvent la supporter. On la définit comme le ratio du nombre de séquences de données qui supportent cette séquence, par le nombre de séquences de données dont on est sûr qu'elles incluent ou non cette séquence (complètes pour cette séquence). Elle est donnée par la formule :

$$Freq(S) = \frac{|\mathcal{O}_S|}{|\mathcal{O}| - |Dis(S)|}$$

Propriété 2.1. En considérant qu'aucun des ensembles O_S, O'_S, $O_{\overline{S}}$ et $O_{\overline{S'}}$ n'est vide, cette définition de la fréquence respecte la propriété d'antimonotonie de la fréquence énoncée dans [AS95].

Démonstration.

$$Freq(S) = \frac{|\mathcal{O}_S|}{|\mathcal{O}| - |Dis(S)|} = \frac{|\mathcal{O}_S|}{|\mathcal{O}_S| + |\mathcal{O}_{\overline{S}}|} = \frac{1}{1 + \frac{|\mathcal{O}_{\overline{S}}|}{|\mathcal{O}_S|}}$$

Or, si $S' \subseteq S$, alors $|\mathcal{O}_{S'}| \geq |\mathcal{O}_S|$. En effet, si une séquence S' est supportée par un objet o, alors soit cet objet supporte sa sur-séquence S, soit il ne la supporte pas, soit on ne sait pas, mais dans tous les cas, aucun objet n'incluant pas S' ne peut inclure sa sur-séquence S.

On montre également que si $S' \subseteq S$, alors $|\mathcal{O}_{\overline{S}}| \geq |\mathcal{O}_{\overline{S'}}|$. En effet,

$$\begin{array}{l} o \in \mathcal{O}_{\overline{S'}} \Rightarrow \exists i | s'_i \not\subset \varsigma_o \quad \text{et} \quad S' \subseteq S \Rightarrow \forall j, \exists a_j | s'_j \subseteq s_{a_j} \\ \Rightarrow \exists k \backslash s_k \not\subset \varsigma_o \\ \Rightarrow o \in \mathcal{O}_{\overline{S}} \end{array}$$

On utilise ensuite les deux inégalités précédentes. En considérant qu'aucun de ces cardinaux n'est nul, on peut multiplier membre à membre les deux inégalités 2.1.2 et 2.1.2. Ce qui nous donne :

$$\begin{array}{rcl} |\mathcal{O}_{\overline{S}}||\mathcal{O}_{S'}| \geq |\mathcal{O}_S||\mathcal{O}_{\overline{S'}}| & \Leftrightarrow & 1 + \frac{|\mathcal{O}_{\overline{S}}|}{|\mathcal{O}_S|} \geq 1 + \frac{|\mathcal{O}_{\overline{S'}}|}{|\mathcal{O}_{S'}|} \\ & \Leftrightarrow & \frac{1}{1+\frac{|\mathcal{O}_{\overline{S'}}|}{|\mathcal{O}_{S'}|}} \geq \frac{1}{1+\frac{|\mathcal{O}_{\overline{S}}|}{|\mathcal{O}_S|}} \\ & \Leftrightarrow & Freq(S') \geq Freq(S) \end{array}$$

La fréquence définie pour les bases de données valides est donc antimonotone. □

La nouvelle définition de la fréquence étant antimonotone, on peut utiliser les différentes propriétés énoncées dans [AS95] afin de réaliser l'extraction de motifs séquentiels sur base de données incomplètes. Toutefois, la notion de ***séquence fréquente*** dépend du calcul de la ***fréquence*** et de la taille de la base de données valide utilisée pour le calculer. On définit un critère de ***représentativité*** minimale : une base de données valide doit être un échantillon significatif de la base de départ pour qu'on considère la séquence comme fréquente si elle valide la fréquence minimale $minFreq$.

Définition 2.4. La ***représentativité*** $Rep(S)$ d'une séquence S est définie comme le ratio du nombre de séquences de données où cette séquence apparaît complète ou n'apparaît pas avec certitude dans la base, par rapport au nombre total de séquences de données dans la base. Elle est donnée par :

$$Rep(S) = \frac{|\mathcal{O}| - |Dis(S)|}{|\mathcal{O}|}$$

Définition 2.5. Une base de données est ***représentative*** si sa représentativité est supérieure à une valeur minimale donnée.

Pour être considérée comme fréquente, une séquence doit donc être trouvée fréquente sur une base de données valide et représentative, c'est-à-dire si sa fréquence est supérieure à la fréquence minimale spécifiée par l'utilisateur $minFreq$ et que sa représentativité est supérieure au seuil minimal de représentativité $minRep$.

Par ailleurs, afin de respecter les conditions nécessaires à l'antimonotonie de la fréquence, nous imposerons une valeur de représentativité minimale strictement positive.

2.1.3 Seuil de représentativité et marge d'erreur

Les statisticiens disposent d'outils d'échantillonnage permettant de considérer un sous-ensemble d'une population afin d'estimer une proportion, à un pourcentage d'erreur près, avec un niveau de confiance suffisant. Ces outils permettent de déterminer la taille optimale d'un échantillon en fonction de la distribution des données. Ainsi, dans le cas d'une distribution au hasard des données, [Toi96] utilise les bornes de Chernoff pour déterminer la taille minimale d'un échantillon tiré au hasard pour l'extraction de règles d'association. Ce résultat est également démontré théoriquement et empiriquement dans [ZPLO96].

Nous proposons donc d'utiliser deux formes de représentativité selon le souhait de l'utilisateur : soit celle-ci sera définie comme une proportion de la base, soit elle sera absolue et calculée à partir de formules statistiques dépendant de la distribution des données, en respectant un pourcentage d'erreur et un niveau de confiance spécifiés par l'utilisateur.
Les expérimentations montrent toutefois que le seuil de représentativité optimale n'est pas absolu mais dépend en fait du nombre et de la répartition des valeurs manquantes contenues dans la base de données.

2.2 Mise en œuvre

2.2.1 Illustration

Reprenons la base incomplète présentée page 96 et rappelée TAB. 2.1, avec $minFreq$=50%. Calculons la fréquence et la représentativité de chacun des items de la base pour déterminer les items fréquents.

L'item a est supporté de manière sûre par les trois objets, donc $Freq(a) = 3/3$, soit 100% et sa représentativité vaut 1. Il en est de même pour les items b et c. Pour l'item d, on a $\mathcal{O}_{<d>} = \{O2\}$ et $Dis(< d >) = \{O1, O3\}$, donc $Freq(d)$ = 1/(3-2) et $Rep(d)$ = (3-2)/3 = 0.33. Si le seuil $minRep$

Obj.	Séquence
O1	$(a\ b)\ (?\ ?\ c)\ (?\ b\ c)$
O2	$(a)\ (?\ c)\ (b\ d)$
O3	$(a\ b)\ (?\ c)\ (?\ ?\ c)$

TAB. 2.1 – Base de données incomplète.

vaut 0.3, alors $Rep(d) > minRep$ et d est un item fréquent. Par contre, si $minRep$ = 0.4, alors $Rep(d) < minRep$ et d n'est pas un item fréquent car la base valide qui permet de calculer sa fréquence n'est pas suffisamment représentative.

On fixe $minRep$=0.3. Prenons maintenant la séquence $S =< (a\ b)(a\ b\ c\ d) >$. Cette séquence ne peut être supportée par aucune des trois séquences de données, car aucune ne comporte d'itemsets de 4 items, complets ou non, donc $\mathcal{O}_S = \{\}$, $Dis(S) = \{\}$ et $\mathcal{O}_{\overline{S}} = \{O1, O2, O3\}$ et $Freq(S)$=0. Cette séquence n'est pas fréquente.
La séquence $S' =< (a\ b)(b\ c) >$, quant à elle, est supportée par $O1$, ne peut l'être par $O2$ mais pourrait l'être par $O3$. On a donc $\mathcal{O}_{S'} = \{O1\}$, $Dis(S') = \{O3\}$ et $\mathcal{O}_{\overline{S'}} = \{O2\}$, ce qui donne :
$Freq(S')$=1/(3-1)=50% et Rep(S')=(3-1)/3 = 0.67. S' est donc représentative et fréquente.

En appliquant la méthode ci-dessus, les motifs extraits pour $minFreq$=50% et $minRep$ = 0.3 sont : <(a b)(c)(b c)> et <(a)(c)(b d)>. Même si ces motifs ne sont pas exactement ceux extraits sur la base complète, on constate qu'ils sont proches de ces motifs.

Les expérimentations présentées dans la section 2.3 montrent qu'il existe une valeur de la représentativité à partir de laquelle l'algorithme SPoID extrait d'une base incomplète l'intégralité des motifs extraits sur la base complète.

2.2.2 Algorithme

Le principe de l'algorithme SPoID est similaire aux algorithmes d'extraction de motifs séquentiels de type générer-élaguer. Il consiste à générer toutes les séquences candidates de longueur k à partir des séquences fréquentes de longueur k-1, puis à scanner l'ensemble de la base pour compter le nombre de séquences de données qui supportent chacune des séquences candidates. La différence réside dans le dénombrement des séquences de données incomplètes pour la séquence candidate considérée. La phase de comptage est décrite par l'algorithme ALG. 1 : pour chaque séquence candidate, pour chaque objet,

- si on trouve la séquence candidate, on incrémente les valeurs absolues de la fréquence,
- si on ne trouve pas la séquence candidate ni de séquence dans laquelle des valeurs manquantes pourraient être remplacées par les items correspondant dans la séquence candidate, alors l'objet ne supporte pas la séquence candidate. On n'incrémente pas la fréquence.
- sinon, on trouve une séquence incomplète dans laquelle des valeurs manquantes pourraient être remplacée par les items correspondant dans la séquence candidate. Dans ce cas, on ajoute cet objet à l'ensemble des objets désactivés.

Une fois l'ensemble de la base parcouru, on divise la valeur absolue de la fréquence par la différence entre le nombre total d'objet s et le nombre d'objets désactivés, puis on calcule la représentativité. Enfin, on procède à la phase d'élagage en supprimant toutes les séquences candidates qui ne sont pas fréquentes puis celles qui ne sont pas représentatives.

SPoID - **Input** : $|\mathcal{O}|$, une base de séquences de données, *minFreq*, fréquence minimale spécifiée par l'utilisateur, *minRep*, représentativité minimum, spécifiée ou calculée
Ouput : $SPList$, liste des séquences fréquente

$C \leftarrow \{i \in \mathcal{I}\}$; $k = 1$;
$F \leftarrow$ getFrepnRep(C,$minFreq$,$minRep$) ;
$SPList$.add(F) ;
Tant que ($C \neq \varnothing$) **faire**
  k++ ;
  $C \leftarrow$ generate(F,k) ;
  Pour chaque séquence candidate $s \in C$ **faire**
    Pour chaque objet $o \in \mathcal{O}$ **faire**
      [On cherche s dans S_o]
      Si ($s \in S_o$) **Alors**
        $Freq(s)$++ ;
        $Dis(s) \leftarrow Dis(s) \backslash$o ;
      Sinon
        Si ($\tilde{s} \in S_o / \tilde{s}$ pourrait être s) **Alors** $Dis(s) \leftarrow Dis(s) \cup o$; **Fin Si**
      Fin Si
    Fin Pour
    $Freq(s) \leftarrow Freq(s)/(|(O)| - |Dis(s)|)$;
    $Rep(s) \leftarrow |(O)| - |Dis(s)|/|(O)|$;
    Si ($(Freq(s) < minFreq) || (Rep(s) < minRep)$) **Alors** prune($s$) ; **Fin Si**
  Fin Pour
Fin Tant que

ALG. 1 : *SPoID - algorithme général.*

2.3 Expérimentations

Ces expérimentations ont été réalisées sur un PC équipé d'un processeur 2,8GHz et de 512Mo de mémoire DDR, sous système Linux, noyau 2.6. Nous utilisons un jeu de données synthétiques dans lequel nous remplaçons certains items par des valeurs manquantes, réparties de manière aléatoire, afin de comparer les motifs séquentiels extraits sur la base complète d'une part, avec les motifs extraits sur la base pré-traitée (dont les items incomplets ont été supprimés) et ceux extraits par notre algorithme SPoID. Les résultats présentés ici ont été obtenus à partir du traitement de plusieurs jeux de données synthétiques comportant environ 2000 séquences de 20 transactions en moyenne. Chacune de ces transactions comportant environ 10 items choisis parmi 100.

Nos analyses sont basées sur le calcul du nombre de bons motifs séquentiels trouvés par SPoID et du nombre de motifs différents extraits par SPoID, ces derniers regroupant les motifs extraits, qui n'existent pas dans la base complète et les motifs non trouvés, mais contenus dans la base complète. Le tableau 2.2 récapitule l'ensemble de ces notations.

β	Nombre de motifs extraits par SPoID, contenus dans la base complète (vrais positifs)
δ	Nombre de motifs différents (faux positifs et faux négatifs)
θ	Nombre de motifs extraits par SPoID sur la base incomplète ($\theta = \beta + \delta$)
τ	Nombre de motifs extraits sur la base de données complète

TAB. 2.2 – Notations pour les différentes catégories de motifs séquentiels extraits.

La FIG. 2.2(a), tout d'abord, montre l'évolution du rapport β/θ, en fonction de la représentativité minimale. On constate que ce taux croît à mesure que $minRep$ augmente, ce qui signifie que parmi les motifs extraits par SPoID, la proportion des motifs également trouvés dans la base complète augmente avec la représentativité minimale.

On peut compléter cette observation par l'analyse de la FIG. 2.2(b), qui présente l'évolution du rapport β/τ (nombre de motifs extraits par rapport aux motifs à extraire) en fonction de la représentativité minimale. On constate que ce ratio diminue à mesure que $minRep$ augmente, ce qui signifie qu'il est nécessaire de choisir une représentativité suffisamment faible pour permettre l'extraction de l'intégralité des motifs présents dans la base complète.

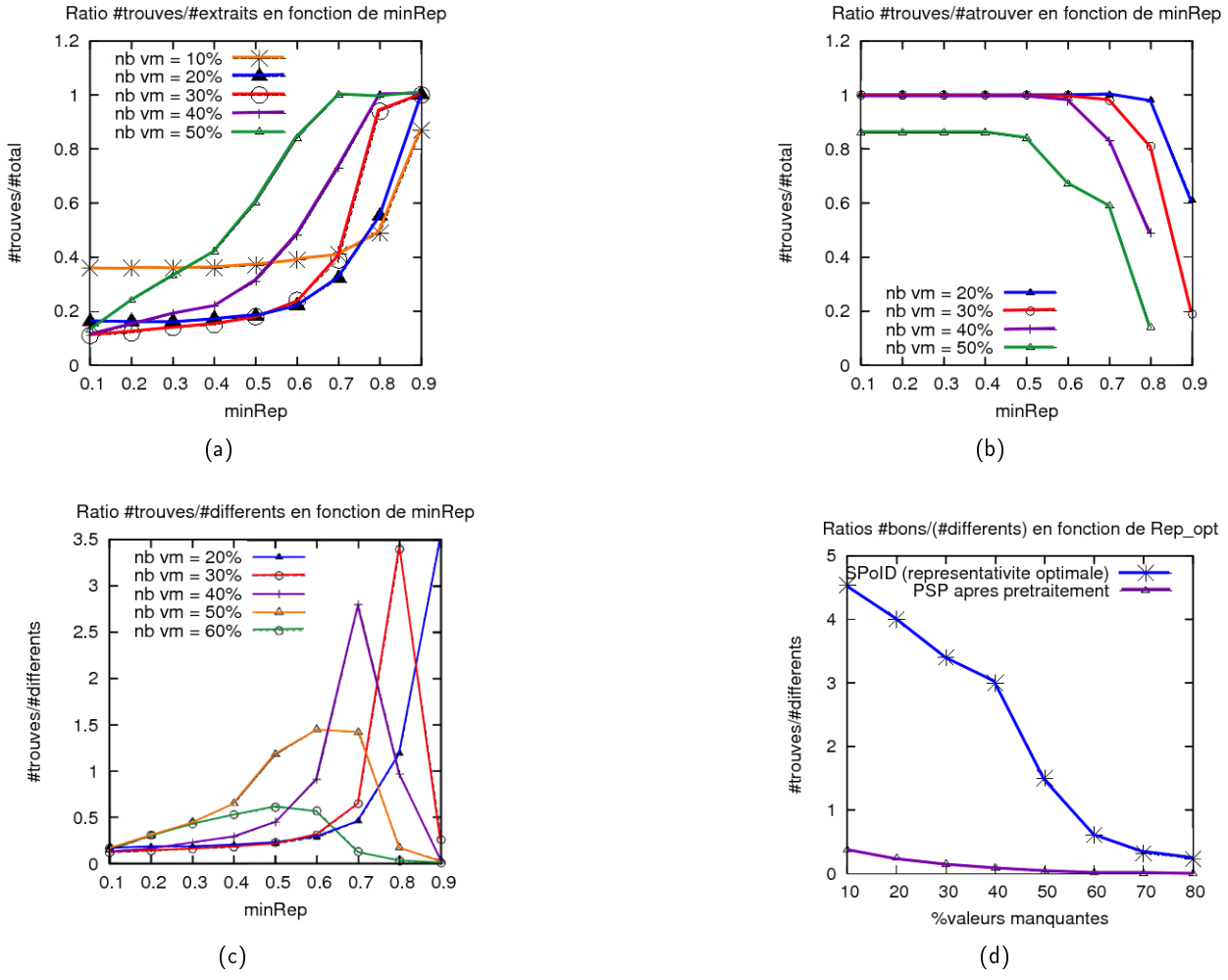


FIG. 2.2 – (a) : Proportion (bons motifs)/(motifs extraits) (β/θ) en fonction de la représentativité minimale ; (b) : Proportion (bons motifs)/(motifs à trouver) (β/τ) en fonction de la représentativité minimale ; (c) : Proportion (bons motifs)/(motifs différents) (β/δ) en fonction de la représentativité minimale ; (d) : Proportion (bons motifs)/(motifs différents) (β/δ) en fonction du pourcentage de valeurs manquantes dans la base.

Nous avons mis en évidence l'existence d'une valeur optimale du seuil de représentativité, pour laquelle les ratios β/θ (bons motifs/motifs extraits par SPoID) et β/τ sont les plus proches possibles de 1. Cette valeur correspond au seuil pour lequel le nombre de bons motifs extraits sur la base incomplète

est le plus élevé possible par rapport au nombre de motifs différents. La FIG. 2.2(c) met en évidence l'existence de cette valeur optimale de $minRep$. Ce graphe montre l'évolution du rapport (β/δ), rapport du nombre de bons motifs extraits par SPoID et du nombre de motifs qui diffèrent des motifs extraits sur la base complète (manquants + supplémentaires). On constate qu'il n'existe pas de valeur absolue de la représentativité minimale, commune à toutes les bases et dépendant d'une marge d'erreur donnée. D'après ces résultats, la représentativité minimale dépend uniquement du taux d'incomplétude de la base.

Quelle que soit la proportion de valeurs manquantes dans la base incomplète, l'allure générale de la courbe est la même : le ratio (β/δ) augmente avant d'atteindre un maximum puis de décroître. Ce point maximal correspond à la représentativité optimale, pour laquelle le nombre de bons motifs extraits par SPoID est le plus élevé et le nombre de motifs différents le plus faible. Le tableau TAB. 2.3 donne la valeur de la représentativité optimale trouvée expérimentalement pour chaque proportion de données incomplètes dans notre jeu de test.

% de valeurs manquantes	10%	20%	30%	40%	50%	60%	70%	80%
$minRep$ optimale	0.97	0.9	0.81	0.74	0.6	0.48	0.39	0.22

TAB. 2.3 – Moyenne des représentativités optimales selon la proportion de valeurs manquantes dans la base de données.

La FIG. 2.2(c) met aussi en évidence l'évolution de comportement de l'algorithme SPoID selon le taux de valeurs manquantes dans la base. On constate une différence entre l'allure de la courbe et la valeur du ratio β/δ pour les bases incomplètes contenant 40% de valeurs manquantes et moins et celles qui comportent 50% d'enregistrements incomplets ou plus.
Ainsi, la FIG. 2.2(d) permet de comparer le taux de réussite de SPoID et d'une extraction sur données préparées (base incomplète dans laquelle les données ont été supprimées). Elle montre l'évolution du ratio optimal de bons motifs trouvés par SPoID selon le pourcentage d'incomplétude de la base. On constate que ce taux chute rapidement lorsque l'on passe de 40 à 50% de valeurs manquantes : le nombre de motifs différents devient proportionnellement plus important par rapport au nombre de bons motifs.
On remarque également que ce ratio devient inférieur à 1, lorsque le pourcentage de valeurs manquantes dépasse 50%. SPoID permet donc d'extraire les motifs séquentiels d'une base incomplète tant que plus de la moitié de ses enregistrements sont complets alors qu'un algorithme classique ne permet plus de trouver tous les motifs fréquents dès 10% de valeurs manquantes. Lorsque le taux d'incomplétude atteint 40%, le nombre de bons motifs extraits est même quasi nul, alors qu'il reste encore relativement élevé pour SPoID.

