RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

FACULTÉ DES SCIENCES
DÉPARTEMENT DES MATHÉMATIQUES ET D'INFORMATIQUE

T H È S E

pour obtenir le titre de

Docteur en Sciences

Spécialité : INFORMATIQUE

Présentée par

Slimane OULAD-NAOUI

Fouille de motifs : formalisation et unification

Soutenue publiquement le 15 mars 2018 devant le jury composé de :

Président :	M. B. YAGOUBI	Prof.	UATL
Examinateurs :	H. DRIAS	Prof.	USTHB
	D.E. ZEGOUR	Prof.	ESI
	Y. OUINTEN	Prof.	UATL
Directeur :	H. CHERROUN	Prof.	UATL
Co-Directeur :	D. ZIADI	Prof.	Normandie Université (France)

Remerciements

Cette thèse est le fruit de la collaboration de plusieurs personnes.

En premier lieu, je suis chaleureusement reconnaissant à Mme. Hadda Cherroun, professeur à l'université de Laghouat, pour d'abord avoir accepté de me diriger, pour sa disponibilité, son suivi inextirpable, ainsi que pour sa compréhension indulgente en dépit des multiples ruptures enregistrées durant l'élaboration de ce travail pour des cas de force majeure. Ses apports et empreintes dans ma carrière tant de chercheur que d'enseignant sont inestimables.

Je suis aussi énormément redevable à M. Djelloul Ziadi, professeur à Normandie université (France), qui m'a guidé à aborder et révéler une problématique de recherche ouverte. Durant ces longues années, j'ai beaucoup appris de sa méthode unique avec laquelle il m'a hautement dirigé, orienté et soutenu dans les différentes phases de l'aboutissement de cette thèse. Je dois avouer que son patronage à plus d'un titre a été, à mon égard, précieux et indéfectible.

Mes vifs remerciements sont adressés également aux distingués membres du jury, pour l'intérêt accordé à mon travail et pour m'avoir honoré en acceptant de l'évaluer.

Toute ma gratitude aux membres du laboratoire d'informatique et de mathématiques de l'université de Laghouat pour leur aimable assistance, et ceux des équipes du professeur Jean-Gabriel Luque du laboratoire LITIS de Rouen, et du professeur Bruce W. Watson de l'université de Stellenbosch en Afrique du sud.

Je ne peux ignorer le soutient capital de mes camarades d'école. Mille mercis à : Slimane Bellaouar, Laredj Chellama, Attia Nehar, Younes Guellouma, Abdellah Lakhdari, Aicha Chorana, Ahlem Belabbassi et toutes/tous les autres.

Je remercie infiniment M. Mohammed Herouini, inspecteur d'Anglais auprès de l'académie de la wilaya de Ghardaia pour ses lectures et remarques pertinentes sur mes papiers en Anglais.

Je dois rendre hommage à mes parents, et mes sœurs et frères qui m'ont toujours aidé malgré tout ce que nous avons éprouvé. Un merci exceptionnel à ma femme qui m'a encouragé et soutenu avec détermination.

Enfin, je voudrais exprimer mon immense regret et mes excuses à toute ma famille, en particuliers mes enfants, qui ont aussi traversés une longue période de stress et de pression. Désolé pour les nombreux week-ends et mêmes vacances et jours fériés passés en lecture, développement/expérimentation ou rédaction.

ملخص

خلال العقدين الماضيين تم إنجاز عمل جبار في ما يتعلق بالجوانب الخوارزمية للتنقيب على الأنماط المتكررة مما أدى إلى ظهور عدد هائل من الخوارزميات و البرمجيات ذات الصلة، كل منها يدعي التفوق. في حين أنه من المعروف أن تطوير رؤى نظرية توحيدية يعد من المسائل المهمة المفتوحة للبحث في مجال التنقيب في البيانات. و عليه فإن المحفز الرئيسي لعملنا هذا هو إنشاء إطارعالي المستوى في هذا المجال و الذي سيفضي إلى رؤية موحدة للمقاربات المقترحة لحد الآن. من أبرز مميّزات النموذج المقترح هوأنه يزاوج في الآن نفسه و بطريقة ذاتية بين الجانب الكيفي و الكمي لهذه المسألة الحيوية، و التي تمت في السابق معالجتها دائما بصفة منفصلة. سنقوم بتكييف المشكلة قيد الدراسة على شكل نموذج يعتمد السلاسل الصورية. بعد ترميز الأنماط على شكل كلمات مستقات من أبجدية مرتبة، سنعبر عن المسألة بسلسلة صورية تعمل في شبه حلقة التعداد $(N, +, \times, 0, 1)$ و التي يمثل مجالها الأنماط في حين تعبر معاملاتها عن دواعم هذه الأخيرة. هدفنا ثلاثي : أولا، إدراج إطار نظري واضح و موحد قابل للتمدد و الذي يمكننا من إعادة تكوين مقاربات التنقيب عن الأنماط المتكررة. ثانيا، إثبات علاقة ملائمة بين تحديد الآلية ذات الأوزان الممثلة لقاعدة البيانات و إستخراج الأنماط المتكررة. أخيرا ، تحقيق و تقييم النموذج المقترح ببرنامج أسميناه وافي. النتائج المتحصل عليها تبرهن صحة نظرة نموذجنا المعتمدة.

كلمات مفتاحية

التنقيب في البيانات، الأنماط المتكررة، السلاسل الصورية، آليات ذات الأوزان، توحيد خوارزميات

Abstract

Over the last two decades, a great deal of work has been devoted to the algorithmic aspects of the Frequent Pattern (FP) mining problem, leading to a phenomenal number of algorithms and associated implementations, each of which claims supremacy. Meanwhile, it is generally well agreed that developing a unifying theory is one of the most important issues in data mining research. Hence, our primary motivation for this work is to introduce a high level formalism for this basic problem, which induces a unified vision of the algorithmic approaches presented so far. The key distinctive feature of the introduced model is that it combines, in one fashion, both the qualitative and the quantitative aspects of this basic problem, which were previously handled separately. In this thesis, we propose a new model for the FP-mining task based on formal series. In fact, we encode the patterns as words over a sorted alphabet and express this problem by a formal series over the counting semiring $(\mathbb{N}, +, \times, 0, 1)$, whose range represents the patterns, and the coefficients are their supports. The aim is threefold : first, to define a clear, unified and extensible theoretical framework through which we can state the main FP-approaches. Second, to prove a convenient connection between the determinization of the acyclic weighted automaton that represents a transaction dataset and the computation of the associated collection of FP. Finally, to devise a first implementation, baptized WAFI (for Weighted Automata Frequent Itemset mining algorithm), of our model by means of weighted automata, which we evaluate against representative leading algorithms. The obtained results show the suitability of our formalism.

Keywords : Data mining, Frequent patterns, Formal series, Weighted automata, Algorithm unification

Résumé

Durant les deux dernières décennies, un travail considérable a été consacré aux aspects algorithmiques de la fouille de motifs fréquents, ce qui a donné naissance à un nombre phénoménal d'algorithmes et d'implémentations associées où chacun prétend la prééminence. Parallèlement, il est généralement admis par la communauté que le développement d'une théorie unificatrice est une des questions ouvertes et les plus intéressantes dans la sphère de recherche en fouille de données. Aussi, la première motivation de notre travail dans cette thèse est l'introduction d'un formalisme de haut niveau pour ce problème fondamental, qui induit une vision unifiée des approches algorithmiques développées jusque-là. Un des traits remarquables de notre modèle est qu'il prend en charge de manière intrinsèque les aspects à la fois qualitatifs et quantitatifs de ce problème basique, traités antérieurement toujours de manière séparée. En effet, nous formalisons ce problème à l'aide d'un modèle reposant sur les séries formelles. Après avoir encodé les motifs comme des mots sur un alphabet ordonné, nous exprimons ce problème par une série formelle sur le semi-anneau de comptage $(\mathbb{N}, +, \times, 0, 1)$, dont l'étendu représente les motifs et les coefficients sont leurs supports. L'objectif est triple : primo, la définition d'un cadre théorique clair, unifié et extensible à travers lequel nous pouvons reproduire les principales approches de fouille de motifs fréquents de l'état de l'art ; secundo, la preuve d'un lien approprié entre la determinisation de l'automate à multiplicité représentant une base de transactions et le calcul de la collection associée de motifs fréquents ; finalement, la mise en œuvre et l'évaluation d'une première implémentation, via des automates à multiplicités, de notre formalisme que nous baptisons WAFI (pour Weighted Automata Frequent Itemset mining algorithm). Les résultats obtenus montrent le bien fondé et l'adéquation de notre modèle.

Mots clés : Fouille de données, Motifs fréquents, Séries formelles, Automates à multiplicité, Unification d'algorithmes

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Liste des abréviations et notations

A	L'ensemble des items
D	Une base de données transactionnelle
T	L'ensemble de transactions d'une base D
s	Le seuil minimal du support
tidlist	Transaction Identifier List
sprt	Le support d'un motif
conf	La confiance d'une règle d'association
MF	Motif Fréquent
$\mathscr{F}$	L'ensemble de tous les MF d'une base D relativement à un support s
MFM	Motif(s) Fréquent(s) Maximal(aux)
MFF	Motif(s) Fréquent(s) Fermé(s)
PSM	Polynôme de sous séquences d'un motif x noté $\mathbb{S}_x$
PSB	Polynôme de sous séquences d'une base de données noté $\mathbb{S}_D$
AMS	Automate à multiplicité de sous séquences $\mathscr{S}_D$
PPM	Polynôme préfixiel d'un motif x noté $\mathbb{P}_x$
PPB	Polynôme préfixiel d'une base de données $\mathbb{P}_D$
AMP	Automate à multiplicité préfixiel $\mathscr{P}_D$
Q	L'ensemble des états d'un automate
C_{wa}	Coût nécessaire pour l'extension du motif w par l'item a
$det(\mathscr{S}_D)$	Automate déterministe de $\mathscr{S}_D$
BFS	Breadth-First Search
DFS	Depht-First Search
BFD	Breadth-First Determinisation
DFD	Depht-First Determinisation
FPTree	Frequent Pattern Tree
FPGrowth	Frequent Pattern Growth : un algorithme de fouille de motifs fréquents
Eclat	Equivalent CLass Transformation : un algorithme de fouille de motifs fréquents
dEclat	diffset Eclat : une variante de l'algorithme Eclat
WAFI	Weighted Automata Frequent Itemset mining algorithm

Première partie

Prologue

Introduction générale

> Si vous n'êtes pas sûr de l'endroit où vous voulez aller, vous risquez de vous retrouver ailleurs… et de ne pas le savoir !
>
> Robert Frank Mager

Sommaire

Contexte et motivations

La fouille de données (Frawley et al., 1991), ou l'extraction de connaissance à partir des données, est un domaine pluridisciplinaire très en vogue pendant maintenant plus de deux décennies, où concourent des spécialistes de : l'algorithmique, l'apprentissage automatique, les mathématiques, les bases de données, etc. Motivé par l'accumulation faramineuse des volumes de données et l'impérieuse nécessité de les valoriser, la fouille de données émargea le début des années quatre-vingt dix comme étant un processus non trivial d'extraction à partir de gros volumes de données de l'information valide, compréhensible, préalablement inconnue et potentiellement utile pour l'utilisateur (Fayyad et al., 1996). C'est un domaine de recherche très fertile qui a suscité l'intérêt des scientifiques, industriels, commerciaux, et de l'ensembles des acteurs de différents champs et organisations au vu de ses impacts scientifiques et socio-économiques importants.

En 2005, les auteurs de (Yang and Wu, 2006) s'engagèrent dans une initiative afin d'identifier les dix problèmes ouverts et les plus *challengeants* [1] dans la recherche en fouille de données. En consultant un groupe de chercheurs éminents du domaine, leur objectif fut de tracer les axes de recherche les plus intéressants dans ce domaine bien réputé. Une question avérée s'émergea alors : le développement d'une théorie unificatrice pour la fouille de données. En effet, la quasi-totalité des participants à ce sondage considérèrent que la plupart des recherches dans le domaine furent si ad hoc, et que les techniques et outils introduits concernent des tâches ponctuelles et des problèmes particuliers en l'absence regrettable de cadres théoriques unificateurs.

En prenant le papier (Yang and Wu, 2006) comme point de départ, cette thèse s'inscrit donc dans le domaine de la théorie de la fouille de données. Nous y tentons de contribuer à la définition de modèles unificateurs pour ce champs de recherche. Elle traite une question philosophique ancienne ; sans, pour autant, se figer dans l'abstrait. Nous développons progressivement un effort de réflexion, réduction puis synthèse de travaux dans une des problématiques de base et la plus connue en fouille de données. Mais avant d'aller plus loin, qu'est-ce que l'unification et pourquoi faire ?

L'unification est l'action d'unifier ou son résultat ; autrement dit, rassembler des

1. Ce qualificatif n'existe pas en français. Toutefois, il peut être remplacé par : excitant, stimulant ou encore motivant.

choses différentes pour en faire un tout cohérent. De cet angle, elle est synonyme d'harmonisation (Jean-Pierre, 1997). Autrefois, l'unification constitua une question philosophique ; puis, nous la retrouvons dans plusieurs autres disciplines scientifiques : mathématiques, physiques, science de la matière, logique, et aussi en informatique mais avec un faible degré (Hoare, 1995).

À titre d'exemples, Henri Poincaré, le matheux célèbre, nota : *« La mathématique est l'art de donner le même nom à des choses différentes. Il convient que ces choses différentes soient semblables, qu'elles puissent se couler dans le même moule»* (Poincaré, 1908). En Physique, nombreux, dont les réputés Isaac Newton, James Clerk Maxwell, et Albert Einstein, qui se sont intéressés à la diversité des phénomènes observés dans l'univers, *«Il ont suggéré que cette variété cachait peut être une unicité sous-jacente, une simplicité, une universalité, ou un concept central qui serait à la base de tout...»*. (Cliche, 2012)

Alors, pour remédier à la complexité des sciences et les détails de ses méthodes, on les divisa en plusieurs branches, qui à leur tours sont scindées en spécialités séparées afin de maîtriser les questions étudiées. Cependant, ce morcellement de la science a provoqué, hélas, des difficultés pour les chercheurs à faire le lien entre leurs travaux et ceux des autres, et conséquemment ignorer l'intégration des résultats réalisés sur chaque domaine pour en tirer des conclusions communes afférentes à plusieurs d'entre eux. Car, chaque spécialité, effectuant une projection de la réalité suivant ce que permet ses outils, prise seule perd une part importante de l'objet étudié (Franck, 1999; Cliche, 2012).

Au fil des progrès et avec les avancés réalisées, il a été trouvé, dans plusieurs domaines, qu'une grande partie des lois qui les régissent ne sont que des cas spéciaux d'autre théories, que ces dernières sont englobées par d'autres théories encore plus générales. L'objet final de cette constante tendance vers des théories unificatrices dans les sciences est la découverte d'interprétation claire et convaincante sur le fonctionnement global de l'univers (Hoare, 1995).

En usant de l'analogie de réunir les disciplines scientifiques séparées, faire concilier les points de vues des travaux d'un domaine en les intégrant constitue un pas important dans l'approfondissent de la question étudiée. Ainsi, le philosophe Henri Bergson écrivit : *« ..., l'on enseignera que l'intelligence est essentiellement unification, que toutes ses opérations ont pour objet commun d'introduire une certaine unité dans la diversité des phénomènes...»* (Bergson, 1907).

Par ailleurs, un autre volet non négligeable nous a inspiré à aborder cette question : l'enseignement de la fouille de données. En effet, il est bien connu, sur le plan pédagogique, qu'installer d'abord les fondements d'un problème en utilisant ou forgeant, si nécessaire, des modèles solides permettrait certes son assimilation facile. Ainsi, dans notre contexte, il est très difficile, voire anti-pédagogique, d'inculquer les algorithmes de fouille de données sans fondation forte et précise ; et que faire accabler les étudiants par des détails ennuyeux d'implémentation et d'algorithmique serait assurément futile. Entreprendre, par contre, en adoptant un modèle de haut niveau contribuera significativement à clarifier non seulement les différents aspects du problème en question, mais également les rapports avec les sujets connexes.

L'élaboration de modèles unificateurs est une question bien établie en fouille de données (Yang and Wu, 2006). Cependant, l'unification n'est pas une mission simple en général et en informatique non plus (Hoare, 1995). Nous postulons que l'unification peut être simplifiée si nous nous concentrons sur une tâche déterminée de fouille de données. Car, il serait absurde de pouvoir rapprocher des tâches de fouille ayant des objectifs divergents. Dans notre travail le choix est porté, commençons par le commencement, sur une des tâches primaires en extraction de connaissance : l'analyse d'association, plus particulièrement la découverte de motifs fréquents.

Algorithmique d'extraction de motifs fréquents

L'extraction des règles d'association premièrement introduite dans (Agrawal et al., 1993) est un problème populaire et commun en fouille de données (Fayyad et al., 1996), avec un large étendu d'applications telles que l'analyse du panier de la ménagère, la classification, la segmentation, le Web mining et une variété d'autres tâches dont l'objectif est la découverte de corrélations et d'associations. Traditionnellement, ce problème est décomposé en deux principaux sous-problèmes. Premièrement, l'énumération de l'ensemble de motifs fréquents, *i.e.*, les motifs dont la co-occurrence dans l'ensemble de données dépasse un seuil minimal de support fourni comme paramètre. Puis, la génération parmi ceux-ci des règles d'association intéressantes. En dépit de sa simplicité, la première phase, c'est-à-dire le calcul de la collection de motifs fréquents, constitue un problème complexe qui a été assez bien traité durant les deux dernières décennies.

Depuis l'introduction de l'algorithme Apriori (Agrawal and Srikant, 1994), une

panoplie d'autres algorithmes ont été proposés pour résoudre le problème de l'extraction de motifs fréquents. Sans prétendre à l'exhaustivité, nous pouvons catégoriser ces travaux en trois grandes classes : (*i*) l'énumération totale de l'ensemble de motifs fréquents (*ii*) l'énumération abrégée (*iii*) l'énumération incrémentale. Pour un approfondissement veuillez se référer à (Hipp et al., 2000a; Goethals and Zaki, 2003; Goethals, 2003; Han et al., 2007; Borgelt, 2012; Aggarwal et al., 2014; Fournier-Viger et al., 2017)

La première classe d'algorithmes visent l'extraction de la totalité des motifs fréquents. Les stratégies d'exploration de l'espace du problème peuvent être communément distinguées par la méthode de parcours et celle de calcul des supports des motifs (Hipp et al., 2000a).

Les techniques par niveaux adoptent un parcours en largeur, où un k-motif est dérivé en étendant un autre de longueur k-1 (Agrawal and Srikant, 1994). Le calcul du support d'un motif est réalisé par scan de la base de données. Dans ces méthodes, l'apport le plus significatif est, incontestablement, la fameuse heuristique-Apriori, qui stipule que tout sous-motif d'un motif fréquent est aussi fréquent, et duallement, tout sur-motif d'un motif infréquent doit être infréquent. Cette observation a permis un élagage considérable de l'espace du problème, et a été par conséquent largement utilisée dans tous les algorithmes ultérieurs. Toutefois, les techniques par niveaux souffrent de deux inconvénients majeurs : la génération d'un nombre excessif de candidats (motifs potentiellement fréquents), et un coût important en terme d'opérations d'entrée/sortie nécessaires pour les calculs des supports.

Dans (Zaki, 2000), l'auteur considère la base de données d'un point de vue vertical. Il associe à chaque motif X la liste de transactions qui le couvrent (transactions contenant le motif) dite sa tidlist, et utilise l'intersection ensembliste entre ces listes comme mécanisme de calcul de support ; une astuce qui a montré son efficacité. Néanmoins, et malgré la relation d'équivalence fondée sur des préfixes communs exploitée pour décomposer l'espace du problème, cette approche exige, en particulier dans le cas de bases denses, des temps et espaces intermédiaires larges pour réaliser les intersections. Une version améliorée qui cible cette dernière limitation a été introduite dans (Zaki and Gouda, 2003).

Afin de réduire la taille de la base de données et passer outre ses multiple scans, Han et al. ont introduit l'algorithme FPGrowth (Han et al., 2000) qui exploite une structure de données compacte appelée FPTree (Frequent-Pattern Tree) ; cette der-

nière structure est un arbre de préfixes (ou trie) augmenté par les supports des motifs. FPGrowth génère récursivement des projections conditionnelles de la base, pour lesquelles des FPTrees correspondants sont aussi construits. En adoptant la stratégie diviser-pour-résoudre, cet algorithme a présenté une idée excellente dans ce contexte offrant ainsi des gains remarquables. Cependant, la version originale de cet algorithme induit, malheureusement, des surcoûts abondants en temps et espace, dûs essentiellement aux opérations répétées de tri et de reconstructions des FPtrees intermédiaires.

Les algorithmes de la deuxième classe se focalisent sur l'extraction de représentations condensées, dites bases, pour les motifs fréquents qui permettent de les dériver tous. C'est ainsi que les concepts de motifs fréquents maximaux (Bayardo, 1998) et fréquents fermés (Pasquier et al., 1998) ont été introduits.

La préoccupation des algorithmes de la troisième classe est l'incrémentalité. C'est-à-dire, comment générer la collection des motifs fréquents, et surtout la maintenir dans le cas des ensembles de données dynamiques (Cheung et al., 1996; Valtchev et al., 2002; Veloso et al., 2002).

Contribution

En récapitulation, après plus de deux décennies de recherche active sur le sujet, menant à de nombreuses techniques incluant des algorithmes efficaces variés et des structures de données judicieuses ayant chacune ses avantages et inconvénients, nous somme persuadés qu'il serait convenable de revenir en arrière et poser une question : outres les modèles connus (Barbut and Monjardet, 1970; Davey and Priestley, 1990; Godin et al., 1995; Zaki and Ogihara, 1998), existe-t-il d'autres formalismes pour ce problème de base ?

Nous souhaitons investiguer une nouvelle tentative de modélisation, et cela afin de surpasser la simple production de nouveaux algorithmes à une appréhension plus profonde de cette tâche. Autrement dit, nous visons le développement d'un modèle générique et unifié capable d'exprimer les travaux antérieurs dans les principales approches de l'état de l'art. Par ailleurs, nous ambitionnons que le formalisme proposé puisse jouir de certains caractéristiques essentielles telles que : la complétude tout en demeurant simple et intuitif, l'extensibilité, et bien évidement l'efficacité.

Pour ce faire, nous introduisons un nouveau modèle pour l'énumération de tous

les motifs fréquents basé sur les séries formelles qui intègre les propriétés convoitées sus-mentionnées. Le modèle proposé définit un cadre théorique unifié offrant ainsi un fondement solide pour les multiple conclusions sur la parité des algorithmes de fouille de motifs fréquents comme cela a été statué, par exemple, dans (Goethals and Zaki, 2003) et confirmé dans l'une des premières études d'évaluation comparatives sur le sujet (Hipp et al., 2000a).

De plus, nous avons constaté que dans la quai-totalité des travaux antérieurs sur le problème, une nette séparation est toujours observée entre la découverte de la collection des motifs fréquents (aspect qualitatif du problème) et le calcul des supports correspondants (aspect quantitatif du problème). À notre modeste connaissance, hormis les approches fondées sur l'algorithme FPGrowth (Han et al., 2000), qui utilise une astuce élégante en étendant, par les supports, la fameuse structure arbre de préfixes, notre modèle (Oulad-Naoui et al., 2015a; Oulad-Naoui et al., 2015b) est le premier qui intègre de façon naturelle ces deux aspects interreliés du problème. Nous présageons, en outre, que notre modèle permet la généralisation de ces algorithmes à la fouille de motifs plus élaborés tels que : les séquences, les arbres, les graphes, etc.

Nous prouvons également un schéma naturel de décomposition, souvent nécessaire dans plusieurs facettes du problème telles que celle de l'incrémentalité et/ou de parallélisation/distribution. Nous expliquons comment ce problème peut être réduit a celui de la réalisation d'une série formelle par un automate à multiplicité (Salomaa et al., 1978), et par conséquent à celui de la reconnaissance de mots, qui est un sujet largement investi avec une algorithmique mature.

Nous proposons, finalement, une transcription de ce problème à un algorithme de déterminisation d'automates à multiplicité particuliers, que nous avons baptisé WAFI (pour Weighted Automata Frequent Itemset mining algorithm) qui énumère tous les motifs fréquents en place sans mémoire additionnelle. Ce dernier résultat est la conséquence d'un lien adéquat que nous avons découvert, au passage, entre la déterminisation d'automates à multiplicité acycliques particuliers représentant un ensemble de données et l'extraction des motifs fréquents associés qui s'avèrent deux problèmes équivalents (Oulad-Naoui and Cherroun, 2015; Oulad-Naoui et al., 2017).

Organisation

Cette thèse est divisée en trois principales parties : la première constitue son prologue et inclut, outre cette introduction I, un chapitre de préliminaires 1 dans lequel nous rappelons brièvement les concepts clés servant à notre travail. Dans la deuxièmement partie, nous dressons un état de l'art sur le sujet à travers deux chapitres. Le chapitre 2 fait le tour sur la problématique et s'étale sur ses nombreuses applications ; nous y présentons également les outils libres qui prennent en charge la fouille de motifs. Le chapitre 3 comprend une taxonomie assez complète des principaux travaux de la littérature. La troisième partie se focalise, quant à elle, sur notre contribution composée de deux chapitres. La construction de notre formalisme, ses propriétés, et les analyses y afférentes sont discutées dans le chapitre 4. Le chapitre 5 est réservé à l'implémentation et l'évaluation du modèle proposé. L'épilogue comprend une conclusion IV et des pistes ouvrant la voie à des extensions plausibles, ainsi qu'une annexe dans laquelle nous avons choisi de reporter de longues preuves.

CHAPITRE 1

Préliminaires

Pour découvrir quelque chose, il faut, en beaucoup de cas, un talent particulier, celui de savoir comment il faut faire pour bien chercher. Le jugement préliminaire qui fait partie de ce talent, est un don qui fait pressentir où la vérité pourrait bien se rencontrer...

Emmanuel Kant

Sommaire

Nous survolerons dans ce chapitre les concepts de base qui ont servi à l'élaboration de notre formalisme. Nous revenons sur quelque notions élémentaires de la théorie des ensembles, les relations, les applications, les treillis, ainsi que les structures algébriques. Les graphes constituent un formalisme et une structure de données omniprésente dans divers domaines et travaux. Aussi, une section est réservée aux graphes et les différents aspects liés à leur implémentations et parcours. Nous abordons enfin les séries formelles et les automates, où nous nous focalisons plus particulièrement sur les versions pondérées. Le lecteur avide est invité à consulter, pour de plus amples informations, les références fondamentales sur ces sujets (Barbut and Monjardet, 1970; Ramis et al., 1977; Salomaa et al., 1978; Bondy, 1976; Mehta and Sahni, 2004; Berstel and Reutenauer, 1988; Frécon, 2002; Caspard et al., 2007).

1.1 Relations et applications

Un *ensemble* fini ou non [1] est un regroupement en une seule entité d'objets variés appelés ses membres ou éléments. Si un objet e est un élément d'un ensemble E, on dit que e appartient à E ou inclut dans E, qu'on note $e \in E$. Dans le cas contraire, e n'appartient pas à E, noté $e \notin E$. Un ensemble est généralement défini par deux méthodes : soit par extension en énumérant ses éléments, ou par la donnée d'un prédicat (critère) que doivent vérifier ses éléments. Le nombre d'éléments d'un ensemble E est son cardinal noté $|E|$. L'ensemble ayant un cardinal nul est l'ensemble vide noté $\emptyset$ ou $\{\}$.

Deux ensembles sont égaux s'ils ont la même extension, sinon ils sont dits différents. Un ensemble A est un sous-ensemble de (ou inclut dans) B si et seulement si (ssi) tout élément de A est élément de B qu'on note $A \subseteq B$. L'ensemble puissance [2] de A est l'ensemble de tous ses sous-ensembles, appelé aussi l'ensemble des parties de A. Son cardinal est égal à $2^{|A|}$. Les opérations courantes sur les ensembles sont définies ci-après : l'union($\cup$), l'intersection($\cap$), la différence($\setminus$) et le complément par rapport à un ensemble référentiel U ($\complement_U$). Si l'intersection de deux ensembles est

1. Dans cette thèse, nous traitons exclusivement des ensembles finis
2. The powerset, en Anglais.

vide, ils sont qualifiés de *disjoints*.

$$\begin{aligned} A \cup B &= \{x \mid x \in A \vee x \in B\} \\ A \cap B &= \{x \mid x \in A \wedge x \in B\} \\ A \setminus B &= \{x \mid x \in A \wedge x \notin B\} \\ \complement_U A &= U \setminus A \end{aligned}$$

Le *produit cartésien* de deux ensembles A et B est l'ensemble formé de leurs couples ordonnés : $A \times B = \{(x,y) \mid x \in A \wedge y \in B\}$. Une *relation binaire* R de A à B est une partie de $A \times B$. Si $A = B$, la relation est en plus dite *interne*. Si le coupe (x,y) vérifie la relation, on écrit $R(x,y)$ ou xRy, où x est l'*origine* et y son *image*. Une relation peut être représentée de différentes façons : en listant ses couples, par une matrice binaire dite la matrice caractéristique de la relation, ou encore par un graphe (un arc lie les éléments des couples de R, qui représentent les sommets du graphe). Une relation R binaire interne sur un ensemble E est :

Réflexive	:	$\forall x \in E$	$R(x,x)$
Symétrique	:	$\forall x,y \in E$	$R(x,y) \Rightarrow R(y,x)$
Antisymétrique	:	$\forall x,y \in E$	$R(x,y) \wedge R(y,x) \Rightarrow x = y$
Transitive	:	$\forall x,y,z \in E$	$R(x,y) \wedge R(y,z) \Rightarrow R(x,z)$

Pour deux relations $R \subseteq A \times B$ et $S \subseteq B \times C$, on appelle la relation $T \subseteq A \times C$ leur *composée* qui se note $T = RoS = \{(a,c) \in A \times C \mid \exists b \in B : R(a,b) \wedge S(b,c)\}$. Une relation binaire interne R sur un ensemble E est une *équivalence* ssi elle est réflexive, symétrique et transitive. Une relation d'équivalence sur un ensemble induit une partition de celui-ci en classes d'équivalence. La *classe d'équivalence* d'un élément e de E par R est l'ensemble des éléments de E dont e est l'image.

Une relation binaire R sur un ensemble E est un *ordre*, noté $\leq$, ssi elle est réflexive, antisymétrique et transitive. Si pour tout couple $(x,y) \in E : x \leq y \vee y \leq x$, l'ordre est dit *total*, dans le cas contraire il est qualifié de *partiel*. Pour une relation R entre E et F de graphe G, une *correspondance* est le triplet (E,F,G), où E est *l'ensemble de départ*, F *l'ensemble d'arrivée*. Une *application* f de E dans F est une correspondance (E,F,G) telle que : $\forall x \in E \; \exists! \; y \in F \mid (x,y) \in G$.

Une application est :

— *Surjective* si tout élément de l'ensemble d'arrivée a au moins un antécédent,

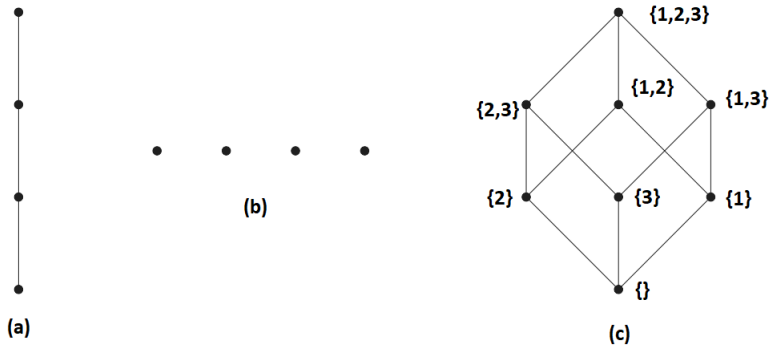


FIGURE 1.1 – Diagrammes de Hasse pour (a) une chaîne (b) une antichaîne (c) le treillis des sous-ensembles $(\{1,2,3\},\subseteq)$

— *Injective* si à deux éléments distincts correspondent toujours deux images distinctes.

— *Bijective* si elle est à la fois surjective et injective

1.2 Ensembles ordonnés et treillis

Un *ensemble ordonné* est un couple $(E,\leq)$ avec E un ensemble non vide et $\leq$ un ordre sur E. Pour deux éléments x et y de E, on dit que x est inférieur à (ou couvert par) y si $x \leq y$ et il n'existe pas un $z \in E$ tel que $x \leq z \leq y$. Si $x \leq y$ ou $y \leq x$, les deux éléments sont dits *comparables*. Dans le cas contraire, ils sont *incomparables*. Une *chaîne* est un ensemble ordonné dans lequel deux éléments quelconque sont toujours comparables. De même, une *antichaîne* est un ensemble ordonné dans lequel deux éléments (distincts) sont toujours incomparables.

Tout ensemble ordonné peut être représenté graphiquement par son diagramme de couverture ou *diagramme de Hasse*. Dans ce diagramme, les éléments de E sont les sommets, et une ligne joint (en montant) x à y si $x \leq y$. Des exemples de diagrammes de Hasse d'une chaîne, une anti-chaîne, et des sous-ensembles de $\{1,2,3\}$ ordonnés par l'inclusion ensembliste sont illustrés dans la figure 1.1.

Un élément m (resp. M) d'un ensemble ordonné $(E,\leq)$ est le *minimum* (resp. le *maximum*) de E si $\forall x \in E : m \leq x$ (resp. $\forall x \in E : x \leq M$). Pour une partie S de l'ensemble ordonné $(E,\leq)$, un élément m de E (resp. M) est un *minorant* (resp. *majorant*) de S si $\forall x \in S : m \leq x$ (resp. $\forall x \in S : x \leq M$). De même, b sera *l'infimum* ou *la borne inférieure* (resp. *le supremum* ou *la borne supérieure*) si b est le maximum

des minorants (resp. le minimum des majorants) de S. Un *treillis* est un ensemble ordonné tel que chaque couple (x,y) d'éléments possède un supremum (noté $x \vee y$) et un infimum (noté $x \wedge y$).

$(E,\leq)$ et $(F,<)$ sont deux ensembles ordonnés, une application $f : E \to F$ est dite *croissante* ou *isoton e* (resp. *décroissante* ou *antitone*) ssi :

$$\begin{array}{lll} \forall (x,y) \in E^2 & : & x \leq y \Rightarrow f(x) < f(y) \\ (resp.)\ \forall (x,y) \in E^2 & : & x \leq y \Rightarrow f(x) > f(y) \\ & & (> \text{ étant la relation opposée de } <) \end{array}$$

Dans un ensemble ordonné $(E,\leq)$, une application $h : E \to E$ est un *opérateur de fermeture* si elle est : isotone, extensive et idempotente. C'est-à-dire, une applications ayant les trois propriétés suivantes :

$$\begin{array}{lllll} \text{Isotone} & : & \forall x \forall y & x \geq y \Rightarrow h(x) \geq h(y) \\ \text{Extensive} & : & \forall x & h(x) \geq x \\ \text{Idempotente} & : & \forall x & h(h(x)) = h(x) \end{array}$$

Un élément x de E est dit *fermé* s'il est égale à sa fermeture ($x = h(x)$). Une *correspondance (ou connexion) de Galois* entre deux ensembles ordonnés E et F est un couple (f,g) où f est une application antitone de E dans F, g une application antitone de F dans E et les composés fog et gof sont extensives. Il est aisé de constater que dans le cas d'une connexion de Galois entre les parties de E et celles de F, l'application gof est un opérateur de fermeture dans E, et que fog est un opérateur de fermeture dans F.

1.3 Structures algébriques

Un *alphabet* est un ensemble fini non vide de symboles ou lettres. Un *mot* sur un alphabet A est une suite finie de symboles de A. Le *mode vide* est celui qui ne contient aucun symbole, noté ε. L'ensemble de tous les mots sur un alphabet A sera noté A^*.

La *concaténation* des mots $u = u_1 \ldots u_n$ et $v = v_1 \ldots v_m$ est le mot $uv = u_1 \ldots u_n v_1 \ldots v_m$, formé en juxtaposant les symboles de u suivis de ceux de v. Dans cette dernière structure, $|w|$ dénote la longueur du mot w. ε est l'unique mot avec une longueur nulle.

Un mot u est un *préfixe* (resp. *suffixe*) d'un autre mot w s'il existe un mot v tel que $w = uv$ (resp. $w = vu$). L'ensemble des préfixes d'un mot u sera noté $\text{Pref}(u)$. Ce dernier concept peut être facilement étendu à un ensemble de mots en formant l'union des préfixes des mots de cet ensemble, i.e, $\text{Pref}(U) = \cup_{u_i \in U} \text{Pref}(u_i)$.
Un mot $u = u_1 \dots u_k$ est une *sous-séquence* d'un mot $w = w_1 \dots w_l$ $(k \leq l)$ s'il existe des mots $v_1, \dots, v_{k+1}$ tel que $w = v_1 u_1 v_2 u_2 \dots v_k u_k v_{k+1}$. Nous écrivons donc $u \preccurlyeq w$.

Un ensemble $\mathbb{M}$ muni d'une opération binaire interne et associative $*$, admettant un élément neutre unique $e \in \mathbb{M}$ forme une structure de *monoïde*, noté $(\mathbb{M}, *, e)$. Lorsque l'opération $*$ est commutative, le monoïde l'est aussi. L'exemple célèbre est le monoïde libre A^* de l'ensemble de mots formés à partir d'un alphabet A équipé de l'opération de concaténation ayant le mot vide ε comme élément neutre.

Un *semi-anneau* est un quintuplet $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ tel que : $(\mathbb{K}, \oplus, \overline{0})$ est un monoïde commutatif, $(\mathbb{K}, \otimes, \overline{1})$ un monoïde, $\otimes$ est distributive dans les deux sens sur $\oplus$, et $\overline{0}$ est un élément absorbant par rapport à $\otimes$. Des exemples courants de semi-anneaux sont : $(\mathbb{N}, +, \times, 0, 1)$ des entiers positifs dit aussi semi-anneau de comptage, $(\mathbb{B}, \vee, \wedge, \bot, \top)$ des booléens, et le semi-anneau tropical $(\mathbb{N} \cup \{\infty\}, min, +, \infty, 0)$. Si $(\mathbb{K}, \oplus, 0)$ et $(\mathbb{K}, \otimes, 1)$ sont seulement des monoïdes sans les autres conditions, la structure $(\mathbb{K}, \oplus, \otimes, 0, 1)$ est appelée un *bi-monoïde*.

1.4 Graphes

Dans sa forme la plus simple, un graphe est un ensemble de points reliés par des lignes. Formellement, un *graphe non orienté* G est une paire (S, A), où S est un ensemble de *sommets* (ou *nœuds*) et A un ensemble de couples $a = (x, y)$, partie de $S \times S$ appelés *arêtes*. Pour une arête $a(x, y)$, x et y sont dits *adjacents* ; de même, a et x (ou y) sont qualifiés d'*incidents*. Deux arêtes sont *adjacentes* si elles sont incidentes à un même sommet.

Une *chaîne* est une suite de sommets $(v_0, v_1 \dots v_n)$ où toute paire de sommets successifs sont adjacents. Un *cycle* est une chaîne fermée, *i.e*, dont les extrémités (v_0 et v_n) coïncident. Si un graphe ne possède aucun cycle, il est qualifié d'*acyclique*. Il est dit *connexe*, si toute paire de sommets est connectée par une chaîne. Un *arbre* est un graphe connexe et acyclique.

Un graphe $G'(S', A')$ est un *sous graphe* de $G(S, A)$ si $S' \subseteq S \wedge A' \subseteq A$. Un graphe est *complet* si tous ses sommets sont adjacents les uns avec les autres. Un sous graphe

complet est dit une *clique*. Un graphe est *biparti* si ses sommets peuvent être partitionnés en deux ensembles S_1 et S_2, telles que toute arête relie un sommet de S_1 à un autre de S_2. Lorsque les paires des arêtes d'un graphe sont ordonnées, le graphe est dit *orienté* où les notions : arête, chaîne et cycle deviennent *arc*, *chemin* et *circuit* respectivement.

Un graphe est *étiqueté* si à ses arêtes sont associées des étiquettes quelconques souvent représentées par des lettres, des symboles ou des mots, ou encore des quantités. Le dernier cas correspond aux graphes *pondérés*. Par convention, le nombre de sommets $|S|$ d'un graphe est noté n et celui des arêtes/arcs $|A|$ par m.

Il existe plusieurs représentations pour les graphes (orienté ou non), qui sont facilement extensibles aux cas étiquetés ; les plus communes sont deux. Par *matrice d'adjacence*, un graphe G est modélisé par une matrice carrée $A[n \times n]$ où la case $A[i, j]$ est à 1 si une arête existe dans G entre i et j, sinon la case contient 0. Cette représentation exige $O(n^2)$ espace, mais un temps constant de recherche de l'existence d'une arête reliant deux sommets. La *liste d'adjacence* consiste, quant à elle, en un tableau de n listes chaînées. Chaque liste i contient les sommets adjacents au sommet numéro i. Cette dernière représentation requière un espace en $O(n+m)$, et un temps de réponse aux requêtes de connectivité en $O(n)$. Le choix d'une représentation est lié à la nature du graphe (dense ou creux), et aussi au traitement envisagé.

Pour répondre à différents traitements concernant un graphe, un parcours est souvent nécessaire permettant de visiter ses nœuds/arêtes. L'exploration d'un graphe peut essentiellement être effectuée selon deux stratégies : en largeur, ou en profondeur.

Le parcours en *largeur (Breadth-First Search (BFS))* consiste, en étant au sommet v, de visiter toutes les arêtes incidentes à v, pour ensuite passer à un sommet adjacent w, à partir duquel nous traitons toutes les arêtes incidentes à w. Ce processus est continué jusqu'à ce que tous les sommets accessibles de v sont traités.

Dans le parcours en *profondeur (Depth-First Search (DFS))*, la stratégie est de ne pas scanner toutes les arêtes incidentes à v, mais plutôt de passer le plutôt possible à un sommet adjacent w non encore visité, laissant ainsi des arêtes non explorées. Ce processus est suivi de manière récursive avec retour arrière, jusqu'à achèvement du parcours. La philosophie du parcours en profondeur est donc de dresser un chemin le plus profond possible dans le graphe avant de passer à un autre possible. Les deux modes d'exploration précédents exigent un temps de l'ordre de $O(n+m)$.

1.5 Séries formelles

Sur le monoïde A^*, nous définissons une *série formelle* $\mathbb{S}$ à coefficients (ou multiplicité) dans un semi-anneau $\mathbb{K}$ comme une application $\mathbb{S} : A^* \to \mathbb{K}$, qui associe à chaque mot w son *coefficient* dans la série noté $\langle \mathbb{S}, w \rangle$. La série $\mathbb{S}$ elle même sera notée comme une somme :

$$\mathbb{S} = \sum_{w \in A^*} \langle \mathbb{S}, w \rangle w \tag{1.1}$$

Grosso modo, une série formelle peut être considérée comme une fonction d'un ensemble d'éléments dans un autre ensemble, plus précisément une structure algébrique : un semi-anneau. Cette fonction attache à chaque élément de l'ensemble de départ une valeur prise du semi-anneau. Par exemple, si l'ensemble de départ est l'ensemble de mots sur un alphabet A, une série formelle généralise la notion de langage formel (Hopcroft et al., 2001) en fournissant en plus une information de quantification pour les mots. En effet, les langages formels classiques sont considérés comme des cas particuliers des séries formelles au vu que leur domaines de quantification sont l'ensemble des booléens $\mathbb{B}$: on peut dire qu'un mot appartient à un langage quand son coefficient est 1 ; sinon, son efficient est nul et le mot n'appartient pas au langage en question.

Dans une série formelle, nous mesurons par un coefficient l'appartenance d'un élément donné à un ensemble. La valeur assignée par une série formelle à un élément peut modéliser un poids, une multiplicité, un coût comme le temps ou la distance ou tout autre quantification ou estimation associée à l'élément dans la série. Dans l' équation 1.1, w dénote un élément et $\langle \mathbb{S}, w \rangle$ symbolise son coefficient ou image par la série $\mathbb{S}$; $\langle \mathbb{S}, w \rangle w$ est, quant à lui, un terme de la série.

Afin d'illustrer ces notions, considérons l'exemple suivant. Supposons la donnée d'un ensemble $R \subset A^*$ de produits achetés d'un magasin dans une période déterminée matérialisée par des estampilles temporelles :

$$R = \{0.ab, 1.a, 2.b, 3.ab, 4.b, 5.ab, 6.ab, 7.c, 8.ab, 9.ab, 10.a\}$$

Associons, de plus, à chaque produit x sa fréquence (nombres d'occurrences) dans l'ensemble R, que nous symbolisons par $\text{Freq}(x)$. Évidemment, et sans entrer dans les détails, Freq définie bien une série formelle de l'ensemble R dans le semi-

anneau des entiers naturels positifs $\mathbb{N}$. Nous pouvons observé aisément que :

$$\langle \text{Freq}, a\rangle = 2, \quad \langle \text{Freq}, ab\rangle = 6, \quad \langle \text{Freq}, b\rangle = 2, \quad \langle \text{Freq}, c\rangle = 1, \quad \langle \text{Freq}, d\rangle = 0$$

La série Freq peut être écrite alors :

$$\text{Freq} = 2a + 6ab + 2b + 1c + 0d \tag{1.2}$$

L'ensemble des séries formelles sur A à coefficients dans $\mathbb{K}$ est noté $\mathbb{K}\langle\langle A\rangle\rangle$. La somme et le produit de deux série formelles sont donnés ci-dessous. Ceci nous conduit à voir qu'une structure d'un semi-anneau est définie sur $\mathbb{K}\langle\langle A\rangle\rangle$: soient $\mathbb{S}$ et $\mathbb{T}$ deux séries formelles sur A à coefficients dans $\mathbb{K}$, et $k \in \mathbb{K}$; nous avons :

$$\langle \mathbb{S} + \mathbb{T}, w\rangle = \langle \mathbb{S}, w\rangle + \langle \mathbb{T}, w\rangle \tag{1.3}$$

$$\langle \mathbb{S}\,\mathbb{T}, w\rangle = \sum_{uv=w} \langle \mathbb{S}, u\rangle\langle \mathbb{T}, v\rangle \tag{1.4}$$

$$\langle k\mathbb{S}, w\rangle = k\langle \mathbb{S}, w\rangle \tag{1.5}$$

$$\langle \mathbb{S}k, w\rangle = \langle \mathbb{S}, w\rangle k \tag{1.6}$$

Pour une série formelle $\mathbb{S}$, l'ensemble des mots ayant des coefficients non nuls est appelé son *étendue*. Mentionnons que le terme commun pour ce dernier ensemble est *support* ; cependant, nous avons préféré étendue (Pin, 1998) [3] afin d'éviter la confusion avec la même appellation utilisée dans le contexte de la fouille de motifs fréquents :

$$\text{Étendue}(\mathbb{S}) = \{w \in A^* |\ \langle \mathbb{S}, w\rangle \neq 0\} \tag{1.7}$$

Si l'étendue d'une série $\mathbb{S}$ est fini, elle est appelée polynôme. Dans l'exemple de l'équation 1.2, la série Freq est un polynôme, et l'ensemble $\{a, ab, b, c\}$ constitue son étendue.

L'ensemble des polynômes sur A à coefficient dans $\mathbb{K}$ est noté $\mathbb{K}\langle A\rangle$. Afin d'alléger la notation, nous dénotons simplement par k, au lieu $k\varepsilon$, le terme correspondant à l'élément neutre ε. De la même manière, pour un mot w de A^*, nous noterons par kw

3. J.E Pin utilise le terme anglais *range* duquel nous avons tiré la traduction étendue.

le terme dont le coefficient pour w est k. Si k est égal à l'élément unité, nous écrivons le terme correspondant dans la série directement w à la place de $1w$.

Soit $\mathbb{S} \in \mathbb{K}\langle\langle A \rangle\rangle$ une série formelle sur A à coefficients dans $\mathbb{K}$, et $u \in A^*$. Nous définissons la dérivée à gauche de la série $\mathbb{S}$ par rapport à u, notée $u^{-1}\mathbb{S}$ dans $\mathbb{K}\langle\langle A \rangle\rangle$ par :

$$\langle u^{-1}\mathbb{S}, v\rangle = \langle \mathbb{S}, uv\rangle \tag{1.8}$$

Similairement, la dérivée à droite de la série $\mathbb{S}$ par rapport à u, notée $\mathbb{S}u^{-1}$ est donnée ci-après :

$$\langle \mathbb{S}u^{-1}, v\rangle = \langle \mathbb{S}, vu\rangle \tag{1.9}$$

Nous donnons ci-dessous quelque propriétés des opérations de dérivation. $\mathbb{S}$ et $\mathbb{T}$ étant deux séries formelles sur A à coefficients dans $\mathbb{K}$ et $k \in \mathbb{K}$:

$$\begin{aligned} u^{-1}(\mathbb{S}+\mathbb{T}) &= u^{-1}\mathbb{S}+u^{-1}\mathbb{T} && (1.10)\\ u^{-1}(k\mathbb{S}) &= k(u^{-1}\mathbb{S}) = u^{-1}(\mathbb{S}k) = (u^{-1}\mathbb{S})k && (1.11)\\ (uv)^{-1}\mathbb{S} &= v^{-1}(u^{-1}\mathbb{S}) && (1.12) \end{aligned}$$

1.6 Automates à multiplicité

Comme les série formelles généralisent les langages formels, les automates à multiplicités (ou pondérés) généralisent les automates classiques. Dans un automate à multiplicité, nous introduisons des quantifications (ou poids) sur les transitions entre les états, et autorisons également chaque état de posséder une multiplicité d'entrée et une autre de sortie. Historiquement, les automates à multiplicité ont été largement utilisés pour modéliser de multiple systèmes dans diverses applications. Quoique les poids d'un automate à multiplicité peuvent a priori être arbitraires, leur nature (le semi-anneau de travail) est généralement dictée par le domaine d'application.

Formellement, un *automate à multiplicité* $\mathscr{A}$ sur un alphabet A à coefficients dans un semi-anneau $\mathbb{K}$ est un tuple $\mathscr{A} = (Q, A, \mu, \lambda, \gamma)$, où Q est un ensemble fini d'états, μ une fonction de Q dans $\mathbb{K}$ des poids initiaux ou d'entrée, λ une fonction de $Q \times A \times Q$ dans $\mathbb{K}$ des poids des transitions, et γ l'application de Q dans $\mathbb{K}$ des poids finaux ou de sortie. La *taille* d'un automate $\mathscr{A}$, notée $|\mathscr{A}|$, est le nombre de ses états.

Nous schématisons les automates à multiplicité de façon similaire que la représentation des automates classiques mettons en évidence les multiplicités sur les états/transitions. La figure 1.2 illustre un automate à multiplicité possible (parmi plusieurs) qui calcule le polynôme de la formule 1.2. Notons que nous avons omis les poids initiaux nuls des états 1 à 4 pour des raisons de clarté.

Un chemin c dans un automate à multiplicité $\mathscr{A}$ est une succession de transitions : $(q_0,a_1,q_1),\ldots,(q_{n-1},a_n,q_n)$ étiqueté par le mot $a_1\ldots a_n$ obtenu par concaténation de symboles des transitions qui le constituent ; son poids $\omega(c)$ est le produit (en général, l'opération $\otimes$ du semi-anneau) des poids de ses transitions, en incluant le poids d'entrée de l'état initial et celui de sortie de l'état final :

$$\omega(c) = \mu(q_0)\lambda(q_0,a_1,q_1)\ldots\lambda(q_{n-1},a_n,q_n)\gamma(q_n) \tag{1.13}$$

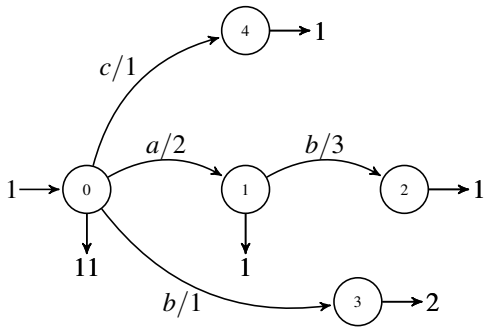


FIGURE 1.2 – Un automate à multiplicité reconnaissant le polynôme Freq de l'équation 1.2

Dans la figure 1.2, $(0,a,1),(1,b,2)$ est un chemin, son étiquette est ab et $(1\times 2\times 3\times 1=6)$ est son poids.

Posons $\mathscr{C}(u)$ l'ensemble des chemins étiquetés u, le poids d'un mot u dans un automate $\mathscr{A}$, noté par $\mathscr{A}(u)$, est la somme (en général, l'opération $\oplus$ du semi-anneau) des poids des éléments de $\mathscr{C}(u)$:

$$\mathscr{A}(u) = \sum_{c\in\mathscr{C}(u)} \omega(c) \tag{1.14}$$

Soient $\mathscr{A} = (Q_A,A,\mu_A,\lambda_A,\gamma_A)$ et $\mathscr{B} = (Q_B,A,\mu_B,\lambda_B,\gamma_B)$ deux automates à multiplicité. Une application h de Q_A dans Q_B est un *homomorphisme d'automates à*

multiplicité si $\forall q \in Q_A$, elle garantit :

$$\begin{cases} h(\mu_A(q)) & = & \mu_B(h(q)) \\ h(\lambda_A(q,a,q\prime)) & = & \lambda_B(h(q),a,h(q\prime)) \\ h(\gamma_A(q)) & = & \gamma_B(h(q)) \end{cases} \tag{1.15}$$

L'extension d'une application h à un ensemble est définie en prenant l'union des images, par h, des éléments de l'ensemble considéré. Autrement dit :

$$h(Q) = \bigcup_{q \in Q} h(q) \tag{1.16}$$

Si h est, en plus, bijective, elle est dite un *isomorphisme d'automates à multiplicité*.

Deuxième partie

État de l’art

CHAPITRE 2

Fouille de motifs fréquents : problématique, applications et outils

Il n'y a pas de problèmes résolus, il y a seulement des problèmes plus ou moins résolus.

Henri Poincaré

Sommaire

Loin de toute exagération, la fouille de motifs est le sujet numéro un dans la sphère de recherche sur la fouille de données. Ce constat est fondé aussi bien sur des aspects historiques que biblio-métriques. D'une part, l'initiateur papier d'Agrawal, Imielinski, et Swami présenté en 1993 et intitulé « *Mining Association Rules between Sets of Items in Large Database* » (Agrawal et al., 1993) est, en fait, considéré comme un des œuvres fondateurs du domaine. Dans ce papier, les auteurs introduisirent pour la première fois la notion des règles d'association qui empocha vite un intérêt grandissant. D'autre part, l'examen du nombre de papiers consacrés au sujet (Giacometti et al., 2013) conduit aisément à reconnaître l'ampleur des efforts et contributions, non seulement fondamentaux mais aussi applicatifs, fournis par la communauté fouille de données à ce problème basique [1].

Le présent chapitre à pour objet d'introduire le problème de la fouille de motifs fréquents. Nous y passons par les nombreuses applications de ce problème et décrivons également de multiples outils et environnements libres qui le prennent en charge.

2.1 Présentation du problème

Nous présentons ici les notions de base relatives au problème de la fouille de motifs comme initialement introduit dans (Agrawal et al., 1993). Nous discutons sa complexité temporelle et spatiale, et terminons par mentionner quelque une de ses applications typiques et outils libres y afférents.

2.1.1 Définitions

Historiquement, le problème de la fouille de motifs fréquents a été proposé dans le cadre de l'analyse du panier de la ménagère (*Market-Basket Analysis*), dont le but fut de rechercher la collection de produits corrélés souvent achetés ensemble, en examinant les tickets de caisse enregistrés par les supermarchés. Cette connaissance des co-occurrences existantes entre produits est ensuite exploitée pour générer

1. Les contributions majeures peuvent être trouvées dans les éminentes conférences : AAAI, SIGMOD, VLDB, SIGKDD, etc.

ce qui est appelé les *règles d'association* qui sont des formes d'implications entre ensembles de produits. La mise en évidence de ces produits reliés sert dans divers usages comme les campagnes de promotion, l'organisation des rayonnages, etc.

Bien qu'élémentaire, ce problème a connu un succès fulgurant et a conséquemment envahi plusieurs autre tâches d'extraction de connaissances telles que la classification, la segmentation et le Web mining pour ne citer que les plus remarquables. Ci-après la formulation de ce problème.

Support et motif fréquent

Étant donné un ensemble fini non vide $A = \{a_1, a_2, \ldots, a_m\}$ de m éléments communément appelés *items*. Ces items peuvent désigner, comme déjà mentionné, des articles commandés d'un supermarché, des pages Web visités en surfant, ou, en général, une collection de variables, d'attributs, ou d'évènements.

Un *motif*[2] ou encore un *itemset* I est un sous-ensemble de A. Il est qualifié de k-motif lorsque son cardinal est égal à k. Dans le contexte du modèle du panier de la ménagère, un k-motif symbolise donc l'ensemble des produits commandés par un client lors de ses courses.

De même, une *transaction* t_i est un sous-ensemble non vide de A. Elle est identifiée par son identificateur unique i. Un *ensemble (ou une base) de données* D est un ensemble de n transactions, qu'on note comme un multiset $D = \{t_1, t_2, \ldots, t_n\}$. Notons que dans ce problème, le nombre n de transactions est généralement supposé très large et que la taille (longueur) d'une transaction est nettement inférieur au nombre total d'items m.

Dans une base de données D, l'ensemble de transactions qui contiennent un motif x est appelé sa *couverture*. Le *support* d'un motif x, dénoté $\text{sprt}(x, D)$, est le nombre de transactions qui contiennent x, i.e., le cardinal de sa couverture.

$$\text{sprt}(x, D)_{abs} = |\{t_k \in D \mid x \subseteq t_k\}| \tag{2.1}$$

Précisons que l'équation 2.1 donne la valeur absolue du support (un entier positif inférieur ou égale à la taille de la base de données), qui peut aussi être exprimée en relatif, c-à-d, par un réel compris entre zéro et un, représentant un pourcentage en le divisant par la taille de l'ensemble de données comme cela est montré dans la

2. Nous avons opté pour le terme motif car il est plus général que itemset.

formule ci-dessous.

$$\text{sprt}(x,D)_{rel} = \frac{|\{t_k \in D \mid x \subseteq t_k\}|}{|D|} \tag{2.2}$$

Malgré qu'historiquement la définition relative du support (une fraction entre 0 et 1) a été retenue dans les premiers travaux (Agrawal et al., 1993; Agrawal and Srikant, 1994; Savasere et al., 1995; Park et al., 1995a), nous adoptons dans la suite de cette thèse, en revanche, la valeur absolue pour faire référence au support d'un motif. Afin d'alléger l'écriture, nous le notons simplement sprt(x) sans la sous-mention *abs* ni référence à une base de transactions qui seront compris du contexte. Ceci est actuellement une tendance largement répondue dans la communauté fouille de données (Bodon, 2006; Borgelt, 2012; Zaki and Wagner Meira, 2014; Leskovec et al., 2014; Fournier-Viger et al., 2017).

Définition 2.1.1 (Motif fréquent)**.** *Un motif x est fréquent*[3] *si son support dépasse un seuil minimal s spécifié d'avance. Formellement, x est fréquent dans D relativement au seuil s ssi :* $\text{sprt}(x) \geq s$

Le problème de la fouille de motifs consiste à énumérer l'ensemble $\mathscr{F}_D(s)$ de tous les motifs fréquents dérivés de A présents dans la base de transactions D au regard du seuil minimal du support s.

$$\mathscr{F}_D(s) = \{x \subseteq A \mid \text{sprt}(x) \geq s\} \tag{2.3}$$

Pour alléger la notation, ce dernier ensemble sera, désormais, noté $\mathscr{F}$ tout court.

Règle d'association et confiance

Définition 2.1.2 (Règle d'association)**.** *Une règle d'association est une implication de la forme $X \Rightarrow Y$, où X et Y sont des motifs non vides et disjoints.*

Une règle d'association exprime la liaison entre l'ensembles d'items de l'antécédent X et ceux du conséquent Y. De manière similaire et vu le nombre prohibitif de règles pouvant être générées, il est généralement admis d'opérer ici aussi un filtrage de façon à ne retenir que les règles *intéressantes*. Par conséquent, outre l'application du seuil de support minimal à toute règle d'association produite[4], une

3. Le papier originel d'Agrawal et al. utilise le terme *large*
4. Le seuil concerne les motifs constituant la règle

deuxième condition de *confiance* est aussi exigée. La confiance d'une règle exprime la proportion des transactions de la base contenant Y parmi celles qui contiennent X. Elle traduit donc la probabilité conditionnelle des transactions contenant Y sachant quelles contiennent X. Cette dernière contrainte permet de retirer les règles qui ne satisfont pas le seuil minimal de confiance c également fixé a priori par l'utilisateur ou l'expert du domaine. Les règles qui vérifient à la fois les deux seuils support et confiance sont parfois qualifiées de *fortes*. Les règles d'association sont écrites alors en spécifiant leurs importances :

$$X \Rightarrow Y[\text{sprt} = vs, \text{conf} = vc] \tag{2.4}$$

Où vs et vc sont les valeurs calculées pour le support et la confiance de la règle respectivement. Ci-dessous les deux formules permettant de mesurer l'intérêt d'une règle, comme définies dans l'énoncé original du problème (Agrawal and Srikant, 1994; Zaki and Ogihara, 1998) :

$$\text{sprt}(\text{X} \Rightarrow \text{Y}) = \text{sprt}(\text{X} \cup \text{Y}) \tag{2.5}$$

$$\text{conf}(\text{X} \Rightarrow \text{Y}) = \frac{\text{sprt}(\text{X} \cup \text{Y})}{\text{sprt}(\text{X})} \tag{2.6}$$

La tâche d'extraction des règles d'association consiste, en considérant les deux paramètres s et c, à générer l'ensemble des règles fortes de la base. Afin d'éviter des temps de calcul importants, et comme nous pouvons le constater de la formule précédente, une règle ne peut être intéressante si elle ne satisfait pas la condition du support ; cela nous permet de filtrer l'ensemble de ces règles avant même d'entamer leurs examens. C'est ainsi que ce problème est souvent effectué, pour fins d'optimisation, en deux phases (Agrawal et al., 1993) :

— L'énumération d'abord des motifs fréquents satisfont le seuil de support s,

— La formation par la suite, parmi ces derniers, des règles intéressantes justifiant le seuil de confiance c.