Chapitre 3

Traitement par estimation

La seconde approche que nous proposons consiste à estimer, pour chaque valeur manquante, les chances qu'elle représente l'un des items présents dans la base, et ainsi, à tenir compte de l'incertitude liée à la présence d'une valeur manquante.

3.1 ApSPoID : estimation des valeurs possibles dans les données incomplètes

3.1.1 Principe

Nous présentons ici la méthode ApSPoID (Approximative Sequential Patterns for Incomplete Database), inspirée de l'algorithme ∼AR, présenté dans [NC01].

Le principe général de notre méthode, comme de la méthode ∼AR, repose sur l'estimation de la valeur des items non-renseignés. Alors que pour les règles d'association, l'algorithme ∼AR calcule la fréquence d'un itemset en tenant compte de la probabilité de chaque valeur possible d'une valeur manquante, nous proposons ici d'utiliser cette estimation lors du calcul de la fréquence d'une séquence.

Ainsi, lors du calcul de la fréquence des items d'une séquence, à chaque valeur manquante, on attribue plusieurs valeurs possibles, chacune avec un ***niveau d'apparition*** possible. La notion de ***fréquence*** doit donc être modifiée afin de prendre en compte ce degré de représentation des items.
Il s'agit alors de pondérer la présence d'un item dans l'itemset par le niveau de certitude que l'on a de sa présence :

- Si un item est présent, le niveau d'apparition est de 1,
- Si un item est absent et qu'il n'y a pas de valeur manquante dans l'enregistrement considéré, alors le niveau d'apparition est nul,
- Sinon, le niveau d'apparition de l'item i dans l'enregistrement r est compris dans l'intervalle $]0, 1[$. Il sera noté $\mu_r(i)$

Définition 3.1. La ***fréquence*** d'un itemset I est le niveau d'apparition de cet itemset pour tous les objets de la base, pour chacun d'eux, on prendra la meilleure représentation (i.e. celle dont on est le plus sûr). Elle est donnée par la formule (3.1) :

$$Freq(I) = \frac{\sum_{o \in \mathcal{O}} \underline{\perp}_{r \in \mathcal{R}_o} \overline{\top}_{i \in I} \mu_r(i)}{|\mathcal{O}|} \tag{3.1}$$

Définition 3.2. La *fréquence* d'une séquence S est le taux d'apparition de cette séquence parmi les séquences de données qui peuvent la supporter. On la définit comme le ratio de la somme du meilleur niveau d'apparition de la séquence pour toutes les séquences de données de la base par le nombre de séquences de données. Elle est donnée par la formule (3.2) :

$$Freq(S) = \frac{\sum_{o \in \mathcal{O}} L(o, S)}{|\mathcal{O}|} \tag{3.2}$$

où $L(o, S)$ est le niveau d'apparition de S dans la séquence de données de l'objet o.

Définition 3.3. Le *niveau d'apparition* d'une séquence S pour un objet o correspond au meilleur taux d'inclusion de la séquence S dans la séquence de données de o. Ce niveau d'apparition est calculé par agrégation du taux d'inclusion de chaque itemset, dans l'ordre de la séquence S. Il est donné par la formule (3.3) :

$$L(S, o) = \perp_{\varsigma \subseteq \zeta_o | S = \varsigma = <r_1 \ldots r_g>} OrdAgr_{<r_1 \ldots r_g>}(\overline{\top}_{j \in r_i} \mu_{r_i}(j)) \tag{3.3}$$

où k est le nombre d'itemsets dans S.

Dans la pratique, nous utiliserons une agrégation par le min, pour son idempotence et pour satisfaire l'antimonotonie de la fréquence.

3.1.2 Détermination des distributions de valeurs pour chaque valeur manquante

La première étape de la méthode consiste à déterminer les valeurs possibles de chaque valeur manquante ainsi que le niveau de certitude de cette valeur.
On souhaite déterminer une approximation de la composition d'un enregistrement incomplet. Une passe sur la base est donc réalisée lors de laquelle le taux d'apparition de chaque item par enregistrement est calculé. La base de données étant incomplète, nous avons choisi d'utiliser la technique de désactivation des transactions sur le principe de RAR [RC98]. Ce calcul nous donne ainsi une distribution de fréquences qui sera ensuite utilisée pour déterminer le taux d'inclusion d'un item dans un enregistrement. La distribution de valeurs est ici calculée en fonction des fréquences d'apparition dans les enregistrements mais elle pourrait l'être en fonction des fréquences d'apparition dans les séquences de données, ou encore, calculée en ne considérant que les enregistrements complets.

Ainsi, lors du scan de la base à la recherche d'une séquence, lorsqu'un itemset est trouvé complet dans un enregistrement, son taux d'inclusion sera 1 et lorsqu'il est trouvé partiellement dans un enregistrement incomplet, on utilise la distribution de valeurs pour déterminer le taux d'inclusion de la partie manquante de l'itemset. Nous présentons dans la section 3.3 quelques résultats expérimentaux qui mettent en évidence l'influence du choix de la distribution sur le résultat de la fouille.

3.2 Mise en œuvre

Dans cette section, nous illustrons les définitions, ainsi que les principes précédemment décrits. Dans la seconde partie, nous présentons le fonctionnement de notre algorithme.

3.2.1 Illustration

Reprenons la base incomplète utilisée dans le chapitre 2 et rappelée dans le tableau 3.1, avec $minFreq$=50%.

Obj.	Séquence
O1	$(a\ b)\ (?\ ?\ c)\ (?\ b\ c)$
O2	$(a)\ (?\ c)\ (b\ d)$
O3	$(a\ b)\ (?\ c)\ (?\ ?\ c)$

TAB. 3.1 – Base de données incomplète.

La première étape consiste à calculer la distribution des taux d'apparition de chaque item grâce à la formule proposée par [RC98] :

$$freq(i) = \frac{|\{r \in \mathcal{R}/i \in r\}|}{|\mathcal{R}| - |\{r \in \mathcal{R}/r.isIncomplete\}|}$$

Tout d'abord, l'ensemble de la base est composé de 9 enregistrements, répartis en trois enregistrements par objet, donc $|R| = 9$. Le calcul de la fréquence se fait alors en distinguant pour chaque item, les enregistrements qui le contiennent de manière sûre, ce qui ne le contiennent pas de façon certaine et ceux pour lesquels on ne sait pas, i.e. les enregistrements incomplets qui ne contiennent pas l'item considéré.

Pour l'item a, par exemple, celui-ci est présent dans le 1er, le 4ème et le 7ème enregistrement, et est absent de l'enregistrement 6 ((bd)) qui est complet. Par contre pour les enregistrements 2, 3, 5, 8 et 9 qui sont incomplets, a n'apparaît pas. Donc, la fréquence de a est $freq(a) = \frac{card(\{r_1,r_4,r_7\})}{card(\mathcal{R})-card(r_2,r_3,r_5,r_8,r_9)}$, soit 3/(9-5) = 0.75.

Pour l'item c, cet item apparaît dans les enregistrements 2, 3, 5, 8 et 9 et n'apparaît pas dans les enregistrements 1, 4, 6 et 7 qui sont tous complets. La fréquence de c est donc : $freq(c) = \frac{card(\{r_2,r_3,r_5,r_8,r_9\})}{card(\mathcal{R})-card(\varnothing)} = 5/9 = 0.56$.

Le tableau 3.2 donne la distribution de valeurs pour chaque valeur possible d'une valeur manquante.

Item	a	b	c	d	e
Fréquence	0.75	0.8	0.56	0.25	0

TAB. 3.2 – Distribution de fréquences pour les valeurs manquantes.

Ainsi, pour le deuxième enregistrement de l'objet 1, le taux d'inclusion de c est de 1, mais pour a, b, d et e, il est donné par la distribution de fréquences ci-dessus. La représentation en sous-ensemble flou de cet enregistrement est donnée par la figure 3.1.

On commence ensuite la recherche de motifs séquentiels, par le calcul des fréquences de chaque item. L'item a apparaît de manière certaine dans la séquence de données de chacun des trois objets, sa fréquence est donc égale à 3/3, soit 100%. Il en est de même pour les items b et c. Pour l'item d, il pourrait être supporté par le deuxième enregistrement de l'objet 1 et également de l'objet 3. Sa fréquence est donc donnée par le calcul suivant :

$$\begin{aligned} Freq(< d >) &= \frac{\perp(0.25,0.25)+\perp(0.25,1)+\perp(0.25,0.25)}{3} \\ &= \frac{0.25+1+0.25}{3} \\ &= 0.5, \text{ soit } 50\% \end{aligned}$$

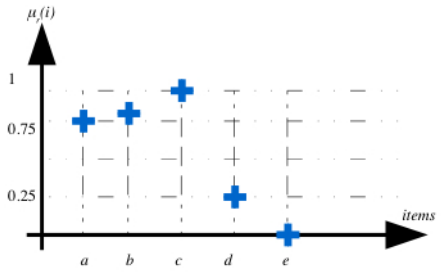


FIG. 3.1 – Représentation de l'enregistrement 2 du premier objet.

Prenons maintenant la séquence $S = < (ab)\ (abcd) >$. Cette séquence ne peut être supportée par aucune des trois séquences de données, car aucune ne comporte d'enregistrements de quatre items. La séquence $S' = < (ab)\ (bc) >$, quant à elle, est supportée par l'objet 1, ne peut l'être par le deuxième, mais peut l'être par l'objet 3. On a donc :

$$Freq(S') = \frac{1 + 0 + \top(1 + \top(0.8, 1))}{3} = \frac{1 + 0.8}{3} = 60\%$$

La séquence S' est donc fréquente.

En appliquant cette méthode, les motifs séquentiels extraits pour $minFreq$=50% sont : $< (a)(bd) >$, $< (ab)(bc)(abc) >$ et $< (ab)(ac)(abc) >$. Même si ces motifs ne sont pas exactement ceux extraits sur la base complète, on constate qu'ils couvrent l'ensemble des items fréquents.

Les expérimentations présentées dans la section 3.3 montrent que globalement, ApSPoID extrait une part importante des motifs séquentiels trouvés sur la base complète ainsi qu'un certain nombre de motifs supplémentaires, qui dépendent de la distribution de valeurs initialement choisie pour les valeurs manquantes.

3.2.2 Algorithme

Le principe de l'algorithme ApSPoID est similaire aux algorithmes d'extraction de motifs séquentiels de type générer-élaguer. Il consiste à générer toutes les séquences candidates de longueur k à partir des séquences fréquentes de longueur k-1, puis à scanner l'ensemble de la base pour compter le nombre de séquences de données qui supportent chacune des séquences candidates. La différence réside dans le dénombrement des séquences de données incomplètes pour la séquence candidate considérée.

La phase de comptage reprend les principes de comptage utilisés par l'algorithme TOTALLYFUZZY, pour la recherche de motifs séquentiels flous (partie I, chapitre 2). Elle est décrite par l'algorithme 1.

Pour chaque séquence, on scanne les enregistrements à la recherche des itemsets de la séquence. Si on trouve le premier, complet, alors on conserve la valeur 1 comme taux d'inclusion du premier itemset, puis on cherche l'itemset suivant. Si on le trouve incomplet, dans un enregistrement incomplet, alors son taux d'inclusion prend la valeur $\min_{i \in I} \mu_r(i)$, puis on cherche la suite de la séquence. Par contre si l'enregistrement ne supporte que partiellement l'itemset et que le nombre de valeur manquante ne permet pas de le compléter, ce premier itemset n'est pas trouvé et on le cherche dans la suite de la séquence de données.

ApSPoID -
Input : $\mathcal{O}$, base de données de séquences ;
minFreq, fréquence minimale
Ouput : $SPList$, liste de séquences fréquentes

[Computation of the frequency distribution]
float[] $freqDist \leftarrow$ calcDist() ;

[Search for frequent items]
$C \leftarrow \{i \in \mathcal{I}\}$; $k = 1$;
$F \leftarrow$ getFrepnRep(C,$minFreq$) ;
$SPList$.add(F) ;

[Search for frequent sequences, size $\geq$ 2]
Tant que ($C \neq \varnothing$) **faire**
  k++ ; $C \leftarrow$ generate(F,k) ;
  Pour chaque sequence candidate $s \in C$ **faire**
    Pour chaque objet $o \in \mathcal{O}$ **faire**
      float[] degTab ; $ag \leftarrow 0$; $id \leftarrow 1$;
      Pour (j=1 to $|\mathcal{R}_o|$ **faire**
        $Trans \leftarrow \{r_j, ...r_{|\mathcal{R}_o|}\}$;
        findSequence(s,$Trans$,$degTab$) ;
        $ag \leftarrow \max(ag, \min i = 1^{id} degTab[i])$;
      Fin Pour
      $tmpFreq \leftarrow tmpFreq + ag$;
    Fin Pour
    $Freq(s) \leftarrow tmpFreq(s)/|(O)|$;
    Si ($(Freq(s) < minFreq)$) **Alors**
      prune(s) ;
    Fin Si
  Fin Pour
  $SPList$.add(F) ;
Fin Tant que
Retourner $SPList$;

ALG. 1 : *ApSPoID* - Algorithme principal.

findSequence -
Input : s, séquence candidate ;
t, une séquence de données ;
d, table des niveaux d'apparition des itemsets de s ;
n, ordre de l'itemset en cours de recherche
Ouput : d

$sTail \leftarrow s \backslash s$.first() ; $tTail \leftarrow t \backslash t$.first() ;
Si (($sTail \neq \varnothing$)&($tTail \neq \varnothing$)) **Alors**
  Si ($\min_{i \in s.\text{first}} \mu_{s.\text{first}}(i)$) **Alors**
    $d[n] \leftarrow \min_{i \in s.\text{first}} \mu_{s.\text{first}}(i)$;
    findSequence($sTail$, $tTail$, d, $n+1$) ;
  Sinon
    findSequence(s, $tTail$, d, n) ;
  Fin Si
Sinon
  Retourner d ;
Fin Si

ALG. 2 : *findSequence* - Algorithme de recherche

Si la séquence est trouvée, on calcule son niveau d'apparition dans la séquence de données. Si elle est trouvée plusieurs fois, on retient le meilleur niveau d'apparition calculé.

Une fois l'ensemble de la base parcouru, on divise la somme des niveaux d'apparition par objet par le nombre total d'objets. Puis on procède à la phase d'élagage en supprimant toutes les séquences candidates qui ne sont pas fréquentes.

La sous-fonction $findSequence$ (ALG. 2) est une fonction récursive qui parcourt une séquence de données t à la recherche de la séquence candidate s. Lorsqu'elle trouve l'itemset de s recherché, alors elle recherche le suivant de la séquence candidate, dans la suite de la séquence t. Sinon, elle continue de rechercher le même itemset de la séquence candidate dans la séquence de données.

Cet algorithme peut être optimisé pour diminuer le nombre de passes sur la base en utilisant la structure de *chemins* présentés dans la partie I, chapitre 2. On conserve alors en mémoire la meilleure représentation courante de la séquence et toutes les amorces possibles de celle-ci lors du parcours de la séquence de données.

3.3 Expérimentations

Ces expérimentations ont été réalisées sur un PC équipé d'un processeur 2,8GHz et de 512Mo de mémoire DDR, sous système Linux, noyau 2.6. Nous utilisons un jeu de données synthétiques, construit aléatoirement selon une loi normale, dans lequel nous remplaçons certains items par des valeurs manquantes, réparties de manière aléatoire, afin de comparer les motifs séquentiels extraits sur la base complète d'une part, avec les motifs extraits sur la base pré-traitée (dont les items incomplets ont été supprimés) et ceux extraits par notre algorithme ApSPoID. Les résultats présentés ici ont été obtenus à partir du traitement de plusieurs jeux de données synthétiques comportant environ 2000 séquences de 20 transactions en moyenne. Chacune de ces transactions comportant en moyenne 10 items choisis parmi 200.

Nos analyses sont basées sur le calcul du nombre de bons motifs séquentiels trouvés par ApSPoID (vrais positifs) et du nombre de motifs différents extraits par ApSPoID, ces derniers regroupant les motifs extraits, qui n'existent pas dans la base complète (faux positifs) et les motifs non trouvés, mais contenus dans la base complète (faux négatifs). Le tableau 3.3 récapitule l'ensemble des notations utilisées dans cette section.

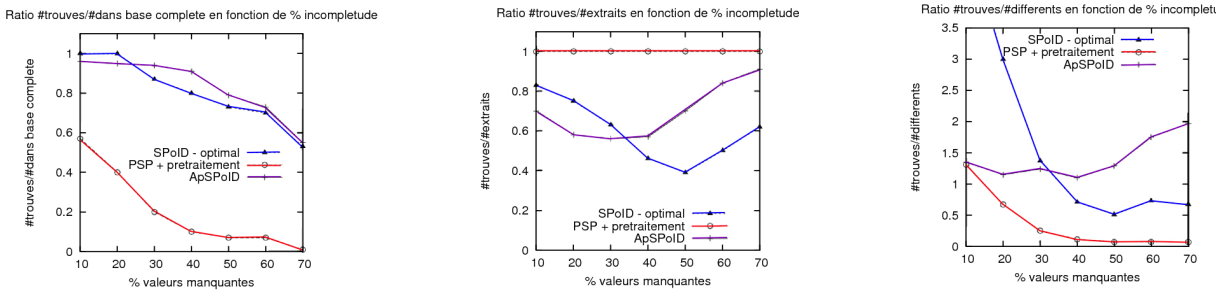
β	Nombre de motifs extraits par ApSPoID, contenus dans la base complète
δ	Nombre de motifs différents (manquants + supplémentaires)
θ	Nombre de motifs extraits par ApSPoID sur la base incomplète
τ	Nombre de motifs extraits sur la base de données complète

TAB. 3.3 – Notations pour les différentes catégories de motifs séquentiels extraits.

3.3.1 ApSPoID vs. SPoID

Comparaison qualitative

Les figures qui suivent présentent une comparaison des motifs extraits par ApSPoID aux motifs extraits avec PSP après prétraitement (élimination des valeurs manquantes) ainsi que ceux extraits par SPoID, méthode développée dans le chapitre précédent (chapitre 2).



(a) (b) (c)

FIG. 3.2 – (a) : Proportion (β/τ) en fonction du taux d'incomplétude de la base ; (b) : Proportion (β/θ) en fonction du taux d'incomplétude de la base ; (c) : Proportion (β/δ) en fonction du taux d'incomplétude de la base.

Tout d'abord, la figure 3.2(a) montrent l'évolution du ratio des bons motifs proportionnellement aux motifs extraits sur la base complète. ApSPoID et SPoID ont des comportements similaires en ce qui

concerne le nombre de bons motifs extraits. Plus particulièrement, ApSPoID offre de meilleurs résultats de 40 à 70% d'incomplétude alors que sur cet intervalle, les résultats de SPoID sont moins intéressants que sur l'intervalle 10 à 20%, sur lequel SPoID extrait tous les motifs de la base complète. On remarque également que quelque soit le pourcentage de valeurs manquantes, la méthode par prétraitement est peu performante.

Ensuite, les figures 3.2(b) et 3.2(c) permettent de comparer la proportion des motifs parasites ainsi que des motifs manquants. La figure 3.2(b) présente la proportion du nombre de bons motifs par rapport aux motifs extraits, en fonction du pourcentage de valeurs manquantes dans la base de données. Ce ratio est considéré comme bon, lorsqu'il est proche de 1 (tous les motifs extraits sont bons). Toutefois, ces résultats sont à confronter avec la figure 3.2(a). En effet, tous les motifs extraits peuvent correspondre à la base complète mais un nombre important d'entre eux peuvent manquer, comme c'est le cas avec PSP.
On remarque alors que les résultats obtenus par ApSPoID sont meilleurs que ceux de SPoID à partir de 40% de valeurs manquantes environ. En effet, à ce taux, ApSPoID extrait 90% des motifs à trouver et ces motifs représentent environ 60% des motifs extraits. De même, lorsque la base contient 50% de valeurs manquantes, ApSPoID extrait 80% des motifs de la base complète et ces motifs représentent 70% des motifs extraits, alors que SPoID n'extrait que 70% des motifs de la base complète, mais la proportion de motifs en trop est plus importante, puisque la part des bons motifs parmi les motifs extraits n'est que de 40%.

Ceci est confirmé par la figure 3.2(c), qui montre l'évolution du ratio β/δ (bons motifs / (manquants+supplémentaires)). Cette figure montre qu'à partir de 35%, la proportion de bons motifs par rapport aux motifs différents est nettement plus importante pour ApSPoID que pour SPoID. De plus, le nombre de bons motifs est toujours supérieur à la part de motifs différents (le ratio est toujours supérieur à un), contrairement à SPoID dont les résultats sont plutôt bons jusqu'à 30% de valeurs manquantes puis s'effondrent rapidement.