Il est facile de remarquer que la phase difficile et gourmande en temps et espace est la première, car elle exige l'exploration de l'espace du problème afin de découvrir les motifs fréquents. En effet, une fois cette dernière achevée, la deuxième est relativement aisée vu que sa mission se résume à dériver de la précédente l'ensemble

des règles possibles en se limitant à celles fortes.

Il est à souligner que l'approche d'extraction des règles d'association basée sur le découpage en deux sous-problèmes mentionnés en haut a été adoptée par la quasi-totalité des travaux qui se sont intéressés au sujet à l'exception de l'algorithme *MagnumOpus* (Webb, 2000) qui génère les règles d'association directement sans passer par les motifs fréquents. Il utilise un algorithme de recherche de type *branch and bound* introduit par G.I. Webb (Webb, 1995). *MagnumOpus* est toutefois limité aux règles ayant un seul item en partie droite de la règle.

Ci-après, nous donnons, dans le pseudo-algorithme 1, une démarche permettant la génération des règles d'association d'une base de transactions (Zaki and Wagner Meira, 2014). Le principe est simple : en scrutant l'ensemble des motifs fréquents trouvés dans la première étape, itérer en séparant à chaque fois un motif en deux parties disjointes. Celles-ci vont produire une règle d'association après la vérification du seuil de confiance.

Algorithme 1 GÉNÉRATION DES RÈGLES D'ASSOCIATION

Entrée : L'ensemble des motifs fréquents $\mathscr{F}$
Sortie : Les règles d'association fortes

1. Soit $\mathscr{F}$ l'ensemble de motifs fréquents trouvés dans la phase une,
2. Itérer en considérant à chaque étape un motif $I \in \mathscr{F}$ avec $|I| \geq 2$
   (a) Choisir X parmi les sous motifs propres de I. Soit Y son complément à I, *i.e.*, $Y = I \setminus X$. (Pour bénéficier de l'optimisation en (b), il est préférable de commencer par les sous motifs ayant les cardinalités maximales)
   (b) Former la règle $X \Rightarrow Y$ et calculer sa confiance, puis l'éditer si la condition de confiance est remplie. Autrement, la négliger et ignorer également tous les sous motifs X' de X dans la suite des itérations (les règles qui en résultent ne peuvent être fortes car $\text{sprt}(\text{X}') \geq \text{sprt}(\text{X})$)
   (c) S'il existe des sous motifs propres non traités : aller à (a)
3. Si $\mathscr{F}$ non épuisé : aller à (2)

Afin d'illustrer les concepts introduits, nous donnons, dans le tableau 2.1, un exemple d'un ensemble de données de dix transactions. Cet ensemble de données servira comme exemple de référence dans la suite du chapitre.

En retenant 3 et 1 comme seuils minimaux de support et de confiance respectivement, nous pouvons remarquer sans difficulté dans l'exemple que :

— Tous les items formant les cinq singletons sont fréquents avec les supports

TABLE 2.1 – Un exemple d'une base de 10 transactions sur l'ensemble d'items $A = \{a,b,c,d,e\}$

i	t_i	i	t_i
1	{a,c,d,e}	6	{a,c,d,e}
2	{a,c,d}	7	{a,b,c,d}
3	{a,b,c,d}	8	{a,b,e}
4	{a,b,e}	9	{a,c,d}
5	{a,c,e}	10	{a,c,d,e}

suivants : $\{(\{a\},10),(\{b\},4),(\{c\},8),(\{d\},7),(\{e\},6)\}$,

— Le motif $\{a,c,e\}$ est fréquent (support = 4). Par contre, $\{b,d\}$ ne l'est pas (support = 2),

— En se référant aux formules 2.5 et 2.6, la seule règle forte dérivable du motif fréquent $\{a,c,e\}$ est : $\{c,e\} \Rightarrow \{a\}[\text{sprt} = 4, \text{conf} = 1]$

Nous donnons ci-dessous l'ensemble $\mathscr{F}$ tout entier (tous les motifs fréquents) de la base de notre exemple relativement à un seuil minimal de support égal à 3.

$\mathscr{F} = \{(\{a\},10),(\{b\},4),(\{c\},8),(\{d\},7),(\{e\},6),(\{a,b\},4),(\{a,c\},8),$
$(\{a,d\},7),(\{a,e\},6),(\{c,d\},7),(\{c,e\},4),(\{d,e\},3),(\{a,c,d\},6),(\{a,c,e\},4),$
$(\{a,d,e\},3),(\{c,d,e\},3),(\{a,c,d,e\},3)\}$

2.1.2 Espace d'états et complexité

La fouille de motifs est un problème complexe qui exige un calcul énorme. En effet, pour un ensemble d'items A, nous devons explorer tous les motifs possibles. L'ensemble puissance de A constitue donc l'espace de recherche de ce problème, qui compte donc $2^{|A|} - 1$ sous motifs candidats (pouvant être fréquents) en excluant le motif vide qui ne véhicule aucune information utile. L'espace du problème est clairement exponentiel avec le nombre d'items, et par conséquent *«intractable»* (Papadimitriou, 1994) même avec une collection faible d'items.

Quant à la complexité temporelle asymptotique, elle dépend en général du nombre de motifs candidats possibles multiplié par le temps nécessaire au calcul du support pour chaque instante. Ainsi, il est évident qu'une solution naïve à ce problème, montrée dans l'Algorithme 2 coûte $O(|D||A|2^{|A|})$, car le calcul du support

d'un motif exige de le comparer avec chacune des transactions de la base, *i.e.*, un temps en $O(|D||A|)$ au pire des cas. Ceci est clairement inefficace voire irréalisable, compte tenu du nombre d'items considérés dans les applications réelles souvent très abondant. Bien entendu, la complexité en pratique est tributaire à plusieurs autres aspects. Outre les caractéristiques des compilateurs et des machines, elle est aussi fonction, en autres, de la nature de la base de transactions (sparse [5] ou dense), de la valeur du seuil du support dont les valeurs trop faibles induisent alors des temps de calcul très prohibitifs, et enfin des structures de données utilisées offrant des gains sur plus d'un titre tels que dans la génération des candidats et le nombre d'entrées/sorties.

Algorithme 2 FOUILLE DE MOTIFS : EXPLORATION NAÏVE

Entrée : Une base de transaction D et le support minimal s
Sortie : L'ensemble $\mathscr{F}$ des motifs fréquents

```
F ← ∅
pour tout x ⊆ A faire                    // x un motif candidat
    supp(x) ← 0
    pour tout t ∈ D faire                // t une transaction de la base
        si x ⊆ t alors
            supp(x)++
        fin si
    fin pour
    si supp(x) ≥ s alors
        F ← F ∪ {(x, supp(x))}
    fin si
fin pour
retourner F
```

Le nombre de règle d'association possibles générées pour un ensemble d'items A est, quant à lui, aussi exponentiel ce qui requière l'élaboration de techniques de génération non triviales. Le théorème suivant donne ce nombre.

Théorème 2.1.1 (Nombre de règles d'association (Tan et al., 2005))**.** *Le nombre de règles d'association possibles dérivables à partir d'un ensemble d'item A est* $3^{|A|} - 2^{|A|+1} + 1$.

Démonstration. En utilisant les combinaisons et le principe de multiplication en combinatoire (Guichard, 2017), il suffit de considérer tous les cas de choix de k items parmi $|A| - 1$ pour l'antécédent et i item parmi $|A| - k$ pour le conséquent. Il y

5. Terme emprunté de l'Anglais, qui signifie creuse ou dispersée

a $\sum_{k=1}^{|A|-1}\left(\binom{|A|}{k}\sum_{i=1}^{|A|-k}\binom{|A|-k}{i}\right)$ possibilités, que nous pouvons développer comme suit. Sachant que : $\sum_{k=0}^{n}\binom{n}{k}=2^n$; $\sum_{k=0}^{n}\binom{n}{k}a^k b^{n-k}=(a+b)^n$; $\binom{n}{0}=\binom{n}{n}=1$

$$\begin{aligned}
\sum_{k=1}^{|A|-1}\left(\binom{|A|}{k}\sum_{i=1}^{|A|-k}\binom{|A|-k}{i}\right) &= \sum_{k=1}^{|A|-1}\left(\binom{|A|}{k}\left(\sum_{i=0}^{|A|-k}\binom{|A|-k}{i}-1\right)\right)\\
&= \sum_{k=1}^{|A|-1}\left(\binom{|A|}{k}\left(2^{|A|-k}-1\right)\right)\\
&= \sum_{k=1}^{|A|-1}\left(\binom{|A|}{k}2^{|A|-k}-\binom{|A|}{k}\right)\\
&= \sum_{k=1}^{|A|-1}\binom{|A|}{k}2^{|A|-k}-\sum_{k=1}^{|A|-1}\binom{|A|}{k}\\
&= \sum_{k=0}^{|A|}\binom{|A|}{k}2^{|A|-k}-2^{|A|}-1-\left(\sum_{k=0}^{|A|}\binom{|A|}{k}-2\right)\\
&= 3^{|A|}-2^{|A|}-1-\left(2^{|A|}-2\right)\\
&= 3^{|A|}-2^{|A|+1}+1
\end{aligned}$$

□

Dans le chapitre suivant, nous étudions quelques optimisations communes introduites afin d'améliorer le processus de fouille de motifs fréquents, et donnons des bornes de complexité plus fines pour ce problème. Néanmoins, signalons d'emblée que l'espace de recherche peut être représenté par un treillis complet $(2^A,\subseteq)$ de l'ensemble 2^A muni de l'ordre partiel induit par la relation d'inclusion ensembliste. Le diagramme de Hasse de cet espace dans le cas d'un ensemble de cinq items $A=\{a,b,c,d,e\}$ est illustré dans la figure 2.1, où les nœuds sont les motifs et les arrêtes correspondent aux relations d'inclusion.

2.2 Applications de la fouille de motifs fréquents

Le framework fouille de motifs est un modèle fondamental que nous pouvons trouver dans plusieurs applications dans des domaines variés. En effet, ce problème générique repose sur une relation de type plusieurs-à-plusieurs entre deux

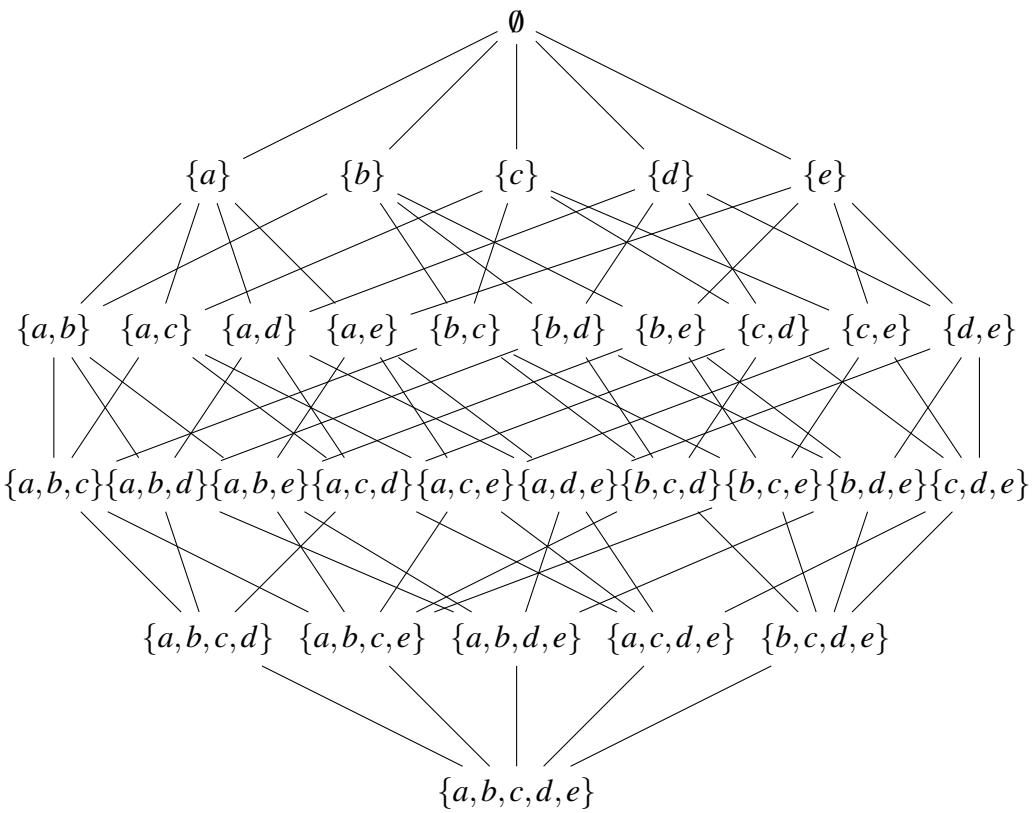


FIGURE 2.1 – Espace de recherche dans la fouille de motifs comme un treillis (inversé) $(2^A, \subseteq)$ des sous ensembles de $A = \{a,b,c,d,e\}$

ensembles : des objets (transactions) décrits par des propriétés (items) qui peuvent être, a priori, quelconques. C'est pour cette raison que ce modèle est souvent représenté par une matrice à deux dimensions : des objets dans les lignes et des items aux niveaux des colonnes. Nous rapportons dans les points suivants une sélection de ses fameuses applications. Ces dernières peuvent être globalement divisées en deux grandes catégories (Aggarwal et al., 2014) : d'une part des applications à des tâches communes de fouille de données telles que la classification, la segmentation ou encore l'analyse d'association, ou d'exceptions (*outliers*), et d'autre part celles qui concernent des applications génériques (Han et al., 2007; Leskovec et al., 2014; Fournier-Viger et al., 2017).

Signalons, pour lever tout équivoque, que cette section et la suivante font référence prématurément à quelques fameux algorithmes de fouille de motifs fréquents à l'instar d'Apriori ou FPGrowth, dont la présentation complète peut être trouvée dans le prochain chapitre.

2.2.1 Analyse d'association

L'analyse du panier de la ménagère ou des habitudes de consommation des clients est l'application archétypique de la fouille de motifs, et c'est d'ailleurs dans cette optique que s'ébaucha les premiers travaux dans le sujet au débuts des années 90 (Agrawal et al., 1993). L'objectif préliminaire est de dégager l'ensemble des produits les plus courants. Le dénouement de cette analyse est la découverte des associations du genre si un client achète l'ensemble de produits A, il a (ou aura) aussi acheté la collection de produits B avec une certaine certitude. Bien entendu, la finalité est surtout de révéler non pas des associations triviales et évidentes telles que le lien lait-pain par exemple, mais plutôt des associations nouvelles et préalablement inconnues à l'instar du célèbre et étonnant rapport bière-couches pour bébés « *beer and diapers* ». La maîtrise de ces associations est utile dans : l'assortiment de produits, l'organisation des rayonnages des magasins, la bonne gestion de la relation avec la clientèle, les campagnes publicitaires, etc.

2.2.2 Classification

La classification est une tâche d'apprentissage supervisé qui consiste en l'affectation, selon leurs propriétés, d'une collection d'objets à un ensemble de classes prédéfinies. Un des premiers travaux combinant la classification à l'analyse d'association est l'algorithme CBA (Classification Based on Association) (Liu et al., 1998). Les auteurs présentent l'idée de restreindre les règles extraites à seulement un sous ensemble réduit dit règles d'association de classes permettant de construire un classifieur «associatif». Ces règles sont celles ayant la classe cible en partie droite de l'implication. Les résultats obtenus exhibent des performances supérieures à des systèmes de classification célèbres à l'instar de C4.5 (Wu et al., 2008).

Dans le cadre de données à hautes-dimension, l'approche bien connue de combinaison de caractéristiques est largement utilisée. Toutefois, cette dernière présente l'inconvénient d'un coté d'être extrêmement coûteuse en terme de calcul vu le nombre exponentiel de combinaisons possibles ; et d'un autre, sa sensibilité à l'incorporation de combinaisons rares de caractéristiques qui se voient détériorer la qualité de la classification. Par conséquent, des recherches ont soulignés l'intérêt de l'exploitation des motifs fréquents comme une technique de classification. C'est ainsi que (Cheng et al., 2007) ont introduit une approche de classification ba-

sée sur les motifs fréquents combinée à une méthode de sélection d'attributs (le gain d'information). L'objectif de cette association de techniques est d'améliorer le pouvoir prédictif d'un motif fréquent ; l'intuition simpliste dernière en est qu'un motif, outre d'être fréquent, représente une forme de combinaison non linéaire de caractéristiques individuelles et véhicule donc une part importante de la sémantique des données. Dans cette approche, l'espace de redescription inclut, outre les caractéristiques individuelles, les motifs fréquents trouvés selon un seuil de support choisi. Les auteurs prouvent que l'approche est efficace aussi bien en terme de précision que du passage à l'échelle, et proposent dans leur algorithme MMRFS une stratégie pour le choix approprié de la valeur minimale du support offrant une meilleure qualité de la classification. Ils établissent que cette dernière est excellente avec une valeur optimale liée à un seuil en gain d'information qu'ils déterminent.

Notons enfin que les motifs fréquents comme moyen de classification est un framework général appliqué à divers types de données. Par exemple, il a été étendu depuis plusieurs années à la classification d'images (Fernando et al., 2012).

2.2.3 Segmentation

Un autre usage important de la fouille de motifs dans l'extraction de connaissances se manifeste dans la segmentation (*clustering*). Cette dernière relève de l'apprentissage non supervisé, puisqu'elle n'exige aucune autre information que les données en question et leurs descriptions. La segmentation vise à explorer les données à la recherche d'ensembles d'objets homogènes dits groupes, segments, catégories, ou encore *clusters*. Elle est fondée sur l'emploi de mesures de similarité via des fonctions de distance entre les données à segmenter qui sont considérées comme des points de l'espace de leurs caractéristiques. Afin d'assurer une segmentation valide, nous veuillons à maximiser ces mesures entre les objets du même groupe (homogénéité interne) et de les minimiser entre ceux appartenant à des groupes différents (séparation externe) (Jain and Dubes, 1988).

Les techniques de segmentation peuvent être répertoriées en trois grandes classes (Xu and II, 2005) : hiérarchiques, partitionnistes, et par voisinage dense. Dans (Zimek and Vreeken, 2015), les auteurs soulignent le lien entre la fouille de motifs et la segmentation : *«Whereas in clustering research we are after identifying sets of objects that are highly similar, in pattern mining we are typically after discovering recurring structures. While at first glance these goals and the accompanying*

problem definitions seem quite different, in the following we will argue that they in fact have quite a lot in common and that researchers in the two fields can learn much from each other.» De même, ils élucident dans (Zimek et al., 2014) comment le modèle de la fouille de motifs a été étendu pour la découverte de clusters locaux dans le contexte de données à hautes dimensions. Dans ce dernier cas, la segmentation par sous-espaces (*subspace clustering*) constitue une adaptation retenue par la communauté. Son principe repose sur le constat que les clusters ne s'accordent forcement pas sur tous les attributs mais seulement sur une partie d'entre eux, i.e., leur similarité peut être restreinte à seulement un sous ensemble des attributs, les autres étant considérés comme inappropriés ou inutiles à la structure des clusters (bruités, à valeurs manquantes, ou uniformément distribués, ce qui les rendent nuisibles au processus). La question est comment choisir les sous-espaces adéquats offrant des clusters valides. En fouille de motifs, la même observation est rencontrée du moment que les données peuvent faire l'objet de différentes segmentation selon l'ensemble de caractéristiques retenues. Cette intuition est illustrée dans la figure 2.2. Sans s'at-

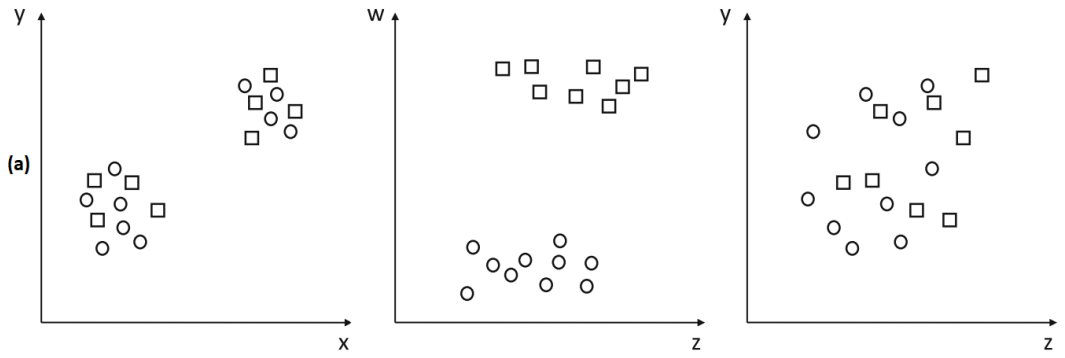


FIGURE 2.2 – (a) Segmentation du même ensemble de données à travers différents sous-espaces en considérant les axes (x,y),(z,w) et (z,y) ; (b) Motifs fréquents en choisissant les items $\{a,c\}$, $\{a,d\}$ et $\{c,d\}$ (Zimek et al., 2014)

tarder sur les détails, le passage segmentation-fouille de motifs considère les items comme des attributs, les motifs en tant que sous-espaces et enfin les motifs fréquents

comme des sous-espaces comprenant des clusters selon un critère de segmentation.

Des exemples de travaux plus anciens sur le sujet incluent : l'algorithme CLIQUE, qui procède par niveaux semblablement à Apriori (Agrawal et al., 1998) ; FIHC, un algorithme de segmentation hiérarchique de documents (Fung et al., 2003), et MineClu (Yiu and Mamoulis, 2003) qui débarque d'une adaptation de l'algorithme de fouille de motifs FPGrowth.

2.2.4 Analyse d'exceptions

Une exception ou une déviation (*outlier* en anglais) est un point de données qui s'écarte de façon significative des instances restantes d'un ensemble de données, au stade où on soupçonne qu'il a été généré par un mécanisme différent[6] (Aggarwal, 2013). L'analyse d'outliers revêt une importance capitale en fouille de données, car elle contribue à la détection d'anomalies ou de conditions d'erreur dans les données. En fait, la détection de ces points aberrants est une condition sine qua non pour une analyse cohérente et valide vu que leur incorporation engendra des estimations biaisées des paramètres observés qui conduira à des résultats trompeurs. C'est la raison pour laquelle ces points sont vus, en fouille de données, comme atypiques ou porteurs de bruits et conséquemment sont filtrés ou, du moins, isolés afin soit de réduire leur impact ou pour une investigation supplémentaire (Ben-Gal, 2010). Néanmoins, il est de coutumes, à contrario, de cibler ces points dans certaines applications telles que la détection de fraude, d'intrusions dans les réseaux ou le diagnostic médical.

Outre la détection visuelle des outliers qui est évidente mais limitée, les approches d'analyse des outliers sont généralement catégorisées en quatre méthodes (Han et al., 2011) : *(i)* Les méthodes statistiques, où les données sont supposées obéir à une distribution de probabilité ; les outliers sont donc repérés par un test de discordance. *(ii)* Les méthodes à base de distances DB (Distance Based) qui emploient une métrique de distance et le principe du faible voisinage. Dans ces méthodes et pour un ensemble de données D un objet o est qualifié de $DB(pct, dmin)$-outlier si au moins $pct\%$ des objets de la base se situent à une distance supérieure à $dmin$ de o. *(iii)* Les méthodes basées sur la densité locale qui améliorent les précédentes dans certaines configurations où des mesures globales échouent. Ici, il n'est plus question de raisonner en binaire (c-à-d outlier ou non), mais plutôt de quan-

6. Cette définition éloquente est attribuée au statisticien Douglas Michael Hawkins de l'université de Minnesota (USA) dans son livre : *Identification of Outliers*, Chapman and Hall, 1980.

tifier le degré d'exception que représente un objet par rapport à un voisinage local. *(iv)* Les méthodes fondées sur les déviations n'emploient ni des tests statistiques ni des distances. Un outlier est reconnu comme tel s'il n'est pas conforme aux normes de certaines caractéristiques du groupe établies à l'avance. Notons, en fin, que la détection d'outliers peut aussi être assurée simplement en opérant une segmentation des données. Les outliers sont donc les points n'appartenant à aucun cluster ; ou de même, les clusters de faible taille (y compris ceux contenant un seul point) (Jiang et al., 2001; Yu et al., 2002).

Nombreux sont les travaux proposant et expérimentant l'adaptation du paradigme fouille de motifs dans l'identification d'outliers (Said et al., 2013). L'idée intuitive derrière ces recherches consiste à considérer une donnée comme normale ou commune pour un ensemble de données si elle inclut suffisamment de motifs fréquents de ceci. Dans le cas contraire, i.e., lorsque elle possède des motifs rares, elle est un bon candidat d'être un outlier. L'algorithme FindFPOF (He et al., 2005) est un exemple de cette famille de méthodes. Il associe à chaque transaction un facteur d'outlier (FPOF : pour Frequent Pattern Outlier Factor), comme mesure de base. Cette dernière est la proportion de la somme des supports relatifs des motifs fréquents présents dans une transaction au cardinal de l'ensemble complet de motifs fréquents. Dans (Otey et al., 2006), les auteurs présentent un algorithme distribué pour la détection d'outliers dans le contexte de données à attributs mixtes (catégoriels et continus), qui se sert des motifs fréquents dans le score des attributs catégoriels, et la covariance pour ceux continus. Il jouit également d'une extension pour le traitement de données évolutives ou en flots. Afin d'éviter de calculer la totalité des motifs fréquents, les auteurs de (Lin et al., 2010) se contentent des motifs fréquents maximaux (présentés dans le chapitre suivant) pour le développement d'un système de détection d'outliers qu'ils utilisent pour l'analyse des flots de séries temporelles à haute dimensions.

2.2.5 Bioinformatique

Les travaux (Atluri et al., 2009; Naulaerts et al., 2015) présentent quelque applications typiques du paradigme fouille de motifs fréquents et règles d'association dans l'analyse de données biologiques. Grosso modo, il s'agit d'identifier des motifs pertinents pouvant prétendre à une interprétation dans un cadre biologique. Ces applications incluent : l'annotation de données biologiques, dans le but de prédire

leurs fonctionnalités, la découverte de motifs structuraux tels que les séquences ou les sous graphes, l'alignements de séquences, etc. Ce type de connaissances s'avère très utile dans plusieurs domaines tels que : la médecine, l'agronomie, et l'industrie. Évidemment, les données biologiques doivent préalablement être traduites sous la forme du modèle transactions/items. Cette transcription n'est pas toujours aisée vue la structure complexe, la nature stochastique, la présence de valeurs manquantes et la taille énorme des données biologiques. A ces difficultés, il faut ajouter la nécessité de définir des mesures d'intérêt spécifiques, prenant en considération les connaissances du domaine.

Quoi que les autres tâches de fouille de données ont été appliquées très tôt en bioinformatique telles que la segmentation (Eisen, 1998; Tamayo, 1999), et la classification (Brown and D., 2000; Kurra et al., 2001), l'analyse d'association n'a trouvée de popularité, dans ce domaine, que plus tard. Des exemples de travaux utilisant l'analyse d'association en bioinformatique incluent (Kotala et al., 2001; Berrar D. and Eils, 2001) et (Ono et al., 1997; Tuzhilin and Adomavicius, 2002).

2.2.6 Recherche d'information et détection de plagiat

La mission des systèmes de recherche d'information (SRI) (Rijsbergen, 1979) est de retrouver l'ensemble de ressources pertinentes à un besoin informationnel d'un utilisateur exprimé souvent par une requête. Ils reposent sur une fonction de correspondance entre un modèle des requêtes et un modèle des ressources (documents, pages web, etc.). Dans cette adaptation du paradigme de la fouille de motifs aux SRI, les termes extraits des documents moyennant des prétraitement courants en recherche d'information (éliminations des mot-vides, lemmatisation, etc.) jouent le rôle des items, alors que les documents sont les transactions. Le principe est de tirer profit du framework des règles d'association pour suggérer, dans le cadre de la technique par expansion de la requête, les termes à injecter dans la requête initiale de l'utilisateur afin d'améliorer la qualité des résultats retournés.

Une première exploitation des règles d'association dans les SRI peut être trouvée dans (Rungsawang et al., 1999), où un système automatique d'expansion de requêtes basé exclusivement sur l'algorithme Apriori est proposé. Les auteurs réalisent par ce système une amélioration de 19% sur la précision. (Martín-Bautista et al., 2004) présentent un système de recherche d'information sur le Web, dont l'architecture est illustrée dan la figure 2.3. Ce système est fondé également sur l'idée de l'exten-

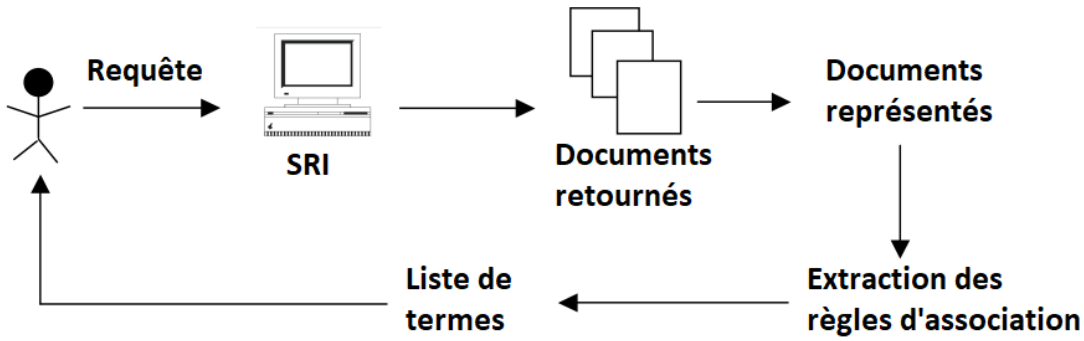


FIGURE 2.3 – Un SRI utilisant l'expansion de requêtes basée sur les règles d'association (Martín-Bautista et al., 2004)

sion de la requête en utilisant les règles d'association combinées à un mécanisme de retour de la pertinence utilisateur. Afin de mieux évaluer l'intérêt des règles exploitées, les auteurs puisent des notions de règles floues ; ils présentent pour cela trois schémas de pondération des termes dans les documents : un schéma booléen (présence/absence du terme dans le document), la fréquence relative, et en fin la fameuse pondération TF-IDF [7]. Dans (Song et al., 2007) le système SemanQE est introduit. Il s'agit d'une technique de recherche consistant en un algorithme hybride d'expansion de la requête combiné à un algorithme d'extraction de règles d'association exploitant une ontologie et des techniques issues du NLP [8].

Le plagiat (Maurer et al., 2006) est un fléau qui va à l'encontre de l'éthique et l'intégrité de la recherche et ses acteurs (étudiants, enseignants chercheurs, établissements, etc.). Il a connu de l'ampleur ces dernières années grâce à la prolifération, la simplicité et le libre accès aux documents numériques et des bases de données ouvertes sur le Web. Les techniques utilisées dans ce domaine sont catégorisées en trois grandes approches : les méthodes à base d'empreintes digitales, d'occurrence de termes, et par analyse de style (Chen et al., 2010). Le lien avec les SRI est donc bel et bien établi, car il est question de repérer des passages dupliqués ou ressourcés de références informationnelles existantes.

Peu de travaux exploitent le framework fouille de motifs dans la détection de plagiat. Toutefois, (Leskovec et al., 2014) la mentionnent parmi les applications de la fouille de motifs dans un chapitre dédié au sujet, mais en inversant cette fois les rôles des items/transactions. Dans le contexte de détection de plagiat, les transactions sont

7. Une pondération connue dans les SRI sigle de : Term Frequency-Inverse Document Frequency
8. Natural Language Processing

les représentations de documents tel que les termes, ou les phrases. Les items sont, quant à eux, les documents à inspecter. Ainsi, si la fouille se restreint à seulement les paires fréquentes par exemple, ceci nous donne les couples de documents qui partagent en commun plusieurs passages, ce qui constitue un signe probable de plagiat. Dans (El-matarawy et al., 2013), les auteurs utilisent, dans un projet de détection de plagiat basé sur Apriori, l'extension de la fouille de motifs à la découverte de séquences fréquentes, qui sont ici les mots et les phrases des documents analysés. Ils visent l'identification d'opérations de copier/coller de textes. Ce projet a montré de bon résultats dans des temps de réponse raisonnables.

2.2.7 Détection de fraudes, intrusions, malwares, et bugs

Fraude, intrusion, malware et bug sont des conditions qui compromettent le fonctionnement normal de systèmes matériels et/ou logiciels, dans le but de rompre la disponibilité d'un environnement, de vol de données privées, ou de sabotage en général. Aussi, différentes techniques de fouille de données dont les motifs fréquents y sont appliquées avec succès (Barbara, 2002). Le principe, comme toute tâche d'extraction de connaissances, est d'abord d'identifier les caractéristiques annotées (bonnes/mauvaises) des systèmes à partir de données d'audit collectées préalablement, puis d'adapter une technique selon la tâche que nous devons accomplir. Dans ce contexte qui relève de la sécurité des systèmes et comme déjà mentionné dans les sous-sections précédentes, une vision soit de classification, de segmentation ou encore d'analyse d'outliers peut être envisagée. Quelque exemples de travaux fondés sur les motifs fréquents pour ce type d'application inclut entre autre : le framework IDS [9] présenté par (Lee and Fan, 2001) pour la détection d'intrusion, PR-Miner de (Li and Zhou, 2005) pour la détection de bugs logiciels, et enfin IMDS [10] développé dans (Ye et al., 2008) qui utilise l'algorithme d'extraction de motifs fréquents FPGrowth.

2.2.8 Web Mining

La fouille du Web (*Web Mining*) (Etzioni, 1996) est l'application des techniques d'extraction de connaissances aux données issues du Web. Il est catégorisé en trois

9. Intrusion Detection System
10. Intelligent Malware Detection System

grandes classes selon le type de données à explorer qu'il s'agisse de données de contenu (*Web Content Mining*), de structure (*Web Structure Mining*) ou d'usage (*Web Usage Mining*) (Kosala and Blockeel, 2000). Le même raisonnement donc a été étendu à la fouille du Web. A titre d'illustration, les motifs fréquents et les règles d'association ont été appliqués avec succès en fouille d'usage du Web (Cooley et al., 1997). Il est question, par exemple, de découvrir les pages souvent accédées ensemble dans les sessions des utilisateurs et les associations intéressantes qui en découlent. Cette connaissance est précieuse dans : la réorganisation et l'adaptation de sites Web, les systèmes de recommandation en e-commerce, et l'implémentation d'heuristiques de pré-récupération de ressources Web, etc.

2.3 Environnements libres de fouille de motifs

La fouille de motifs a conquis une attention capitale depuis son introduction. Ceci s'est clairement manifesté par la production d'un nombre phénoménal d'algorithmes où chacun prétend prendre le devant. En dépit de l'engouement observé, la communauté a enregistré, quand même, des avis divergents sur plusieurs convictions relatives aux comportements de plusieurs algorithmes, voire sur le même algorithme sous différentes configurations. Cet état de fait a incité les chercheurs du domaine à l'organisation de meetings pour débattre et échanger les idées sur cette question. Les workshops FIMI'03 (Goethals and Zaki, 2003) et FIMI'04 (Bayardo et al., 2004) ont été organisés dans cette optique. Les objectifs visés furent d'apporter des réponses et contributions aux multiples défaillances constatées. Parmi ces dernières nous citons, en particulier, le manque d'implémentations publiques, de bases de données réelles, et d'études expérimentales sérieuses. Les organisateurs ont insisté surtout sur la nécessité d'adopter la stratégie open-source.

Dans cette section, nous explorons quelques outils et bibliothèques libres de fouille de données de l'angle de la découverte de motifs, que nous avons partiellement tirés de (Chen et al., 2007; Naulaerts et al., 2015; Altalhi et al., 2017; Fournier-Viger et al., 2017). Nous voulons en profiter pour, en plus, mettre en exergue certain outils généraux de fouille de données peu ou méconnus.

2.3.1 Weka

Weka (*Waikato Environment for Knowledge Analysis*) (Hall et al., 2009) est un environnement open source, multi-plateforme, extensible écrit en Java et développé à l'université de Waikato en New Zealand [11]. Ce projet populaire offre plusieurs algorithmes pour les tâches usuelles d'apprentissage et de fouille de données : classification, segmentation, analyse d'association, régression, et sélection d'attributs, avec une variété d'autres outils de prétraitement et de visualisation de données. Il est exploitable via une interface graphique ou en se servant de la ligne de commande.

Le support de la fouille de motifs est, hélas, limité à seulement trois composants : les deux algorithmes d'extraction de motifs Apriori et FPgrowth, ainsi qu'un filtre d'association permettant différents filtrages de données avant d'attaquer un algorithme d'association. Toutefois, Weka possède plusieurs autres traits intéressants, ce qui lui a permis d'occuper les premiers rangs dans les études d'évaluation et de comparaison à l'instar de celle présentée dans (Altalhi et al., 2017).

2.3.2 SPMF

SPMF (*Sequential Pattern Mining Framework*) est une bibliothèque de fouille de données open source écrite en Java par Philippe Fournier-Viger et ses collaborateurs (Fournier-Viger et al., 2016). Cette suite d'algorithmes est entièrement spécialisée en fouille de motifs ; elle est pour cela considérée la suite la plus riche sur le sujet (Fournier-Viger et al., 2017) [12], car elle inclut 138 implémentations de différents algorithmes de fouille d'itemsets, de séquences, classifications, etc. SPMF est conçue de façon à offrir une intégration simple à d'autres projet de fouille de données. Elle est offerte aussi comme une application indépendante en deux modes : par interface graphique ou via la ligne de commande. Notons que le site de cette bibliothèque inclut plusieurs autres données utiles telles qu'une documentation, des bases de données réelles et synthétiques, des études de performance et une mailing-liste, etc.

11. Disponible sur : http://www.cs.waikato.ac.nz/ml/weka/downloading.html
12. Actuellement, SPMF est à la version v2.20 sortie le 6 Novembre 2017 et disponible sur le site http://www.philippe-fournier-viger.com/spmf/

2.3.3 KNIME

KNIME (*Konstanz Information Miner*) (Berthold et al., 2009) est un environnement interactif d'analyse de données, modulaire en pipeline à travers des composants prédéfinis appelés nœuds, extensible et multi-plateformes basé sur Eclipse. Fondé par Michael Berthold comme nouveau venu du monde exploration de données et bien qu'originellement orienté bioinformatique, il connaît un succès fleurissant aussi bien dans les sphères professionnelles que celles d'enseignement et de recherche. KNIME offre des solutions d'entrée/sortie, de préparation, d'analyse et de visualisation avancées simples et rapides avec une intégration parfaite à d'autres langages et technologies et une variété de sources de données. Compte tenu de ses atouts, il est sélectionné parmi les quatre top outils de fouille de données (Altalhi et al., 2017). Outre l'intégration aux bases de données via des ports dédiés et à des outils connus d'exploration de données tels que Weka et R, KNIME offre deux options pour la fouille de motifs : l'Item Set Finder node et Association Rule Learner implémentant les algorithmes : Apriori, FPGrowth et bien d'autre, et l'Association Rule node qui produit à la fois les motifs fréquents et les règles d'association.

2.3.4 Orange

Orange (Demšar et al., 2013) est un environnement libre livré sous la licence GPL, destiné à l'apprentissage automatique et l'exploration et la visualisation de données. Écrit en Python par le laboratoire de Bioinformatics de l'université de Ljubljana (Slovénie), il adopte la philosophie de la programmation visuelle basée composants. Des widgets sont donc disponibles pouvant être connectés pour créer des workflows qui constituent l'usage standard. Les utilisateurs expérimentés peuvent user sa bibliothèque Python pour différentes sortes de traitements et de modifications de composants. Le support de la fouille de motifs est limité dans Orange à deux variantes de l'algorithme Apriori.

2.3.5 Rattle et R

«*The R Analytical Tool To Learn Easily*» (Williams, 2011) est une interface graphique orientée onglets pour la fouille de données en utilisant le langage de programmation statistique R (R Core Team, 2013). Elle est libre et inclut de multiple fonctionnalités de la préparation, la transformation et le résumé des données à la création

de modèles supervisés ou non (dits paradigmes dans Rattle), et de présentation graphique et évaluation. Une propriété remarquable de cet outil est que l'ensemble des actions opérées en interagissant avec l'interface peuvent être capturées sous la forme d'un script R invoquable alors de façon indépendante de l'outil. Rattle support la fouille de motifs via l'onglet *Associate* du paradigme *Unsupervised*, il implémente l'algorithme Apriori avec les mesures d'intérêt suivants : le support/confiance, le *lift* et le *leverage*. Par ailleurs, notons que le langage R inclut *ARules* (Hahsler et al., 2011) une bibliothèque dédiée complètement à l'extraction de motifs fréquents (tous, fermés et maximaux) et les règles d'association associées.

2.3.6 Tanagra

Tanagra est un logiciel open source écrit en C++ par Ricco Rakotomalala (Rakotomalala, 2005). Fonctionnant exclusivement sous Windows et seulement via une interface graphique, il propose des modules pour l'analyse exploratoire et statistique de données, ainsi que l'apprentissage automatique. Tanagra offre une implémentation de l'algorithme Apriori pour l'extraction de motifs fréquents. Un paramétrage simple permet de limiter la sortie aux motifs fréquents fermés ou maximaux. L'absence de maintenance et de mise à jours depuis décembre 2013, ainsi que l'importabilité et l'inaccessibilité de son site sont des points faibles de cet outil.

2.3.7 Mahout

Mahout est une bibliothèque open source d'apprentissage automatique et de fouille de données développée en Java par la fondation Apache à partir de 2008 comme une partie du projet du moteur de recherche Lucene de la même fondation (Owen et al., 2011). Elle est destinée à des projets énormes impliquant des données très volumineuses dont le traitement est insupportable voire impossible par une seule machine quelle que soit sa puissance. La force de Mahout est attribuée à ses capacités de mise à l'échelle, de parallélisme et de distribution fondées sur le paradigme *MapReduce* de Hadoop (Dean and Ghemawat, 2008). Cette bibliothèque n'offre aucune interface graphique et est utilisable exclusivement par la ligne de commande et exige plusieurs autre outils, qui sont à adapter par les développeurs. Elle a offert initialement des algorithmes de classification, de segmentation et de recommandation. Actuellement, la fouille de motifs y est intégrée à travers PFP (Li

et al., 2008) une implémentation parallèle de l'algorithme FPGrowth.

2.3.8 ELKI

ELKI (*Environment for deveLoping KDD-Applications supported by Index-structures*) est un logiciel de fouille de données open source indépendant de toute plateforme écrit en Java (Achtert et al., 2008), dont la finalité est la recherche en algorithmique. Bien maintenu et à jour, cet outil est spécialisé dans les tâches non supervisées, plus particulièrement l'analyse de clusters et la détection d'outliers. Il est conçu dans l'optique d'être paramétrable et extensible afin de permettre des expérimentations et des évaluations rapides et simples. Outre les fameuses techniques de segmentation et de détection d'outliers, ELKI offre deux algorithmes de classification et les trois algorithmes de fouille de motifs les plus célébrés : Apriori, Eclat et FPGrowth.

2.3.9 Autres outils

Les outils de fouille de données disponibles sont nombreux, où chacun offre des fonctionnalités intéressantes. Les présenter tous, outre que n'est pas notre objectif, est un travail onéreux. Afin d'alléger le document, nous citons ici d'autres environnements qui prennent en charge, avec bien d'autre tâches, la fouille de motifs. Coron [13], Lucs-KDD [14], AdaM [15], Adams [16], DataMelt [17], KEEL [18], PyFIM [19], ML-Flex [20] et RapidMiner [21] sont aussi des outils libres de fouille de données, ils offrent plusieurs algorithmes d'extraction de motifs tels que Apriori, Eclat, Pascal, etc., et leurs extensions pour les motifs fermés et rares. Une description et une comparaison de certain de ces outils en considérant plusieurs caractéristiques peut être trouvée dans (Altalhi et al., 2017; Fournier-Viger et al., 2017). Trois autres bibliothèques dignes d'être mentionnées dans ce contexte sont : DMTL (Chaoji et al., 2008), iZi (Flouvat et al.,

13. http://coron.loria.fr
14. https://cgi.csc.liv.ac.uk/~frans/KDD/Software/
15. http://projects.itsc.uah.edu/datamining/adam/
16. https://adams.cms.waikato.ac.nz/
17. http://jwork.org/dmelt/
18. http://www.keel.es/
19. http://www.borgelt.net/pyfim.html
20. https://github.com/srp33/ML-Flex
21. https://rapidminer.com/

2008), et en fin mais moins libre la collection d'algorithmes Illimine [22]. Pour un suivi des évolutions et des nouveautés sur les outils et logiciels, le lecteur est invité à consulter les pages http://sci2s.ugr.es/keel/links.php#sub10, http://www.kdnuggets.com/software/suites.html et http://www.borgelt.net/software.html qui listent plus de 90 outils libres de fouille de données.

2.4 Conclusion

Dans ce premier chapitre de l'état de l'art, nous avons défini la problématique de la fouille de motifs fréquents puis avons cerné son espace d'états et déterminé sa complexité. En deuxième lieu, une présentation détaillée de ses applications multiples a été effectuée. Nous avons aussi passé en revue les outils libres qui prennent en charge cette tâche importante pour l'extraction de connaissances. Dans le chapitre suivant, nous nous focaliserons sur l'étude des approches et travaux algorithmiques introduits pour l'énumération des motifs fréquents d'une base de transactions.

22. http://http://illimine.cs.uiuc.edu/

CHAPITRE 3

Approches de fouille de motifs fréquents

> We cannot solve our problems with the same thinking we used when we created them.
>
> Albert Einstein

Sommaire

Un état de l'art sur les méthodes de fouille de motifs fréquents fera l'objet du présent chapitre. Nous commençons par évoquer les propriétés permettant une taxonomie des travaux de la littérature, pour ensuite s'attaquer à les distinguer. Pour chaque approche, nous donnons les traits primaires et présentons l'algorithme type, souvent l'initiateur, comme représentant des travaux qui y sont affectés. La présentation évoque : l'idée derrière le papier originel, le pseudo-code, un exemple d'exécution, pour terminer avec une analyse et quelques fameuses extensions. La discussion ne prétend pas à l'exhaustivité vu le nombre important de travaux, mais se contentera plutôt des principales approches introduites, étant donné que la quasi-totalité des algorithmes sont dérivés de celles-ci moyennant des adaptations.

Comme cette thèse a essentiellement une vocation formelle, il est à mentionner que seules les approches de fouille exactes et complètes sont explorées. Ceci dit, les travaux qui visent une fouille approchée ou partielle ne sont pas considérés ici, même si un bref passage leur sera réservé en fin du chapitre. Notons également que le même exemple de la base de données introduit dans le chapitre précédent est retenu ici à des fins d'illustration.

3.1 Critères de catégorisation

En dépit de l'apparente simplicité, le modèle de la fouille de motifs fréquents a exhibé, pour la communauté fouille de données, des défis importants qui lui ont procuré des attraits de recherche marquants. En effet, devant un espace de problème trop large exigeant des coûts (en temps et espace) immenses, une pléthore de méthodes et approches ont été introduites afin de proposer des solutions raisonnables à ce problème intéressant qui est devenu rapidement un bloc de base dans beaucoup d'autre tâches d'extraction de connaissances. Dans cette section, nous étudions les principales approches algorithmiques de la littérature proposées pour la résolution de ce problème. Nous avons délibérément séparé ces approches en trois grandes classes, suivant la philosophie d'extraction adoptée, à savoir :

— l'énumération totale,

— l'énumération abrégée,

— l'incrémentalité.

Chacune de ces classes d'approches est divisée à son tour en de multiples autres sous approches que nous allons décrire. Les présentes sections tirent leur origines du nombreuses sources que le lecteur intéressé est invité à puiser en se référant aux références abondantes dédiées au sujet. À titre indicatif, nous citons les papiers de comparaisons et de *survey* suivants : (Hipp et al., 2000a; Goethals and Zaki, 2003; Goethals, 2003; Ceglar and Roddick, 2006; Han et al., 2007; Borgelt, 2012; Aggarwal et al., 2014; Fournier-Viger et al., 2017; Altalhi et al., 2017), et les chapitres consacrés au problème des livres de référence ci-après : (Tan et al., 2005; Han et al., 2011; Zaki and Wagner Meira, 2014; Leskovec et al., 2014; Aggarwal, 2015).

Étant donné que notre problème est exponentiel, les approches proposées ont introduit des idées et astuces afin de faire face aux défis présents. Ces approches se ressemblent relativement aux points ayant trait à l'efficacité comme a été clairement conclu dans de multiple études comparatives (Hipp et al., 2000a; Goethals and Zaki, 2003; Bayardo et al., 2004) ; toutefois, elles peuvent être distinguées sur plusieurs autres aspects. Parmi ceux-ci, nous retenons les critères suivants : la représentation de données, les méthodes d'exploration de l'espace de recherche, de génération de candidats et de calcul des supports.

3.1.1 Représentation de données

Dans la fouille de motifs, un ensemble de données D est modélisé par une relation binaire entre un ensemble de transactions T et un ensemble d'items A. Le modèle *binaire* pur (le produit $T \times A$) peut être représenté par une matrice binaire à deux dimensions, où une case (i, j) prend la valeur 1 ou 0 selon que la transaction i contient ou non l'item j. Pour des raisons de gain d'espace, une vue *horizontale* (resp. *verticale*) est souvent employée, permettant de ne considérer (à l'aide de listes ou tableaux) que les éléments (items/transactions) présents effectivement dans la base. La représentation horizontale donne les items présents pour chaque transaction. Elle est alors une vue centrée transactions ou lignes, tandis que celle verticale énumère la liste de transactions où apparaît un item donné, dite *tidlist* (pour *transaction identifier list*) de l'item considéré. La représentation verticale est, quant à elle, orientée items ou colonnes et peut être vue comme la transposée de la matrice binaire initiale. D'autres algorithmes ont adopté une combinaison des deux représentations afin de tirer profit des avantages de chacune. Cette dernière représentation est

qualifiée d'*hybride*, dont l'implémentation célèbre est celle qui utilise les arbres de préfixes ou les tries (Mehta and Sahni, 2004), après l'adoption d'un ordre fixe sur les items. Le choix de l'une des représentations citée ci-dessus est en lien étroit avec (ou du moins supporte) les autres aspects tels que la stratégie d'exploration de l'espace de recherche ou le calcul des supports des motifs. Une illustration des principales représentations de données pour la fouille de motifs, à l'égard de notre exemple de référence présenté dans la table 2.1, est montrée dans la figure 3.1.

Trans.	a	b	c	d	e
t_1	1	0	1	1	1
t_2	1	0	1	1	0
t_3	1	1	1	1	0
t_4	1	1	0	0	1
t_5	1	0	1	0	1
t_6	1	0	1	1	1
t_7	1	1	1	1	0
t_8	1	1	0	0	1
t_9	1	0	1	1	0
t_{10}	1	0	1	1	1

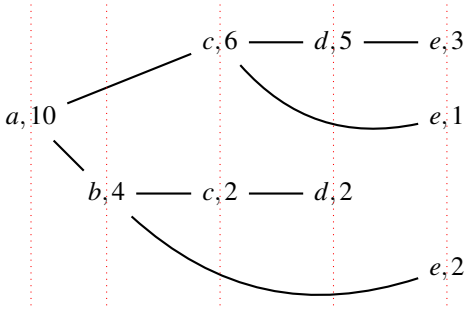
(a) Binaire

Trans.	Motifs
1	$a\ c\ d\ e$
2	$a\ c\ d$
3	$a\ b\ c\ d$
4	$a\ b\ e$
5	$a\ c\ e$
6	$a\ c\ d\ e$
7	$a\ b\ c\ d$
8	$a\ b\ e$
9	$a\ c\ d$
10	$a\ c\ d\ e$

(b) Horizontale

Items	Tidlists
a	1,2,3,4,5,6,7,8,9,10
b	3,4,7,8
c	1,2,3,5,6,7,9,10
d	1,2,3,6,7,9,10
e	1,4,5,6,8,10

(c) Verticale



(d) Hybride

FIGURE 3.1 – Différentes représentations de la base de l'exemple de référence

3.1.2 Méthode d'exploration de l'espace de recherche

L'espace de recherche de notre problème est un treillis semblable à celui de la figure 2.1. Les algorithmes de fouille de motifs adoptent différents schémas de par-

cours de cette structure. Il est évident que la méthode naïve est inefficace car elle permet la génération redondante de motifs. Pour parer à ce problème, les approches adoptent un ordre quelconque mais fixe sur les items afin d'éviter de générer les même motifs plusieurs fois. Ceci est simplifié par la réduction du treillis en un *arbre de préfixes* en considérant l'ordre choisi sur les items. Dans cette structure, dite aussi *arbre d'énumération*, les nœuds des k-motifs partagent le même père (représentant un $(k-1)$-motif) s'ils le possèdent comme un préfixe commun ; les nœuds associés aux motifs singletons constituent les fils de la racine de l'arbre qui dénote le motif nul. La figure 3.2 illustre cette structure, où les items sont traités comme des symboles de l'ensemble A.

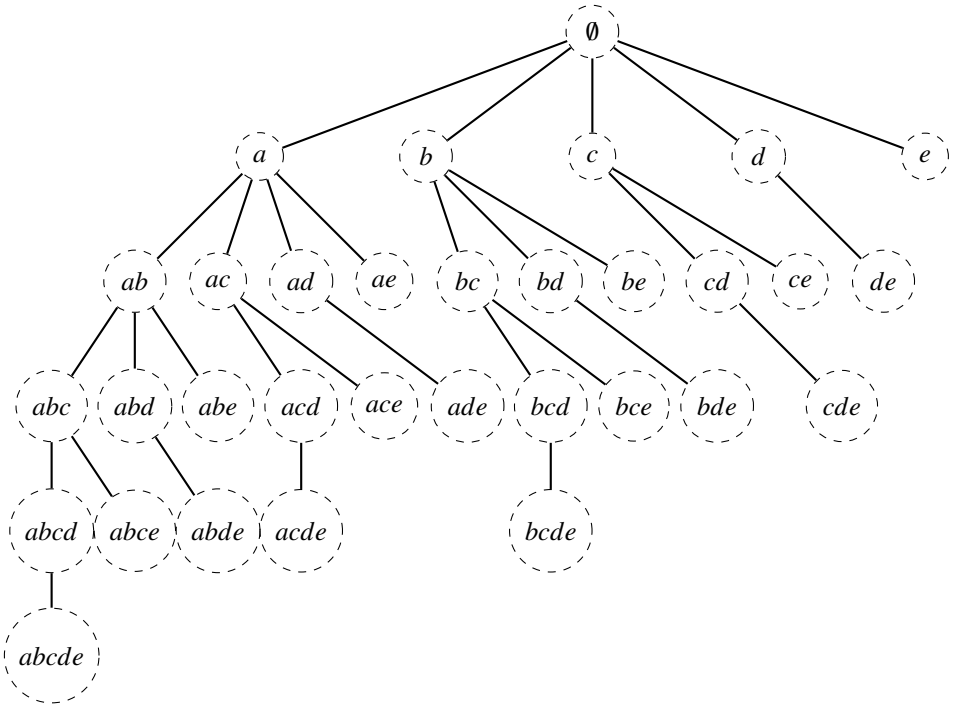


FIGURE 3.2 – Arbre d'énumération (de préfixes) dérivé de l'espace de recherche dans la fouille de motifs le treillis $(2^A, \subseteq)$ des sous ensembles de $A = \{a, b, c, d, e\}$ en adoptant l'ordre lexicographique des items

L'exploration de l'arbre peut être effectuée selon les schémas bien connus en théorie des graphes. La première est l'exploration en *largeur* d'abord (*Breadth-First Search (BFS)*) qui examine, en partant de la racine, les nœuds de l'arbre par niveaux. La mission à l'étape k est de reporter les k-motifs fréquents. C'est une méthode qui scanne l'arbre des motifs des plus généraux à ceux plus spécifiques (les 1-motifs, puis le 2-motifs, etc.), qualifiée également d'*horizontale* ou descendante. Le même

principe peut être envisagé mais dans le sens inverse ascendant, où les motifs plus spécifiques sont visités en premier pour remonter progressivement aux plus généraux. La deuxième manière de scruter l'espace du problème s'engage dans un parcours en *profondeur* d'abord (*Depth-First Search (DFS)*). L'espace est décomposé en sous-problèmes plus petits relativement à une relation de préfixes communs, en progressant à chaque fois par un nœud qui est traité souvent de façon récursive. Il est à noter que d'autre approches combinent les deux méthodes dans une stratégie *hybride* basée sur des heuristiques afin d'optimiser le processus global d'exploration.

3.1.3 Génération de candidats

À la recherche de motifs fréquents, la majorité des algorithmes suivent une stratégie connue sous le slogan *générer et tester* consistant à proposer dans une première phase des motifs *candidats*, *i.e.*, potentiellement ou vraisemblablement fréquents, puis à valider ce caractère dans une deuxième phase en se référant à la base de transactions. Le nombre et la manière avec laquelle sont suggérés les candidats pour le test de fréquence influent considérablement sur l'efficacité de l'algorithme de fouille de motifs. Outre le procédé naïf, il existe principalement deux façons pour proposer des motifs candidats selon qu'elle utilise la *jointure* ou l'*extension*. Dans la première, deux k-motifs fréquents partageant un $(k-1)$-préfixe commun sont joints pour générer un $(k+1)$-motif. Par exemple, la jointure des deux 3-motifs $\{a,b,c\}$ et $\{a,b,d\}$ qui partagent le 2-préfixe $\{a,b\}$ engendre le 3-motif $\{a,b,c,d\}$. Dans la deuxième, un k-motif est étendu par des items successeurs, selon un ordre établit à l'avance sur les items, pour donner naissance à un nouveau $(k+1)$-candidat. Pour le dernier exemple, le 3-motif $\{a,b,c\}$ peut être étendu en considérant l'ordre lexicographique des items par les item $\{d,e\}$, ceci fournira les deux 4-motifs $\{a,b,c,d\}$ et $\{a,b,c,e\}$. Les deux méthodes sont équivalentes et constituent différents moyens de construction de l'arbre d'énumération que tout algorithme de fouille est censé générer.

3.1.4 Calcul des supports

Plusieurs techniques ont été proposées pour mesurer le support d'un motif, étant donné qu'il représente le facteur le plus pesant dans l'efficacité des algorithmes de fouille de motifs. Nous nous limitons ici à seulement deux versions. La première et la plus triviale, employée surtout dans les représentations horizontales, utilise le *comp-*

tage par scans de la base. Un candidat (souvent un ensemble d'entre eux en pratique) est affronté aux transactions (toutes ou une partie d'entre elles selon l'approche d'exploration), où un compteur est incrémenté à chaque fois qu'une occurrence de celui-ci est détectée dans une transaction. L'autre technique emploie l'*intersection* des listes de transactions (tidlists). Dans le cadre d'une représentation verticale, pour connaître le support d'un motif $X \cup Y$ il suffit, en partant des tidlists associées aux motifs X et Y, de calculer le cardinal de l'intersection $|tidlist(X) \cap tidlist(Y)|$. Plusieurs autres optimisations ont été proposées afin d'améliorer l'efficacité de cette phase capitale. Certaines d'entre elles seront évoquées ultérieurement.

3.2 Énumération totale

L'énumération totale vise la découverte complète de l'ensemble $\mathscr{F}$ tout entier, sans omission ni erreur comme défini dans la formule 2.3. La sortie de l'algorithme doit inclure tout motif fréquent et en même temps omettre tout motif infréquent. La question est de savoir si une exploration exhaustive de l'espace de recherche est toujours exigée ? En effet, la réponse est négative comme a été noté par (Agrawal and Srikant, 1994; Mannila et al., 1994; Mannila and Toivonen, 1997). Une propriété clé permettant d'optimiser considérablement le processus de recherche de motifs fréquents et ainsi passer outre l'exploration de la totalité de l'espace du problème a été introduite. Cette propriété, pourtant évidente, est indéniablement perçue comme une avancée considérable dans le sujet et a été par conséquent adoptée plus tard par la totalité des algorithmes. Cette propriété connue sous l'appellation d'*anti-monotonie du support* ou de *fermeture descendante de l'ensemble de motifs fréquents*, stipule que si un motif est fréquent alors tous ses sous-motifs le sont aussi ; et dualement, si un motif est infréquent alors tous ses sur-motifs ne peuvent être fréquents. L'impact de cette astuce dans les implémentations se traduit à localiser, dans l'espace du problème, ce qui communément appelé la *bordure de fréquence* puis de restreindre ainsi la génération seulement aux motifs situés en dessus de cette frontière. Les autres motifs qui se situent en bas de cette limite sont donc infréquents et doivent être ignorés. Ces notions sont introduites ci-dessous et illustrées dans les figures 3.3 et 3.4.