Performances temporelles

ApSPoID apparaît beaucoup plus constant en terme de temps d'exécution (FIG. 3.3). Alors que le temps d'exécution de SPoID reste inférieur à celui de ApSPoID jusqu'à environ 40% de valeurs manquantes, au-delà, ApSPoID devient nettement plus efficace. Si on compare ces performances par rapport à la qualité des motifs extraits, on peut dire que SPoID est la méthode qui convient aux bases comportant moins de 40% de valeurs manquantes, les résultats obtenus étant proches des motifs de la base complète et le temps d'exécution restant acceptable. Par contre, si la base contient plus de 40% de valeurs manquantes, il vaut mieux utiliser ApSPoID, qui est à la fois plus efficace en temps d'exécution et fournit des motifs de meilleure qualité.

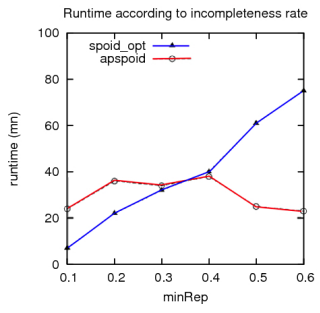


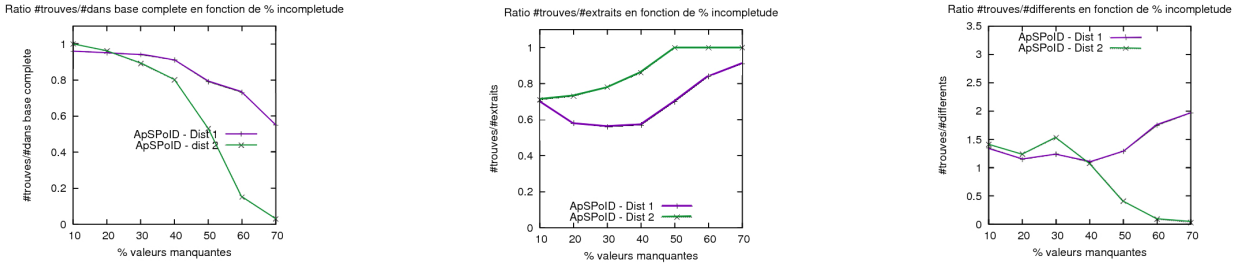
FIG. 3.3 – Comparaison des temps d'exécution.

3.3.2 Choix de la distribution

L'analyse du comportement de ApSPoID selon le choix de la distribution de valeurs pour les itemsets incomplets montre que celle-ci a une influence sur les résultats obtenus. Nous présentons ici une comparaison des performances de ApSPoID utilisé avec deux distributions :

- une distribution calculée par la méthode RAR (comme dans l'exemple de la section 3.2). Celle-ci est utilisée par "ApSPoID - Dist 1" dans la figure 3.4,
- une distribution calculée en incrémentant uniquement lors de la présence mais en considérant la totalité des enregistrements pour le calcul de la fréquence. Celle-ci est utilisée par "ApSPoID - Dist 2" dans la figure 3.4,

Alors que la première distribution est plutôt optimiste (chaque valeur manquante est ignorée et n'intervient pas dans le calcul de la fréquence des items), la deuxième est plutôt pessimiste (chaque valeur manquante est considérée comme n'étant pas l'item dont on calcule la fréquence). Le choix de l'une ou de l'autre a donc un impact sur le résultat final.



(a) (b) (c)

FIG. 3.4 – (a) : Proportion (β/τ) en fonction du taux d'incomplétude de la base ; (b) : Proportion (β/θ) en fonction du taux d'incomplétude de la base ; (c) : Proportion (β/δ) en fonction du taux d'incomplétude de la base.

Ainsi, la figure 3.4(a) montre que la seconde distribution ne permet de trouver l'ensemble des motifs de la base complète que jusqu'à 40% de valeurs manquantes dans la base, alors que la distribution optimiste trouve un nombre important de motifs jusqu'à 60% d'incomplétude. Par contre la figure 3.4(b) montre que globalement, la seconde distribution extrait moins de motifs parasites. La figure 3.4(c) permet de conclure que la distribution pessimiste donne de meilleurs résultats que la distribution optimiste jusqu'à un taux d'incomplétude d'environ 40%.
Sur l'intervalle 20 à 40% d'incomplétude, les résultats obtenus par ApSPoID avec la deuxième distribution sont donc comparables à ceux obtenus par SPoID.

On retiendra que le choix d'une distribution plus ou moins optimiste déterminera quelle quantité de motifs séquentiels de la base complète ApSPoID pourra extraire mais également le pourcentage de motifs parasites. Il faudra alors admettre un compromis entre le pourcentage de bons motifs extraits et le pourcentage de motifs parasites (compromis précision/rappel).

Ces expérimentations nous montrent donc que globalement, ApSPoID est plus fiable que SPoID à partir de 30% de valeurs manquantes et que ces résultats sont intéressants jusqu'à environ 60% de données incomplètes. Par ailleurs, les motifs extraits par ApSPoID - Dist 1 contiennent en moyenne 80 à 90 % des motifs extraits sur la base complète pour un taux d'incomplétude de 10 à 40%. Ces résultats sont légèrement moins bons que ceux de SPoID, toutefois l'utilisation de ApSPoID ne requiert pas la spécification d'autres paramètres que la fréquence contrairement à SPoID, l'utilisation d'ApSPoID est

donc facilitée. Par ailleurs, les résultats obtenus par ApSPoID pourront être améliorés en affinant la distribution de fréquences à chaque itération, en considérant de plus en plus précisément le contenu de chaque enregistrement. Enfin, en ce qui concerne la comparaison d'ApSPoID avec SPoID en terme de temps d'exécution, nous avons pu constater que les deux méthodes ont des performances équivalentes.

Chapitre 4

Processus de complétion des valeurs manquantes

Comme nous l'avons vu précédemment, les valeurs manquantes peuvent être gérées à différentes étapes du processus d'extraction de connaissances : lors de la fouille de données, ou plus avant, lors de la préparation des données. Lorsque les valeurs manquantes sont prises en compte lors du pré-traitement, différentes techniques peuvent être mises en œuvre [HK00, Wri98, Pea06] :

- La première solution consiste à ignorer les enregistrements incomplets. Toutefois, la qualité des schémas extraits risque d'être faible si les données incomplètes sont trop nombreuses ;
- La seconde option consiste à inférer les valeurs des attributs incomplets en utilisant un remplissage manuel, une valeur statistique ou une méthode d'induction ou de régression. Cependant ces approches ne conduisent pas toujours à des résultats satisfaisants, et elles ne sont pas toujours applicables sur de gros volumes de données.

Dans ce chapitre, nous proposons d'utiliser l'information temporelle contenue dans les motifs séquentiels afin d'inférer les valeurs possibles pour chaque valeur manquante. La méthode proposée a été implémentée afin de pouvoir gérer de grandes bases de données. Nous comparons les résultats obtenus en utilisant différents critères de choix des motifs séquentiels qui sont utilisés pour compléter chacune des valeurs manquantes.

4.1 Motifs séquentiels pour compléter des valeurs manquantes

Le principal intérêt des motifs séquentiels, par rapport aux règles d'association ou à d'autres méthodes, réside dans la prise en compte des informations temporelles contenues dans la base de données, de l'ordre liant les enregistrements de celle-ci et les informations inter-enregistrements contenues dans les séquences fréquentes. Cette information, inutilisée jusqu'à maintenant lors d'un complétion de valeurs manquantes, pourrait améliorée les résultats obtenus par les règles d'association.

4.1.1 Apport de l'information temporelle

Jusqu'ici, les méthodes proposées pour la complétion et basée sur les règles d'association utilisent les corrélations intra-enregistrements ainsi que les mesures de support et de confiance. Nous proposons ici, à l'aide des motifs séquentiels, d'utiliser également les corrélations inter-enregistrements, afin d'affiner les estimations faites pour la complétion d'enregistrements incomplets.

Prenons la base de données incomplètes du tableau Tab. 4.1. Les règles d'association ne permettent pas de proposer une valeur de remplacement pour l'item manquant de la transaction 2 puisqu'elles

Tid	Client	Date	Items
1	1	1	10 20
2	1	2	?
3	1	3	40
4	2	1	10 30
5	2	2	50
6	2	3	30 40
7	3	1	10 20
8	3	2	50
9	3	3	30 40

TAB. 4.1 – Base de données exemple

s'appuient sur des corrélations. Or, l'utilisation des motifs séquentiels permet de trouver une solution possible. En effet, en prenant par exemple $minFreq$=60%, on peut extraire deux séquences fréquentes maximales : <(10)(50)(30 40)> et <(10 20)(40)>. La première de ces deux séquences fréquentes maximales s'applique à la séquence à compléter et permet de conclure que la valeur manquante dans la transaction Tid 2 est l'item 50.

4.1.2 Approche proposée

Le processus général de complétion d'un base incomplète à partir des schémas extraits par une méthode de fouille de données est souvent basé sur le principe décrit par la figure 4.1.

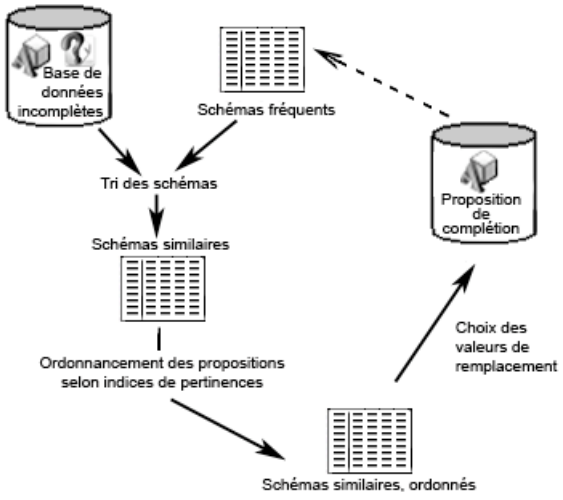


FIG. 4.1 – Processus général de complétion d'un base incomplète

La première étape consiste à obtenir les schémas fréquents caractéristiques de la base, soit en fouillant uniquement les enregistrements complets (s'il constitue une sous-base significative de la base principale), soit en utilisant toute la base avec un méthode de fouille prenant en compte les valeurs manquantes, sans estimation de celles-ci. Ensuite, pour chaque donnée incomplète, on recherche le ou les schémas extraits les plus similaires, qui permettraient de conclure sur une valeur de remplacement. Ces schémas peuvent conclure sur une valeur unique de complétion ou au contraire ils peuvent aboutir à plusieurs solutions. Dans ce cas, il est nécessaire de faire intervenir l'expert des données et d'utiliser un ou plusieurs indices

de pertinence afin d'ordonner les solutions et de choisir la valeur de remplacement la plus appropriée. La dernière étape de ce processus consiste à fournir la base de données complétée avec, pour chaque enregistrement incomplet, la ou les valeurs possibles, selon les besoins de l'utilisateur final et de son application.

La méthode de complétion des valeurs manquantes basées sur les motifs séquentiels que nous proposons fonctionne également sur ce principe. Pour chaque séquence de données incomplète dans la base de données, pour chaque valeur manquante, nous identifions les motifs séquentiels similaires à la séquence de données à compléter. Notre première implémentation de la recherche de séquences similaires est une approche de comparaison d'itemsets et de recherche d'inclusion. Dans un deuxième temps, nous envisageons de nous appuyer sur la recherche d'alignement de motifs proposés dans [KPWD02], pour améliorer la rapidité de notre système de complétion.

Puis, afin de proposer une solution ou un ensemble ordonné de solutions, nous utilisons différents indices pour trouver les motifs les plus pertinents pour chacune des valeurs manquantes. Nous utilisons pour le moment des indices génériques tels que la fréquence et le taux de correspondance entre les itemsets de la séquence incomplète et ceux du motif. La prochaine étape de ce travail aura pour but la mise en œuvre de différentes mesures de qualité pour les motifs séquentiels.

4.1.3 Un exemple

Considérons la séquence de données incomplète S_i = <(a b c) (? h) (e f g)>, ainsi que l'ensemble de motifs séquentiels, présentés dans le tableau TAB. 4.2.

	Motifs séquentiels	fréquence
1	<(a e) (h) (j)>	70%
2	<(a b) (u) (e f)>	60%
3	<(a c) (w) (e f g)>	50%
4	<(c) (t) (e b)>	50%

TAB. 4.2 – Liste de motifs séquentiels

Une fois la liste des motifs séquentiels obtenue, ceux qui correspondent le mieux à la séquence incomplète S_i sont identifiés. La similarité entre les deux séquences est basée sur la similarité entre les itemsets ordonnés. Sur la figure 4.2, on constate que les motifs les plus proches sont les motifs (2) et (3), et que les motifs (1) et (4) sont peu adéquats pour la complétion de cette séquence incomplète, car ces séquences sont peu similaires à la séquence de données.

	<(a b c)	(? h)	(e f g)	>
(1)	<(a e)	(h)		(j)>
(2)	<(a b)	(u)	(e f)	>
(3)	<(a c)	(w)	(e f g)	>
(4)	<(c)	(t)	(e b)	>

FIG. 4.2 – Similarités entre la séquence de données et les motifs séquentiels

L'étape suivante consiste à déterminer laquelle des valeurs u ou w, respectivement proposées par les motifs (2) et (3), est la plus appropriée, puisque les deux séquences sont aussi proches que possible de

S_i (tous les itemsets correspondent). Un premier indice de pertinence, intuitif, est la longueur du motif, auquel cas, la valeur retenue sera celle du motif (3), w. Une autre possibilité pourrait être l'ordonnancement par rapport à la fréquence et dans ce cas, c'est le motif (2) qui sera retenu et donc la valeur u. Il apparaît donc nécessaire de définir et d'utiliser plusieurs indices de pertinence, afin de proposer une valeur de complétion optimale, correspondant au mieux à la base de données.

4.2 Expérimentations

Afin de tester cette approche, nos expérimentations ont pour but d'évaluer la sélection des motifs à utiliser pour la complétion (motifs similaires) et la qualité de la complétion de la base incomplète selon les indices de pertinence retenus. Nous générons donc plusieurs jeux de données synthétiques correspondant à la même base complète mais comportant une proportion variable d'enregistrements incomplets. Nous pourrons ainsi évaluer dans un premier temps l'efficacité de la complétion selon le nombre de valeurs manquantes : quelle proportion se voit attribuer une valeur de remplacement ? Puis dans un deuxième temps, nous pourrons évaluer la qualité de nos propositions d'indices de pertinence : dans quelle mesure les propositions correspondent-elles aux valeurs de la base complète ?

4.2.1 Qualité de la complétion selon l'incomplétude de la base

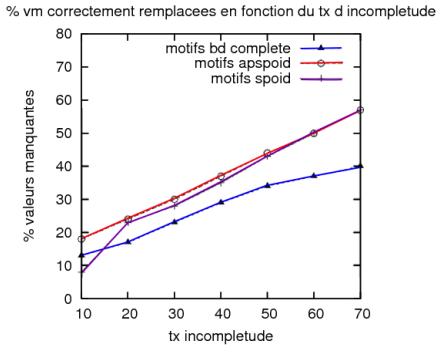
Nous commençons ici par évaluer la qualité des remplacements proposés par l'utilisation des motifs séquentiels selon le taux d'incomplétude de la base. Le principe de complétion est le suivant, si tous les motifs séquentiels permettent de conclure à la même valeur, celle-ci est choisie, sinon, les différentes propositions sont ordonnées en fonction de la fréquence des motifs.

Nous avons utilisés trois jeux de motifs séquentiels différents, pour chaque base incomplète. Puis, nous avons comparé les propositions de complétion aux valeurs correspondant aux valeurs manquantes dans la base de données complète.

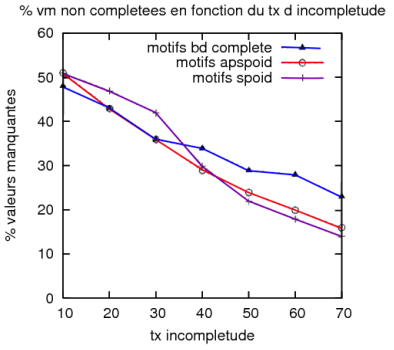
Les différents jeux de motifs séquentiels correspondent aux motifs extraits sur la base complète ainsi qu'aux motifs extraits sur la base à compléter, par SPoID et par ApSPoID.

La figure 4.3(a) donne le pourcentage de valeurs manquantes pour lesquelles la première proposition est la bonne valeur, i.e. celle présente dans la base complète. On constate que plus le taux d'incomplétude est élevé et plus les bonnes propositions sont nombreuses. On peut également noter que les motifs séquentiels extraits par les méthodes ApSPoID et SPoID, robustes aux valeurs manquantes, donnent globalement de meilleurs résultats que les motifs extraits sur la base complète. Cela peut s'expliquer par l'extraction d'items et de motifs fréquents, inexistants dans la base complètes mais tout de même extraits par les méthodes approximatives.

Grâce à la figure 4.3(b), on peut ainsi observer que certaines valeurs manquantes ne peuvent être complétées et que leur nombre est plus élevé lorsque les motifs séquentiels utilisés sont les motifs extraits sur la base complète. Enfin, la proportion de valeurs manquantes non complétées diminue à mesure que le taux d'incomplétude augmente, car le nombre de possibilités faisant coïncider une séquence incomplète avec des séquences fréquentes augmente.



(a)



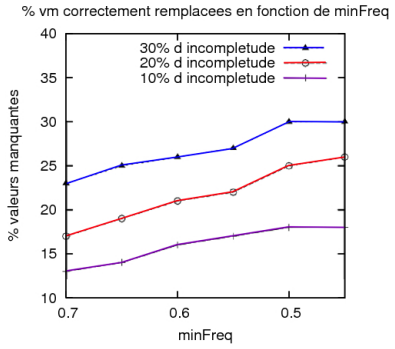
(b)

FIG. 4.3 – (a) : Pourcentage de valeurs manquantes correctement complétées en fonction du taux d'incomplétude de la base ; (b) : Pourcentage de valeurs manquantes non remplacées en fonction du taux d'incomplétude de la base.

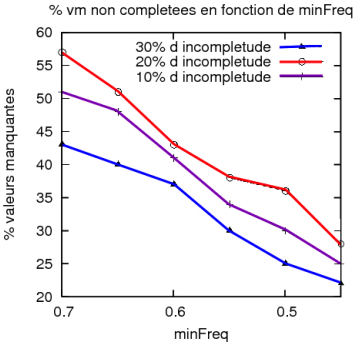
4.2.2 Qualité de la complétion selon la fréquence des motifs

La seconde partie de nos tests consiste à analyser la qualité de la complétion selon la fréquence minimum des motifs séquentiels utilisés. Ces motifs sont ceux extraits sur la base incomplète par la méthode SPoID à la représentativité optimale.

La figure 4.4(a) indique que le nombre de valeurs manquantes correctement complétées augmente à mesure que la fréquence des motifs utilisés pour la complétion diminue. En effet, la valeur de $minFreq$ diminuant, le nombre d'items fréquents augmente. Les motifs séquentiels extraits couvrent donc une plus grande partie de la base de données à compléter.



(a)



(b)

FIG. 4.4 – (a) : Pourcentage de valeurs manquantes correctement complétées en fonction de la fréquence minimum utilisée pour l'extraction des motifs ; (b) : Pourcentage de valeurs manquantes non remplacées en fonction de la fréquence minimum utilisée pour l'extraction des motifs.

Par ailleurs, on constate sur la figure 4.4(b) qu'à mesure que la fréquence diminue, le nombre de valeurs manquantes pour lesquelles aucune valeur de remplacement n'est trouvée diminue également. Là encore, plus de motifs séquentiels étant extraits, plus coïncident avec les séquences incomplètes de la

base à compléter.

Enfin, la figure 4.5 confirme ces résultats en montrant que le nombre moyen de propositions de remplacement pour chaque valeur manquante augmente alors que la fréquence minimum d'extraction des motifs séquentiels utilisés diminue.

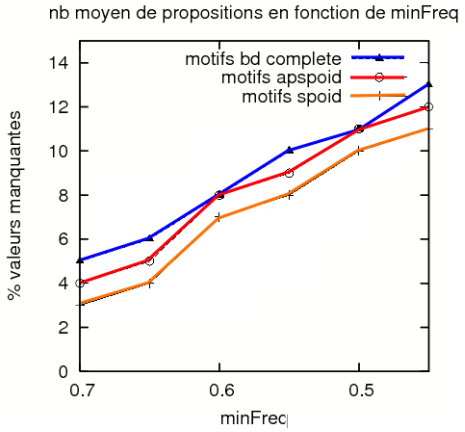


FIG. 4.5 – Nombre moyen de propositions par valeur manquante en fonction de la fréquence minimum utilisée pour l'extraction des motifs.

4.2.3 Conclusion

Ces expérimentations montrent que les résultats d'une complétion basée sur la similarité entre séquence incomplète et motifs séquentiels dépend à la fois du taux d'incomplétude de la base à compléter, ainsi que des motifs utilisés pour cette complétion. De plus, on ne peut pas considérer ces résultats comme entièrement satisfaisants.