Lemme 3.2.1 (Apriori ou anti-monotonie du support (Agrawal and Srikant, 1994))**.** *Soit D une base de transactions, x et y deux motifs sur l'ensemble d'items A, nous*

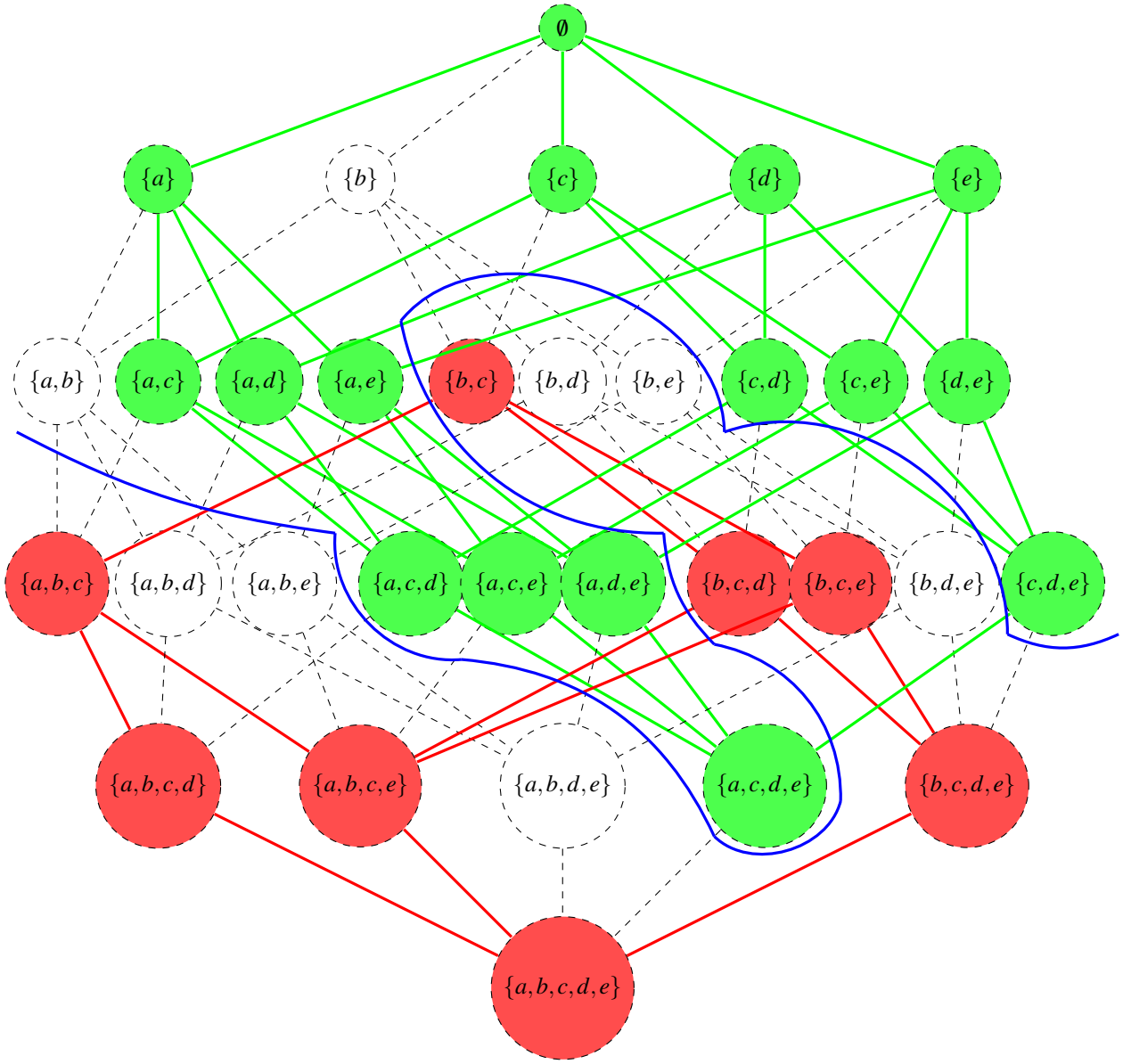


FIGURE 3.3 – Illustration du principe d'anti-monotonie du support : $\{a,c,d,e\}$ est fréquent, tous ses sous motifs le sont aussi (couleur verte). De même, $\{b,c\}$ est infréquent ; tous ses sur-motifs sont également infréquents (couleur rouge)

avons :

$$x \subseteq y \implies \text{sprt}(x) \geq \text{sprt}(y)$$

Démonstration. Triviale. Elle découle de la définition 2.1 relative à la couverture d'un motif et de la transitivité de l'inclusion. En effet, si une transaction t_i contient un motif y, elle inclut forcément tout sous-motif x de y. □

De ce lemme, il découle le corollaire utile suivant.

Corollaire 3.2.2 (Fermeture descendante des motifs fréquents)**.** *Soit D une base de transactions sur un ensemble d'item A, et s un seuil de support, nous avons :*

$$x \text{ fréquent} \implies \forall y \subseteq x,\ y \text{ est aussi fréquent}$$

Corollaire 3.2.3. *Soit D une base de transactions sur un ensemble d'item A, et s un seuil de support, nous avons :*

$$x \text{ infréquent} \implies \forall y \supseteq x,\ y \text{ est aussi infréquent}$$

Par ailleurs, la complexité d'une instance du problème étudié, et par conséquent de l'efficacité de l'algorithme proposé pour le résoudre, est liée à la notion de bordure de l'ensemble de motifs fréquents qui obéit au principe de fermeture descendante. Cette notion est introduite ci-dessous.

Définition 3.2.1 (Bordure de $\mathscr{F}$)**.** *La bordure d'un ensemble $\mathscr{F}$, noté $\mathscr{B}d(\mathscr{F})$, est l'ensemble des motifs de A dont tous les sous-motifs sont dans $\mathscr{F}$ et tous les sur-motifs ne sont pas dans $\mathscr{F}$.*

$$\mathscr{B}d(\mathscr{F}) = \{x \subseteq A \mid \forall y \subset x, y \in \mathscr{F} \wedge \forall z \supset x, z \notin \mathscr{F}\}$$

Les motifs de la bordure qui font ou non partie de $\mathscr{F}$ sont dits bordure positive et négative respectivement et notés $\mathscr{B}d^+(\mathscr{F})$ et $\mathscr{B}d^-(\mathscr{F})$. Ils correspondent, comme nous verrons dans la section 3.3 du prochain chapitre, aux motifs fréquents maximaux et ceux infréquents minimaux. Formellement, ces deux notions sont spécifiées ci-après.

Définition 3.2.2 (Bordure positive)**.**

$$\mathscr{B}d^+(\mathscr{F}) = \{x \subseteq A \mid \forall y \subseteq x, y \in \mathscr{F} \wedge \forall z \supset x, z \notin \mathscr{F}\}$$

Définition 3.2.3 (Bordure négative)**.**

$$\mathscr{B}d^-(\mathscr{F}) = \{x \subseteq A \mid \forall y \supseteq x, y \notin \mathscr{F} \wedge \forall z \subset x, z \in \mathscr{F}\}$$

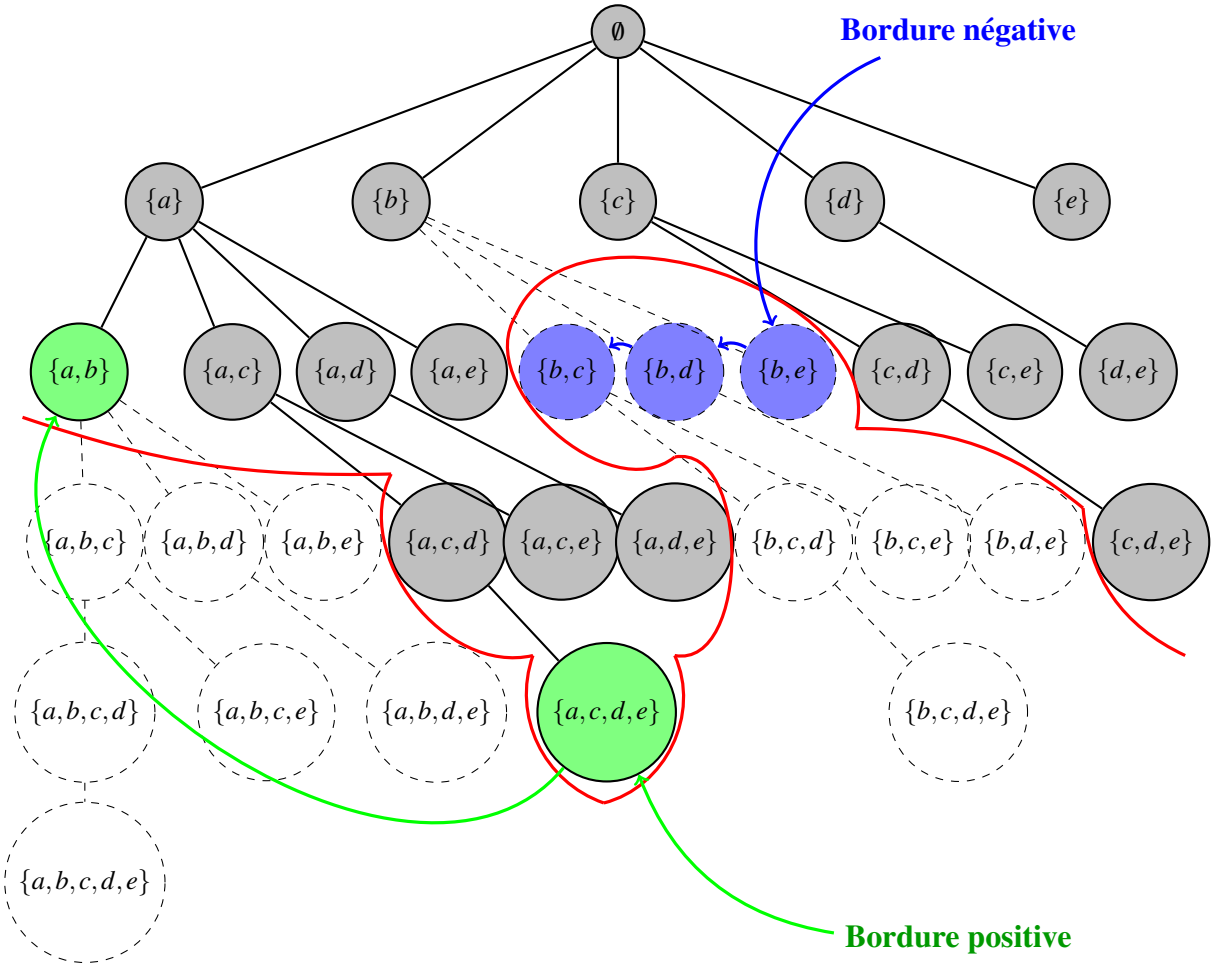


FIGURE 3.4 – Bordure de fréquence de la base de l'exemple de référence en ligne rouge : tous les nœuds en dessus sont fréquents dont la bordure positive en vert, et ceux en bas infréquents dont la bordure négative en bleu.

3.2.1 Approches par niveaux

Comme leur nom indique, ces méthodes extraient les motifs fréquents niveau par niveau. En premier lieu, les singletons (1-motifs) fréquents sont d'abord déterminés et ceux infréquents écartés. Ensuite, la mission de l'algorithme consiste à générer les paires fréquentes en combinant les singletons fréquents, puis les triplets fréquents à partir des doublets fréquents, et ainsi de suite. Souvent itératives, cette classe de méthodes utilisent une représentation horizontale de la base de transactions et le principe générer-tester par une stratégie d'exploration de type BFS. Ainsi, les candidats potentiels à l'étape k, notés par l'ensemble $\mathbb{C}_k$, sont construits par jointure de l'ensemble des $(k-1)$-motifs fréquents, dénoté $\mathbb{F}_{k-1}$, trouvés dans l'étape $k-1$. Le

processus continue en alternance entre génération de candidats et calcul de support, via des balayages de la base de données, et s'arrête lorsque il n'est plus possible de découvrir de nouveaux motifs fréquents de longueurs supérieures. Le résultat de l'algorithme est alors l'union des différents k-motifs fréquents trouvés.

3.2.1.1 Apriori

Apriori (Agrawal and Srikant, 1994) est incontestablement l'algorithme pionnier de cette classe de méthodes et d'ailleurs pour les autres aussi. Il est introduit comme une amélioration de son prédécesseur AIS (Agrawal et al., 1993), baptisé selon les noms de ses inventeurs, qui n'emploie aucune optimisation. La force d'Apriori, dont le principal pseudo-code [1] est donné en algorithme 3, est dû à l'heuristique d'anti-monotonie du support dite aussi la propriété Apriori tout court, qui a offert une réduction considérable du nombre de candidats à examiner. Cette dernière est exploitée selon deux optimisations : *(i)* à l'étape k, un k-candidat x est éliminé s'il possède un sous-motif infréquent de taille $k-1$ *(ii)* la totalité du sous arbre enraciné au motif x est ignoré si ce dernier s'avère infréquent. Notons que le principe Apriori fût indépendamment découvert également dans (Mannila et al., 1994). Les deux groupes de recherche consolident plus tard leur travaux dans un seul article (Agrawal et al., 1996).

De plus, pour accélérer le calcul des supports réalisé par balayages de la base, les motifs candidats sont consignés dans un arbre de hachage (*hash tree*). Dans cette structure, les motifs sont stockés dans les feuilles, et les nœuds internes au niveau d contiennent des tables de hachage qui pointent sur des nœuds du niveau suivant $d+1$. Ainsi, lors du calcul des supports et pour comparer les candidats à une transaction, les items de cette dernière sont utilisés pour descendre (par application de la fonction de hachage aux différents items) dans l'arbre et atteindre les candidats possibles, pour lesquels les supports associés sont incrémentés.

Le déroulement de l'algorithme Apriori est illustré ci-dessous pour la base de notre exemple de référence. Les motifs barrés avec un point entre parenthèse sont les candidats éliminés en raison qu'ils ne respectent pas la clause *(i)* de la propriété Apriori (contiennent un sous-motif infréquent) ; ceux barrés avec le support entre parenthèses sont des candidats retenus mais identifiés comme infréquents à la suite

1. Le code de la procédure de génération de candidats par auto-jointure étant trivial, il n'est pas donné ici. Les exemples présentés sont jugés suffisants pour l'expliquer.

du calcul de leurs supports, c'est-à-dire ne vérifiant pas la clause *(ii)*.

Algorithme 3 APRIORI(D,s)

Entrée : Une base de transactions D, et un seuil de support s
Sortie : L'ensemble $\mathscr{F}$ des motifs fréquents

```
k ← 1
F_1 ← {Les sigletons fréquents}
tant que F_k ≠ ∅ faire
    Générer C_{k+1} par jointure des motifs de F_k
    Éliminer de C_{k+1} les éléments qui violent la propriété Apriori
    Déterminer F_{k+1} par calcul des supports des candidats retenus
    k ← k + 1
fin tant que
retourner ∪_i F_i
```

$\mathbb{C}_1 = \{a,b,c,d,e\} \Rightarrow \mathbb{F}_1 = \{a(10),b(4),c(8),d(7),e(6)\}$
$\mathbb{C}_2 = \mathbb{F}_1 \times \mathbb{F}_1 = \{ab(4),ac(8),ad(7),ae(6),bc(2),bd(2),be(2),cd(7),ce(4),de(3)\}$
$\Rightarrow \mathbb{F}_2 = \{ab(4),ac(8),ad(7),ae(6),\cancel{bc(2)},\cancel{bd(2)},\cancel{be(2)},cd(7),ce(4),de(3)\}$
$\mathbb{C}_3 = \mathbb{F}_2 \times \mathbb{F}_2 = \{abc(.),abd(.),abe(.),acd(.),ace(.),ade(.),bcd(.),bce(.),bde(.),cde(.)\}$
$\Rightarrow \mathbb{F}_3 = \{\cancel{abc(.)},\cancel{abd(.)},\cancel{abe(.)},acd(7),ace(4),ade(3),\cancel{bcd(.)},\cancel{bce(.)},\cancel{bde(.)},cde(3)\}$
$\Rightarrow \mathbb{C}_4 = \mathbb{F}_3 \times \mathbb{F}_3 = \{acde(.)\} \Rightarrow \mathbb{F}_4 = \{acde(3)\}$
$\mathscr{F} = \mathbb{F}_1 \cup \mathbb{F}_2 \cup \mathbb{F}_3 \cup \mathbb{F}_4$

3.2.1.2 Analyse d'Apriori

La preuve de correction des algorithmes par niveaux dont Apriori est un représentant est donnée dans (Mannila and Toivonen, 1997), en considérant ce problème comme celui de la recherche des instances d'un langage satisfaisant le prédicat de fréquence. Comme est indiqué dans le pseudo-code d'Apriori, le temps est dominé par les étapes de génération de candidats et de calcul des supports. Cet algorithme suit une génération en largeur, ceci borne son calcul par le plus long motif fréquent dans la collection, qui représente également le nombre de scans nécessaires à l'algorithme. Aussi, sa complexité asymptotique se réduit à $O(l|D|2^l)$, où l est la taille du plus long motif fréquent. Donc, malgré l'amélioration apparente par rapport à l'approche naïve, la complexité de ces algorithmes demeure toujours exponentielle de l'ordre de $2^{|A|}$. En pratique, outre les détails d'implémentations, plusieurs autres facteurs influent sur le temps de réponse de ces algorithmes (Tan et al., 2005).

— Bien que n'apparaît pas directement sur l'estimation de la complexité, une valeur trop faible du *seuil minimal du support s* induit des calculs énormes, vu qu'elle élargit la bordure de fréquence et pousse à la hausse le nombre possible de motifs fréquents,

— Les algorithmes sont directement sensibles au *nombre d'items* $|A|$, qui représente la dimensionnalité des données et détermine la taille de l'espace de recherche,

— La *taille de la base de données* $|D|$ exprimant le nombre de transactions est aussi capitale dans cette complexité,

— La *longueur moyenne d'une transaction*, car des transactions plus larges qui caractérisent les bases de données denses tendent à compliquer les calculs en raison de la présence de nombreux candidats à examiner.

En tout état de cause, (Gunopulos et al., 2003; Angiulli et al., 2001) précisent le caractère difficile du problème, étant donné qu'il est énumératif. En fait, il a été prouvé que le comptage du nombre de motifs fréquents est #P-difficile (Papadimitriou, 1994) en fournissant une réduction polynômiale au problème de comptage d'assignations pour la satisfaction d'instances de problèmes 2SAT [2], et que la question de décider de l'existence, pour un k donné, d'un k-motif fréquent dans une base de transactions est NP-complet. C'est pour ces raisons que souvent l'examen des algorithmes de fouille de motifs préconise des complexités sensibles à la sortie. Il a été démontré par ailleurs dans (Mannila and Toivonen, 1997) que tout algorithme de fouille de motifs doit au moins évaluer la bordure de fréquence. Mais, à quel ordre cette frontière peut être large ? En effet, cette frontière est bornée comme stipule le lemme suivant.

Lemme 3.2.4 (Borne de la bordure de fréquence)**.** *Dans un contexte de fouille de motif, la bordure négative (positive) est bornée par* $\binom{|A|}{\lfloor |A|/2 \rfloor}$.

Démonstration. Il suffit d'étudier la valeur extrême, selon k, de la grandeur $\binom{|A|}{k}$. L'examen par exemple de la fraction $\frac{\binom{|A|}{k+1}}{\binom{|A|}{k}}$, montre que ce rapport est supérieur à l'unité tant que k est en dessous de $\frac{|A|-1}{2}$. Une autre façon d'aboutir à ce résultat consiste à remarquer que cette frontière correspond à l'ensemble des motifs fréquents

2. Cas particulier des problèmes SAT de satisfaction de formules logiques en forme normale conjonctive à clauses ayant deux littéraux. 2SAT est dans P.

maximaux et infréquents minimaux, présentés plus loin dans la section 3.3, qui sont des anti-chaînes de l'ensemble puissance de A, où celles de tailles maximales sont connues en combinatoire par le théorème de Sperner (Guichard, 2017). □

Apriori est un algorithme intéressant qui a bénéficié de nombreuses implémentations publiées [3] (Bodon, 2003; Borgelt, 2003; Kosters and Pijls, 2003), dont certaines sont réalisées dans divers outils de fouille de données comme nous avons vu à la section 2.3. Il est particulièrement efficace dans les bases de données sparses avec des motifs peu longs (Pei et al., 2001; Zaki and Gouda, 2003). Toutefois, deux inconvénients sont souvent infligés à Apriori. Le premier est le fait de générer un nombre considérable de candidats, ce qui dégrade significativement, comme déjà mentionné, ses performances. Le second à trait au surcoût causé par les multiples et couteuses opérations d'entrée/sortie nécessaires pour le calcul des supports. Aussi, plusieurs améliorations ont été introduites afin de l'améliorer. Ci-après, nous présentons brièvement quelques perfectionnements notables.

3.2.1.3 Quelques extensions à Apriori

— **AprioriTid** et **AprioriHybrid** sont deux extensions mineures proposées dans le même papier d'Apriori (Agrawal and Srikant, 1994). Le premier permet de réduire le nombre de balayages de la base à seulement un seul. En effet, AprioriTid construit, après la première passe, des tables de transactions réduites aux motifs candidats, générées de la même manière qu'Apriori, que contiennent celles-ci. Ces structures remplacent la base qui n'est donc plus utilisée dans les calculs. Il a été montré que AprioriTid donne des résultats inférieurs comparés à Apriori dans les toutes premières itérations, vu que les nouvelles structures induisent des surcoûts en mémoire (et en temps car les données sont à remplacer sur diuesque). Toutefois, au fur à mesure que l'algorithme progresse les résultats dépassent ceux d'Apriori, pour les raisons inverses. Ce constat a incité les auteurs à proposer AprioriHybrid qui combine les deux, dans lequel Apriori est invoqué en premier, puis dès que les structures d'AprioriTid peuvent se loger en mémoire AprioriHybrid commute à AprioriTid de manière à tirer profit des avantages des deux algorithmes.

3. Parmi les autres implémentations célèbres, nous signalons celle de Bart Goethals de l'université d'Antwerp (belgique) disponible dans *Frequent Pattern Mining Implementations* à l'url : http://www.adrem.ua.ac.be/~goethals/software/files/apriori.tgz

— **DHP** (*Dynamic Hashing and Pruning*) (Park et al., 1995a) est introduit pour réduire le nombre de candidats. Dans DHP, les auteurs ont remarqué que durant la première passe d'Apriori une bonne partie de la mémoire reste libre et inexploitée. Pour ce faire, cet algorithme prépare, durant le calcul des supports des k-motifs, des indicateurs renseignant sur les fréquences des $(k+1)$-motifs. Une table de hachage est utilisée pour stocker des compteurs des candidats, regroupés en blocs, représentant les sous-motifs de la transaction en cours. Ainsi, dans l'étape de calcul des supports des $(k+1)$-motifs, seuls les candidats des blocs dont les compteurs dépassent le seuil minimal du support sont pris en considération, le reste sera ignoré. Cette astuce s'est avérée efficace précisément dans l'étape de calcul des pairs (2-motifs) fréquents qui dominent le calcul dans ce problème. De plus, DHP réduit progressivement la taille de la base et tronque les transactions en s'appuyant sur la condition nécessaire que tout item i peut être supprimé s'il n'apparait pas au moins k fois dans les k-motifs candidats d'une transaction.

— **Partition** (Savasere et al., 1995) est un algorithme qui réduit le nombre de scans de la base à seulement deux. Dans la première, il fractionne la base en plusieurs parties disjointes choisies telle que leurs tailles tiennent en mémoire, qui les traite séparément en extrayant leurs motifs fréquents locaux. Dans la seconde passe, les résultats locaux sont consolidés et vérifiés en confirmant les motifs fréquents globaux et écartant le reste (les résultats de la première passe peuvent inclure des faux positifs). L'idée sur lequel repose Partition est qu'un motif ne peut être fréquent dans la base s'il n'est pas fréquent dans au moins une des partitions.

— **Sampling** (Toivonen, 1996) travaille sur un échantillon aléatoire plus petit de la base et nécessite deux balayages au pire sur cette dernière. La technique a montrée sa précision et son efficacité notamment pour les bases trop larges. Dans le premier scan, l'algorithme extrait les motifs fréquents dans l'échantillon avec leur bordure négative. Les résultats (motifs et bordure) sont ensuite validés par un second balayage. Pour éviter de sauter des motifs fréquents dans la base, le seuil de support est diminué, lors de la première passe, à une valeur qui garantit avec une forte probabilité d'éviter ces oublis.

— **DIC** (Dynamic Itemset Counting) (Brin et al., 1997) est motivé par la réduction du nombre de passes sur la base de données. La philosophie de cet algo-

rithme entremêle génération de candidats et calcul des supports en anticipant la formation de motifs candidats plus long des motifs trouvés fréquents sans même attendre de voir toutes les transactions. Par analogie au train, la base est divisée en wagons de M transactions chacun, à chaque wagon de nouveau motif sont ajoutés/supprimés dynamiquement. La dynamicité est suivie en catégorisant les motifs en quatre classes : confirmés fréquents/infréquents et présumés fréquents/infréquents.

3.2.2 Approches verticales

Une représentation verticale de la base de transactions associée à un mécanisme de calcul de support via des intersections de tidlists et un parcours de l'espace du problème en profondeurs (DFS) sont les traits caractéristiques des approches verticales de fouille de motifs. Bien qu'Eclat (Zaki, 2000), proposé par Zaki et al. et décrit dans la suite, est considéré comme l'algorithme type de cette famille de méthodes, les fondements et les origines de cette approche remontent, à vrai dire, à bien avant. En effet, la représentation verticale comme structures inversées a été étudiée dans la communauté bases de données depuis les années 80 (Copeland and Khoshafian, 1985). En outre, l'astuce de calcul des supports par intersection des tidlists a été utilisée très tôt par Savasere et al. pour ce problème (Savasere et al., 1995) dans le cadre de l'algorithme horizontale *Partition*. Les origines de l'astuce intersection peuvent être aussi trouvées dans les théories des treillis (Barbut and Monjardet, 1970) et l'analyse formelle de concepts (Wille, 1982; Zaki and Ogihara, 1998).

3.2.2.1 Eclat

Les atouts de l'algorithme ECLAT (Equivalence CLAss Transformation) sont attribués, outre l'heuristique Apriori, essentiellement à trois principes clés. D'une part, l'algorithme indexe la base de données par les items en fournissant leurs tidlist afin d'améliorer son efficacité. Ce principe en plus qu'il diminue les nombreux scans de la base, il offre aussi un calcul rapide des supports. En effet, Eclat exploite ces dernières listes pour la détermination des supports, qui sont alors les cardinaux des intersections résultantes. Par exemple, le support de $\{a,b\}$ est 4 représentant le cardinal de l'intersection des deux tidlists des items a et b qui vaut $\{1,2,3,4,5,6,7,8,9,10\} \cap \{3,4,7,8\} = \{3,4,7,8\}$. Le dernier principe sur lequel

repose Eclat, est une stratégie de décomposition de l'espace de recherche en parties disjointes via la définition de classes d'équivalence sur cet espace. Cette décomposition permet d'alléger les traitements en réduisant ainsi la mémoire requise. La relation d'équivalence introduite est basée sur des préfixes communs (deux motifs x et y son équivalents, s'ils partagent un k-préfixe). L'ensemble des principes précédents sont combinés dans un algorithme récursif qui explore l'espace du problème en profondeur, comme il est montré par le pseudo-code de l'algorithme 4.

Algorithme 4 ECLAT$(L, s, \mathscr{F})$

Entrée : Une base de données transactionnelle D, et un seuil de support s
Sortie : L'ensemble $\mathscr{F}$ des motifs fréquents

```
F ← ∅
L ← {les singletons fréquents}
pour tout P_i ∈ L faire
    F ← F ∪ P_i
    FP_i ← ∅
    pour tout P_j ∈ L | j > i faire
        si |tidlist(P_i) ∩ tidlist(P_j)| ≥ s alors
            FP_i ← FP_i ∪ {P_i ∪ P_j}
        fin si
    fin pour
    si FP_i ≠ ∅ alors
        ECLAT(FP_i, s, F)
    fin si
fin pour
```

Pour simplifier la présentation, une partie seulement du déroulement de l'algorithme Eclat relative aux motifs de la classe de l'item a (exceptés les singletons) est illustrée pour l'exemple de référence dans la figure 3.5

3.2.2.2 Analyse d'Eclat

L'avantage d'Eclat par rapport aux méthodes horizontales est la rapidité dans le calcul des supports via des intersections intelligentes d'ensembles. En effet, pour accélérer le calcul des intersections, les tidlists dans Eclat sont, d'un coté, ordonnées de façon croissante ce qui produit un temps linéaire borné par la taille des deux listes. D'un autre coté, le calcul de cette intersection peut être rapidement abandonné prématurément par une technique dite intersection court-circuit, dès qu'on est certain que l'intersection en cours d'examen ne peut atteindre le seuil minimal du support.

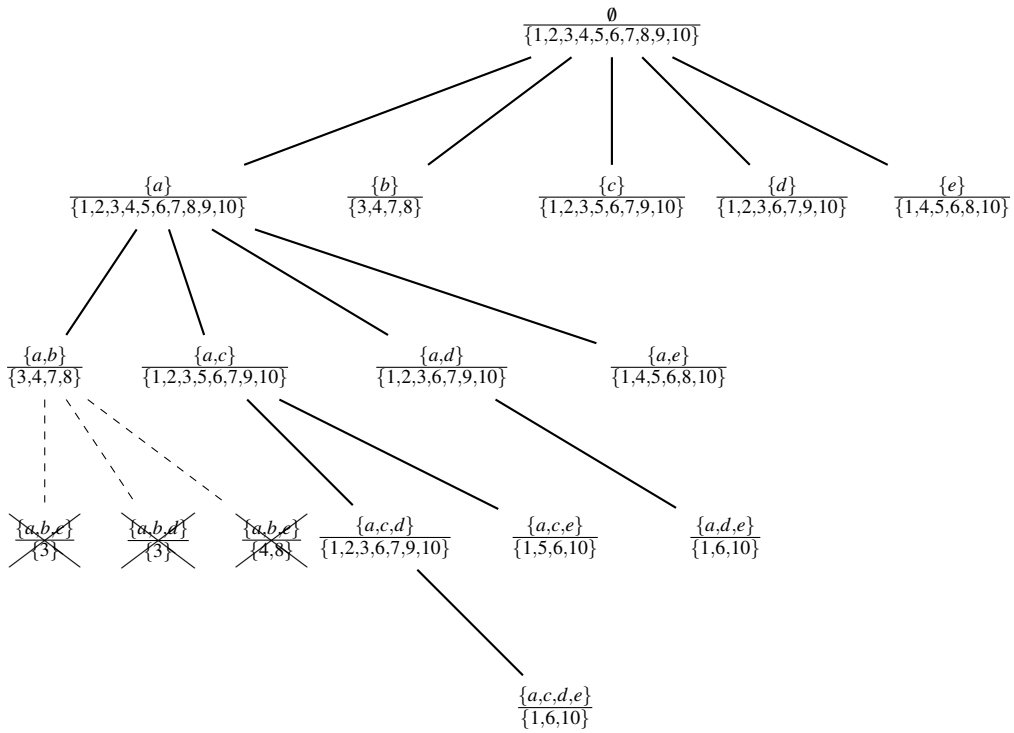


FIGURE 3.5 – Eclat partiellement déroulé pour l'exemple de référence sur la classe d'équivalence de a. Les tidlists résultants des différentes intersections sont en parties basses des nœuds.

Pour toutes ces raisons, il a été trouvé qu'Eclat se comporte très bien devant des contextes de fouille pour des bases de données sparses qui impliquent souvent des intersections légères. Cependant, dans le cas des bases denses les résultats intermédiaires générés par Eclat deviennent énormes où il soufre alors de manque de mémoire. Sa complexité est de l'ordre de $O(2^{|A|}|D|)$ au pire des cas, car l'intersection de deux tidlists est bornée par la taille de la base $|D|$.

3.2.2.3 dEclat

Motivé par l'espace mémoire énorme qu'exige Eclat, notamment dans les cas de fouille complexes (bases de transactions denses/seuil de support trop faible), une version améliorée a été proposée appelée dEclat pour (diffset Eclat) (Zaki and Gouda, 2003). Ce dernier algorithme part de l'idée qu'au lieu de considérer les listes complètes de transactions, il suffit de suivre leurs changements (différences) par rapport au parent en cours de traitement, ce qui permet de réaliser des

gains importants en mémoire. Formellement, le *diffset* d'un motif $X = x_1 \dots x_{k-1} x_k$, noté $d(X)$, est la différence entre les transactions supportant son préfixe et celles supportant son dernier item, *i.e*, $d(X) = tidlist(x_1 \dots x_{k-1}) \setminus tidlist(x_k)$. Cette définition conduit au fait que le *diffset* de deux motifs $X \cup Y$ partageant le même préfixe est la différence entre leur *diffsets* : $d(X \cup Y) = d(Y) - d(X)$, et que le support d'un motif X est la différence entre le support de son préfixe et le cardinal de son *diffset*. Ainsi dans notre exemple, le support du motif $\{a,b\}$ est : $\text{sprt}(\{a,b\}) = \text{sprt}(a) - |d(\{a,b\})| = 10 - |\{1,2,5,6,9,10\}| = 10 - 6 = 4$, puisque $d(\{a,b\}) = d(\{b\}) - d(\{a\}) = \{1,2,5,6,9,10\} \setminus \emptyset = \{1,2,5,6,9,10\}$.

3.2.3 Approches projectives

Afin de réduire la taille de la base de données transactionnelles, parer à ses multiples balayages, et atténuer l'effet de génération de candidats, Han et al. ont proposé l'algorithme *FPGrowth* (Han et al., 2000). Ce dernier exploite une structure de données compacte appelée FPTree (*Frequent Pattern tree*) en guise de représentation de la base de données, qui est une sorte d'arbre de préfixes ou trie (Mehta and Sahni, 2004) augmenté par les informations de support. Ainsi, chaque nœud de l'arbre contient un item et un entier qui représente le support du motif formé des items trouvés sur le chemin depuis la racine à ce nœud. Reposant sur le principe diviser pour régner, FPGrowth adopte une exploration récursive en profondeur de l'espace de recherche via des extensions de suffixes (dans l'ordre inverse) et nécessite seulement deux passes sur l'ensemble de données. Durant la première passe, l'ensemble des items fréquents sont identifiés (il constitue le nouveau ensemble d'items de la base) puis triés par ordre décroissant de leurs supports, et ceux infréquents sont tout simplement supprimés. La mission dans la deuxième passe est de construire l'arbre FPTree (qui contient initialement la racine modélisant le motif nul) de manière analogue à l'insertion de chaînes dans un trie, mais en respectant l'ordre décroissant des supports. En lisant la base ligne par ligne, chaque transaction est d'abord réécrite selon le nouveau ensemble d'items retenus et l'ordre retenu sur celui-ci puis insérée dans l'arbre. Les transactions partageant le même préfixes sont alors juxtaposés (les supports des nœuds concernés sont incrémenté à chaque nouvelle occurrence) ; de nouveaux nœuds sont créés pour les items restants (première occurrence) de la transaction où les supports associés sont initialisés à 1.

L'idée élégante de la structure compacte FPTree représente un pas remarquable

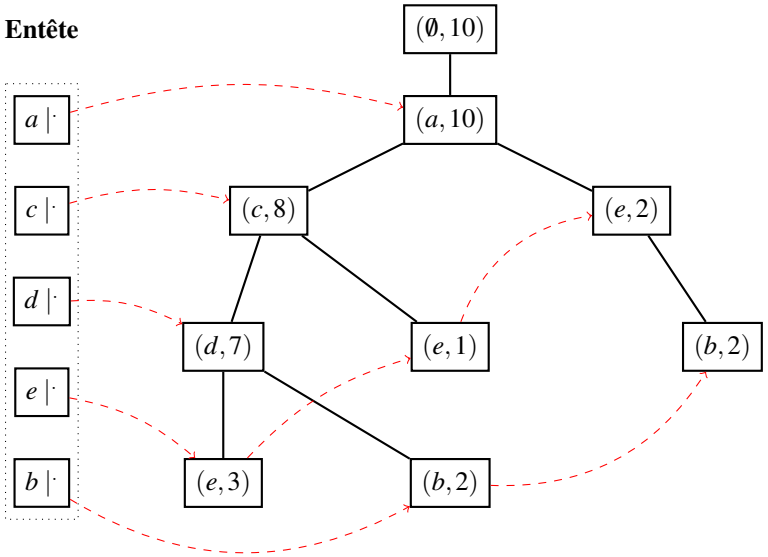


FIGURE 3.6 – FPTree représentant la base de l'exemple de référence

pour cette classe de méthodes. En effet, l'heuristique de réécriture des transactions de la base selon les items fréquents par ordre décroissant du support a permis de réduire significativement la taille de la base à fouiller. D'une part, cette structure inclut uniquement des items fréquents (propriété Apriori), où les transactions sont alors tronquées des items infréquents. D'une autre part, les items les plus fréquents, c'est-à-dire, communs à plusieurs transactions, sont regroupés dans les niveaux supérieurs de l'arbre, ce qui produit un gain en espace en réalisant un taux important de compressions de données.

Par ailleurs, en vue d'accélérer le processus de fouille, une table entête avec des pointeurs intra-nœuds sont ajoutés à l'arbre FPTree. Cette table maintient une liste pointée des items (fréquents) de la base. À chaque case correspond un pointeur, qui la lie au nœud le plus à gauche du même item dans l'arbre. Tous les nœuds de l'arbre portants le même item sont aussi connectés par des pointeurs pour faciliter leur accessibilité. En se référant à la table entête et commençant des items moins fréquents, FPGrowth réalise une succession de projections de la base de données sur le suffixe en cours d'extension de façon à produire une base conditionnelle (base conditionnée par l'item en question). La base résultante de la projection avec le suffixe considéré forment un nouveau contexte sur le quel FPGrowth, dont le pseudo code est montré dans l'algorithme 5, travaille de la même façon (reconstruction d'un nouveau FP-Tree, avec tri (sur le même critère) des items qui les composent et élimination des

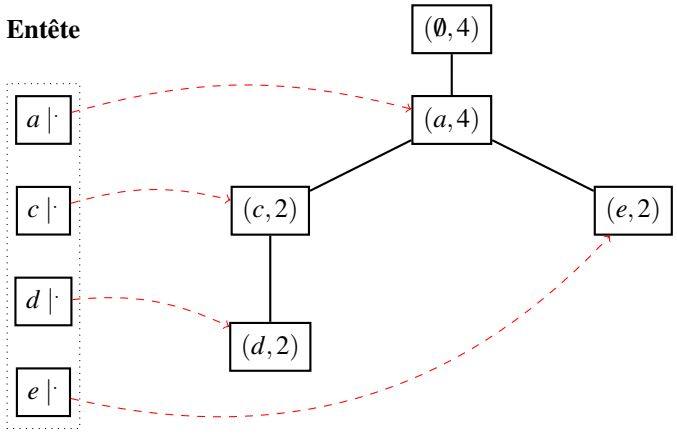


FIGURE 3.7 – FPTree conditionnel de l'item b

infréquents. Ce schéma de traitement est effectué dans une récursion. Le cas de base de cette récursion correspond à une base conditionnelle simple où l'arbre obtenu est vide ou forme un chemin simple. Dans ce cas, tous ses sous motifs possibles sont trouvés et les extensions associés au suffixe en cours sont reportées. Un exemple de construction de l'FPTree de la base de l'exemple de référence est illustré ci-dessous dans la figure 3.6. De même, il facile de voir que la base conditionnelle de b est $\{acd(2), ae(2)\}$; son FPTree conditionnel est montré dans la figure 3.7. L'application de l'algorithme sur cette dernière délivre seulement l'item $\{a\}$ avec un support égal à 4. Ce qui en résulte que le seul motif fréquent du suffixe $\{b\}$ est $\{a,b\}$.

3.2.3.1 Analyse de FPGrowth

La représentation de données par FPTree adoptée dans FPGrowth bien qu'elle a permis de réaliser des économies notables sur la taille des données, elle reste juste une heuristique assez bonne mais qui ne garantit pas toutefois la minimalité (il a été trouvé dans plusieurs exemples de bases de transaction que leur représentations horizontales sont inférieurs à celles en utilisant FPTree). Certains chercheurs ont même considéré la structure FPTree comme une représentation horizontale compacte et d'autre comme une variante de la représentation verticale (Borgelt, 2012). Ceci a été illustré, tout en début du chapitre, dans la figure 3.1

En dépit des gains réalisés en adoptant la structure compressée FPTree, l'ensemble du traitement dans cette approche reste équivalent à celui des algorithmes verticaux et par niveaux. De plus, les auteurs de FPGrowth avancent que leur algo-

Algorithme 5 FPGROWTH(FPT, s, P)

Entrée : FPT l'arbre compact d'une base de données transactionnelle D, un seuil de support s et le suffixe courant P
Sortie : L'ensemble $\mathscr{F}$ des motifs fréquents

```
si FPT est un chemin simple alors
    pour tout combinaison c de nœuds de FPT faire
        Écrire (c ∪ P)
    fin pour
sinon
    pour tout item a dans FPT selon l'ordre croissant du support faire
        Q = {a} ∪ P
        F ← F ∪ Q
        Temp ← ∅
        pour tout chemin d depuis la racine à a faire
            d1 ← d tronqué de a
            insérer d1 dans l'arbre Temp
        fin pour
        si Temp ≠ ∅ alors
            FPGROWTH(Temp, s, Q)
        fin si
    fin pour
fin si
```

rithme n'effectue aucune génération de candidats, ce qui parait peu vrai (Goethals, 2003; Dexters et al., 2006), puisque le processus récursif de projection est équivalent à une sélection de candidats qui est validé ensuite par l'algorithme. Plusieurs travaux conséquents ont constatés que la première version de l'algorithme FPGrowth souffre, hélas, de surcoûts considérables devant, en particulier, des bases de données sparses et très volumineuses (Pei et al., 2001; Sucahyo and Gopalan, 2004; El-Hajj and Zaïane, 2005). En fait, dans ces cas l'arbre FPTree devient énorme, et l'espace nécessaire pour garantir les niveaux profonds de la récursion représente un grand défi. De plus, les multiples reconstructions des différents FPTree et les opérations onéreuses de tri induisent des charges additionnelles sur l'algorithme. La complexité asymptotique de l'algorithme demeure, quant à elle, toujours de l'ordre de $O(2^{|A|}|D|)$ au pire, bien qu'en pratique cette borne est loin d'être atteinte, car les arbres construits sont en général, comme expliqué en haut, nettement inférieurs à la base.

3.2.3.2 Améliorations à FPGrowth

Plusieurs travaux ont suivi la philosophie de FPGrowth en y incorporant des extensions remarquables (Rácz, 2004; El-Hajj and Zaïane, 2005). Les brèches par lesquelles a été attaqué FPGrowth sont, en premier, la structure FPTree qui peut être optimisée d'avantage, et deuxièmement son inconvénient dû aux nombreuses opérations couteuses de reconstruction et re-tri qu'il exige. Ainsi, dans (Sucahyo and Gopalan, 2004) les auteurs présentent un algorithme similaire à FPGrowth mais cette fois-ci avec un arbre dit CPFTree pour *Compressed FPTree* dont la taille peut être réduit à la moitie de celle d'un FPTree, sur lequel travaille un algorithme semblable à FPGrowth mais itératif cette fois-ci, dont la confrontation avec Apriori et FPGrowth a montré une nette supériorité. Une implémentation efficace de l'algorithme FPGrowth qui élimine les lourdes opérations de reconstruction a été présentée dans (Rácz, 2004).

3.2.4 Approches hybrides

À chaque algorithme ses avantages et ses inconvénients et un *one-fits-all* n'existe pas toujours. Par conséquent, des solutions hybrides ont été proposées dans le but d'éliminer les défauts des approches introduites et tirer profit, au même temps, de leurs avantages. Comme nous l'avons vu, AprioriHybrid (Agrawal and Srikant, 1994) est la première hybridation proposée pour la fouille de motifs fréquents. Elle combine Apriori et AprioriTid. Dans (Hipp et al., 2000b) un algorithme hybride est conçu et analysé ; il démarre avec Apriori pour ensuite commuter à Eclat. Récemment, une solution hybride associant l'approche projective représentée par FPGrowth avec l'approche verticale a été suggérée dans (Deng and Lv, 2015). Les auteurs introduisent l'algorithme Prepost$^+$ qui étend la structure FPTree en ajoutant des étiquettes aux nœuds de l'arbre. Cette extension offre des intersections efficaces pour le calcul des supports. L'évaluation de cet algorithme a montré des performances extraordinaires en ce qui concerne le temps de réponse.

3.3 Énumération abrégée

Le nombre de motifs fréquents dans une base de transaction est, comme nous l'avons mentionné, exponentiel avec le nombre d'items. Ainsi, les reporter tous via

une énumération exhaustive est une tâche inextricable notamment pour les contextes de fouille complexes (bases denses et très volumineuses et/ou seuil de support trop bas). Ceci a vivement incité les chercheurs à prospecter des solutions permettant de réduire la complexité de cette tâche. Comme la compression de données est un sujet mature en technologie de l'information, l'appliquer à ce problème constitue une alternative prometteuse. Aussi, plusieurs représentations compactes pour les motifs fréquents qui permettent de les restaurer tous ont été proposées dans la littérature. Ces représentations condensées outre qu'elles permettent de rationner les ressources de calcul et de stockage nécessaires en éliminant les redondances présentes, elles offrent une aisance dans l'analyse des connaissances (règles) découvertes à partir de ces motifs. Dans cette section et afin d'adoucir le manuscrit, nous nous limiterons seulement aux deux représentations compactes les plus courantes à savoir les motifs fréquents maximaux et les motifs fréquents fermés ; pour une étude plus approfondie des représentations compactes des motifs fréquents veuillez vous référez aux références-survey cités en début du chapitre ou les papiers dédiés suivants : (Calders et al., 2004; Mielikäinen, 2005). Afin d'apprécier le gain obtenu en considérant ce type de représentation, la figure. 3.8 anticipe et montre, pour une base de donnée transactionnelle commune (voir le chapitre sur l'expérimentation), les rapports entre le nombre de motifs fréquents, fréquents fermés et fréquents maximaux.

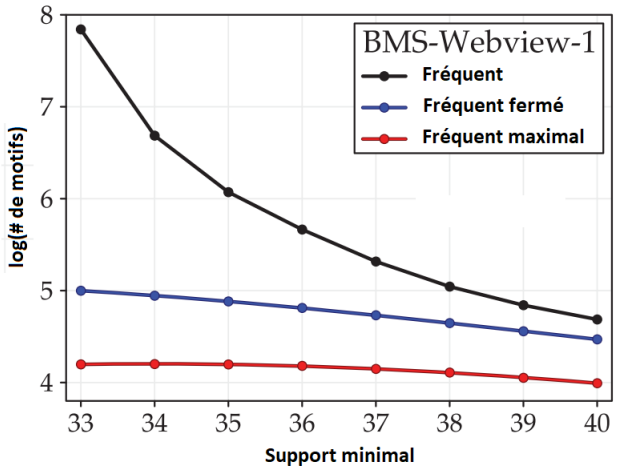


FIGURE 3.8 – Nombre (en échelle logarithmique) de motifs fréquents, fréquents fermés et fréquents maximaux pour la base de données BMS-Webview-1 (Borgelt, 2012)

3.3.1 Motifs fréquents maximaux (MFM)

Un motif x est dit fréquent maximal dans une base D par rapport à un seuil de support s si et seulement si x est fréquent et il n'existe pas un sur-motif de x qui est aussi fréquent ; autrement dit, tous ses sur-motifs sont infréquents. Une définition formelle est donnée ci-dessous.

Définition 3.3.1 (Motif fréquent maximal)**.** *Dans une base D et un seuil de support s, un motif x est fréquent maximal ssi x est fréquent et* $\nexists y \mid y \supset x \wedge \text{sprt}(y) \geq s$

Partant de la propriété Apriori stipulant que tout sous-motif d'un motif fréquent est lui même fréquent, il suffit donc de connaitre les motifs fréquents maximaux (MFM) d'un ensemble de données. Ce dernier permet de dériver le reste en considérant leurs sous-motifs sans référence à la base. Notons que l'ensemble de MFM, noté $\mathscr{M}$, correspond exactement à la bordure positive définie dans 3.2.2, qui une fois identifiée tout ce qui est situé en dessous d'elle est forcement fréquent. L'ensemble de MFM bien qu'il offre une réduction importante de l'espace du problème, il ne préserve pas cependant l'information relative au supports des motifs fréquents trouvés, ce qui requière un balayage additionnel de la base pour renseigner les supports associés aux sous-motifs fréquents qu'on pourra en dérivés. Les MFM, illustrés dans la figure 3.9, pour notre exemple de référence sont : $\mathscr{M} = \{\{a,b\},\{a,c,d,e\}\}$.

En principe, tous les algorithmes d'extraction totale de motifs fréquents sont adaptables à l'extraction de ceux fréquents maximaux. L'idée naïve est d'ajouter un test de maximalité après chaque motif fréquent trouvé de façon à ne retenir que les maximaux entre eux. Ainsi, si un motif fréquent extrait est un sous-motif d'un autre déjà marqué maximal il sera ignoré ; de même, s'il représente un sur-motif d'un autre classé maximal, ce dernier sera supprimé de la sortie. Nombreux sont les algorithmes introduits pour l'extraction de MFM. Ils sont fondés sur l'approche de base précédente à laquelle ils ajoutent d'autre optimisations afin d'accélérer le processus de fouille. Ci-après une sélection non intégrale d'algorithmes de cette catégorie : Max-Miner (Bayardo, 1998), Pincer-Search (Lin and Kedem, 1998), Depth Project (Agarwal et al., 2000), MAFIA (Burdick et al., 2001), FPMax (Grahne and Zhu, 2003), LCM-Max (Uno et al., 2004), GenMax (Gouda and Zaki, 2005), Charm-MFI (Szathmary, 2006), etc. Loin d'une présentation exhaustive, nous décrivons brièvement, dans ce qui suit, trois algorithmes représentatifs (un pour chaque approche) de découverte de MFM.

MaxMiner

Cet algorithme (Bayardo, 1998) suit la même philosophie d'Apriori en adoptant un parcours en largeur de l'espace de recherche représenté par l'arbre d'énumération d'ensemble défini par Rymon (Rymon, 1992), qui l'implémente aussi par un arbre de hachage. Les points forts de *MaxMiner* sont les optimisations qu'il introduit (fréquence de sur-motifs/infréquence de sous-motifs) et un mécanisme d'anticipation (*lookahead*) permettant de découvrir rapidement des motifs fréquents plus longs. Pour ce faire, l'auteur associe à chaque nœud correspondant à un groupe candidat deux motifs : le premier est la tête du groupe noté $h(g)$ qui symbolise le motif que code le nœud et $t(g)$ sa queue qui est un ensemble ordonné d'items pouvant constituer une extension possible de la tête du groupe. L'examen d'un nœud correspond au traitement du groupe candidat codé par celui ci, qui implique le calcul des supports pour les motifs suivants : $h(g)$, $h(g) \cup t(g)$, et $h(g) \cup \{i\}$ pour tout $i \in t(g)$.

Dans *MaxMiner*, l'optimisation fondée sur la fréquence de sur-motifs est implémenté en ignorant l'extension de tout groupe pour lequel $h(g) \cup t(g)$, qui représente le plus long sur-motif de celui du nœud, est fréquent puisque dans ce cas tous les descendants de ce groupe sont évidemment fréquents mais non maximaux. De plus, si le résultats de l'extension fréquente $h(g) \cup t(g)$ représente un sous-motif d'un autre connu déjà comme tel, ce candidat sera éliminé. D'un autre coté, l'optimisation fondée sur l'infréquence de sous-motifs est implémentée pour tous les items i où le motif $h(g) \cup \{i\}$ s'avère infréquent. Ces branches sont tout simplement abandonnées, car leurs expansions futures seront certainement infréquentes. Enfin, *MaxMiner* propose deux autres techniques permettant une optimisation plus fine. D'une part, les descendants d'un nœud sont triés par ordre croissant de leurs supports afin de permettre une efficacité de la première optimisation basée sur la fréquence des sur-motifs, car les items les plus fréquents se trouvent dans la majorité des nœuds ce qui accroit la chance de bénéficier de cette optimisation. Une borne inférieure sur le support d'un motif en exploitant les informations déjà disponibles sur les supports de ses sous-motifs permet, d'autre part, d'estimer quand est-ce qu' il serait fréquent avant même d'accéder à la base, offrant ainsi la possibilité de limiter le nombre de candidats à examiner.

GenMax

GenMax (Gouda and Zaki, 2005) est un algorithme récursif de découverte de MFM qui utilise le format de données vertical et un parcours en DFS de l'espace du problème. Cet algorithme génère exactement la collection de MFM et ne nécessite aucun post-traitement ; il profite de certaines optimisations connues déjà à l'instar de celles utilisées dans MaxMiner et MAFIA. Il définit, similairement à MaxMiner, pour chaque motif x son *combine-set* composé des items pouvant participer à une extension possible à x, qu'il ordonne selon l'ordre croissant des supports. Cette astuce, permet un élagage très-tôt de l'espace de recherche car les motifs moins fréquents ont peu de chance de former des combine-set plus larges dans le niveau suivant. En commençant par un ensemble $\mathscr{M}$ de MFM vide, à chaque étape l'union du motif en cours avec son combine-set est examiné. S'ils sont contenus dans un motif maximal, la branche toute entière sera alors élaguée ; sinon, GenMax passe à la formation de nouveaux motifs candidats et calcule leurs supports. Si l'ensemble des extensions possibles est non vide, un appel récursif est lancé pour trouver d'autres extensions plus longues. Le cas contraire correspondant à un motif fréquent maximal probable ; ce dernier n'est inséré dans $\mathscr{M}$ que s'il ne constitue pas un sous-motif d'un autre déjà dans $\mathscr{M}$. De plus, GenMax utilise d'autres heuristiques pour réduire davantage son temps de réponse tels que : la propagation des diffsets introduit dans dEclat afin d'accélérer l'estimation des supports, et la «progressive focusing» consistant à diminuer le nombre des tests d'appartenance inutiles entre motifs en se limitant uniquement à un sous ensemble réduit de $\mathscr{M}$, dit MFM locaux, et en fin, le maintient d'un indicateur de suivi de changements dans l'ensemble $\mathscr{M}$ pour signaler si ce dernier a été ou non modifié de manière à annuler toutes les vérifications d'inclusion dans le cas où $\mathscr{M}$ n'a pas changé.

FPMax

FPMax (Grahne and Zhu, 2003) est une extension, pour l'extraction de motifs fréquents maximaux, de l'algorithme FPGrowth duquel il emprunte plusieurs aspects. D'un coté, il est récursif et opère un parcours en DFS de l'espace du problème. Il utilise, d'un autre coté, en addition de l'arbre FPTree introduit par FPGrowth, une structure semblable dite MFI-Tree (Maximal Frequent Itemset Tree) dans les tests de maximalité et le suivi de la collection des MFM trouvés. L'idée sur laquelle repose FPMax est l'observation que si un motif fréquent n'est pas un sous-motif d'aucun des

MFM existants, il est alors un nouveau MFM. Ainsi, cet algorithme crée l'arbre MFI-Tree de façon similaire à l'arbre FPTree, qui inclut donc une table entête (ordonnée de la même manière) et des pointeurs de liens internes, mais sans les informations relatives aux supports. Une fois la structure initialisée, le processus d'extraction est entamé par FPMax durant lequel les cas de bases des récursions de FPGrowth sont exploités pour insérer le motif fréquent trouvé (qui représente un chemin simple de l'FPTree) auquel est ajouté l'item en cours si bien évidement le motif fréquent en question est absent dans la structure MFI-Tree. A la fin du traitement, l'MFI-Tree contient la collection des MFM de la base de données.

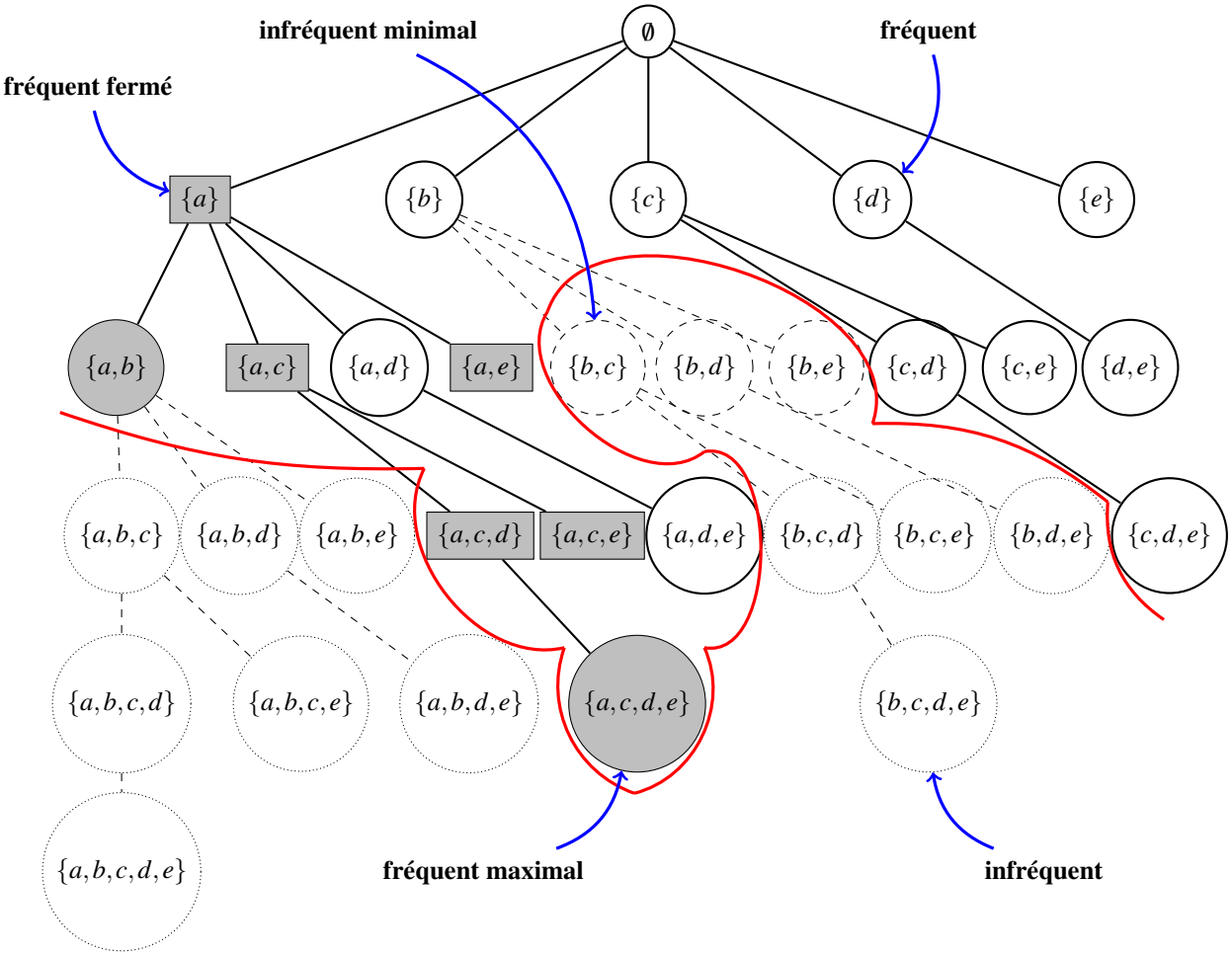


FIGURE 3.9 – Représentations condensées de la base de l'exemple de référence : motifs fréquents maximaux (2 nœuds gris en cercles) et motifs fréquents fermés (7 nœuds gris)

3.3.2 Motifs fréquents fermés (MFF)

Un motif x est dit fréquent fermé dans une base D par rapport à un seuil de support s ssi x est fréquent et il n'existe pas un sur-motif de x qui a le même support. Intuitivement, un motif fermé est un ensemble maximal d'items commun à un ensemble de transactions.

Définition 3.3.2 (Motif fréquent fermé)**.** *Dans une base D et un seuil de support s, un motif x est fréquent fermé ssi x est fréquent et* $\nexists y \mid y \supset x \wedge \text{sprt}(y) = \text{sprt}(x)$

Formellement, un motif x est fermé par rapport à un opérateur γ si $x = \gamma(x)$. Cette notion de fermeture a été empruntée à la fouille de motifs premièrement mais séparément par (Pasquier et al., 1998) en France et (Zaki and Ogihara, 1998) aux USA. Les auteurs ont soulignés le lien entre la fouille de motifs fréquents et l'analyse formelle de concepts (AFC) (Barbut and Monjardet, 1970; Wille, 1982; Davey and Priestley, 1990), dans lequel les transactions représentent les objets, les items sont les attributs et la base de données joue le rôle du contexte d'analyse (la relation binaire entre les deux) noté par le triplet (T,A,D). Dans les papiers de Pasquier et al. et M.Zaki et al., il a été montré que la fouille de motifs fréquents peut être réduite à la construction seulement du treillis des motifs fréquents fermés dit treillis des concepts ou de Galois dans l'AFC. Ce dernier treillis est un sous ordre du treillis complet des sous ensembles de A, souvent de taille moins réduite. L'opérateur γ est ici la composition des deux applications $f : 2^T \to 2^A$ et $g : 2^A \to 2^T$ suivantes qui lient les transactions aux items et inversement, c'est-à-dire les fonctions définies comme suit où $X \subseteq A$ et $Y \subseteq T$:

$$f(Y) = \{a \in A \mid \forall y \in Y \ : \ y \text{ contient a}\} \tag{3.1}$$

$$g(X) = \{t \in T \mid \forall a \in X \ : \ t \text{ contient a}\} \tag{3.2}$$

Le couple de fonctions (f,g) est une connexion de Galois où $\gamma = fog$ (resp. $\gamma = gof$) représente l'opérateur de fermeture (vérifie les conditions de fermeture vues en section 1.1) pour 2^A (resp. 2^T) ; il associe des motifs à eux même (resp. des transactions à elles-mêmes). La fermeture d'un motif x correspond donc à l'unique sur-motif maximal ayant le même support que x. La collection de MFF constitue donc une autre représentation compressée pour les motifs fréquents d'un ensemble de données. Cette représentation est complète et sans perte d'information (Pasquier et al., 1998), puisque sa connaissance permet de dériver le reste

des motifs fréquents avec les valeurs exactes de leurs supports à l'opposé de l'ensemble de MFM qui, comme déjà mentionné, ne préserve pas les supports. Il est aisé de constater que tout motif fréquent maximal est aussi fréquent fermé (l'inverse n'est pas vrai), ce qui signifie que l'ensemble des MFM $\mathscr{M}$ est inclut dans l'ensemble des MFF $\mathscr{C}$, qui est à son tour inclut dans l'ensemble de tous les motifs fréquents $\mathscr{F}$ ($\mathscr{M} \subseteq \mathscr{C} \subseteq \mathscr{F}$). Le support d'un motif x est celui de sa fermeture : $\text{sprt}(x) = \max\{\text{sprt}(y) \mid x \subseteq y \in \mathscr{C}\}$. Notons enfin que sur le plan théorique, il a été prouvé l'équivalence des problèmes suivants : la recherche de rectangles maximaux d'une relation binaire, l'énumération de toutes les cliques bipartites maximales d'un graphe et l'identification de motifs fermés d'une base transactionnelle (Zaki and Ogihara, 1998; Li et al., 2005). Pour la base de notre exemple de référence, il existe sept MFF $\mathscr{C} = \{\{a\},\{a,b\},\{a,c\},\{a,e\},\{a,c,d\},\{a,c,e\},\{a,c,d,e\}\}$ qui sont montrés dans la figure 3.9 (notons la présence des deux MFM).