En effet, dans le meilleur cas, il est possible de compléter correctement, sans intervention humaine, environ 50% des valeurs manquantes d'un base incomplète. Or, ces performances sont nettement inférieures à celles obtenues lors d'une complétion basée sur les règles d'association utilisant la confiance comme indice d'ordonnancement des propositions.

Il apparaît donc nécessaire d'utiliser pour la complétion des séquences extraites selon un autre critère que la fréquence. L'utilisation d'un autre indice permettrait en effet la sélection d'autres motifs séquentiels, qui contiendraient d'autres items. Ces nouvelles valeurs possibles permettraient alors d'améliorer les propositions de remplacement.

Discussion

La découverte de motifs séquentiels est une méthode de fouille de données intéressante lorsqu'il s'agit d'extraire des connaissances dans une base de données historisée, telle que des relevés de processus industriel ou de fonctionnement de machines. Or, dans ce type de bases de séquences, la présence de valeurs manquantes est inévitable.

Dans cette partie, nous avons présenté deux techniques destinées à extraire des motifs séquentiels dans des bases de données de séquences incomplètes. Ces approches, basées sur une redéfinition de la notion de fréquence, permettent de traiter les valeurs manquantes, distribuées au hasard, directement pendant la fouille plutôt que de supprimer les enregistrements incomplets lors d'un pré-traitement, comme cela était le cas avec les algorithmes existants.

La première méthode proposée, SPoID, basée sur la désactivation partielle et temporaire d'une partie de la base, est robuste jusqu'à un taux d'incomplétude de 40%. La seconde méthode, ApSPoID, utilise un système d'estimation multivaluée des valeurs manquantes. Les expérimentations ont montré que cette approche, selon la méthode d'estimation utilisée, conduit à des résultats cohérents pour un taux d'incomplétude de l'ordre de 40 à 50%. Ces résultats pourraient être améliorés en utilisant le principe de l'algorithme Expectation-Maximization : les résultats obtenus et les calculs de fréquence seraient utilisés pour réestimer les valeurs possibles des valeurs manquantes à chaque itération, jusqu'à convergence du modèle. De plus, l'utilisation d'une double distribution possibilité/nécessité permettrait le calcul de deux bornes de la fréquence des motifs séquentiels extraits, fournissant ainsi une liste optimiste de schémas et une seconde plus pessimiste.

Par ailleurs, une combinaison des deux techniques ApSPoID et SPoID, pourrait s'avérer être une solution optimale. Ainsi, lors du parcours de la base afin de déterminer les items fréquents, le taux d'incomplétude est également calculé, ainsi que la distribution des valeurs manquantes dans la base. Si les valeurs manquent au hasard, l'algorithme le plus approprié, SPoID ou ApSPoID, selon le taux d'incomplétude, est utilisé pour extraire les motifs séquentiels. Ces méthodes devront aussi être étendues afin de pouvoir prendre en compte d'autres types de valeurs manquantes (non distribuées au hasard, par exemple), après avoir détecté les différents types d'informations incomplètes présentes dans la base.

Une autre amélioration de ce travail consiste en la mise en place de mesures permettant de trier les motifs séquentiels en fonction du taux d'incomplétude des séquences de données qui permettent de les trouver. Une fois ces extensions mises en œuvre, des expérimentations sur données réelles, ainsi qu'avec différentes distributions de données, complètes ou incomplètes, devront être réalisées.

En ce qui concerne le processus de complétion des valeurs manquantes que nous avons mis en œuvre, les résultats de nos expérimentations montrent qu'une complétion basée sur la similarité entre séquence incomplète et motifs séquentiels, telle que nous l'avons envisagée, n'est pas entièrement satisfaisante.

En effet, dans le meilleur cas, il est possible de compléter correctement, sans intervention humaine, environ 50% des valeurs manquantes d'un base incomplète. Or, ces performances sont nettement inférieures à celles obtenues lors d'une complétion basée sur les règles d'association utilisant la confiance

comme indice d'ordonnancement des propositions.

Il serait donc intéressant d'étudier la qualité d'une complétion basée non plus sur des motifs séquentiels sélectionnés par rapport à un seuil de fréquence d'apparition minimum, mais sur des séquences fréquentes à forte corrélation entre les items et itemsets qui la composent. Par ailleurs, les résultats présentés ici sont obtenus en utilisant une correspondance entre motifs séquentiels et séquences incomplètes. Ils pourraient sans doute être améliorés en autorisant des correspondances partielles, ainsi qu'une mesure de similarité entre le motif et la séquence à compléter.

L'utilisation d'une telle mesure, ou d'une combinaison de plusieurs mesures, pourrait également permettre l'amélioration des performances algorithmiques de la complétion. Pour cela, on peut par exemple envisager un rapprochement des séquences incomplètes et des séquences fréquentes grâce aux principes développés par l'algorithme ApproxMAP [KPWD02], utilisant des techniques de clustering.

Enfin, les résultats d'une telle complétion devraient également pouvoir être améliorés en prenant en compte les apports des extensions proposées pour les motifs séquentiels, telle que la gestion de certaines paramètres temporels ou la prise en compte de valeurs numériques.

Bilan, perspectives et conclusion

Prévoir consiste à projeter dans l'avenir ce qu'on a perçu dans le passé.
Henri Bergson (1859-1941) — *L'évolution créatrice*

La masse d'informations disponibles, collectées et stockées a augmenté de façon exponentielle au cours des vingt dernières années, rendant impossibles l'analyse, la description ou l'extraction manuelle de ces connaissances potentielles. Afin de remédier à ce problème, de nombreuses techniques ont été proposées, aussi bien en apprentissage automatique qu'en statistique, permettant la découverte de schémas et modèles de ces données. Ces outils, regroupés sous le terme de fouille de données, fournissent différents types d'informations et de connaissances.

Dans le présent manuscrit, nous nous sommes intéressés à la recherche de corrélations fréquentes dans des bases de données temporelles, grâce à l'extraction de motifs séquentiels. Cette technique développée il y a une quinzaine d'années dans le but d'analyser le comportement de clients supermarché est aujourd'hui appliquée dans de nombreux domaines. En effet, les données biologiques, médicales, textuelles ou industrielles comportent très souvent une notion d'ordre, assimilable à une chronologie temporelle.

Cependant, les données de ces nouveaux champs d'application comportent de nouvelles caractéristiques. En particulier, ce type de bases de données contient de nombreux attributs numériques, alors que la technique originale d'extraction de motifs séquentiels visait à exploiter des données symboliques, de type binaire. Par ailleurs, ces nouvelles bases de données comportent un certain nombre d'imperfections dont la présence de nombreux champs non renseignés, les valeurs manquantes.

Notre travail a donc eu pour objectif de faciliter le traitement de ces données, en proposant dans un premier temps un cadre général pour l'extraction de motifs séquentiels dans des données numériques puis des méthodes de gestion des valeurs manquantes lors de la découverte de motifs séquentiels.

Toutefois, si le travail que nous avons réalisé permet de répondre à ces problématiques et offre à l'utilisateur la possibilité de traiter de nouveaux types de données de façon plus précise, certains problèmes demeurent non résolus. Nous présentons donc, pour conclure ce manuscrit, un bilan du travail réalisé ainsi qu'un certain nombre de perspectives issues plus ou moins directement des propositions que nous avons formulées.

La première section dresse un bilan de nos propositions ainsi que de leurs améliorations éventuelles. La section 2 développe ensuite quelques pistes pour la définition de mesures d'intérêt pour les motifs séquentiels. Enfin, la section 3 s'intéresse à de travaux futurs sur l'utilisation de techniques permettant de guider la fouille, pour en améliorer l'efficacité et faciliter parallèlement l'interprétation des motifs extraits.

1 Travail réalisé

Dans le cadre de cette thèse, nous nous sommes intéressés à la gestion des données numériques puis incomplètes lors de l'extraction de motifs séquentiels. Nous dressons dans cette section un bilan de notre travail, ainsi que des perspectives résultant directement de celui-ci.

1.1 Données numériques

Dans la première partie de ce manuscrit, nous avons mis en œuvre un cadre général pour l'extraction de motifs séquentiels en présence de données numériques quantitatives, basé sur la théorie des sous-ensembles flous. Nous avons montré que notre proposition permet de choisir entre trois niveaux de fuzzification et ainsi entre trois niveaux d'information : fréquence de certains attributs quantitatifs, fréquences des séquences composées d'items flous suffisamment significatifs, et enfin, fréquence pondérée par la pertinence des attributs quantitatifs composant les séquences. Ce cadre général permet également de considérer les attributs quantitatifs de façon binaire et d'extraire ainsi des motifs séquentiels symboliques. Enfin, avec un certain paramétrage, il permet également la gestion d'intervalles discrets.

Nos expérimentations ont montré que l'utilisation de ces différentes techniques pour l'extraction de motifs séquentiels flous permet effectivement la découverte de différentes connaissances. Toutefois, les motifs découverts et leur qualité dépendent directement du partitionnement réalisé sur les attributs quantitatifs. De plus, selon le niveau de fuzzification utilisé, les motifs séquentiels peuvent être très nombreux.

Il pourrait donc être intéressant d'étendre notre cadre en intégrant une première étape, basée sur une approche de clustering flou, à l'issue de laquelle on dispose d'une partition optimale des attributs numériques. La fouille serait ensuite réalisée selon le niveau de fuzzification choisi. Par ailleurs, une telle approche permettrait également, en choisissant un algorithme de clustering adéquat, de prendre en compte les données incomplètes et d'estimer les degrés d'appartenance des attributs non-renseignés.

1.2 Contraintes de temps étendues

Nous avons ensuite développé dans la deuxième partie notre définition de contraintes temporelles relâchées pour l'extraction de motifs séquentiels sur données symboliques. Ces travaux ont pour but d'autoriser l'utilisateur à spécifier des contraintes temporelles approximatives. La mise en œuvre de telles contraintes permet également de calculer un indice de précision temporelle, qui peut être utile lors de l'analyse des motifs extraits.

Toutefois, on peut envisager d'utiliser ces contraintes étendues afin, par exemple, de détailler l'information portée par chaque motif séquentiel. Ceux-ci ne seraient plus seulement accompagnés de la mesure de précision temporelle, mais également décrit par le degré "typique" de respect de $mingap$ et $maxgap$ entre deux itemsets des séquences de données supportant ce motif.

L'analyse des schémas extraits pourraient alors se faire à deux niveaux : un premier tri pourra être utilisé par rapport au critère de précision temporelle ; dans un deuxième temps, une analyse plus précise permettra d'analyser plus en détail les durées de transition entre les itemsets de séquences fréquentes.

1.3 Séquences de données incomplètes

La dernière partie du travail que nous présentons nous a permis de développer deux approches pour l'extraction de motifs séquentiels sur des données symboliques incomplètes au hasard. La première approche utilise l'information partielle disponible dans les enregistrements incomplets, les ignorant quand l'information recherchée n'est pas complète, tandis que la seconde utilise une estimation multivaluée

des valeurs manquantes. Les résultats obtenus sur des bases de données synthétiques ont montré que chacune des deux approches peut fournir une bonne approximation des motifs séquentiels extraits sur la base complète, selon les paramètres spécifiés en entrée des algorithmes ainsi que de la distribution et surtout de la proportion des valeurs manquantes.

Il s'agit donc désormais de définir une approche hybride, combinaison des deux techniques, qui permettrait d'améliorer les résultats en déterminant automatiquement les valeurs optimales des paramètres (représentativité, distribution de valeurs) ainsi que la méthode la plus appropriée en fonction du taux d'incomplétude de la base. Cette approche pourrait ensuite être étendue afin de prendre en compte d'autres types de valeurs manquantes, non distribuées au hasard, par exemple.

L'utilisation de l'algorithme SPoID, ignorant partiellement les séquences incomplètes, inclut le calcul de la représentativité des motifs séquentiels. Celle-ci peut être vue comme une première mesure de qualité des motifs extraits, puisqu'elle indique, pour chaque motif séquentiel, combien de séquences de données complètes sont utilisées pour le calcul de sa fréquence.
Au contraire, ApSPoID n'inclut pas ce calcul puisque toutes les séquences de données, mêmes incomplètes, sont utilisées pour le calcul de la fréquence de tous les motifs séquentiels. On peut alors envisager d'inclure la représentativité lors de l'exécution d'ApSPoID, afin de fournir un indice sur la proportion de séquences incomplètes permettant de générer chaque fréquent.

Par ailleurs, une autre mesure pourrait être mise en place afin de rendre compte du taux moyen d'incomplétude des séquences de données utilisées pour calculer la fréquence des motifs séquentiels par l'algorithme ApSPoID. La qualité des motifs séquentiels pourra ainsi être partiellement évaluée en considérant les taux d'incomplétude des séquences de données support des motifs et de la base complète pour chaque motif.

Dans un second temps, nous avons étudié la possibilité d'utiliser les motifs séquentiels extraits afin de proposer des valeurs de remplacement pour les valeurs manquantes. Les premiers résultats expérimentaux ne sont pas concluants, car trop de propositions de remplacement sont formulées pour chaque valeur manquante. De plus, seuls les items considérés fréquents sont des valeurs éventuelles.

Il nous semble donc nécessaire d'extraire d'autres types de séquences afin de pouvoir continuer à utiliser la notion de temporalité mais de baser les propositions de complétion sur d'autres propriétés que la fréquence. Ces propriétés pourraient être, par exemple, une forte corrélation entre les itemsets d'une séquence ou la mise en évidence d'une forte implication des itemsets successifs d'une séquence, les premiers impliquant les suivants et ainsi de suite. Les propriétés de ces séquences seraient alors caractérisées par des indices de pertinence permettant de sélectionner les séquences utilisées lors du processus de complétion des valeurs manquantes.

1.4 Synthèse

Le travail que nous avons réalisé facilite donc le traitement des données numériques, incomplètes ou la spécification de contraintes temporelles, en offrant à l'utilisateur la possibilité de traiter de nouveaux types de données, de façon plus précise. Toutefois, certains problèmes demeurent non résolus, notamment, concernant l'analyse des motifs séquentiels extraits. En effet, l'extraction de séquences fréquentes aboutit souvent à la découverte de très nombreux motifs qui doivent ensuite être analysés par l'utilisateur, expert des données, avant de devenir des connaissances exploitables. Pourtant, il n'existe que peu de travaux proposant de simplifier cette dernière étape d'analyse.

Ainsi, une partie de notre travail a consisté en la spécialisation de la notion de fréquence selon le type de données traitées : nous avons défini différents niveaux pour l'information que l'on peut obtenir d'une

base de données numériques. Nous avons également introduit la notion de précision temporelle pour les séquences extraites sous contraintes temporelles ainsi que la représentativité des motifs séquentiels extraits sur des bases de données incomplètes. Ces premières mesures peuvent permettre de trier les motifs séquentiels afin de répondre à certains besoins, mais de nombreuses problématiques ne peuvent être résolues par l'utilisation de ces mesures.

Il paraît ainsi nécessaire de proposer des indices de pertinence ou mesures d'intérêt afin de trier ou de réduire le nombre de motifs séquentiels extraits et des mesures de la qualité des motifs, d'autre part, afin de rendre compte des différentes imperfections contenues dans les données.

Une partie de nos travaux avait pour but de formuler des propositions de remplacement pour les valeurs manquantes, grâce aux motifs séquentiels extraits et à leur similarité avec les séquences incomplètes. Nous avons donc étudié les critères existants qui permettraient d'ordonner les motifs séquentiels, selon entre autre, les corrélations et lien de causalité existant entre les itemsets des séquences fréquentes. Toutefois, nous n'avons trouvé que peu de travaux utilisant une mesure de confiance, rarement formalisée et le plus souvent utilisée à des fins de classification, en indiquant la probabilité pour une séquence de données d'appartenir à une classe si elle inclut un motif séquentiel particulier.

Nous présentons maintenant les perspectives de notre travail relatives à la notion de causalité dans les séquences fréquentes.

2 Temporalité, séquentialité, causalité

Contrairement aux règles d'association, les motifs séquentiels ne comportent pas de relation intrinsèque de cause à conséquence. En effet, les règles d'association sont extraites parmi les itemsets fréquents, en décomposant ceux-ci en deux sous-ensembles disjoints et en calculant les probabilités conditionnelles qui les lient. Il existe donc un lien de causalité entre les itemsets antécédent et conséquent d'une règle d'association. Plus précisément, la confiance indique à quel point le conséquent résulte de l'itemset antécédent.

Or, il n'existe pas de liens évidents entre les évènements composant une séquence fréquente ou un motif séquentiel. En effet, la recherche de tels motifs se base sur la fréquence d'apparition des séquences et sur leur maximalité au sens de l'inclusion. Mais aucun lien de causalité n'existe dans cette définition. En effet, prenons l'exemple de comportement suivant : "50% des clients achètent un lecteur DVD, puis du lait, puis des DVD". L'expert des données pourra conclure que l'achat des DVD résulte de l'achat du lecteur de DVD et non de celui du lait, mais la seule indication portée par ce motif séquentiel est que les achats de lecteur DVD, de lait et de DVD sont souvent réalisés dans cet ordre.

2.1 Règles d'association séquentielles

La première approche consiste à reproduire le schéma générique de découverte de règles d'association : tout d'abord les motifs fréquents sont extraits puis on isole ceux dont la "confiance" est plus élevée que le seuil minimum.

Afin de construire les règles à tester, chaque motif séquentiel S, composé de p itemsets, avec $p \geq 2$, est partitionné en deux sous-séquences distinctes S_1 et S_2, telles que :

- la sous-séquence S_1 précède S_2, i.e., tous les itemsets de S_1 précèdent tous les itemsets de S_2 dans S,
- les itemsets de S_1 et S_2 sont exactement les itemsets contenus dans S.

Exemple 4.1. Considérons le motif séquentiel <(a b) (c) (d e)>, il peut être divisé en deux sous-séquences telles que définies précédemment de 5 façons différentes :

- S_1=<(a b)> et S_2 = <(c)>, S_1=<(a b)> et S_2 = <(d e)>, S_1=<(a b)> et S_2 = <(c) (d e)>,
- S_1=<(c)> et S_2 = <(d e)>,
- S_1=<(a b) (c)> et S_2 = <(d e)>,

Des contraintes peuvent alors être envisagées, comme par exemple que tout le motif séquentiel soit représenté dans S_1 et S_2, dans l'exemple précédent, seules les combinaisons (S_1=<(a b)>, S_2 = <(c) (d e)>) et (S_1=<(a b) (c)>, S_2 = <(d e)>) pourraient alors être conservées.

Cette contrainte, qui correspond au traitement réalisé sur les itemsets pour la découverte de règles d'association, diminue nettement la complexité du problème. En effet, sans contrainte autre que les inclusions respectives de S_1 et S_2 dans S, il est possible de générer $\sum_{i=0}^{p-2} \binom{p}{i}$ règles à partir d'un seul motif composé de p itemsets.

Ce nombre peut être réduit en ne considérant que des itemsets consécutifs de chaque côté de la règle, par exemple, le premier et le deuxième itemset impliquant le troisième, ou bien le deuxième et le troisième impliquant le quatrième. Dans ce cas, seulement $\sum_{i=1}^{p-1}(\sum_{j=0}^{i} j)$ seront générées. Enfin, si tout le motif doit se retrouver dans S_1 et S_2, seules $p-1$ règles seront générées pour un motif séquentiels composé de p itemsets.

Les règles générées seront alors de la forme $S_1 \twoheadrightarrow S_2$, ce qui signifie que la séquence S_1 précède et implique S_2 avec une confiance séquentielle supérieure à $minConfSeq$, seuil spécifié par l'utilisateur.

Exemple 4.2. Prenons le motif séquentiel $S =< s_1 s_2 s_3 s_4 >$ où les s_i sont des itemsets, les règles générées et testées selon la dernière approche seront : $< s_1 > \twoheadrightarrow < s_2 s_3 s_4 >$, $< s_1 s_2 > \twoheadrightarrow < s_3 s_4 >$ et $< s_1 s_2 s_3 > \twoheadrightarrow < s_4 >$.
Auxquelles s'ajoutent lorsque l'on impose des itemsets consécutifs de chaque côté de la règle :

- $< s_1 > \twoheadrightarrow < s_2 s_3 >$, $< s_1 > \twoheadrightarrow < s_2 s_4 >$, $< s_1 > \twoheadrightarrow < s_2 >$,
- $< s_1 s_2 > \twoheadrightarrow < s_3 >$, $< s_1 s_3 > \twoheadrightarrow < s_4 >$, $< s_2 s_3 > \twoheadrightarrow < s_4 >$,
- $< s_2 > \twoheadrightarrow < s_3 s_4 >$, $< s_2 > \twoheadrightarrow < s_3 >$ et $< s_3 > \twoheadrightarrow < s_4 >$.