Dans la suite, nous décrivons brièvement trois algorithmes représentatifs d'extraction de MFF.

A-Close

A-Close (Pasquier et al., 1999) est un algorithme horizontal itératif qui étend Apriori pour découvrir les MFF. En exploitant les propriétés de fermeture, A-close identifie, niveau par niveau, d'abord l'ensemble de motifs *générateurs* qui permettent de dériver tous les MFF de la base par application de l'opérateur de fermeture. Un motif g est un générateur du motif fermé c s'il est le motif minimal (selon l'inclusion) dont la fermeture est c, autrement dit, le motif ayant le même support que c. À l'étape k , les k-générateurs candidats sont construits à partir des $(k-1)$-générateurs fréquents en examinant les supports calculés par balayage de la base. Tout générateur infréquent ou non minimal est alors supprimé. Par la suite, un ultime scan est effectué afin de calculer les fermetures des générateurs trouvés qui constituent la collections de MFF contenus dans la base.

Charm

Charm (*Closed Association Rule Mining, avec un H sans correspondance*) (Zaki and Hsiao, 2002) est un algorithme de la famille des méthodes verticales initiées avec Eclat. Il utilise donc, d'une part, des paires motifs-listes de transactions associées et décompose l'espace de recherche, représenté par un arbre, en plusieurs

classes d'équivalence qui sont traitées séparément. D'autre part, il exploite l'astuce des diffsets proposée dans dEclat pour les calculs des supports. Étant donné que la découverte de MFF requière des tests de fermeture, cet algorithme introduit plusieurs mécanismes d'élagage permettant de sauter des niveaux complets de l'espace du problème et ainsi de générer moins de candidats à vérifier. Pour ce faire, Charm examine pour chaque cas les tidlists impliquées dans l'extension d'un candidat en cours et distingue trois cas de figures impliquant les deux paires $(X_i, tidlist(X_i))$ et $(X_j, tidlist(X_j))$ selon que leurs listes de transactions sont égales, l'une contenue dans l'autres ou différentes : *(i)* le cas où $tidlist(X_i)) = tidlist(X_j)$, qui conduit aussi à l'égalité des fermetures de X_i, X_j et celle de leur union $X_i \cup X_j$. Pour ce cas toute occurrence de X_i est remplacée par $X_i \cup X_j$ et la branche entière de X_j est élaguée, *(ii)* le cas où $tidlist(X_i) \subset tidlist(X_j)$ (resp. $tidlist(X_j) \subset tidlist(X_i)$) qui mène à l'inégalité des fermetures des deux motifs, mais à l'égalité de celle de X_i (resp. X_j) avec la fermeture de l'union $X_i \cup X_j$ qui implique que toute occurrence de X_i (resp. X_j) est remplacée par $X_i \cup X_j$ tout en gardant cette fois la branche de X_j (resp. X_j), *(iii)* ce dernier cas est rencontré lorsque les deux listes de transactions sont différentes qui n'offre aucune optimisation. En opérant ainsi, il est prouvé que Charm produit correctement la liste des motifs fréquents fermés. En effet, il aisé de constater que cet algorithme explore son espace et que les branches éliminées sont celles correspondantes à des motifs non fréquents, ou fréquents mais non fermés.

LCM

LCM (*Linear time Closed itemset Miner*) (Uno et al., 2004) est un algorithme efficace de fouille de motifs fréquents. Cet algorithme n'appartient pas, comme on peut le croire de par sa position, à l'approche projective ; mais il fait partie plutôt des algorithmes hybrides (LCM est considéré à la fois comme un algorithme vertical et projectif). Nous l'avons présenté ici en raison de sa popularité, étant donné qu'il a remporté la deuxième compétition du *workshop* organisé en 2004 (Bayardo et al., 2004). LCM est récusrsif et fondé sur une relation entre les motifs fréquents fermés tirée du domaine des cliques bipartites. Son efficacité est due à de simples structures de données exploitée de façon intelligente qu'il associe à plusieurs techniques pour améliorer le temps de calcul. L'espace du problème étant un arbre de préfixe parcouru ici en DFS, et ne nécessite aucun stockage des résultats précédents trouvés. Par ailleurs, LCM utilise des tableaux comme structures de données pour

représenter les transactions et les items qu'il ordonne afin d'accélérer les recherche. Il bénéficie également de deux autres idées : d'un coté, l'*occurrence deliver* consistant à anticiper le calcul de tous les successeurs du motif en cours durant le même scan, et la technique des diffsets proposée dans dEclat. Vu que ces deux dernières heuristiques sont contradictoires, LCM les combine dans un schéma hybride de façon à trouver un compromis selon que la base à fouiller est dense ou sparse. Notons en fin, que cet algorithme propose aussi deux adaptations LCMfreq et LCMmax permettant, respectivement, l'extraction de tous les motifs fréquents et ceux fréquents maximaux.

3.4 Énumération incrémentale

Cette classe de méthodes concerne les projets de fouille pour les base de données dynamiques. Il est évident que les connaissances extraites deviennent caducs si l'ensemble de données est évolutif objet de mises à jour (insertion, suppression ou modification), ou si les paramètres du problèmes sont ajustés. Dans ces cas de figure, l'approche naïve consistant à relancer le processus de fouille à partir de zéro est manifestement impraticable. Il serait, par contre, judicieux de maintenir les résultats obtenus par l'exploitation des connaissances extraites actuelles en les confrontant seulement avec les fragments altérés de la base. Dans cette catégorie, plusieurs algorithmes ont été proposés. Pour citer des exemples, nous mentionnons les tous premiers travaux de Chen et al. (Cheung et al., 1996; Cheung et al., 1997), suivis par l'approche qui utilise les treillis de Galois de (Valtchev et al., 2002), puis l'algorithme efficace ZIGZAG (Veloso et al., 2002) qui s'attaque à l'extraction et la maintenance des motifs fréquents maximaux.

3.5 Aspects avancés

La fouille de motifs fréquents est un sujet très vaste qui a suscité d'innombrables contributions et il est hors de question de pouvoir le cerner par un rapport si limité. Dans cette dernière section, nous soulignons sommairement quelques aspects que nous pouvons qualifiés d'avancés dans la mesure où ils traitent des extensions au problème standard de la fouille de motifs.

3.5.1 Retour sur l'intérêt de motifs/règles

La totalité des solutions discutées dans cette thèse concernent le framework de fouille de motifs classique basé sur les mesures support et confiance. Il a été noté, dans plusieurs études, que se limiter aux règles fortes vérifiant les deux seuils support et confiance, s'il permet de générer certes des connaissances dignes d'intérêt, peut souvent conduire à l'omission d'autres connaissances utiles ; liées à un support plus bas par exemple. De plus, le réglage insouciant de ses paramètres est critique et peu produire des résultats étranges. En conclusion, le choix d'une configuration de fouille est fortement tributaire au type de données et au domaine d'application. Par conséquent, d'autre mesures pour quantifier l'importance d'un motif ou d'une règle ont été proposés. Certains ont pour but de renforcer le framework de base par des mesures de corrélation comme le test χ^2, lift (pour deux motifs, représente le rapport entre la probabilité de lunion sur le produit des probabilités individuelles) etc. (Han et al., 2011; Han et al., 2007), d'autres suggèrent de nouveaux frameworks plus objectifs (Steinbach et al., 2007).

3.5.2 Motifs et fouilles complexes

Un motif dans le framework classique est simplement un ensemble d'items. Or dans plusieurs applications réelles, nous avons en face des motifs non triviaux comme les catégories de produits dans les grands magasins, des structures (données multidimensionnelles, sections de documents, pages web, réseaux sociaux, données biologiques, etc.) qui constituent des terrains fertiles à explorer. Il est donc naturel d'envisager l'extraction de ce genre de motifs plus élaborés. Ainsi, le même framework a été vite étendu à, par exemple, la fouille de séquences (Agrawal and Srikant, 1995), d'arbres (Zaki, 2002), de graphes (Inokuchi et al., 2000), et enfin la fouille de motifs quantifiés, c'est-à-dire des bases ayant des quantités associées aux items dans les transactions (Wang et al., 2000).

Par ailleurs, certains contextes imposent, pour le framework classique ou ses extensions, des conditions de travail trop contraignantes. Il s'agit, à titre d'exemple, de fouille en situation de mémoire limitée, ou la présence de données en ligne, temps réel, ou en flots. Ces situations nécessitent une prise en charge poussée, étant donné que ces contextes ne tolèrent ni repassage sur les données, ni leurs sauvegarde ou temporisation (Han et al., 2007).

3.5.3 Parallélisation et distribution

Il a été observé très tôt (Park et al., 1995b; Agrawal and Shafer, 1996) que la complexité du problème de la fouille de motifs fréquents dépasse les schéma de traitements séquentiels et les architectures conventionnelles. Dès lors, le recours aux paradigmes de traitements parallèles et/ou distribués constitue légitimement une alternative pour faire face à l'espace énorme impliquant des données massives et les traitements intensifs qu'engendre le problème étudié. Ce courant de recherche a récemment été stimulé par l'apparition d'architectures et modèles de calcul plus adéquats. Outre les premiers travaux des équipes de Park et Agrawal (Park et al., 1995b; Agrawal and Shafer, 1996), plusieurs autres travaux se sont intéressés à l'extraction parallèle/distribuée de motifs fréquents. Par exemple, *FPF* (Li et al., 2008) est un des algorithmes célèbres de fouille parallèle de motifs fréquents ; il étend l'algorithme FPGrowth en se basant sur le modèle MapReduce (Dean and Ghemawat, 2008). Sur les nouveaux processeurs disposant du multicore, $PLCM_{QS}$ (Négrevergne et al., 2010) est une extension parallèle de l'algorithme *LCM*. Un algorithme distribué et parallèle pour la fouille incrémentale de motifs fréquents destiné aux bases de données dynamiques est présenté dans (Otey et al., 2004). Ce dernier parallélise l'algorithme ZIGZAG (Veloso et al., 2002) de fouille incrémentale de motifs fréquents.

3.5.4 Métaheuristiques pour l'extraction de règles d'association

Pour clore ce chapitre, nous avons jugé opportun de mentionner un autre courant de recherche sur le sujet, qui vise plus particulièrement la phase de découverte de règles d'association en exploitant les métaheuristiques (Osman and Laporte, 1996). Ces dernières, sont des classes de méthodes générales et approximatives conçues pour résoudre des problèmes complexes d'optimisation, où les techniques d'optimisation classiques ont échoué. Les métaheuristiques tirent leur origines de différents phénomènes observés dans plusieurs domaines tels que : l'évolution en biologie, la dynamique de certain systèmes physiques, la propagation de la chaleur en métallurgie, la mécanique statistique, etc. Ces approches ont connu un succès grandissant ces dernières années et un large étendu d'applications ; elles incluent, entre autres, : le recuit simulé, la recherche tabou, les réseaux de neurones, les algorithmes évolutionnaires, la programmation génétique, les colonies de fourmis, les optimisations par essaims de particules, et les approches hybrides (Talbi, 2002), etc. Le principe étant

d'explorer efficacement (en temps raisonnable) l'espace de recherche, sans pour autant songer à l'énumération totale de l'ensemble de solutions, afin de trouver des solutions quasiment optimales.

En assimilant la question à un problème d'optimisation combinatoire, la stratégie est de démarrer avec une(des) solution(s) réalisable(s), selon que l'approche est à base de solution unique ou de population d'entre elles, puis de progresser selon une(des) fonctions objectif(s) à optimiser (maximiser/minimiser). Le recours à ces méthodes pour l'extraction des règles d'association est justifié par la nature combinatoire du problème et sa complexité exponentielle (Dhaenens and Jourdan, 2016).

Suite aux extensions introduites au problème standard, où les règles d'association quantitatives et floues ont été d'abord explorées (Kuok et al., 1998; Freitas, 2013), plusieurs métaheuristiques ont été appliquées à l'extraction des règles d'association. Celles-ci introduisent des mesures autre que le support et la confiance (compréhension, intérêt, etc.) et les combinent dans des optimisations mono ou multi-objectifs. C'est ainsi que les algorithmes génétiques ont été utilisés dans (Martínez-Ballesteros et al., 2016) et des métaheuristiques multi-objectifs exploitées par (Ghosh and Nath, 2004) et (Kaya, 2006). Des stratégies évolutionnaires multi-objectifs basées sur des populations BSO (Bee Swarm optimization) sont aussi proposées par (Djenouri et al., 2012) ; récemment, un algorithme de type Bat est introduit dans (Heraguemi et al., 2016). Pour finir, le papier (Khabzaoui et al., 2008) propose une d'hybridation entre ces méthodes optimisations évolutionnaires et celles exactes pour la découverte de règles d'association intéressantes.

3.6 Conclusion

Dans ce chapitre nous avons dressé un panorama des travaux algorithmiques en fouilles de motifs fréquents. Nous avons vu que l'ensembles des travaux de la littérature sont classés dans trois principales approches : horizontale par niveau, verticales ou projectives. Dans le chapitre suivant, nous introduisons notre contribution dans ce domaine. Nous élaborons un modèle formel générique qui permet d'unifier ces approches. Nous étudions ses propriétés et prouvons nos propositions.

Troisième partie

Contribution

CHAPITRE 4

Un modèle unificateur basé sur les séries formelles

Vous voyez les choses et vous dites : pourquoi ?
Moi, je rêve de choses qui n'ont jamais existé et je dis : pourquoi pas ?

George Bernard Shaw

Sommaire

Dans le présent chapitre, nous allons nous pencher sur notre contribution et expliquons notre démarche d'adoption du modèle des séries formelles pour attaquer le problème de la fouille de motifs fréquents. À ce niveau, il paraît tout à fait naturel de s'interroger sur la raison de cette transposition d'un problème élémentaire à un modèle qui semble, a priori, trop compliqué devant la tâche à accomplir. Rappelons, comme déjà tracé dans l'introduction de ce manuscrit, que notre objectif est la formalisation de cette tâche primaire en extraction de connaissances de manière à permettre son approfondissement, et d'unifier les approches algorithmiques proposées pour sa résolution. Il est capital de souligner que le modèle que nous proposons instaure une vision formelle de haut niveau claire et générique pour ce problème basique mais surtout extensible à d'autres problèmes similaires. Le développement, tout au long de ce chapitre, nous conduit progressivement à voir ce problème comme celui du calcul d'une série formelle. Nous élaborons pas à pas nos constructions et prouvons leurs propriétés les plus remarquables. Dans la dernière section du chapitre, nous confrontons notre modèle aux travaux existants et montrons son caractère unificateur entrainant à la conviction que ces derniers peuvent en être dérivés comme instances particulières. Parallèlement, nous argumentons aussi le lien avec les automates à multiplicité qui constituent l'outil permettant de passer à la réalisation.

4.1 Modélisation

Avant d'entamer notre modèle, nous nous permettons de revoir sommairement les concepts de base afférents au problème de la fouille de motifs fréquents et d'introduire les notations qui seront utilisées dans la suite de cette étude.

Soit $A = \{a_1, a_2, \ldots, a_m\}$ un alphabet de m symboles appelés *items*. Ceux-ci peuvent désignés, selon le domaine d'application, des produits achetés d'un supermarché, des pages visités lors des surfs ou généralement une collection d'attributs ou d'événements.

Un *motif* est un sous ensemble de A. Il est appelé k-motif si son cardinal est k. Une *transaction* t_i est un ensemble non vide d'items distinguée par son unique identificateur i. Une *base de données* D est un ensemble de n transactions que nous dénotons comme un multi-ensemble : $D = \{t_1, t_2, \ldots, t_n\}$.

Dans une base de données D, le *support* d'un motif x est le nombre de transactions contenant x, i.e., pour lesquelles x est un sous-ensemble.

$$\text{sprt}(x,D) = |\{t_k \in D \mid x \subseteq t_k\}| \tag{4.1}$$

Un motif x est *fréquent* si son support dépasse un seuil minimal s donné comme paramètre ;

$$\text{sprt}(x,D) \geq s \tag{4.2}$$

Un motif x est *fréquent maximal* si aucun des ses sur-motifs n'est fréquent. Le problème de la fouille de motifs consiste à extraire l'ensemble $\mathscr{F}$ de tous les motifs fréquents.

$$\mathscr{F} = \{x \subseteq A \mid \text{sprt}(x,D) \geq s\} \tag{4.3}$$

4.1.1 Fouille de motifs comme un polynôme

Dans les préliminaires, nous avons vu qu'un polynôme est une série formelle finie. Partant du constat que l'univers que nous modélisons est formé par des composants finis (alphabet, motifs, transactions, base de données), nous choisissons donc une modélisation reposant sur les polynômes.

Dans ce qui suit, nous considérons des éléments du monoïde libre A^* et les coefficients associés pris dans le semi-anneau des entiers naturels ou de comptage $(\mathbb{N}, +, \times, 0, 1)$. L'idée principale derrière notre formalisme est de coder un motif par un mot sur l'alphabet A et, de même, tous ses sous-motifs par un polynôme sur le semi-anneau de comptage. Après avoir déduit le polynôme d'une base de données en effectuant la somme de ceux correspondant aux transactions les constituant, la question est alors d'extraire de ce polynôme l'ensemble des termes où la condition du support de l'inégalité (4.2) est remplie.

Pour commencer, supposons, sans perte de généralité, que l'alphabet A est trié selon un ordre total quelconque [1], où nous pouvons écrire : $A = \{a_1, \ldots a_m\}$ avec $\varepsilon < a_1 < \ldots < a_m$.

Maintenant, nous représentons un k-motif $x = \{a_{i_1}, \ldots, a_{i_k}\}$ par le mot $w(x)$ de longueur k construit par concaténation de ses items suivant l'ordre prédéfini adopté. Nous écrivons :

$$w(x) = a_{i_1} \ldots a_{i_k}, \text{ tel que } a_{i_1} < \ldots < a_{i_k} \tag{4.4}$$

1. Dans nos exemples, nous admettons pour simplifier l'ordre lexicographique.

Par la suite, nous confondons un motif x et le mot $w(x)$ qui l'encode. Autrement dit, au lieu d'écrire, par exemple, $x = \{a,b,c\}$ nous adoptons simplement la notation $x = abc$ si l'ordre lexicographique est choisi. Évidemment, le motif nul $\emptyset$ sera représenté par le mot vide ε.

Il est aussi important de mentionner que puisque nous traitons essentiellement des motifs, qui sont rien d'autre que des ensembles d'items, nous suivons dans la suite une concaténation particulière de motifs. Cette opération doit en fait se conformer à deux conditions triviales : d'une part, l'absence de répétition d'items, et d'autre part le respect de l'ordre fixé au départ. À titre d'exemple, la concaténation de $a_i a_k$ et $a_j a_k$ génère le motif $a_i a_j a_k$ (supposons l'ordre $a_i < a_j < a_k$) et non pas $a_i a_k a_j a_k$. Plus formellement, la concaténation de deux items de a et b produit le résultat suivant :

$$ab = \begin{cases} ab & \text{si} \quad a < b \\ ba & \text{si} \quad b < a \\ a & \text{si} \quad a = b \end{cases} \tag{4.5}$$

Définition 4.1.1 (Polynôme de sous-séquences de motif (PSM))**.** *Soit $x = a_{i_1} a_{i_2} \dots a_{i_k}$ un k-motif. Le polynôme de sous-séquences de motif (PSM) $\mathbb{S}_x$ associé à x est défini comme suit :*

$$\mathbb{S}_x = \begin{cases} 1 & si\ x = \varepsilon \\ \prod\limits_{1 \le j \le k} (a_{i_j} + 1) = (a_{i_1} + 1)(a_{i_2} + 1) \dots (a_{i_k} + 1) & sinon \end{cases} \tag{4.6}$$

La formule (4.6) ci-dessus constitue la brique de base de notre formalisation à base de polynômes. En effet, les racines de cette formule découlent de notre encodage. Premièrement, le motif nul est codé par le terme 1ε, qui peut être simplifié selon la section 1 relatives aux préliminaires par le terme constant 1. Identiquement, un singleton $\{a\}$ est représenté, en deuxième lieu, par le terme $1a$ ou simplement a. Dans le même ordre d'idées, et afin de pister l'existence d'un item donné a dans nos polynômes, nous devons considérer soit son absence matérialisée par le motif nul ou sa présence marquée par le terme associé au singleton en question. Ainsi, nous aboutissons au polynôme $(a+1)$. Notons au passage, que ce dernier polynôme est une généralisation de l'expressions $(a+\varepsilon)$ bien connue dans les langages rationnels

pour dénoter les sous-séquences d'un mot (Hopcroft et al., 2001).

Dans ce qui suit, nous dénotons, pour chaque $a \in A$, par $\overline{a}$ le polynôme $(a+1)$, et par $\overline{a_{i_1} a_{i_2} \dots a_{i_k}}$ le PSM $\mathbb{S}_x$ associé au motif $x = a_{i_1} a_{i_2} \dots a_{i_k}$. Le polynôme $\mathbb{S}_x$ est donc le polynôme qui représente tous les sous motifs de x. Par exemple, nous associons au motif $x = abc$ son polynôme de sous-séquences $\mathbb{S}_x = \overline{abc} = (a+1)(b+1)(c+1)$, ce qui nous donne le polynôme : $1 + a + b + c + ab + ac + bc + abc$. Notons qu'étant donné que ε est une sous-séquence de tout mot, chaque polynôme doit alors inclure au moins le terme constant 1.

Le passage au polynôme de sous-séquences d'une base de données D est simple. Ce dernier sera dérivé à partir de ceux associés aux transactions qui la forment.

Définition 4.1.2 (Polynôme de sous-séquences de base de donnée (PSB))**.** *Soit $D = \{t_1, \dots, t_n\}$ une base de n transactions. Le polynôme de sous-séquences de base de donnée (PSB) $\mathbb{S}_D$ associé à D est la somme des n polynômes de sous-séquences de ses transactions :*

$$\mathbb{S}_D = \sum_{i=1}^{n} \mathbb{S}_{t_i} \tag{4.7}$$

Commençons par illustrer ces concepts à l'aide d'un exemple de référence. La table 4.1 montre une base de données de huit transactions, où la troisième colonne donne, à travers l'équation (4.6), le PSM de chaque transaction [2]. Nous avons calculé aussi dans la dernière ligne le polynôme de sous-séquences de la base toute entière via l'équation (4.7).

Comme début, il découle de la fondation que nous venons d'élaborer le premier résultat trivial suivant. La proposition ci-après établit le lien entre le support d'un motif et son coefficient dans le polynôme de sous-séquences de la base de données.

Proposition 4.1.1. *Soit $D = \{t_1, \dots, t_n\}$ une base de données de n transactions, et $\mathbb{S}_D$ le PSB associé. Si nous considérons la forme canonique réduite de $\mathbb{S}_D$ comme somme de termes $\mathbb{S}_D = \sum_{w \in A^*} \langle \mathbb{S}_D, w \rangle w$ alors :*

$$\langle \mathbb{S}_D, w \rangle = \mathrm{sprt}(w, D) \tag{4.8}$$

2. Rappelons qu'une transaction est un motif.

TABLE 4.1 – Base de données de transactions et les polynômes associés

i	t_i	$\mathbb{S}_{t_i}$
1	abc	$1+a+b+c+ab+ac+bc+abc$
2	abce	$1+a+b+c+e+ab+ac+ae+bc+be+ce+abc+abe+ace+bce+abce$
3	abd	$1+a+b+d+ab+ad+bd+abd$
4	acd	$1+a+c+d+ac+ad+cd+acd$
5	acde	$1+a+c+d+e+ac+ad+ae+cd+ce+de+acd+ace+ade+cde+acde$
6	ade	$1+a+d+e+ad+ae+de+ade$
7	bcd	$1+b+c+d+bc+bd+cd+bcd$
8	bce	$1+b+c+e+bc+be+ce+bce$

$$\mathbb{S}_D = \mathbf{8}+\mathbf{6a}+\mathbf{5b}+\mathbf{6c}+\mathbf{5d}+\mathbf{4e}+\mathbf{3ab}+\mathbf{4ac}+\mathbf{4ad}+\mathbf{3ae}+\mathbf{4bc}+2bd+2be+\mathbf{3cd}+\mathbf{3ce}$$
$$+2de+2abc+2abd+abe+2acd+2ace+2ade+bcd+2bce+cde+abce+acde$$

Démonstration. Il est facile de constater que w est un motif et que $\langle \mathbb{S}_D, w \rangle$ est un coefficient dans $\mathbb{N}$ qui représente son support dans la base de données. En effet, un motif w possède, selon la définition 4.1.1, l'unité (1) comme coefficient dans tout transaction t_i où il apparait ; par conséquent son coefficient dans le PSB, qui ne fait que sommer via la formule 4.1.2 ceux de ses transactions, est exactement le nombre de toutes les transactions qu'elles le couvrent, c-à-dire, son support. □

Dans notre exemple, nous pouvons, sans peine, observer que les motifs suivants : ε, d, bc, $acde$ ont respectivement $8, 5, 4$, et 1 comme valeurs de support.

4.1.2 Algorithme général

Étant donné à présent un polynôme $\mathbb{S}$ sur un alphabet A et à coefficients dans un semi-anneau $\mathbb{K}$, et un seuil de support minimal s spécifié par l'utilisateur ; nous désirons extraire à partir du polynôme $\mathbb{S}$ le polynôme $\mathbb{F}$ défini ainsi :

$$\langle \mathbb{F}, w \rangle = \begin{cases} \langle \mathbb{S}, w \rangle & \text{si } \langle \mathbb{S}, w \rangle \geq s \\ \\ 0 & \text{sinon} \end{cases} \tag{4.9}$$

Nous cherchons donc tous les mots de l'étendue du polynôme $\mathbb{S}$ ayant des coefficients supérieurs à s. Dans notre exemple, et si nous fixons à trois le seuil minimal du support, ces termes sont la collection montrée en gras dans la table 4.1.

Notons que puisque la notion de fréquence est une propriété relative ; car un item fréquent/infréquent peut, suite à des mises à jour opérées sur la base, devenir infréquent/fréquent, nous conservons dans notre modèle, similairement à de nombreux travaux (Cheung and Zaïane, 2003; Goethals, 2004), tous les items indépendamment de leurs supports initiaux. Nous avons estimé pour plusieurs raisons telles que les capacités de mémoires vives offertes par les machines modernes que ce choix rend le modèle plus flexible notamment en cas de bases de données dynamiques. Si ce choix s'avère, toutefois, à écarter pour divers arguments, nous pouvons toujours revenir à une phase de prétraitement dont la mission serait de filtrer du modèle, qui sera utilisé dans la phase de fouille, les items inutiles soit en amont ou en val de sa construction.

L'exploration de l'espace du problème, de nature exponentielle, est assurée par l'algorithme générique 6. Ce dernier liste les motifs ciblés en invoquant DISCOVER-FI$(\mathbb{S}, s, \varepsilon, \emptyset)$, qui exploite la propriété Apriori de la proposition 4.1.2 permettant d'élaguer significativement l'espace du problème.

Algorithme 6 DISCOVER-FI$(\mathbb{S}, s, w, \mathscr{F})$

Entrée : Le polynôme $\mathbb{S}$, le seuil de support s, et un motif $w = w_1 w_2 \dots w_{|w|}$
Sortie : L'ensemble des motifs fréquents.

```
pour tout a > w_|w| faire
    si ⟨S, wa⟩ ≥ s alors                    // si l'extension wa est fréquente
        F ← F ∪ {(wa, ⟨S, wa⟩)}             // l'ajouter dans la sortie
        DISCOVER-FI(S, s, wa, F)            // continuer l'exploration
    fin si
fin pour
```

Proposition 4.1.2 (Apriori (Agrawal and Srikant, 1994))**.** *Soit D une base de transactions et w_1, w_2 deux motifs. Si $w_1 \subseteq w_2$ alors* $\text{sprt}(w_1, D) \geq \text{sprt}(w_2, D)$

Dans notre modèle l'heuristique Apriori est la suivante :

Corollaire 4.1.3. *Soit D une base de transactions et w_1, w_2 deux motifs. Si $w_1 \preccurlyeq w_2$ alors* $\text{sprt}(w_1, D) \geq \text{sprt}(w_2, D)$

Il est évident que la complexité de l'Algorithme 6 dépend du nombre de motifs fréquents et du coût du test de fréquence. Ce dernier est tributaire à son tour de la

longueur du motif w concerné et du temps nécessaire au calcul de son coefficient $\langle \mathbb{S}, w \rangle$ dans le modèle adopté.

Afin d'élaborer une implémentation efficace de l'Algorithme 6, il est primordial d'utiliser une structure de données optimale, qui outre qu'elle doit jouir d'une taille réduite, elle devait également offrir un coût minimal pour le calcul des coefficients.

Dans la suite, nous allons montré que le problème de la fouille de motifs peut prétendre à une formulation par des séries formelles (Salomaa et al., 1978; Berstel and Reutenauer, 1988). Ce formalisme se prête parfaitement à une implémentation à l'aide d'automates à multiplicité, que nous proposons de la montrer ici.

4.2 Automate à multiplicité de motifs fréquents

Soit $\mathscr{S}_D$ un automate à multiplicité reconnaissant le polynôme de sous-séquences $\mathbb{S}_D$ associé à la base de transactions D comme défini en haut. Le calcul du coefficient $\langle \mathbb{S}_D, w \rangle$ d'un motif w dans le polynôme $\mathbb{S}_D$ est équivalent à la détermination de sa multiplicité (poids) dans l'automate $\mathscr{S}_D$. Par conséquent, la complexité de ce calcul est soumise au type du dernier automate (déterministe, non déterministe, asynchrone, etc.) et de sa taille. Dans ce qui suit, nous proposons un automate particulier réduit par rapport à la taille de la base D qui réalise le polynôme $\mathbb{S}_D$.

Afin de construire l'automate $\mathscr{S}_D$ et puisque la technique de la juxtaposition des préfixes communs a prouvé son intérêt dans ce problème (Zaki, 2000; Han et al., 2000; Valtchev et al., 2002; Cheung and Zaïane, 2003; Totad et al., 2012), nous allons passer par un autre type d'automate qui va nous aider dans la définition de l'automate cible $\mathscr{S}_D$. Cet automate intermédiaire est l'automate à multiplicité préfixiel $\mathscr{P}_D$ introduit ci-après. Pour ce faire, commençons d'abord par définir le concept de polynôme préfixiel.

Définition 4.2.1 (Polynôme préfixiel de motif (PPM))**.** *Soit $x = a_{i_1} a_{i_2} \dots a_{i_k}$ un k-motif, le polynôme préfixiel de motif (PPM) associé à x, noté $\mathbb{P}_x$, est défini comme suit :*

$$\mathbb{P}_x = \sum_{u \in \text{Pref}(x)} u \tag{4.10}$$

Cela signifie que le polynôme préfixiel d'un motif est la somme de tous ses préfixes. Par exemple, le polynôme préfixiel du motif abc est $\mathbb{P}_{abc} = 1 + a + ab + abc$.

De façon similaire, nous pouvons passer au polynôme préfixiel d'une base de données.