Enfin, la génération des règles sans aucune contrainte aboutirait aux règles précédentes, auxquelles s'ajouteraient : $< s_1 > \twoheadrightarrow < s_3 s_4 >$, $< s_1 > \twoheadrightarrow < s_3 >$, $< s_1 > \twoheadrightarrow < s_4 >$, $< s_1 s_2 > \twoheadrightarrow < s_4 >$ et $< s_2 > \twoheadrightarrow < s_4 >$.

Une fois ces règles séquentielles trouvées, il est alors possible de les analyser selon les nombreuses mesures déjà existantes pour les règles d'association.

Considérant la complexité algorithmique d'une telle approche, il paraît essentiel de définir un algorithme efficace de post-traitement des motifs séquentiels découverts ou bien d'intégrer cette notion de causalité lors de la recherche des motifs, afin de n'extraire que des séquences à forte causalité interne.

2.2 Séquences à forte causalité : motifs conséquentiels

Cette deuxième approche repose sur un filtrage des motifs séquentiels lors de la fouille, et non plus lors de la phase de post-traitement. Il s'agit en effet d'éliminer au fur et à mesure les séquences dont le début n'implique pas la fin de la séquence. Le but de la fouille n'est plus simplement d'extraire des séquences fréquentes, mais des séquences dont les itemsets successifs sont liés par une implication, dont la confiance est calculée en même temps que la fréquence et est utilisée comme condition d'élagage.
Une séquence causale $< s_1 s_2 ... s_n >$ aurait alors pour signification que

$$\forall i \in [1, n-1], \forall j \in [i+1, n], < s_i > \twoheadrightarrow < s_j > \text{ et } \frac{freq(< s_i s_j >)}{freq(s_i)} \geq minConfSeq$$

Se pose alors la question du calcul de la "confiance séquentielle" de ces séquences. En effet, chaque couple d'itemsets peut satisfaire cette notion de causalité, il faut ensuite réussir à exprimer cette mesure et sa valeur au niveau de la séquence complète. On ne parlera alors plus de motifs séquentiels mais de *motifs conséquentiels*.

L'avantage de cette approche par rapport aux règles d'association séquentielles présentées précédemment, est de réduire fortement le nombre de séquences candidates et donc le nombre de séquences extraites. Toutefois, afin d'assurer la complétude d'une telle approche, il sera nécessaire de définir la notion de causalité et de confiance, et de vérifier que les critères d'élagage respectent la propriété d'antimonotonie, qui garantit la complétude des résultats, tel que le permet le critère de fréquence lors de l'extraction de motifs séquentiels.

Ces différents liens de causalité qui pourront être isolés au sein des séquences de données permettront de mettre en évidence de nouvelles connaissances. Ainsi, on ne se contentera plus de savoir que "60% des clients achètent une télévision et un lecteur de DVD puis reviennent acheter des DVD", mais on cherchera des connaissances de la forme "90% des clients qui achètent une télévision et un lecteur de DVD reviennent plus tard pour acheter des DVD".

Dans de nombreux domaines, les informations sont stockées sous une forme assimilable à des séquences de données. Or, dans certains cas, les connaissances intéressantes n'apparaissent pas dans ce qui est fréquent, mais plutôt dans ce qui est rare et sûr ; par exemple, certaines mutations dans les séquences d'ADN sont des évènements rares, mais les observations ont montrées qu'elles conduisent nécessairement à des dysfonctionnements.

Il peut également être intéressant d'analyser les conséquences à moyen et long terme de traitements de patients souffrant d'affections de longue durée. Les effets secondaires, qui sont peu fréquents, devraient pouvoir être mis en évidence par la recherche de liens de causalité.

Dès lors, de nombreuses autres applications sont envisageables, dans des domaines variés, telles que la recherche de causes de défaillances dans des systèmes de production, la détection d'attaques de serveur internet ou encore la mise en évidence de symptômes de maladie.

3 Schémas nouveaux, valides et potentiellement utiles

Comme nous l'avons montré dans la deuxième partie de ce manuscrit, une des possibilités offertes à l'utilisateur consiste à appliquer un certain nombre de contraintes lors de l'extraction des motifs séquentiels. Une autre possibilité consiste à introduire différentes mesures afin de trier les schémas extraits. Enfin, l'inclusion des connaissances du domaine, formalisées par exemple sous la forme d'une ontologie, pourrait permettre de filtrer les motifs à extraire ou de guider la fouille de données.

En effet, la masse de données aujourd'hui disponible dans certains domaines tels que la biologie ou la santé, requiert la spécification de bases de données prenant en compte, dès leur conception, les connaissances qui devront en être extraites, ainsi que les techniques de fouille de données qui seront utilisées. Ainsi, l'intégration des connaissances des experts des domaines applicatifs permettrait d'améliorer simultanément la structure des bases de données destinées à l'extraction de connaissances, ainsi que l'interprétation et la compréhension des schémas extraits.

De plus, la connaissance d'un expert du domaine lors d'un processus d'extraction de connaissances peut également être utilisée afin de mettre en évidence les schémas "nouveaux", par élimination ou complétion des connaissances déjà acquises.

Par ailleurs, les ontologies peuvent également être vues comme un moyen de simplifier certains problèmes liés à l'hétérogénéité des données. En effet, il peut être envisagé de stocker dans une base de connaissances, dérivées d'une ontologie, les conversions réalisées entre des données numériques et leur valeur symbolique, que ce soit lors d'un précédent processus de fouille ou d'après les connaissances d'un expert. De même, les raisonnements sur des variables linguistiques permis par la logique floue, couplés à une ontologie de domaine pourraient également être exploité lors de la recherche de séquences spécifiques d'une requête, de contraintes ou d'une problématique donnée.

Ainsi, la combinaison de la connaissance experte, afin d'extraire des nouveautés, de mesures de qualité objectives ou résultant de l'expression du savoir d'un expert, afin d'extraire des informations valides, ainsi que d'indices de pertinence pour isoler des connaissances utiles, permettra sans doute l'extraction plus efficace de schémas plus spécifiques et plus proches des besoins des utilisateurs des données.

4 Conclusion

Dans ce mémoire, nous avons développé plusieurs extensions des algorithmes d'extraction de motifs séquentiels afin de fouiller des sources de données hétérogènes, sous contraintes ou encore incomplètes.

Pour cela, nous avons mis en œuvre un cadre pour l'extraction de motifs séquentiels en présence de données numériques quantitatives. Basé sur la théorie des sous-ensembles flous, ce cadre permet d'extraire des motifs séquentiels selon trois niveaux d'information, plus ou moins précis, selon les besoins de l'utilisateur.

Dans un second temps, nous avons défini une extension des contraintes temporelles afin d'autoriser à l'utilisateur la spécification de contraintes temporelles approximatives, limitant ainsi le nombre d'essais nécessaires pour obtenir des motifs séquentiels intéressants sous contraintes. Ces contraintes souples nous permettent également de calculer un indice de précision temporelle, utilisé pour trier les résultats obtenus.

Enfin, la présence de nombreux enregistrements incomplets dans les bases de données industrielles nous a conduits à développer deux algorithmes pour l'extraction de motifs séquentiels sur des données symboliques incomplètes au hasard, l'un se basant sur la distribution des données renseignées dans la base ainsi que sur une estimation multivaluée des valeurs manquantes lors de l'extraction des motifs, le second utilisant l'information partielle, mais sûre, contenue dans les enregistrements incomplets pour extraire les séquences fréquentes.

Ainsi, nous avons proposé des améliorations des algorithmes permettant l'extraction de séquences fréquentes afin de pouvoir prendre en compte des données hétérogènes, incomplètes, incertaines, ou mal connues de leur utilisateur, tout en minimisant les pertes éventuelles d'information.

Désormais, un travail important devrait porter sur la conception d'outils d'analyse et de l'évaluation de la qualité des schémas extraits. En effet, l'extraction de séquences fréquentes aboutit souvent à la découverte de très nombreux motifs qui doivent ensuite être analysés par l'utilisateur expert des données afin de devenir des connaissances exploitables. Or, l'utilisation d'indices de pertinence ou mesures d'intérêt pourra permettre de trier ou de réduire le nombre de motifs extraits. Des mesures de qualité, d'autre part, pourront rendre compte des différentes imperfections contenues dans les bases de données exploitées.

Bibliographie

[AC98] W.-H. Au et K. Chan : An effective algorithm for discovering fuzzy rules in relational databases. *In IEEE World Congress on Computational Intelligence*, pages 1314–1319, 1998.

[AGYF02] J. Ayres, J. Gehrke, T. Yiu et J. Flannick : Sequential PAttern Mining using bitmaps. *In 8th ACM International Conference on Knowledge Discovery and Data Mining (KDD'02)*, 2002.

[AIS93] R. Agrawal, T. Imielinski et A. N. Swami : Mining Association Rules between Sets of Items in Large Databases. *In the ACM SIGMOD International Conference on Management of Data*, pages 207–216, 1993.

[AL99] Y. Aumann et Y. Lindell : A statisitical theory for quantitative association rules. *In ACM International Conference on Knowledge Discovery and Data Mining (KDD'99)*, pages 261–270, 1999.

[ALB03] H. Albert-Lorincz et J.-F. Boulicaut : Mining Frequent Sequential Patterns under Regular Expressions : a Highly Adaptative Strategy for Pushing Constraints. *In 3rd SIAM Int. Conf. on Data Mining (SIAM DM'03)*, pages 316–320, 2003.

[AMS+96] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen et A. I. Verkamo : Fast Discovery of Association Rules. *In Advances in Knowledge Discovery and Data Mining*, pages 307–328, 1996.

[AO04] C. Antunes et A. L. Oliveira : Constraint relaxations for discovering unknown sequential patterns. *In 3rd International Workshop Knowledge Discovery in Inductive Databases (KDID'04)*, Lecture Notes in Computer Science, pages 11–32, 2004.

[AS94] R. Agrawal et R. Srikant : Fast Algorithms for Mining Association Rules. *In 20th International Conference on Very Large Data Bases, (VLDB'94)*, pages 487–499, 1994.

[AS95] R. Agrawal et R. Srikant : Mining Sequential Patterns. *In the 11th IEEE International Conference on Data Engineering*, pages 3–14, 1995.

[BA96] R. J. Brachman et T. Anand : The process of knowledge discovery in database. *Advances in knowledge discovery and data mining*, pages 37 – 57, 1996.

[BB05] S. Bistarelli et F. Bonchi : Interestingness is not a dichotomy : Introducing softness in constrained pattern mining. *In 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'05)*, volume 3721 de *Lecture Notes in Computer Science*, 2005.

[Bez81] J. C. Bezdeck : Pattern recognition with fuzzy objective function algorithms, 1981.

[BFOS84] L. Breiman, J. H. Friedman, R. A. Olshen et C. J. Stone : Classification and Regression Trees, 1984.

[BGMP03a] F. Bonchi, F. Giannotti, A. Mazzanti et D. Pedreschi : Adaptive constraint pushing in frequent pattern mining. *In 7th European Conference on Principles and Practice of*

Knowledge Discovery in Databases (PKDD'03), volume 2838 de *Lecture Notes in Computer Science*, 2003.

[BGMP03b] F. Bonchi, F. Giannotti, A. Mazzanti et D. Pedreschi : Pre-processing for constrained pattern mining. *In 11th Italian Symposium on Advanced Database Systems (SEBD'03)*, 2003.

[BH97] M. R. Berthold et K.-P. Huber : Tolerating Missing Values in a Fuzzy Environment. *In 7th IFSA World Congress, Prag*, volume 1, pages 359–362, 1997.

[BH98] M. R. Berthold et K.-P. Huber : Missing Values and Learning of Fuzzy Rules. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 1998.

[BHLK06] J. C. Bezdek, R. J. Hathaway, C. Leckie et R. Kotagiri : Approximate data mining in very large relational data. *In 17th Australasian Database Conference (ADC'06)*, pages 3–13, 2006.

[BJ00] J. Boulicaut et B. Jeudy : Using constraints during set mining : Should we prune or not ? *In Bases de Données Avancées (BDA'00)*, pages 221–237, 2000.

[BM99] B. Bouchon-Meunier : *La logique floue*. Presses Universitaires de France, 1999.

[CA97] K. Chan et W.-H. Au : Mining fuzzy association rules. *In 6th International Conference on Information and Knowledge Management (CIKM '97)*, pages 209–215, 1997.

[CCK04] M. De Cock, C. Cornelis et E. E. Kerre : A clear view on quality measures for fuzzy association rules. *In International Conference on Fuzzy Sets and Soft Computing in Economics and Finance (FSSCEF'04)*, pages 54–61, 2004.

[CGM07] T. Calders, B. Goethals et M. Mampaey : Mining itemsets in the presence of missing values. *In ACM Symposium on Applied Computing (SAC'07)*, 2007.

[CH02] R.-S. Chen et Y.-C. Hu : A Novel Method for Discovering Fuzzy Sequential Patterns Using the Simple Fuzzy Partition Method. *Journal of the American Society for Information Science*, 54(7):660–670, 2002.

[CH06] Y.-L. Chen et T. C.-K. Huang : A new approach for discovering fuzzy quantitative sequential patterns in sequence databases. *Fuzzy Sets and System*, 157:1641–1661, 2006.

[CMB02] M. Capelle, C. Masson et J.-F. Boulicaut : Mining Frequent Sequential Patterns under a Similarity Constraint. *In 3rd Int. Conf. on Intelligent Data Engineering and Automated Learning (IDEAL'02)*, pages 1–6, 2002.

[CTCH01] R.-S. Chen, G.-H. Tzeng, C.-C. Chen et Y.-C. Hu : Discovery of Fuzzy Sequential Patterns for Fuzzy Partitions in Quantitative Attributes. *In ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)*, pages 144–150, 2001.

[CWK00] G. Chen, Q. Wei et E. Kerre : Fuzzy data mining : Discovery of fuzzy generalized association rules. *Recent Research Issues on Management of Fuzziness in Databases*, 2000.

[CYH04] H. Cheng, X. Yan et J. Han : IncSpan : Incremental mining of sequential patterns in large database. *In ACM International Conference on Knowledge Discovery and Data Mining (KDD'04)*, 2004.

[DHP03] D. Dubois, E. Hüllermeier et H. Prade : A note on quality measures for fuzzy association rules. *In 10th International Fuzzy Systems Association World Congress (IFSA'03)*, volume 2715 de *Lecture Notes in Computer Science*, pages 677–648, 2003.

[DHP06] D. Dubois, E. Hüllermeier et H. Prade : A systematic approach of the assessment of fuzzy association rules. *Data Mining and Knowledge Discovery*, 13(2):167–192, 2006.

[DLR77] A. P. Dempster, N. M. Laid et D. B. Rubin : Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[DOQT01] E. Damiani, B. Oliboni, E. Quintarelli et L. Tanca : Modeling users' navigation history. *In Workshop on Intelligent Techniques for Web Personalisation, 7th International Joint Conference on Articial Intelligence (IJCAI'01)*, 2001.

[DP80] D. Dubois et H. Prade : *Fuzzy Sets and Systems - Theory and Applications*. Academic press, 1980.

[DSV02] M. Delgado, D. Sanchez et M.-A. Vila : Acquisition of fuzzy association rules from medical data. *Fuzzy Logic in Medicine, Studies in Fuzziness and Soft Computing Series*, 83:286–310, 2002.

[Dun73] J. C. Dunn : A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3:32–57, 1973.

[FKY96] J. H. Friedman, R. Kohavi et Y. Yun : Lazy Decision Trees. *In the 13th National Conference on Artificial Intelligence and the 8th Innovative Applications of Artificial Intelligence Conference*, pages 717–724, 1996.

[FPSS96a] U. M. Fayyad, G. Piatetsky-Shapiro et P. Smyth : From Data Mining to Knowledge Discovery : an Overview. *Advances in knowledge discovery and data mining*, pages 1–34, 1996.

[FPSS96b] U. M. Fayyad, G. Piatetsky-Shapiro et P. Smyth : Knowledge Discovery and Data Mining : Towards a unifying framework. *In Knowledge Discovery and Data Mining*, pages 82–88, 1996.

[FWS+98] A. Fu, M. Wong, S. Sze, W. Wong, et W. Yu : Finding Fuzzy Sets for the Mining of Fuzzy Association Rules for Numerical Attributes. *In the 1st International Symposium on Intelligent Data Engineering and Learning (IDEAL)*, pages 263–268, 1998.

[GJ94] Z. Ghahramani et M. I. Jordan : Learning from Incomplete Data. Rapport technique, MIT - Artificial Intelligence Laboratory, USA, 12 1994.

[GNP06] F. Giannotti, M. Nanni et D. Pedreschi : Efficient mining of temporally annotated sequences. *In 6th SIAM Int. Conf. on Data Mining (SDM'06)*, 2006.

[GNPP06] F. Giannotti, M. Nanni, D. Pedreschi et F. Pinelli : Mining sequences with temporal annotations. *In ACM symposium on Applied computing (SAC '06)*, pages 593–597, 2006.

[GRS02] M. Garofalakis, R. Rastogi et K. Shim : Mining Sequential Patterns with Regular Expression Constraints. *IEEE Transactions on Knowledge and Data Engineering*, 14(3):530–552, 2002.

[Gye00a] A. Gyenesei : A fuzzy approach for mining quantitative association rules. Rapport technique TUCS-TR-336, Turku Centre for Computer Science, 2000.

[Gye00b] A. Gyenesei : Fuzzy partitioning of quantitative attribute domains by a cluster goodness index. Rapport technique TUCS-TR-368, Turku Centre for Computer Science, 2000.

[Har75] J. Hartigan : *Clustering Algorithms*. John Wiley & Sons, Inc., 1975.

[HCTS03] Y.-C. Hu, R.-S. Chen, G.-H. Tzeng et J.-H. Shieh : A Fuzzy Data Mining Algorithm for Finding Sequential Patterns. *International Journal of Uncertainty Fuzziness Knowledge-Based Systems*, 11(2):173–193, 2003.

[HK00] J. Han et M. Kamber : *Data Mining : Concepts and Techniques*. Morgan Kaufmann Publishers, 2000.

[HK01] F. Höppner et F. Klawonn : A new approach to fuzzy partitioning. *In Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, volume 3, pages 1419–1424, 2001.

[HKCW00] T.-P. Hong, C.-S. Kuo, S.-Chai C. et S.-L. Wang : Mining fuzzy rules from quantitative data based on the Apriori Tid algorithm". *In ACM symposium on Applied computing (SAC'00)*, pages 534–536, 2000.

[HL01] N. J. Horton et R. S. Lipsitz : Multiple imputation in practice : Comparison of software packages for regression models with missing variables. *The American Statistician*, 55(3): 244–254, 2001.

[HLW01] T.P. Hong, K.Y. Lin et S.L. Wang : Mining Fuzzy Sequential Patterns from Multiple-Items Transactions. *In Proceedings of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, pages 1317–1321, 2001.

[HLW03] T.-P. Hong, K.-Y. Lin et S.-L. Wang : Fuzzy data mining for interesting generalized association rules. *Fuzzy Sets and Systems*, 138:255–269, 2003.

[HT03] Y.-C. Hu et G.-H. Tzeng : Elicitation of classification rules by fuzzy data mining. *Engineering Applications of Artificial Intelligence*, 8:709 – 716, 2003.

[HTC04] Y.-C. Hu, G.-H. Tzeng et C.-M. Chen : Deriving Two-Stage Learning Sequences from Knowledge in Fuzzy Sequential Pattern Mining. *Information Sciences*, 159:69–86, 2004.

[Hui00] M. Huisman : Poststratification to correct for nonresponse : Stratification of zip code areas. *In Computational Statistics (COMPSTAT'00)*, pages 325–330, 2000.

[ID01] S. P. Imberman et B. Domanski : Finding association rules from quantitative data using data booleanization. *In 7th Americas Conference on Information Systems*, 2001.

[JJL+05] S. Jami, T.Y. Jen, D. Laurent, G. Loizou et O. Sy : Extraction de règles d'association pour la prédiction de valeurs manquantes. *Revue Africaine de la Recherche en Informatique et Mathématique appliquée (Numéro spécial CARI'04)*, pages 103–124, 2005.

[JKK99] M. Joshi, G. Karypis et V. Kumar : A universal formulation of sequential patterns. Rapport technique 99-021, University of Minnesota, Computer Science and Engineering, 1999.

[Kac88] J. Kacprzyk : Fuzzy logic with linguistic quantifiers : A tool for better modeling of human evidence aggregation processes ? *Fuzzy Sets in Psychology*, pages 233—263, 1988.

[Kau73] A. Kaufmann : *Introduction à la théorie des sous-ensembles flous*, volume 1 : Eléments théoriques de base. Masson, 1973.

[KFW98] C. M. Kuok, A. W.-C. Fu et M. H. Wong : Mining Fuzzy Association Rules in Databases. *ACM SIGMOD Record*, 27(1):41–46, 1998.

[KLNS04] C. Kim, J.-H. Lim, R. Ng et K. Shim : SQUIRE : Sequential pattern mining with quantities. *In 20th International Conference on Data Engineering (ICDE'04)*, page 827, 2004.

[KPWD02] H. Kum, J. Pei, W. Wang et D. Duncan : ApproxMAP : Approximate mining of consensus sequential patterns. Rapport technique TR02-031, UNC-CH, 2002.

[Kry99] M. Kryszkiewicz : Association rules in incomplete databases. *In 3rd Pacific-Asia Conference on Methodologies for Knowledge Discovery and Data Mining (PAKDD'99)*, volume 1574 de *Lecture Notes in Computer Science*, pages 84–93. Springer, 1999.

[KYZ00] J. Kacprzyk, R.R. Yager et S. Zadrozny : A Fuzzy Logic Based Approach to Linguistic Summaries of Databases. *International Journal of Applied Mathematics and Computer Science*, 10:813–834, 2000.

[LAS97] B. Lent, R. Agrawal et R. Srikant : Discovering Trends in Text Databases. *In 3rd ACM International Conference on Knowledge Discovery and Data Mining (KDD'97)*, pages 227–230, 1997.

[Lau02] A. Laurent : *Bases de données multidimensionnelles floues et leur utilisation pour la fouille de données*. Thèse de doctorat, Université Paris 6, rapport LIP6 2002/22, 2002.

[LC92] K. J. Lynch et H. Chen : Knowledge discovery from historical data : An algorithmic approach. *In 25th Hawaii International Conference on System Sciences*, pages 70–79, 1992.

[LL99a] M. Levene et G. Loizou : Database design for incomplete relations. *ACM Transactions on Database Systems*, 24(1), 1999.

[LL99b] M. Levene et G. Loizou : *A Guided Tour of Relational Databases and Beyond*. Springer-Verlag, 1999.

[LL05] M.-Y. Lin et S.-Y. Lee : Fast discovery of sequential patterns through memory indexing and database partitioning. *Journal of Information Science and Engineering*, 21(1):109–128, 2005.

[LLW02] M.-Y. Lin, S.-Y. Lee et S.-S. Wang : DELISP : Efficient discovery of generalized sequential patterns by delimited pattern-growth technology. *In 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD '02)*, pages 198–209, 2002.

[LR87] R. J. A. Little et D. B. Rubin : *Statistical analysis with missing data*. John Wiley, 1987.

[LRBE03] M. Leleu, C. Rigotti, J.-F. Boulicaut et G. Euvrard : Constraint-Based Mining of Sequential Patterns over Datasets with Consecutive Repetitions. *In the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'03)*, pages 303–314, 2003.

[LWTB97] W. Z. Liu, A. P. White, S. G. Thompson et M. A. Bramer : Techniques for Dealing with Missing Values in Classification. *In the 2nd International Symposium on Advances in Intelligent Data Analysis, Reasoning about Data*, volume 1280 de *Lecture Notes in Computer Science*, 1997.

[Mas02] F. Masseglia : *Algorithmes et applications pour l'extraction de motifs séquentiels dans le domaine de la fouille de données : de l'incrémental au temps réel*. Thèse de doctorat, Université de Montpellier, 2002.

[MB06] I. Mitasiunaite et J.-F. Boulicaut : About softness for inductive querying on sequence databases. *In 7th International Baltic Conference on Databases and Information Systems (DBIS'06)*, pages 77–82, 2006.

[MBM99] C. Marsala et B. Bouchon-Meunier : An adaptable system to construct fuzzy decision trees. *In the North American Fuzzy Information Processing Society Conference (NAFIPS'99)*, pages 223 – 297, 1999.

[MCP98] F. Masseglia, F. Cathala et P. Poncelet : The PSP Approach for Mining Sequential Patterns. *In Principles of Data Mining and Knowledge Discovery*, pages 176–184, 1998.

[Mel] G. Melli : Synthetic Classification Data Set Generator.

[MI96] H. Mannila et T. Imielinski : A database perspective on knowledge discovery. *Communications of the ACM*, 39(11):58–64, 1996.

[MPC99] F. Masseglia, P. Poncelet et R. Cicchetti : An efficient algorithm for web usage mining. *Networking and Information Systems Journal*, 2(5-6), 1999.

[MPM02] S. Mitra, S. K. Pal et P. Mitra : Data mining in soft computing framework : A survey. *IEEE Transactions on Neural Networks*, 13(1):3–14, 2002.

[MPT03] F. Masseglia, P. Poncelet et M. Teisseire : Incremental Mining of Sequential Patterns in Large Databases. *Data and Knowledge Engineering*, 46(1):97–121, 01 2003.

[MPT04] F. Masseglia, P. Poncelet et M. Teisseire : Pre-processing time constraints for efficiently mining generalized sequential patterns. *In 11th International Symposium on Temporal Representation and Reasoning (TIME '04)*, pages 87–95, 2004.

[MR04] N. Meger et C. Rigotti : Constraint-based mining of episode rules and optimal window sizes. *In the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'04)*, pages 313–324, 2004.

[MT96] H. Mannila et H. Toivonen : Multiple uses of frequent sets and condensed representations (extended abstract). *In ACM International Conference Knowledge Discovery and Data Mining (KDD'96)*, pages 189–194, 1996.

[MTV97] H. Mannila, H. Toivonen et A. I. Verkamo : Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

[NC01] J. Nayak et D. Cook : Approximate association rule mining. *In Florida Artificial Intelligence Research Symposium*, 2001.

[NH94] R. Ng et J. Han : Efficient and Effective Clustering Methods for Spatial Data Mining. *In the 20 th Very Large Database Conference (VLDB'94)*, 1994.

[NL98] V. Ng et J. Lee : Quantitative association rules over incomplete data. *In IEEE International Conference*, pages 2821–2826, 1998.

[OPS04] S. Orlando, R. Perego et C. Silvestri : A new algorithm for gap constrained sequence mining. *In ACM Symposium on Applied computing (SAC'04)*, pages 540–547, 2004.

[OW03] C. Olaru et L. Wehenkel : A complete fuzzy decision tree technique. *Fuzzy Sets and Systems*, 138(2):221–254, 2003.

[Pea06] R. Pearson : The problem of disguised missing data. *ACM SIGKDD Explorations Newsletter*, 8(1):83–92, 2006.

[Ped98] W. Pedrycz : Fuzzy set technology in knowledge discovery. *Fuzzy Sets and Systems*, 98:279 – 290, 1998.

[PH00] J. Pei et J. Han : Can we push more constraints into frequent pattern mining ? *In ACM International Conference Knowledge Discovery and Data Mining (KDD'00)*, pages 350–354, 2000.

[PHMA+01] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal et M.-C. Hsu : Prefixspan : Mining sequential patterns efficiently by prefix-projected pattern growth. *In 17th International Conference Data Engineering (ICDE'01)*, pages 215–226, 2001.

[PHW02] J. Pei, J. Han et W. Wang : Mining sequential patterns with constraints in large databases. *In the International Conference on Information and Knowledge Management (CIKM'02)*, pages 18–25, 2002.

[PLT06] M. Plantevit, A. Laurent et M. Teisseire : HYPE : Prise en compte des hiérarchies lors de l'extraction de motifs séquentiels multidimensionnels. *In Entrepôts de Données et Analyse en ligne (EDA'06)*, pages 155–172, 2006.

[PZOD99] S. Parthasarathy, M. J. Zaki, M. Ogihara et S. Dwarkadas : Incremental and interactive sequence mining. *In 8th International Conference on Information and Knowledge Management (CIKM'99)*, pages 251–258, 1999.

[Qui86] J. R. Quinlan : Induction of Decision Trees, 1986.

[Qui89] J. R. Quinlan : Unknown Attribute Values in Induction. *In Proc. 6th Int. Machine Learning Workshop Cornell*. Morgan Kaufmann, 1989.

[RC98] A. Ragel et B. Crémilleux : Treatment of Missing Values for Association Rules. *In Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 258–270, 1998.

[RC99] A. Ragel et B. Crémilleux : MVC — a Preprocessing Method to Deal with Missing Values. *Knowledge-Based Systems Journal*, pages 285–291, 1999.

[RC04] F. Rioult et B. Crémilleux : Représentation condensée en présence de valeurs manquantes. *In XXIIème Congrès Informatique des organisations et systèmes d'information et de décision (INFORSID'04)*, pages 301–317, 2004.

[RPT05] C. Raissi, P. Poncelet et M. Teisseire : Need for speed : Mining sequential patterns in data streams. *In 21 ème Journée Bases de Données Avancées (BDA '05)*, 2005.

[SA96a] R. Srikant et R. Agrawal : Mining Quantitative Association Rules in Large Relational Tables. *In Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 1–12, 1996.

[SA96b] R. Srikant et R. Agrawal : Mining sequential patterns : Generalizations and performance improvements. *In 5th International Conference on Extending Database Technology (EDBT '96)*, pages 3–17, 1996.

[SF99] M. Spiliopoulou et L. C. Faulstich : WUM : A tool for web utilization analysis. *In EDBT Workshop WebDB'98*, pages 184–203, 1999.

[SG05] R. B. V. Subramanyam et A. Goswami : A fuzzy data mining algorithm for incremental mining of quantitative sequential patterns. *International Journal of Uncertainty Fuzziness Knowledge-Based Systems*, 13(6):633–652, 2005.

[Sou06] A. Soulet : *Un cadre générique de découverte de motifs sous contraintes fondées sur des primitives*. Thèse de doctorat, Université de Caen, 2006.

[SVA97] R. Srikant, Q. Vu et R. Agrawal : Mining association rules with item constraints. *In 3rd International Conference Knowledge Discovery and Data Mining (KDD'97)*, pages 63–73, 1997.

[TDK03] H. Timm, C. Doring et R. Kruse : Different Approaches to Fuzzy Clustering of Incomplete Datasets. *International Journal of Approximate Reasoning*, 35, 2003.

[TK99] H. Timm et F. Klawonn : Different Approaches for Fuzzy Cluster Analysis with Missing Values. *In 7th European Congress on Intelligent Techniques and Soft Computing*, 1999.

[Toi96] H. Toivonen : Sampling large databases for association rules. *In VLDB '96 : Proceedings of the 22th International Conference on Very Large Data Bases*, pages 134–145, 1996.

[VCB06] H. Verlinde, M. De Cock et R. Boute : Fuzzy versus quantitative association rules : a fair data-driven comparison. *IEEE Transactions on Systems, Man and Cybernetics - Part B*, 36(3):679–684, 2006.

[WF00] I. H. Witten et E. Frank : Data Mining : Practical Machine Learning Tools with Java Implementations, 2000.

[WH04] J. Wang et J. Han : Bide : Efficient mining of frequent closed sequences. *In 20th International Conference on Data Engineering (ICDE'04)*, pages 79–90, 2004.

[Whi86] A. P. White : Probabilistic Induction by Dynamic Path Generation in Virtual Trees. *In the 6th Annnual Conference on Research and Development in Expert Systems*, pages 35–46, 1986.

[WI00] S. M. Weiss et N. Indurkhya : Decision-Rule Solutions for Data Mining with Missing Values. *In the 7th International Joint Ibero-American Conference on AI : Advances in Artificial Intelligence*, volume 1952 de *Lecture Notes In Computer Science*, pages 1–10, 2000.

[WMZ02] M. Wojciechowski, T. Morzy et M. Zakrzewicz : Efficient constraint-based sequential pattern mining using dataset filtering techniques. *In Baltic Conference on Databases and Information Systems (BalticDB-IS'02)*, pages 213–224, 2002.

[Woj01] M. WOJCIECHOWSKI : Interactive constraint-based sequential pattern mining. *In 5th ADBIS Conference*, volume 2151 de *Lecture Notes in Computer Science*, 2001.

[Woj03] M. WOJCIECHOWSKI : Supporting interactive sequential pattern discovery in databases. *In Emerging Database Research in East Europe (VLDB'03 Workshop)*, 2003.

[Wri98] P. WRIGHT : The significance of the missing data problem in knowledge discovery, 1998.

[WWC04] C.-H. WU, C.-H. WUN et H.-J. CHOU : Using association rules for completing missing data. *In 4th International Conference on Hybrid Intelligent Systems (HIS'04)*, pages 236–241, 2004.

[Yag82] R. R. YAGER : A new approach to the summarization of data. *Information Sciences*, 28(1):69–86, 1982.

[Yag96] R. R. YAGER : Database discovery using fuzzy sets. *International Journal of Intelligent Systems*, 11:691 – 712, 1996.

[Yan04] G. YAN : The complexity of mining maximal frequent itemsets and maximal frequent patterns. *In 10th ACM International Conference on Knowledge Discovery and Data Mining (KDD'04)*, pages 344–353, 2004.

[YHA03] X. YAN, J. HAN et R. AFSHAR : Clospan : Mining closed sequential patterns in large datasets. *In 3rd SIAM Int. Conf. on Data Mining (SIAM DM'03)*, 2003.

[YISI00] M. YOSHIDA, T. IIZUKA, H. SHIOHARA et M. ISHIGURO : Mining sequential patterns including time intervals. *In Data Mining and Knowledge Discovery : Theory, Tools, and Technology*, volume 4057 de *SPIE Proceedings series*, pages 213–220, 2000.

[YJGMD96] T.-W. YAN, M. JACOBSEN, H. GARCIA-MOLINA et U. DAYAL : From user access patterns to dynamic hypertext linking. *In 5th International World Wide Web Conference on Computer networks and ISDN systems*, pages 1007–1014, 1996.

[Zad65] L.A. ZADEH : Fuzzy sets. *Information and Control*, 3(8):338–353, 1965.

[Zak00] M. J. ZAKI : Sequence mining in categorical domains : incorporating constraints. *In the 9th International Conference on Information and Knowledge Management (CIKM '00)*, pages 422–429, 2000.

[Zak01] M. J. ZAKI : SPADE : An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1/2):31–60, 2001.

[ZB03] Qiankun ZHAO et S. S. BHOWMICK : Sequential pattern mining : A survey. Rapport technique, Centre for Advanced Information Systems, Nanyang Technological University, Singapore, 2003.

[ZKCY02] M. ZHANG, B. KAO, D. W.-L. CHEUNG et C. L. YIP : Efficient algorithms for incremental update of frequent sequences. *In Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 186–197, 2002.

[ZPLO96] M. J. ZAKI, S. PARTHASARATHY, W. LI et M. OGIHARA : Evaluation of sampling for data mining of association rules. Rapport technique TR617, University of Rochester, 1996.

[ZXH98] O.-R. ZAIANE, M. XIN et J. HAN : Discovering web access patterns and trends by applying olap and data mining technology on web logs. *Advances in Digital Libraries Conference*, 1998.

Cette bibliographie contient **140 références**.

Articles publiés dans le cadre de cette thèse

Chapitre dans un ouvrage à audience internationale

- C. Fiot. Fuzzy Sequential Patterns for Quantitative Data Mining. In Galindo, J. (Ed.), Handbook of Research on Fuzzy Information Processing in Databases. Hershey, PA, USA : Information Science Reference, à paraître (2008).

Revues internationales avec comité de lecture

- C. Fiot, A. Laurent et M. Teisseire. From Crispness to Fuzziness : Three Algorithms for Soft Sequential Pattern Mining. *IEEE Transactions on Fuzzy Systems*, à paraître (2007).

- C. Fiot, A. Laurent et M. Teisseire. Softening the Blow of Frequent Sequence Analysis : Soft Constraints and Temporal Accuracy. *International Journal of Web Engineering and Technology, special issue on Web-based Knowledge Representation and Management*, à paraître (2008).

Revue nationale avec comité de lecture

- C. Fiot. Extension des contraintes de temps : précision et flexibilité pour les motifs séquentiels généralisés. *Revue I3, numéro spécial*, 2007.

Conférences internationales avec comité de lecture

- C. Fiot, A. Laurent et M. Teisseire. SPOID : Do not Throw Meaningful Sequences Away !. In *5th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT'07)*, 09 2007.

- C. Fiot, A. Laurent et M. Teisseire. Approximate Sequential Patterns for Incomplete Sequence Database Mining. In *16th IEEE International Conference on Fuzzy Systems (FuzzIEEE'07)*, 07 2007.

- C. Fiot, A. Laurent et M. Teisseire. Extended Time Constraints for Sequence Mining. In *14th IEEE International Symposium on Temporal Representation and Reasoning (TIME'07)*, 06 2007.

- C. Fiot, A. Laurent et M. Teisseire. Web Access Log Mining with Soft Sequential Patterns. In *7th International FLINS Conference on Applied Artificial Intelligence (FLINS'06)*, 08 2006.

- C. Fiot, A. Laurent, M. Teisseire et B. Laurent. Why Fuzzy Sequential Patterns can Help Data Summarization : an Application to the INPI Trademark Database. In *15th IEEE International Conference on Fuzzy Systems (FuzzIEEE'06)*, 07 2006.

Conférences nationales avec comité de lecture

- C. Fiot. Motifs séquentiels et approximation des valeurs manquante. In *25ème Congrés Informatique des organisations et systèmes d'information et de décision (InforSID'07)*, 05 2007.

- C. Fiot, A. Laurent et M. Teisseire. SPolD : Extraction de motifs séquentiels pour les bases de données incomplètes. In *7èmes journées d'Extraction et Gestion des Connaissances (EGC'07)*, 01 2007.

- C. Fiot. Motifs séquentiels pour la complétion des valeurs manquantes. *4ème Manifestation des Jeunes Chercheurs STIC (MajecSTIC'06)*, 11 2006.

- C. Fiot, A. Laurent et M. Teisseire. Des motifs séquentiels généralisés aux contraintes de temps étendues. In *6èmes journées d'Extraction et Gestion des Connaissances (EGC'06)*, 01 2006.

- C. Fiot, A. Laurent et M. Teisseire. Motifs séquentiels flous : un peu, beaucoup, passionnément. In *5èmes journées d'Extraction et Gestion des Connaissances (EGC'05)*, 01 2005.

- C. Fiot, G. Dray, A. Laurent et M. Teisseire. A la recherche des motifs séquentiels flous. In *12èmes rencontres francophones sur la Logique Floue et ses Applications (LFA'04)*, 11 2004.

Annexes

Annexe A

Quelques éléments de la théorie des sous-ensembles flous

Dans la théorie classique des ensembles, un objet appartient ou n'appartient pas à un ensemble. En prenant comme exemple, pour U l'ensemble des individus, pour A l'ensemble des personnes petites, en logique classique, on a $A \cup \overline{A} = U$ et $A \cap \overline{A} = \varnothing$. Une différence essentielle avec la théorie classique est que, dans la théorie des sous-ensembles flous, introduite par Zadeh en 1965 [Zad65], un objet peut appartenir à un ensemble et en même temps à son complément. Ainsi, un individu de 1m63 pourra à la fois être grand et petit. Soit une variable x (la taille) et un univers de référence ou de discours U (les individus). Un sous-ensemble flou A est défini par une fonction d'appartenance μ_A qui décrit le degré avec lequel l'élément x appartient à A [BM99]. Dans ce système, on parlera de *variable linguistique* taille et de *valeurs linguistiques* petit et grand. Le tableau A.1 illustre ces différences entre la logique classique, pour laquelle un degré d'appartenance a pour valeur 0 ou 1, et la théorie des sous-ensembles flous, où le degré d'appartenance est dans l'intervalle [0, 1].

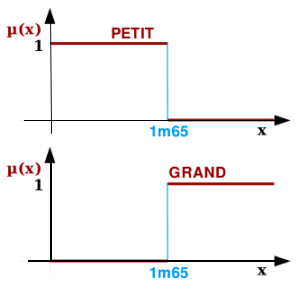
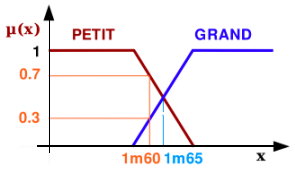
Théorie classique	**Théorie floue**
$\mu_A(x) \in \{0, 1\}$	$\mu_A(x) \in [0, 1]$
	
Un individu qui mesure 1m63 appartient au sous-ensemble PETIT (avec un degré 1)	Un individu de 1m63 appartient au sous-ensemble PETIT et au sous-ensemble GRAND (avec un degré de 0.7 et de 0.3).

FIG. A.1 – Comparaison logique floue - logique classique

Soit une fonction d'appartenance $\mu_A(x)$ de forme trapézoïdale (fig. A.2-a), il est possible de définir sa hauteur, son support et son noyau. Un sous-ensemble classique (fig. A.2-b) est un sous-ensemble flou

particulier avec une hauteur de 1 et dont le support est égal au noyau.

$$noy(A) = \{x \in X | \mu_A(x) = 1\}$$
$$supp(A) = \{x \in X | \mu_A(x) > 0\}$$
$$hgt(A) = max_{x \in X} \mu_A(x)$$

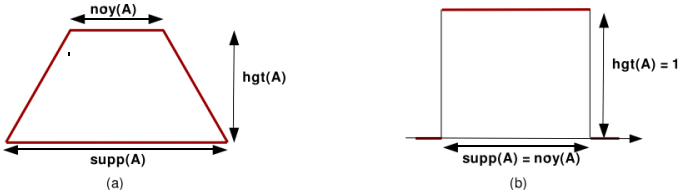


FIG. A.2 – Représentation d'une fonction d'appartenance à un ensemble flou (a) ; représentation d'une fonction d'appartenance en logique classique (b)

Opérateurs en logique floue

Il s'agit de la généralisation des opérateurs négation, intersection et union de la théorie classique des ensembles.

L'opérateur NON (complément) est défini par : $\overline{A} = \{x | x \notin A\}$. Il est représenté par la fonction $non(\mu_A(x)) = \mu_{\overline{A}}(x) = 1 - \mu_A(x)$.

L'opérateur t-norme (intersection, norme triangulaire), noté $\top$, est défini par : $A \cap B = \{x | x \in A \wedge x \in B\}$. Il est représenté par la fonction $\mu_{A \cap B}(x) = \top(\mu_A(x), \mu_B(x))$.
Une *norme triangulaire* est une fonction $\top : [0,1] \times [0,1] \rightarrow [0,1]$ commutative, associative, monotone, d'élément neutre 1.

L'opérateur t-conorme (union, conorme triangulaire), noté $\perp$, est défini par : $A \cup B = \{x | x \in A \vee x \in B\}$. Il est représenté par la fonction $\mu_{A \cup B}(x) = \perp(\mu_A(x), \mu_B(x))$.
Une *conorme triangulaire* est une fonction $\perp : [0,1] \times [0,1] \rightarrow [0,1]$ commutative, associative, monotone, d'élément neutre 0.

Il existe de nombreux opérateurs $\top$ et $\perp$ [Kau73]. Le plus souvent, on utilise les fonctions définies par Zadeh, c'est-à-dire le MIN comme T-norme et le MAX comme T-conorme. Le tableau proposé figure A.3 récapitule plusieurs fonctions fréquemment utilisées comme opérateurs $\top$ et $\perp$ [BM99].

Appellation	$\top$	$\perp$	**NON**
Zadeh [Zad65]	$min(x,y)$	$max(x,y)$	$1-x$
Probabiliste [DP80]	xy	$x+y-xy$	$1-x$
Lukasiewicz	$max(x+y-1,0)$	$min(x+y,1)$	$1-x$
Hamacher ($\beta > 0$)	$\frac{xy}{\beta+(1-\beta)(x+y-xy)}$	$\frac{x+y-xy-(1-\beta)xy}{1-(1-\beta)xy}$	$1-x$
Weber	$\begin{cases} x & \text{si } y=1 \\ y & \text{si } x=1 \\ 0 & \text{sinon} \end{cases}$	$\begin{cases} x & \text{si } y=0 \\ y & \text{si } x=0 \\ 1 & \text{sinon} \end{cases}$	$1-x$

FIG. A.3 – Principales t-normes et t-conormes duales [BM99].

Cardinalité d'un sous-ensemble flou

La cardinalité d'un ensemble (non-flou) est le nombre d'éléments qui appartiennent à cet ensemble.

Considérant un ensemble flou, le problème de comptabiliser le nombre d'éléments qu'il contient revient à définir quand un élément appartient ou non à un ensemble flou.

Prenons par exemple le problème suivant. Le tableau présentés sur la figure A.4 montre le degré d'appartenance de chaque achat au sous-ensemble flou *"beaucoup de chocolat"*.

Achats	Beaucoup de chocolat
12/01	0.8
16/01	0.3
20/01	1
21/01	0

FIG. A.4 – Exemple de base de degrés d'appartenance

On cherche à déterminer *le nombre* **N** *d'achats qui comportent beaucoup de chocolat*.

On peut considérer tout d'abord que l'on comptabilise tous les achats pour lesquels le degré d'appartenance est non nul et que, quelle que soit la quantité achetée, tous les achats sont équivalents. Formellement, le nombre d'éléments d'un sous-ensemble flou A sera donné par $card(supp(A))$.
Dans ce cas, on aura **N = 3**, puisque 3 achats parmi les 4 de la base comportent *"beaucoup de chocolat"*.

On peut également considérer que seuls les achats pour lesquels le degré d'appartenance dépasse un certain seuil comportent réellement le produit*"beaucoup de chocolat"* et que, quelle que soit la quantité achetée, tous les achats sont équivalents. Ce comptage est appelé *comptage seuillé*. Formellement, le nombre d'éléments d'un sous-ensemble flou A sera donné par $card(supp(A_\alpha))$.
Dans ce cas, on aura, en prenant pour seuil la valeur 0.5, **N = 2**. En effet, on ne tiendra compte que des achats du 12/01 et du 20/01.

On peut aussi considérer que tous les achats n'ont pas la même importance, selon leur degré d'appartenance. Le nombre d'achats est alors donné par Σ-*comptage*, c'est-à-dire en sommant les degrés d'appartenance de chaque achat. Formellement, le nombre d'éléments d'un sous-ensemble flou A sera donné par $\Sigma_{x \in supp(A)} \mu_A(x)$.
On aura alors **N = 0.8 + 0.3 + 1 + 0 = 2.1**.

Enfin, on peut combiner les deux comptages précédents, en considérant que tous les achats n'ont pas la même importance mais qu'ils ne sont assez significatifs pour être comptabilisés que si leur degré d'appartenance dépasse un certain seuil. On réalise alors un Σ-*comptage seuillé*. Formellement, le nombre d'éléments d'un sous-ensemble flou A sera donné par $\Sigma_{x \in supp(A)} \mu_{A_\alpha}(x)$.
Dans ce cas, en prenant pour seuil la valeur 0.5, on ne tiendra que des achats du 12/01 et du 20/01, on aura alors **N = 0.8 + 1 = 1.8**.

L'utilisation de chacun de ces comptages est fonction des objectifs du comptage et de sa signification.

Partitionnement en sous-ensembles flous
Le traitement des attributs quantitatifs pour l'extraction de règles d'association floues ou de motifs séquentiels flous nécessite une étape supplémentaire, intégrée au prétraitement, le partitionnement. Cette étape consiste à découper l'univers de référence en plusieurs sous-ensembles flous qui constituent une

partition floue.

Une partition floue définit le découpage de l'univers de référence d'une variable linguistique ou d'un attribut quantitatif par des sous-ensembles flous. Prenons la variable linguistique *"taille"*, elle pourra par exemple être partitionnée en trois sous-ensembles flous *"Petit"*, *"Moyen"* et *"Grand"*.
Contrairement au partitionnement d'un ensemble non-flou, où les sous-ensembles sont disjoints deux à deux, l'intersection de deux sous-ensembles flous d'une partition floue n'est pas vide.

Il existe plusieurs définitions d'une partition floue. Une première impose comme seule restriction, que chaque élément du domaine partitionné doit appartenir à un sous-ensemble au moins. Autrement dit, chaque élément doit avoir au moins un degré d'appartenance non nul, comme le montre la figure A.5(a).

Définition A.1. La *partition floue* d'un univers de discours U consiste à définir n sous-ensembles flous F_i de façon à recouvrir U. C'est-à-dire que pour tout élément x de U, il faut assurer une appartenance minimale non nulle à l'union des F_i.

$$\forall x \in U, \exists F_i \ / \ \mu_{F_i}(x) > 0$$

Les partitions floues fortes sont des partitions floues auxquelles on impose des contraintes supplémentaires. Parmi ces contraintes, celle que nous imposerons à notre découpage : pour un élément, la somme de ses degrés d'appartenance aux différents sous-ensembles flous est égale à 1, par exemple la figure A.5(b) montre une partition floue forte du domaine de variation de la variable "taille".

Définition A.2. Soient n sous-ensembles flous F_i définissant une partition sur le domaine U. Une *partition floue forte* est définie par :

$$\forall x \in U, \sum_{i=1}^{n} \mu_{F_i}(x) = 1$$

(a) (b)

FIG. A.5 – (a) : Exemple de partition floue ; (b) : Exemple de partition floue forte.

Le partitionnement flou consiste à trouver un découpage en sous-ensembles flous. Une technique assez répandue consiste à utiliser des algorithmes de segmentation afin d'extraire des classes, puis des sous-ensembles flous, pour chacun des attributs, regroupant assez d'enregistrements pertinents.

Notamment, certaines techniques de fouille de données [Gye00b, HK01] utilisent ce principe avec l'algorithme de clustering flou fuzzy c-means pour définir des sous-ensembles flous en vue de l'extraction de motifs. La méthode consiste tout d'abord à trouver les différents groupes par l'algorithme de clustering flou, puis à construire les sous-ensembles flous grâce aux centres des groupes mis en évidence, pour

chaque attribut quantitatif. Enfin, la fonction d'appartenance correspondante est dérivée à partir des sous-ensembles trouvés.

Cette technique est celle utilisée par un certain nombre de propositions formulées afin d'utiliser des partitions floues d'attributs numériques en vue d'une extraction de connaissances symboliques. On peut notamment citer des travaux concernant la recherche de règles d'association floue [KFW98, Gye00b] ou la découverte de motifs séquentiels flous dans des bases de données de type relationnelles [HTC04].

Pour certaines applications, ce découpage peut également être défini a priori par un expert du domaine. Enfin, certaines expérimentations sur données synthétiques homogènes utilisent les découpages *equi-depth*[1] et *equi-width*[2] de Weka [WF00].

Résumés flous

L'extraction de descriptions concises et compréhensibles est un des challenges pour les utilisateurs de grandes bases de données. L'une des techniques à leur disposition repose sur l'utilisation de la théorie des sous-ensembles flous. En effet, l'utilisation de variables linguistiques permet la formulation de règles et de résumés humainement compréhensibles. L'approche classique consiste à formuler des résumés linguistiques [Yag82] ou des résumés flous [Kac88, KYZ00]. Toutefois, cette formulation nécessite l'interaction d'un utilisateur afin de sélectionner les connaissances intéressantes et utiles, grâce à des mesures de validité et de qualité de ces résumés.

Plus précisément, les résumés flous sont des propositions comportant des quantificateurs linguistiques généralement formulées de la façon suivante "Q Y sont B", où Q est un quantificateur linguistique, Y est un ensemble d'objets et B est une propriété. Un tel résumé pourrait être, par exemple, "La plupart des chats sont noirs". Les résumés flous sont caractérisés par un degré de vérité indiquant à quel point le résumé est vérifié par les données.

Ces résumés sont généralement construits de façon partiellement automatique, par combinaison de quantificateurs, objets et propriétés connus a priori. Chacune des propositions ainsi formulées est ensuite vérifiée sur la base et retournée accompagnée d'un degré de vérité. A l'inverse, les résumés flous issus de règles d'association sont construits de façon totalement automatique.

[1] Equi-répartition en nombre d'éléments
[2] Intervalles de largeur équivalente

Annexe B

SpeedyFuzzy, MiniFuzzy et TotallyFuzzy

B.1 L'algorithme SpeedyFuzzy

Le comptage SpeedyFuzzy utilise la fonction *FindSpeedySeq*, Alg. 1, qui recherche la séquence candidate dans la séquence d'enregistrements d'un objet. Cela implique de rechercher chaque itemset de la séquence en respectant l'ordre de celle-ci. Lorsque le premier itemset est trouvé dans l'enregistrement, l'itemset suivant est recherché dans les enregistrements suivants et ainsi de suite.

FindSpeedySeq - **Input** : gS, candidate g-k-sequence ; $\mathcal{R}$, record set to run
Ouput : $found$, 1 if gS is found, 0 else

$currIS \leftarrow g$S.first() ;
$found \leftarrow 0$;
$r \leftarrow \mathcal{R}$.first() ;
Tant que (((!$found$) || (r !=$\varnothing$))) **faire**
  Si (($\overline{\top}_{[x,a]\in currIS} \alpha_a(r[x]) \geq 0$)) **Alors**
    Si (($currIS$!= gS.last())) **Alors**
      $currIS \leftarrow g$S.next() ;
    Sinon
      $found \leftarrow 1$;
    Fin Si
  Fin Si
  $r \leftarrow \mathcal{R}$.next() ;
Fin Tant que
Retourner $found$

Alg. 1 : *FindSpeedySeq*

B.2 L'algorithme MiniFuzzy

Le comptage MiniFuzzy utilise la fonction *FindMiniSeq*, Alg. 2, qui recherche la séquence candidate dans la séquence d'enregistrements d'un objet. Cela implique de rechercher chaque itemset de la séquence en respectant l'ordre de celle-ci. Lorsque le premier itemset est trouvé avec un degré supérieur au seuil ω, l'itemset suivant est recherché dans les enregistrements suivants et ainsi de suite.

FindMiniSeq - **Input** : gS, candidate g-k-sequence ; $\mathcal{R}$, record set to run
Ouput : $found$, 1 if gS is found, 0 else

$currIS \leftarrow g$S.first() ;
$found \leftarrow 0$;
$r \leftarrow \mathcal{R}$.first() ;
Tant que (((!$found$) || (r !=$\varnothing$))) **faire**
  Si (($\overline{\top}_{[x,a]\in currIS}\alpha_a(r[x]) \geq \omega$)) **Alors**
    Si (($currIS$!= gS.last())) **Alors**
      $currIS \leftarrow g$S.next() ;
    Sinon
      $found \leftarrow 1$;
    Fin Si
  Fin Si
  $r \leftarrow \mathcal{R}$.next() ;
Fin Tant que
Retourner $found$

ALG. 2 : *FindMiniSeq*

B.3 L'algorithme TOTALLYFUZZY

L'algorithme TOTALLYFUZZY utilise la fonction *FindTotallySeq* pour parcourir dans l'ordre les enregistrements d'un objet afin de trouver la meilleure occurrence d'une séquence candidate par cette séquence de données. Lorsque le premier itemset de la séquence candidate est trouvé, un chemin est créé avec le degré de l'itemset. Les enregistrements suivants sont ensuite parcourus afin de trouver : la suite de la séquence candidate, à nouveau le début de la séquence, ou encore une amélioration des chemins déjà créés. Tous les chemins possibles sont ainsi complétés pas à pas à chaque enregistrement. Le chemin complet pour la séquence candidate ayant le meilleur degré est conservé pour le calcul de la fréquence. La fonction *Update* permet la mise à jour de chaque chemin et leur clôture lorsqu'ils contiennent toute la séquence candidate. La fonction *Optimize* permet de limiter l'espace mémoire utilisé en supprimant tous les chemins qui ne permettront pas une solution optimale.

Optimize - **Input** : $Paths$, list of paths to reduce

Pour each $pth \in Paths$ **faire**
  Tant que (($Paths$.hasnext())) **faire**
    $pth' \leftarrow Paths$.next() ;
    Si ($pth'.currIS = pth.currIS$) **Alors**
      Si ($\odot(pth.curDeg) \geq \odot(pth'.curDeg)$) **Alors**
        delete(pth') ;
      Sinon
        delete(pth) ;
      Fin Si
    Fin Si
  Fin Tant que
Fin Pour

ALG. 3 : *Optimize*

Update - **Input** : pth, path to update

pth.seq $\leftarrow$ pth.seq $\cup pth$.curIS ;
Si pth.curIS.next $\neq \varnothing$ **Alors** pth.curIS $\leftarrow$ pth.curIS.next ;
Sinon close(pth) ;
Fin Si

ALG. 4 : *Update*

FindTotallySeq - **Input** : g-S, candidate g-k-sequence ; R, record set to run
Ouput : m, frequency degree of the best g-S representation instanciated in the record set R

$Paths$: **list of paths** $\rightarrow$ **(seq, curIS, curDeg)**
$Paths \leftarrow$Path($\varnothing$, gS.first, 0)
Pour each transaction $r \in R$ **faire**
  Pour each path $pth \in Paths$, not updated at r **faire**
    Si (pth not closed) **Alors**
      Si (pth.curIS $\in r$) **Alors**
        pth.curDeg $\leftarrow$ pth.curDeg | $\overline{\top}_{[x,a] \in pth.\text{curIS}} \alpha_a(r[x])$;
        Update(pth) ;
      Fin Si
    Fin Si
    Pour j **de** 2 **à** pth.curIS -1 **faire**
      [Search for an improvement of the current path]
      Si ((gS.get(j) $\in r$) &
      ($\overline{\top}_{[x,a] \in gS.get(j)} \alpha_a(r[x]) > pth$.curDeg[j])) **Alors**
        nCurIS $\leftarrow$ gS.get(j) ;
        Pour i **de** 1 **à** j-1 **faire**
          nSeq $\leftarrow$ nSeq | gS.get(i) ;
          nCurDeg $\leftarrow$ nCurDeg | pth.curDeg[i] ;
        Fin Pour
        nCurDeg $\leftarrow$ nCurDeg | $\overline{\top}_{[x,a] \in gS.get(j)} \alpha_a(r[x])$;
        $Paths \leftarrow Paths \cup$ Update((nSeq, nCurIS, nCurDeg)) ;
      Fin Si
    Fin Pour
  Fin Pour
  Si ((gS.first $\in r$) & (not(FirstPass))) **Alors**
    $pth \leftarrow$Path($\varnothing$, gS.first, $\overline{\top}_{[x,a] \in gS.first} \alpha_a(r[x])$) ;
    $Paths \leftarrow Paths \cup pth$;
    Update(pth) ;
  Fin Si
  Paths.Optimize() ; *[deletion of less pertinent paths]*
Fin Pour
Pour each path $pth \in Paths$ **faire**
  Si (pth not closed) **Alors** delete(pth) ; *[deletion of paths not containing the whole sequence]*
  Fin Si
Fin Pour
$pth \leftarrow Paths.first$; *[$Paths$ only contains the best complete path]*
$m \leftarrow \odot$(pth.curDeg) ; *[Agregation to return the frequency degree]*
Retourner m ;

ALG. 5 : *FindTotallySeq*

Annexe C

Complétude de GETC : Démonstrations

Nous allons montrer que GETC extrait exactement toutes les séquences maximales supportées par la séquence d'entrée.

Théorème C.1. L'inclusion de séquence est impossible après l'exécution de l'algorithme GETC.

Démonstration. Supposons qu'il existe, dans le graphe de séquences, deux séquences s_1 et s_2 telles que $s_1 \subset s_2$.
Cela signifie que le graphe de séquences contient un sommet y tel que l'un des prédécesseurs x de y est accessible par t, également prédécesseur de y, c'est-à-dire qu'il existe un chemin $(t, ..., y)$ de longueur supérieure ou égale à 2 et un arc (t,y).
Par construction, ce type de cas ne se présentent que lors de l'exécution de $propagate$. L'existence d'un tel chemin implique une intersection entre les sommets qui précèdent y et ceux qui suivent t. Or si une telle situation se présente, $propagate$ ne crée pas de nouvel arc entre t et y. Ainsi GETC ne construit bien que le chemin maximal et élimine bien les chemins inclus. □

Théorème C.2. L'algorithme GETC construit exactement toutes les solutions de la plus grande taille possible pour les séquences respectant $minGap$ et $maxGap$.

Démonstration. L'algorithme GETC parcourt tous les sommets du graphe, ce qui implique que si un chemin respectant $minGap$ et $maxGap$ contient le sommet x, alors ce chemin fait partie du graphe après l'exécution de GETC (même s'il est restreint au seul sommet x).
Tous les sommets font partie d'un chemin, l'inclusion de chemin est impossible et si deux chemins $(x, ..., y)$ et $(y', ..., z)$ peuvent être fusionnés (i.e. si y'.date() - y.date() $> g_\rho$ et z.date() - x.date() $> G$), ils le seront, car l'algorithme, en explorant le sommet y, va construire l'arc (y,y').
L'algorithme GETC construit donc exactement toutes les solutions de la plus grande taille possible pour les séquences respectant $minGap$ et $maxGap$. □

Théorème C.3. L'algorithme $addWindowSize$ construit exactement tous les sommets susceptibles de contribuer à la construction de toutes les solutions de la plus grande taille possible pour les séquences respectant $minGap$ et $maxGap$.

Démonstration. Supposons que l'on construise tous les sommets résultant des combinaisons d'itemsets respectant la contrainte $windowSize$. Certains sommets ne sont pas utiles pour le problème de la recherche des séquences les plus longues.

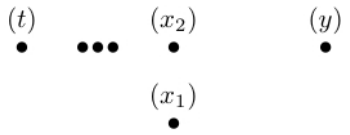


FIG. C.1 – Illustration, théorème

Tout d'abord considérons le cas où le graphe contient à un niveau donné deux sommets x_1 et x_2 tels que les itemsets des deux sommets sont les mêmes et que x_1.end() = x_2.end() et x_1.begin() < x_2.begin(). On considère également un sommet t tel que t.begin() $\leq$ t.end() < x_1.begin() et un sommet y tel que x_1.end() < y.begin() $\leq$ y.end() (figure C.1).
Nous allons montrer que toute séquence incluant x_1 peut également être obtenue en incluant le sommet x_2 et que, par conséquent, seul le sommet x_2 est nécessaire pour rechercher les séquences maximales.

Supposons qu'il existe un arc entre t et x_1, cela signifie que l'on respecte les contraintes $mingap$ et $maxgap$, on a donc :

$$\begin{cases} x_1.\text{end}() - t.\text{begin}() & \leq & maxgap & \text{a} \\ x_1.\text{begin}() - t.\text{end}() & > & mingap & \text{b} \end{cases} \quad \text{(C.1)}$$

Or, x_1.end() = x_2.end(), l'équation C.1-a devient $x_2.\text{end}() - t.\text{begin}() \leq maxgap$. La contrainte sur $maxgap$ est donc également respectée entre t et x_2.
On a également x_1.begin() < x_2.begin(), c'est-à-dire :

$$\begin{array}{lcl} x_2.\text{end}() - t.\text{begin}() & > & x_1.\text{end}() - t.\text{begin}() \\ \text{et donc :} & & \\ x_2.\text{begin}() - t.\text{end}() & > & mingap \end{array} \quad \text{(C.2)}$$

La contrainte sur $mingap$ est donc également respectée entre t et x_2. On aura donc un chemin qui passera par t puis x_2.

Supposons maintenant que l'on ne peut construire d'arc entre t et x_1. Cela signifie que l'une des deux contraintes sur $mingap$ et $maxgap$ n'est pas respectée.
Si c'est la contrainte sur $maxgap$ qui n'est pas respectée, comme x_1.end() = x_2.end(), elle ne sera pas non plus respectée entre t et x_2. Si au contraire, c'est la contrainte sur $mingap$ qui n'est pas satisfaite, on aura plus de chance de la satisfaire avec x_2. En effet :

$$\begin{array}{lcl} x_1.\text{begin}() & < & x_2.\text{begin}() \\ x_1.\text{begin}() - t.\text{end}() & < & x_2.\text{begin}() - t.\text{end}() \end{array} \quad \text{(C.3)}$$

Ce qui signifie que même si $x_1.\text{begin}() - t.\text{end}() \leq mingap$, on ne peut pas en conclure que $x_2.\text{begin}() - t.\text{end}() \leq mingap$.

Nous avons montré que tout séquence incluant x_1 est une sous-séquence d'une séquence incluant x_2 et aussi que si un chemin ne passe pas par x_1, il peut passer par x_2. Le sommet x_1 est donc redondant par rapport au sommet x_2 pour la construction d'un chemin aboutissant à l'itemset représenté par les

sommets x_1 et x_2.

Supposons maintenant qu'il existe un arc entre x_1 et y, cela signifie que l'on respecte les contraintes $mingap$ et $maxgap$, on a donc :

$$\begin{cases} y.\text{end}() - x_1.\text{begin}() & \leq & maxgap & \text{a} \\ y.\text{begin}() - x_1.\text{end}() & > & mingap & \text{b} \end{cases} \tag{C.4}$$

Or, x_1.end() = x_2.end(), l'équation C.4-b devient $y.\text{begin}() - x_2.\text{end}() > mingap$. La contrainte sur $mingap$ est donc également respectée entre x_2 et y.
On a également x_1.begin() < x_2.begin(), c'est-à-dire :

$$\begin{array}{lcl} -x_1.\text{begin}() & > & -x_2.\text{begin}() \\ y.\text{end}() - x_1.\text{begin}() & > & y.\text{end}() - x_2.\text{begin}() \\ \text{et donc :} & & \\ y.\text{begin}() - x_2.\text{end}() & \leq & maxgap \end{array} \tag{C.5}$$

La contrainte sur $maxgap$ est donc également respectée entre x_2 et y. On aura donc un chemin qui passera par x_2 puis y.

Supposons maintenant que l'on ne peut construire d'arc entre x_1 et y. Cela signifie que l'une des deux contraintes sur $mingap$ et $maxgap$ n'est pas respectée.
Si c'est la contrainte sur $mingap$ qui n'est pas respectée, comme x_1.end() = x_2.end(), elle ne sera pas non plus respectée entre x_2 et y. Si au contraire, c'est la contrainte sur $maxgap$ qui n'est pas satisfaite, on aura plus de chance de la satisfaire avec x_2. En effet :

$$\begin{array}{lcl} x_1.\text{begin}() & < & x_2.\text{begin}() \\ -x_1.\text{begin}() & > & -x_2.\text{begin}() \\ y.\text{end}() - x_1.\text{begin}() & > & y.\text{end}() - x_2.\text{begin}() \end{array} \tag{C.6}$$

Ce qui signifie que même si $t.\text{end}() - x_1.\text{begin}() > maxgap$, on ne peut pas en conclure que $t.\text{end}() - x_2.\text{begin}() > maxgap$.

Nous avons montré que tout séquence incluant x_2 est une sous-séquence d'une séquence incluant x_2 et aussi que si un chemin ne passe pas par x_1, il peut passer par x_2. Le sommet x_1 est donc redondant par rapport au sommet x_2 pour la construction d'un chemin partant de l'itemset représenté par les sommets x_1 et x_2.

Conclusion : pour deux sommets représentant le même itemset, et ayant la même date de fin, seul celui qui a la date de début la plus tardive est nécessaire pour le problème de la recherche de chemin de longueur maximale.

On considère maintenant le cas où le graphe contient à deux niveaux successifs deux sommets x_1 et x_2 tels que les itemsets des deux sommets sont les mêmes et que x_1.begin() = x_2.begin() et x_1.end() < x_2.end(). On considère également un sommet t tel que t.begin() ≤ t.end() < x_1.begin() et un sommet y tel que x_2.end() < y.begin() ≤ y.end() (figure C.2).
Nous allons montrer que toute séquence incluant x_2 peut également être obtenue en incluant le sommet x_1 et que, par conséquent, seul le sommet x_1 est nécessaire pour rechercher les séquences maximales.

Supposons qu'il existe un arc entre t et x_2, cela signifie que l'on respecte les contraintes $mingap$ et $maxgap$, on a donc :

$$\begin{cases} x_2.\text{end}() - t.\text{begin}() & \leq & maxgap & \text{a} \\ x_2.\text{begin}() - t.\text{end}() & > & mingap & \text{b} \end{cases} \tag{C.7}$$

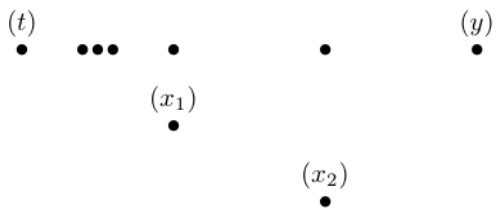


FIG. C.2 – Illustration, théorème

Or, x_1.begin() = x_2.begin(), l'équation C.7-b devient $x_1.\text{begin}() - t.\text{end}() > mingap$. La contrainte sur $mingap$ est donc également respectée entre t et x_1.
On a également x_1.end() < x_2.end(), c'est-à-dire :

$$\begin{array}{lcl} x_1.\text{end}() - t.\text{begin}() & < & x_2.\text{end}() - t.\text{begin}() \\ \text{et donc :} & & \\ x_1.\text{begin}() - t.\text{end}() & \leq & maxgap \end{array} \quad \text{(C.8)}$$

La contrainte sur $maxgap$ est donc également respectée entre t et x_1. On aura donc un chemin qui passera par t puis x_1.

Supposons maintenant que l'on ne peut construire d'arc entre t et x_2. Cela signifie que l'une des deux contraintes sur $mingap$ et $maxgap$ n'est pas respectée.
Si c'est la contrainte sur $mingap$ qui n'est pas respectée, comme x_1.begin() = x_2.begin(), elle ne sera pas non plus respectée entre t et x_1. Si au contraire, c'est la contrainte sur $maxgap$ qui n'est pas satisfaite, on aura plus de chance de la satisfaire avec x_1. En effet :

$$\begin{array}{lcl} x_1.\text{end}() & < & x_2.\text{end}() \\ x_1.\text{end}() - t.\text{begin}() & < & x_2.\text{end}() - t.\text{begin}() \end{array} \quad \text{(C.9)}$$

Ce qui signifie que même si $x_2.\text{end}() - t.\text{begin}() > maxgap$, on ne peut pas en conclure que $x_1.\text{end}() - t.\text{begin}() > maxgap$.

Nous avons montré que tout séquence incluant x_2 est une sous-séquence d'une séquence incluant x_1 et aussi que si un chemin ne passe pas par x_2, il peut passer par x_1. Le sommet x_2 est donc redondant par rapport au sommet x_1 pour la construction d'un chemin aboutissant à l'itemset représenté par les sommets x_1 et x_2.

Supposons maintenant qu'il existe un arc entre x_2 et y, cela signifie que l'on respecte les contraintes $mingap$ et $maxgap$, on a donc :

$$\left\{ \begin{array}{lcll} y.\text{end}() - x_2.\text{begin}() & \leq & maxgap & \text{a} \\ y.\text{begin}() - x_2.\text{end}() & > & mingap & \text{b} \end{array} \right. \quad \text{(C.10)}$$

Or, x_1.begin() = x_2.begin(), l'équation C.10-a devient $y.\text{end}() - x_1.\text{begin}() \leq maxgap$. La contrainte sur $maxgap$ est donc également respectée entre x_1 et y.
On a également x_1.end() < x_2.begin(), c'est-à-dire :

$$\begin{array}{lcl} -x_1.\text{begin}() & > & -x_2.\text{begin}() \\ y.\text{end}() - x_1.\text{begin}() & > & y.\text{end}() - x_2.\text{begin}() \\ \text{et donc :} & & \\ y.\text{end}() - x_2.\text{begin}() & \leq & maxgap \end{array} \quad \text{(C.11)}$$

La contrainte sur $maxgap$ est donc également respectée entre x_1 et y. On aura donc un chemin qui passera par x_1 puis y.

Supposons maintenant que l'on ne peut construire d'arc entre x_2 et y. Cela signifie que l'une des deux contraintes sur $mingap$ et $maxgap$ n'est pas respectée.
Si c'est la contrainte sur $maxgap$ qui n'est pas respectée, comme x_1.begin() = x_2.begin(), elle ne sera pas non plus respectée entre x_1 et y. Si au contraire, c'est la contrainte sur $mingap$ qui n'est pas satisfaite, on aura plus de chance de la satisfaire avec x_1. En effet :

$$\begin{array}{lcl} x_1.\mathrm{end}() & < & x_2.\mathrm{end}() \\ -x_1.\mathrm{end}() & > & -x_2.\mathrm{end}() \\ y.\mathrm{begin}() - x_1.\mathrm{end}() & > & y.\mathrm{begin}() - x_2.\mathrm{end}() \end{array} \tag{C.12}$$

Ce qui signifie que même si $y.\mathrm{begin}() - x_2.\mathrm{end}() \leq mingap$, on ne peut pas en conclure que $y.\mathrm{begin}() - x_1.\mathrm{end}() \leq mingap$.

Nous avons montré que tout séquence incluant x_2 est une sous-séquence d'une séquence incluant x_1 et aussi que si un chemin ne passe pas par x_2, il peut passer par x_1. Le sommet x_2 est donc redondant par rapport au sommet x_1 pour la construction d'un chemin partant de l'itemset représenté par les sommets x_1 et x_2.

Conclusion : pour deux sommets représentant le même itemset, et ayant la même date de début, seul celui qui a la date de fin la plus petite est nécessaire pour le problème de la recherche de chemin de longueur maximale.

L'algorithme $addWindowSize$ construit donc exactement tous les sommets susceptibles de contribuer à la construction de toutes les solutions de la plus grande taille possible pour les séquences respectant $minGap$ et $maxGap$. □

Théorème C.4. L'algorithme GETC construit exactement toutes les solutions de la plus grande taille possible pour les séquences respectant $windowSize$, $minGap$ et $maxGap$.

Démonstration. D'après le théorème C.2, GETC construit exactement toutes les solutions de la plus grande taille possible pour les séquences respectant $minGap$ et $maxGap$. Il reste à vérifier que le traitement de la contrainte $windowSize$ ne permet pas non plus d'inclusion.
Supposons que le graphe de séquences calculé par GETC contient deux séquences s_1 et s_2 tels que $s_1 \subset s_2$. Cela signifie que le graphe de séquences contient un sous-graphe tel que l'un des sommets y inclus dans un autre sommet z et tel que y.next() $\subseteq z$.next().
Or les algorithmes $addEdge$ et $propagate$ marquent un tel sommet y lors de la construction des chemins et l'algorithme $pruneMarked$ supprime les sommets marqués. Par construction, une telle inclusion est donc impossible. □

Annexe D

Types de contraintes pour la recherche de motifs fréquents

Afin de réduire la quantité de motifs générés et de fournir des connaissances réellement intéressantes pour l'utilisateur final, de nombreuses propositions ont été formulées afin d'extraire des motifs (itemsets ou séquences) sous contraintes. Nous présentons ici les contraintes le plus fréquemment rencontrées, ainsi que les propriétés permettant leur intégration lors d'un processus de recherche de motifs fréquents.

D.1 Différents types de contraintes

D'un point de vue applicatif, la plupart des contraintes peuvent être regroupées en sept catégories selon leur forme et leur sens [PHW02] :

1. Une *contrainte d'item*, tout d'abord, spécifie quel item ou groupe d'items doivent se trouver ou non dans les motifs extraits.
2. Une *contrainte de longueur* impose des limitations concernant la longueur du motif en terme de répétition d'un item ou du nombre d'itemsets.
3. Une *contrainte de super-motif* permet de n'extraire que les motifs qui incluent un sous-motif spécifié.
4. Une *contrainte d'agrégation* est une contrainte posée sur un groupe d'items composant un motif. Ces items sont agrégés selon une fonction d'agrégation `sum, avg, min, max, std_dev, etc...`
5. Une *contrainte d'expression régulière* est une contrainte spécifiée sous la forme d'une expression régulière sur un ensemble d'items, utilisant un ensemble défini a priori d'opérateurs d'expressions régulières.
6. Une *contrainte de durée* ne peut être définie que sur une base de séquences. Elle impose que le motif doit apparaître fréquemment et que chaque occurrence doit se produire sur une certaine durée.
7. Une *contrainte de laps de temps*, enfin, ne peut également être définie que sur des bases de séquences. Elle impose une durée minimale ou maximale entre les itemsets d'une même séquence.

Parmi ces contraintes, seules les deux dernières ont un impact sur le calcul du support. Les autres contraintes peuvent en effet être traitées comme des filtres sur les fréquents une fois que ceux-ci ont été déterminés. Toutefois, de nombreux travaux se sont intéressés à les intégrer lors de la fouille afin de réduire l'espace de recherche et d'améliorer les performances algorithmiques. C'est pourquoi, ces contraintes ont été classées selon certaines propriétés. Celles-ci permettent en effet de déterminer à quelle étape de la

fouille la contrainte peut être intégrée et quelles contraintes peuvent être combinées afin d'obtenir un résultat complet.

D.2 Classes des contraintes

Lors de l'extraction de motifs fréquents non séquentiels, les contraintes peuvent être classées selon qu'elles sont *monotones*, *antimonotones*[1], *succinctes*, ou *convertible (monotone ou antimonotone)* [PH00].

Une contrainte $\mathcal{C}_{AM}$ est *antimonotone* si toutes les sous-séquences s' d'une séquence s satisfaisant $\mathcal{C}$ satisfont également $\mathcal{C}$.

Une contrainte $\mathcal{C}_M$ est *monotone* si toutes les séquences s incluant une séquence s satisfaisant $\mathcal{C}$ satisfont également $\mathcal{C}$.

Une contrainte $\mathcal{C}_S$ est dite *succincte* lorsque l'ensemble des items satisfaisant $\mathcal{C}$ peut être calculée sans que cela ne nécessite de générer puis tester toutes les possibilités.

Une contrainte $\mathcal{C}_{CAM}$ est convertible antimonotone si et seulement s'il existe un ordre sur les items tel que la contrainte soit anti-monotone (resp. monotone) sur les préfixes.

La dernière propriété, la convertibilité, a tout d'abord été définie et utilisée lors de la recherche d'itemsets, elle a ensuite été étendue sous le terme de *monotonie préfixée* pour l'extraction de séquences fréquentes [PHW02].

Ces classes de contraintes se recoupent [Sou06], comme décrit par la figure D.1.

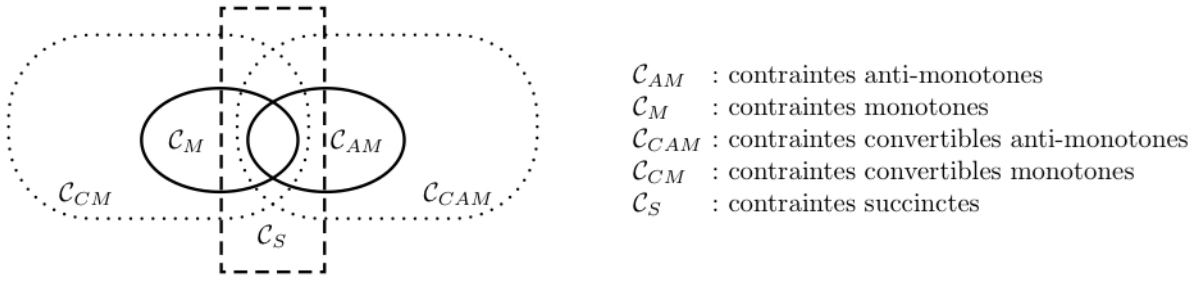


FIG. D.1 – Comparaisons des différentes classes de contraintes

[1] Les classes de contraintes monotones et antimonotones sont définies pour tous les langages, et par conséquent, dans le cadre de la découverte de motifs, également pour les séquences ainsi que les graphes.

Extraction de séquences fréquentes : des données numériques aux valeurs manquantes

La quantité de données aujourd'hui emmagasinées dans tous les domaines ainsi que leur diversité d'origines et de formats rendent impossibles l'analyse, le résumé ou l'extraction manuelle de connaissances. Pour répondre à ces besoins, diverses communautés se sont intéressées à la conception et au développement d'outils permettant d'extraire automatiquement de la connaissance de ces grandes bases. Désormais ces travaux visent à prendre en compte l'hétérogénéité de ces données, de leur format et de leur qualité. Notre travail s'inscrit dans cet axe de recherche et, plus précisément, dans le contexte de la découverte de schémas fréquents à partir de données regroupées sous la forme de séquences ordonnées. Ces schémas, appelés motifs séquentiels, n'étaient jusqu'alors extraits que sur des bases de données de séquences symboliques et parfaites, c'est-à-dire des bases ne contenant que des informations binaires ou pouvant être traitées comme telles et ne contenant aucun enregistrement incomplet. Nous avons donc proposé plusieurs améliorations des techniques d'extraction de séquences fréquentes afin de prendre en compte des données hétérogènes, incomplètes, incertaines ou mal connues de leur utilisateur, tout en minimisant les pertes éventuelles d'informations. Ainsi, le travail présenté dans cette thèse comporte la mise en œuvre d'un cadre pour l'extraction de motifs séquentiels en présence de données numériques quantitatives, la définition de contraintes temporelles relâchées autorisant l'utilisateur à spécifier des contraintes temporelles approximatives et permettant un tri des résultats obtenus selon un indice de précision temporelle, enfin, le développement de deux approches pour l'extraction de motifs séquentiels sur des données symboliques incomplètes.

Frequent Sequence Discovery : from Numerical Data to Missing Values

The large amount of data stored in any areas as well as the diversity of their format and origin make manual analysis or knowledge discovery impossible. For this reason, various communities have been interested for several years in the conception and implementation of tools that can automatically extract knowledge from such large databases. Nowadays these works aim at considering heterogeneity of data, format and quality. Our own work is part of this research axis. More particularly, we consider the context of frequent pattern discovery from data ordered as data sequences. Until now, such patterns, called sequential patterns, could be extracted only from sequence databases containing symbolic and perfect data, i.e. databases consisting of binary information or data that can be processed as binary and only containing complete data. So we propose several improvement of frequent sequence discovery techniques in order to take into account heterogeneous, incomplete or uncertain data, while minimizing possible information loss. Thus, the work described in this thesis consists of the implementation of a global framework for fuzzy sequential pattern discovery within numerical quantitative data, the definition of soft temporal constraints allowing flexibility for the user and sorting of uncovered patterns, last the implementation of two approaches for sequential pattern discovery from incomplete data.

Mots-clés : Extraction de connaissances, fouille de données, logique floue, sous-ensembles flous, motifs séquentiels, séquences fréquentes, base de données de séquences, données numériques, données incomplètes, valeurs manquantes, contraintes temporelles, règles d'association.

Keywords : Knowledge discovery, data mining, fuzzy logic, fuzzy sets, sequential patterns, frequent sequences, sequence database, numerical data, incomplete data, missing values, temporal constraints, association rules.

Discipline : Informatique
Laboratoire : Laboratoire d'Informatique de Robotique et de Micro-électronique de Montpellier
Université Montpellier II - CNRS (UMR 5506)
161 rue Ada - 34392 Montpellier cedex 5 - France