Définition 4.2.2 (Polynôme préfixiel de base de données (PPB)). *Soit $D = \{t_1, \ldots, t_n\}$ une base de n transactions. Le polynôme préfixiel de base de données (PPB) associé à D, dénoté $\mathbb{P}_D$, est la somme des n polynômes préfixiels associés à ses transactions :*

$$\mathbb{P}_D = \sum_{i=1}^{n} \mathbb{P}_{t_i} \tag{4.11}$$

Notons que cette dernière définition implique que Étendue$(\mathbb{P}_D) = \text{Pref}(D)$. Autrement dit, l'étendue du polynôme préfixiel d'une base D représente l'ensemble des préfixes de ses transactions. Nous donnons ci-dessous le polynôme préfixiel de la base de transactions de notre exemple de référence après quelque développements : $\mathbb{P}_D = 8 + 6a + 2b + 3ab + 2ac + ad + 2bc + 2abc + abd + 2acd + ade + bcd + bce + abce + acde$.

4.2.1 Automate à multiplicité préfixiel

À ce niveau, nous estimons que la construction d'un automate à multiplicité pour le polynôme des sous-séquences $\mathbb{S}_D$ de la base passe par la définition d'un autre automate qui calcule le polynôme préfixiel $\mathbb{P}_D$. Il existe plusieurs automates à multiplicité qui réalisent ces polynômes. Ici, nous donnons un automate particulier crisp-déterministe (Droste et al., 2010) qui reconnaît le polynôme préfixiel $\mathbb{P}_D$, que nous allons légèrement modifié afin d'obtenir un autre qui réalise notre polynôme initial $\mathbb{S}_D$ des sous-séquences de la base.

Définition 4.2.3 (Automate à multiplicité préfixiel (AMP)). *Soit $\mathbb{P}_D$ le polynôme préfixiel d'une base de transactions D. L'automate à multiplicité préfixiel associé $\mathscr{P}_D = (Q, A, \mu, \lambda, \gamma)$ est défini comme suit :*

- $Q = \text{Étendue}(\mathbb{P}_D)$,
- $\mu(u) = \begin{cases} 1 & si\ \ u = \varepsilon \\ & \forall u \in Q \\ 0 & sinon \end{cases}$
- $\lambda(u, a, ua) = \begin{cases} 1 & pour\ u\ et\ ua \in Q,\ et\ a \in A, \\ 0 & sinon \end{cases}$

— $\gamma(u) = \langle \mathbb{P}_D, u \rangle, \forall u \in Q$.

Notons que les chemins réussis dans notre automate à multiplicité préfixiel $\mathscr{P}_D$, c'est à dire ceux ayant des multiplicités ou des poids non nuls, sont uniquement ceux émanant de l'unique état initial ε. Dans ces cas, la multiplicité d'un chemin étiqueté u est égale à $\gamma(u)$, étant donné que les deux fonctions μ et λ sont crisps (Droste et al., 2010). Dans cet automate, μ est égale à 1 pour l'unique état initial et 0 dans le cas contraire, et également $\lambda(v,a,va) = 1$ pour tout $v, va \in Q$.

Afin de simplifier la lecture, nous appelons par la suite tout automate $\mathscr{A}$ qui réalise un polynôme $\mathbb{P}_D$ un AMP si et seulement si il est isomorphe à $\mathscr{P}_D$(noté $\mathscr{A} \cong \mathscr{P}_D$). Un automate isomorphe à l'automate à multiplicité préfixiel associé à la base de donnée de notre exemple de référence est montré dans la figure 4.1, où les multiplicités d'entrée (ou initiales) nulles sont omises pour des raisons de clarté.

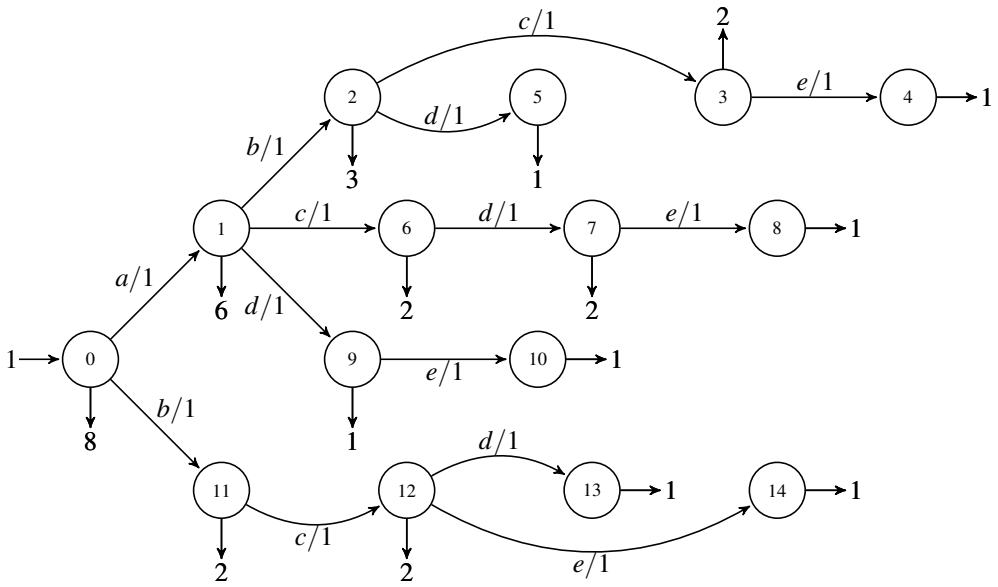


FIGURE 4.1 – Un AMP associé à l'exemple de référence

Lemme 4.2.1. *Pour une base de données transactionnelles D, l'AMP $\mathscr{P}_D$ réalise le polynôme $\mathbb{P}_D$.*

Démonstration. Par construction. Il est aisé de constater que l'automate booléen dérivé de $\mathscr{P}_D$ (ce dernier dépourvu de ses multiplicités) reconnaît l'étendue du polynôme préfixiel $\mathbb{P}_D$. En effet, $\mathscr{P}_D$ a seulement un seul état initial ε, et tous les états sont finaux et associés à des mots de l'étendue de $\mathbb{P}_D$. De plus, une transition, si elle

existe, entre un état u est effectuée par des items de l'alphabet A menant à ua, qui est toujours un mot de l'étendue de $\mathbb{P}_D$. Par ailleurs, la multiplicité de chaque mot de l'étendue de $\mathbb{P}_D$ dans l'automate est exactement le coefficient qui lui correspond vu que $\gamma(u) = \langle \mathbb{P}_D, u \rangle$. □

La définition 4.2.3 introduit l'automate à multiplicité d'une base de transactions à partir de son polynôme préfixiel associé. Dans la suite, nous donnons une procédure de construction pour cet automate, qui peut être réalisée de différentes façons : progressivement en considérant une transaction à la fois ou un ensemble de transactions, ou encore en bloc. Cette procédure constitue un algorithme incrémental et général de construction d'un AMP associé à une base de transactions D. L'idée est d'élaborer cet automate en l'assimilant à une déterminisation en utilisant une extension de la fameuse procédure de constructions des sous ensembles accessibles pour les automates classiques (Hopcroft et al., 2001).

Proposition 4.2.2. *Soit $\mathscr{A}$ et $\mathscr{B}$ deux AMP associés respectivement aux bases de données X et Y. Il existe un AMP $\mathscr{C}$, dérivé de $\mathscr{A}$ et $\mathscr{B}$, associé à la base $X \cup Y$.*

Démonstration. Nous construisons l'automate $\mathscr{C}$ en déterminisant les automates $\mathscr{A}$ et $\mathscr{B}$ vus comme un seul ayant deux états initiaux. Autrement dit, déterminiser l'automate $\mathscr{A} \cup \mathscr{B}$. La preuve complète est dans l'annexe A □

Nous illustrons, dans la figure 4.2, cette dernière construction par un exemple de fusion et de déterminisation de deux AMP simples associés aux deux bases de transactions suivantes : $X = \{ab, ac\}, Y = \{ac, ad, e\}$

4.2.2 Analyse de la construction de l'automate à multiplicité préfixiel

La procédure de la proposition 4.2.2 précédente introduit une méthode de construction d'un AMP associé à une base de données de transactions. Le plus intéressant dans cette procédure est qu'elle n'impose aucune condition sur les fragments X et Y. Elle fournit donc un algorithme de construction flexible de l'union de deux AMP ou plus, transaction par transaction ou par bloc de manière incrémentale. Par ailleurs, cette construction présente quelque propriétés de complexité remarquables que nous développerons dans les points ci-dessous.

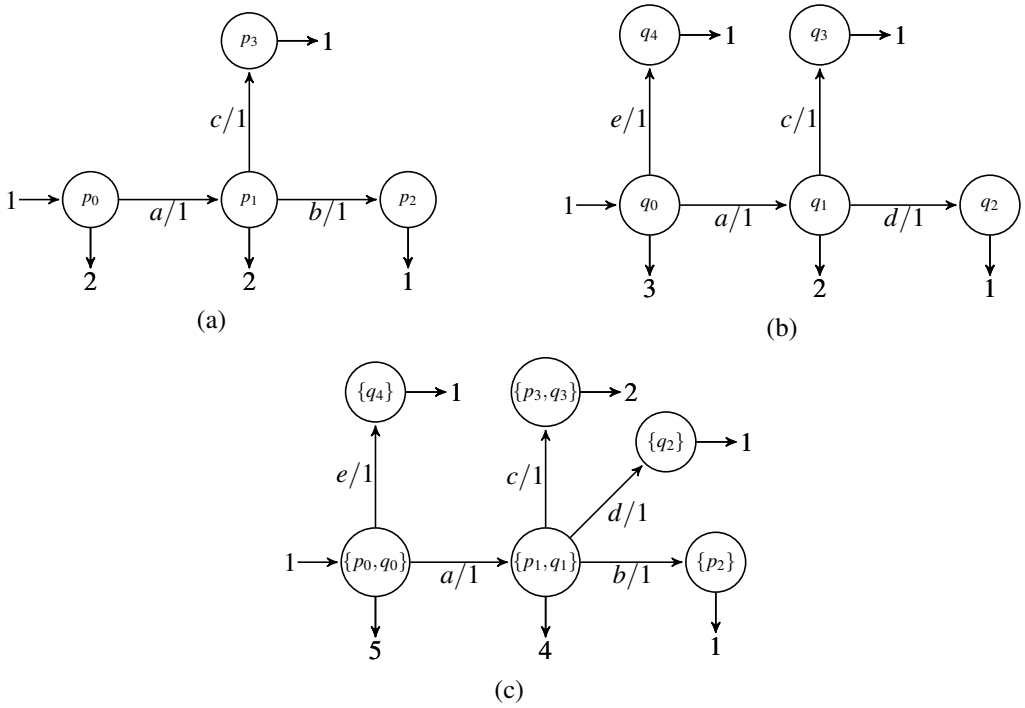


FIGURE 4.2 – Deux automates à multiplicité (a) et (b) et leur fusion par déterminisation (c)

Le lemme suivant est induit du principe d'inclusion-exclusion [3] (Guichard, 2017).

Lemme 4.2.3. *Soient X et Y deux bases de transactions. Alors :*

$$|\mathscr{P}_{X\cup Y}| \leq |\mathscr{P}_X| + |\mathscr{P}_Y| \tag{4.12}$$

Ce dernier résultat nous conduit aux corollaires suivants concernant la taille et la complexité de la construction d'un AMP associé à une base de données.

Corollaire 4.2.4. *Soit D une base de transactions. Alors :*

$$|\mathscr{P}_D| \leq |D| \tag{4.13}$$

De même, et puisque la construction des sous-ensembles durant la déterminisation d'un AMP associé à la base de données $X \cup Y$, dérivée des AMP de X et Y, est guidée par les transitions de l'automate le plus petit, nous pouvons introduire le

3. Ce principe stipule dans sa forme la plus triviale que $|A \cup B| = |A| + |B| - |A \cap B|$

lemme suivant.

Lemme 4.2.5. *Un AMP associé à la base de données $X \cup Y$ peut être construit des AMP associés à X et Y en temps $O(\min(|\mathscr{P}_X|, |\mathscr{P}_Y|))$.*

Le passage du plan réduit de deux fragments à celui global au niveau d'une base de transactions nous amène à la proposition suivante.

Proposition 4.2.6. *Soit D une base de transactions. Un AMP associé à D peut être construit en $O(|D|)$ en temps et espace.*

De plus, nous pouvons naturellement généraliser ces résultats à k ensembles de données. Cette faculté à la généralisation constitue un critère important offrant ainsi un schéma flexible de partitionnement des données avec un paramétrage fluide, qui s'avère extrêmement utile dans plusieurs aspects du problème. En effet, elle facilite la prise en charge de diverses exigences et contraintes telles que celles relatives à la mémoire, la parallélisation, et/ou l'incrémentalité, étant donnée que la granularité du partitionnement ne présente plus une difficulté : transaction par transaction (Cheung and Zaïane, 2003), ou par lots (Totad et al., 2012) considérant deux ou plusieurs fragments de données. Cette extension est introduite dans le corollaire ci-dessous.

Corollaire 4.2.7. *Soient $\mathscr{A}_1, \ldots, \mathscr{A}_k$ k AMP associés respectivement aux bases de données $X_1, \ldots, X_k$. Nous pouvons construire un AMP associé à l'union $X_1 \cup \ldots \cup X_k$ en $O(|X_1| + \ldots + |X_k|)$ en temps et espace.*

4.2.3 Vers l'automate à multiplicité de motifs

Le travail effectué jusqu'ici représente un pas manifeste vers notre objectif. Rappelons que notre but est la construction d'un automate à multiplicité qui réalise le polynôme de sous-séquences $\mathbb{S}_D$ associé à une base de données D. Nous nous permettons à ce stade de définir un nouveau polynôme, que nous appelons le polynôme préfixiel étendu. Ce dernier sert comme un intermédiaire qui nous guide à l'obtention de notre polynôme cible $\mathbb{S}_D$.

Définition 4.2.4. *Soit D une base de transactions et $\mathbb{P}_D$ le polynôme préfixiel associé. Le polynôme préfixiel étendu $\overline{\mathbb{P}_D}$ est défini comme suit :*

$$\overline{\mathbb{P}_D} = \langle \mathbb{P}_D, \varepsilon \rangle + \sum_{\substack{u \in A^* \\ a \in A}} \langle \mathbb{P}_D, ua \rangle \overline{u}a \tag{4.14}$$

Évidemment, le polynôme préfixiel étendu d'une base de données D dépend de son polynôme préfixiel. La proposition suivante résume le lien entre les trois polynômes d'une base de transactions. Les polynômes : préfixiel, préfixiel étendu et de sous-séquences.

Proposition 4.2.8. *Soient $D = \{t_1, \ldots, t_n\}$ une base de transactions, $\overline{\mathbb{P}_D}$ et $\mathbb{S}_D$ respectivement ses polynômes préfixiel étendu et de sous-séquences de motifs, nous avons :*

$$\overline{\mathbb{P}_D} = \mathbb{S}_D \tag{4.15}$$

Démonstration. Cette proposition signifie que nous pouvons dériver le polynôme de sous-séquences d'une base de données directement à partir du polynôme préfixiel en ajoutant des transitions spontanées. La preuve commence par vérifier que la proposition est vraie pour une transaction t_i d'une base de données D de n transactions. Nous assurons, ensuite, que l'égalité demeure correcte lorsque nous passons à la somme. Soit $t_i = a_{i_1} a_{i_2} \ldots a_{i_k}$ un k-motif. Selon les définitions des sections 3 et 4, et la convention $\overline{a_i} = a_i + 1$, nous avons :

$$\begin{aligned}
\mathbb{P}_{t_i} &= 1 + a_{i_1} + a_{i_1}a_{i_2} + \ldots + a_{i_1}a_{i_2}a_{i_3}\ldots a_{i_k} \\
\text{Donc,}\quad \overline{\mathbb{P}_{t_i}} &= 1 + a_{i_1} + \overline{a_{i_1}}a_{i_2} + \ldots + \overline{a_{i_1}a_{i_2}\ldots a_{i_{k-1}}}a_{i_k} \\
\text{Comme}\quad \overline{a_i} &= 1 + a_i, \text{ nous pouvons écrire :} \\
\overline{\mathbb{P}_{t_i}} &= \overline{a_{i_1}} + \overline{a_{i_1}}a_{i_2} + \ldots + \overline{a_{i_1}a_{i_2}a_{i_3}\ldots a_{i_{k-1}}}a_{i_k} \\
&= \overline{a_{i_1}}(1 + a_{i_2}) + \ldots + \overline{a_{i_1}a_{i_2}a_{i_3}\ldots a_{i_{k-1}}}a_{i_k} \\
&= \overline{a_{i_1}a_{i_2}} + \ldots + \overline{a_{i_1}a_{i_2}a_{i_3}\ldots a_{i_{k-1}}}a_{i_k} \\
&= \overline{a_{i_1}a_{i_2}}(1 + a_{i_3}) + \ldots + \overline{a_{i_1}a_{i_2}a_{i_3}\ldots a_{i_{k-1}}}a_{i_k} \\
&\ldots \\
&= \overline{a_{i_1}a_{i_2}a_{i_3}\ldots a_{i_{k-1}}a_{i_k}} \\
&= \mathbb{S}_{t_i}
\end{aligned}$$

Vérifions maintenant l'égalité entre la somme des polynômes préfixiel étendu et le

polynôme de sous-séquences de la base toute entière D.

$$\begin{aligned}
\overline{\mathbb{P}_{t_i}} &= \langle \mathbb{P}_{t_i}, \varepsilon \rangle + \sum_{\substack{u \in A^* \\ a \in A}} \langle \mathbb{P}_{t_i}, ua \rangle \overline{u}a \\
\sum_{i=1}^{n} \overline{\mathbb{P}_{t_i}} &= \sum_{i=1}^{n} (\langle \mathbb{P}_{t_i}, \varepsilon \rangle + \sum_{\substack{u \in A^* \\ a \in A}} \langle \mathbb{P}_{t_i}, ua \rangle \overline{u}a) \\
\sum_{i=1}^{n} \overline{\mathbb{P}_{t_i}} &= \sum_{i=1}^{n} \langle \mathbb{P}_{t_i}, \varepsilon \rangle + \sum_{i=1}^{n} \sum_{\substack{u \in A^* \\ a \in A}} \langle \mathbb{P}_{t_i}, ua \rangle \overline{u}a \\
&= \sum_{i=1}^{n} \langle \mathbb{P}_{t_i}, \varepsilon \rangle + \sum_{\substack{u \in A^* \\ a \in A}} \sum_{i=1}^{n} \langle \mathbb{P}_{t_i}, ua \rangle \overline{u}a \\
&= \langle \mathbb{P}_D, \varepsilon \rangle + \sum_{\substack{u \in A^* \\ a \in A}} \langle \mathbb{P}_D, ua \rangle \overline{u}a \\
&= \overline{\mathbb{P}_D}
\end{aligned}$$

Nous avons trouvé que : $\overline{\mathbb{P}_{t_i}} = \mathbb{S}_{t_i}$, donc $\sum_{i=1}^{n} \overline{\mathbb{P}_{t_i}} = \sum_{i=1}^{n} \mathbb{S}_{t_i}$, qui conduit à $\overline{\mathbb{P}_D} = \mathbb{S}_D$. □

La construction d'un automate qui calcule le PSB $\mathbb{S}_D$ devient maintenant plus aisée. Notons que le polynôme $\overline{\mathbb{P}_D}$ peut être réécrit de manière à montrer le lien avec le polynôme $\mathbb{P}_D$ par l'ajout de termes nuls :

$$\overline{\mathbb{P}_D} = \langle \mathbb{P}_D, \varepsilon \rangle + \sum_{u \in \text{Étendue}(\mathbb{P}_D)} 0 \times \overline{u} + \sum_{\substack{ua \in \text{Étendue}(\mathbb{P}_D) \\ a \in A}} \langle \mathbb{P}_D, ua \rangle \overline{u}a \tag{4.16}$$

En rapprochant l'expressions de $\mathbb{P}_D$ et celle de $\overline{\mathbb{P}_D}$, Il est facile de noter la bijection entre les termes des deux polynômes, c'est-à-dire, entre chaque u dans $\mathbb{P}_D$ et $\overline{u}$ dans $\overline{\mathbb{P}_D}$. Conséquemment, comme $\overline{u}$ encode les sous-séquences de u (voir la définition 4.1.1), il suffit donc d'ajouter des ε-transitions dans les chemins étiquetés u dans l'automate $\mathscr{P}_D$ afin d'obtenir le PSB. Toutefois, nous devons être plus prudents en ce qui concerne précisément les coefficients, car l'insertion des ε-transitions multiplie les chemins de reconnaissance d'un motif, et peut ainsi induire en erreur en générant des résultats incorrects. Pour pallier à ces anomalies, nous proposons la duplication d'états. A cet effet, pour chaque état $u \neq \varepsilon$, nous créons un second état $(\overline{u}a)$ avec le bon coefficient $\langle \mathbb{P}_D, ua \rangle$, l'état original devient un état de non acceptation avec une multiplicité nulle $(0 \times \overline{u})$. Mentionnons que l'astuce de duplication d'états/transitions

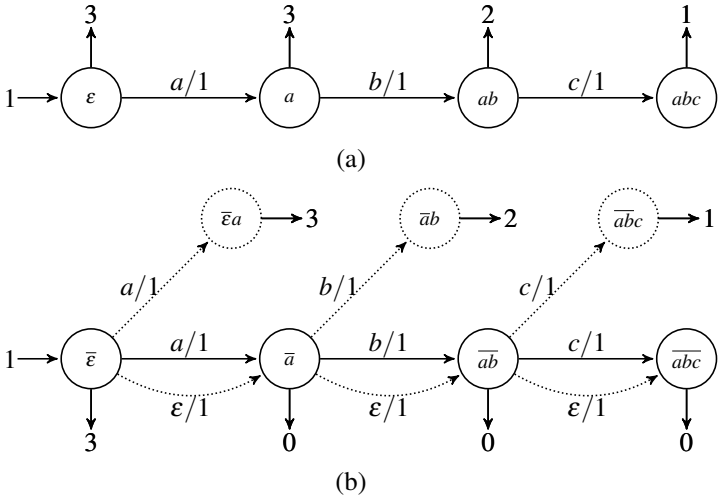


FIGURE 4.3 – Un AMP pour $D_1 = \{a, ab, abc\}$ (a) et son automate étendu (b)

est ici artificielle et sera simulée comme cela est illustré dans l'Algorithme 8. Cette idée assure aussi les valeurs des autres termes du polynôme $\overline{\mathbb{P}_D}$. Nous éclaircissons cette extension par un exemple simple d'une base de données D_1 contenant seulement trois transactions $D_1 = \{a, abc, ab\}$. La figure 4.3 montre deux automates : un AMP de la base D_1 et celui après extension.

4.3 Algorithme de fouille

Une fois l'AMP associé à une base de données D a été construit en utilisant un des procédés introduits par la proposition 4.2.2 ou le corollaire 4.2.7, il servira comme une structure pour l'exploration de l'espace du problème. Dans la phase de fouille nous explorons l'automate en adoptant un parcours en DFS comme cela est montré dans l'Algorithme 7. Un schéma de fouille naïf est comme suit : l'exploration commence par invoquer DISCOVER-FI$(\mathbb{S}, s, \{q_0\}, \varepsilon, \emptyset)$, où q_0 est l'état initial de l'automate. Dans chaque étape, en démarrant de l'ensemble d'états Q_w, un motif w est étendu par concaténation pour obtenir ses successeurs selon l'ordre défini sur les items. Ceci est assuré par l'appel de la fonction EXTEND(Q_w, w, a) de l'algorithme 8, qui renvoie l'ensemble d'états Q_{wa} de tous les chemins étiquetés wa avec leurs coefficients. le support du motif concerné wa est alors la somme (l'opération $\oplus$ selon le semi-anneau) des coefficients des éléments de l'ensemble retourné Q_{wa}, étant donné que $\gamma(R) = \bigoplus_{r \in R} \gamma(r)$. Si la dernière extension réussira avec un motif

Algorithme 7 DISCOVER-FI$(\mathbb{S}, s, Q_w, w, \mathscr{F})$

Entrée : un AMP d'une base D, un seuil du support s, un ensemble d'états Q_w, et un motif w.
Sortie : l'ensemble $\mathscr{F}$ de tous les motifs fréquents dans D.

```
pour tout a > w_{|w|} faire
    (Q_wa, ⟨S_D, wa⟩) ← EXTEND(Q_w, w, a)
    si ⟨S_D, wa⟩ ≥ s alors
        F ← F ∪ {(wa, ⟨S, wa⟩)}
        DISCOVER-FI(S, s, Q_wa, wa, F)
    fin si
fin pour
```

fréquent, le processus continuera en prenant en considération le dernier ensemble d'états atteint Q_{wa} ; dans le cas contraire, le couple retourné est $(\emptyset, 0)$ où un retour arrière implicite est réalisé pour continuer l'exploration d'autres motifs fréquents. Notons que dans l'algorithme 8, $i(q)$ représente l'étiquette de la transition menant à l'état q, avec $i(q_0) = \varepsilon$, et $\delta^+(q)$ dénote les états successeurs de l'état q.

Algorithme 8 EXTEND(Q_w, w, a)

Entrée : Un ensemble d'états Q_w, un motif w et un item a.
Sortie : l'ensemble d'états étendu Q_{wa} avec le coefficient associé.

```
P ← Q_w
R ← ∅
tant que P ≠ ∅ faire
    q ← Retirer un état parmi P
    si i(q) < a alors
        P ← P ∪ δ+(q)
    sinon si i(q) = a alors
        R ← R ∪ {q}
    fin si
fin tant que
retourner (R, γ(R))
```

Proposition 4.3.1. *L'Algorithme 7 peut être exécuté en temps* $O(\sum_{\substack{w\in\mathscr{F} \\ a>w_{|w|}}} C_{wa})$ *et* $O(|Q|)$ *espace, où* Q *est l'ensemble des états de l'AMP,* $\mathscr{F}$ *l'ensemble des motifs fréquents de la base et* C_{wa} *est le temps nécessaire au calcul de l'ensemble* Q_{wa} *en étendant l'ensemble* Q_w.

Démonstration. Notre automate est acyclique défini sur un alphabet ordonné. Par

conséquent, La longueur de tout chemin est bornée par $|Q|$ (dans notre cas $|A|$ la taille de l'alphabet est une borne plus fine).

C_{wa} est le temps nécessaire à l'appel de la fonction EXTEND, qui calcule l'ensemble Q_{wa} en prenant en compte le dernier ensemble d'état obtenu Q_w.

Posons, sans perte de généralité, $C_{wa} = |Q_{wa}|$, d'où pour chaque motif $w = w_1 \dots w_k$ dans $\mathscr{F}$, nous avons : $C_{w_1} + C_{w_1 w_2} + \dots + C_{w_1 w_2 \dots w_k} \leq |Q|$. Dès lors, si $\mathscr{F}_M$ dénote l'ensemble de motifs fréquents maximaux, et $\mathscr{F}_L$ celui des motifs fréquents les plus longs selon la relation de préfixes ($\mathscr{F}_M \subseteq \mathscr{F}_L \subseteq \mathscr{F}$), nous obtenons l'inégalité suivante : $|\mathscr{F}_M||Q| \leq \sum_{w \in \mathscr{F}} C_{wa} \leq |\mathscr{F}_L||Q| \leq |\mathscr{F}||Q| \leq |\mathscr{F}||D|$.

Par ailleurs, la mémoire nécessaire à l'exploration récursive est aussi bornée par $|Q|$ quelle que soit la longueur du motif à reconnaître et le niveau courant d'exploration, étant donné que les ensembles retournés lors de la fouille sont deux à deux disjoints et leur union est au pire égale à Q. □

Le travail développé dans cette section constitue une transcription préliminaire de notre formalisme en algorithmes. Nous montrons par la suite que cette adaptation peut être améliorée d'avantage lorsque nous évoquons le lien avec la déterminisation des automates.

4.4 Comparaison et unification

Pour la découverte des règles d'association, un framework théorique basé sur l'analyse formelle de concepts et la théorie des treillis (Barbut and Monjardet, 1970; Davey and Priestley, 1990) a été présenté très tôt dans (Godin et al., 1995; Zaki and Ogihara, 1998). DMTL (Chaoji et al., 2008) est une bibliothèque unifiée pour la fouille de différents sortes de motifs développée en C++, fondée sur les graphes comme modèle de données, qui offre ainsi une vision générique, paramétrable et extensible. Elle constitue, au vu de ces traits, un effort considérable concernant le volet développement et implémentation des algorithmes de fouille de motifs. Récemment, dans (Pijls and Kosters, 2010) une tentative a été effectuée afin d'unifier les algorithmes communs de fouille de motifs par rapport aux paradigmes de parcours bien connus dans la communauté recherche opérationnelle. Dans (Achar et al., 2012), les auteurs proposent une unification des algorithmes de calcul des supports en se servant des automates classiques. Leur unification concerne différents schémas de

calcul de fréquences pour l'extraction des épisodes fréquents (Mannila et al., 1997), et elle est, néanmoins, limitée aux algorithmes horizontaux.

Notre modèle utilise les séries formelles qui sont des correspondances entre un monoïde $\mathbb{M}$ et un semi-anneau $\mathbb{K}$ pouvant être à priori quelconques. Le choix approprié de $\mathbb{M}$ et $\mathbb{K}$, et les caractéristiques de l'automate le réalisant sont dictés par l'application ciblée et les performances souhaitées.

Notre framework peut être naturellement généralisé pour la fouille de motifs plus élaborés. Cette généralisation aux : séquences, les arbres et les graphes est possible ; étant donné que ces objets sont des structures rationnelles, pourvu qu'un travail plus approfondi doit être effectué pour définir des monoïdes de ces éléments avec les opérations adéquates. L'étude d'automates spécifiques reconnaissant ces structures est une condition pour atteindre cet objectif.

Avant d'entamer la comparaison avec les principales techniques de la littérature, et expliquer comment celles-ci peuvent être dérivées de notre modèle comme des cas particuliers, nous proposons d'introduire notre dernier résultat.

4.4.1 Fouille de motifs et déterminisation d'automates à multiplicité acycliques

Dans ce qui suit, nous révélons que la découverte de l'ensemble des motifs fréquents pour une base de transactions D revient à une déterminisation, compte pris du critère du support, de l'automate à multiplicité des sous-séquences associé (AMS), que nous allons définir ci-après.

Définition 4.4.1 (Automate à multiplicité des sous-séquences (AMS))**.** *Soit $D = \{t_1, \ldots, t_n\}$ une base de transactions et $\mathscr{P}_D$ son AMP associé. Nous définissons l'automate à multiplicité des sous-séquences (AMS) associé à la base D, noté $\mathscr{S}_D$, comme l'automate $\mathscr{P}_D$ augmenté par des ε-transitions conformément à la proposition 4.2.8*

En d'autre termes, l'automate à multiplicité des sous-séquences AMS, associé à une base de données D, est son automate à multiplicité préfixiel AMP auquel nous incorporons des ε-transitions. L'AMS associé à la base de notre exemple de référence est illustré dans la figure 4.4. Comme déjà expliqué dans la section 4.2.3, nous signalons qu'afin de maintenir la taille de l'automate inchangée nous n'ajoutons pas

réellement de transitions additionnelles à l'AMP de la base à fouiller ; toutes les ε-transitions sont donc implicites et artificielles.

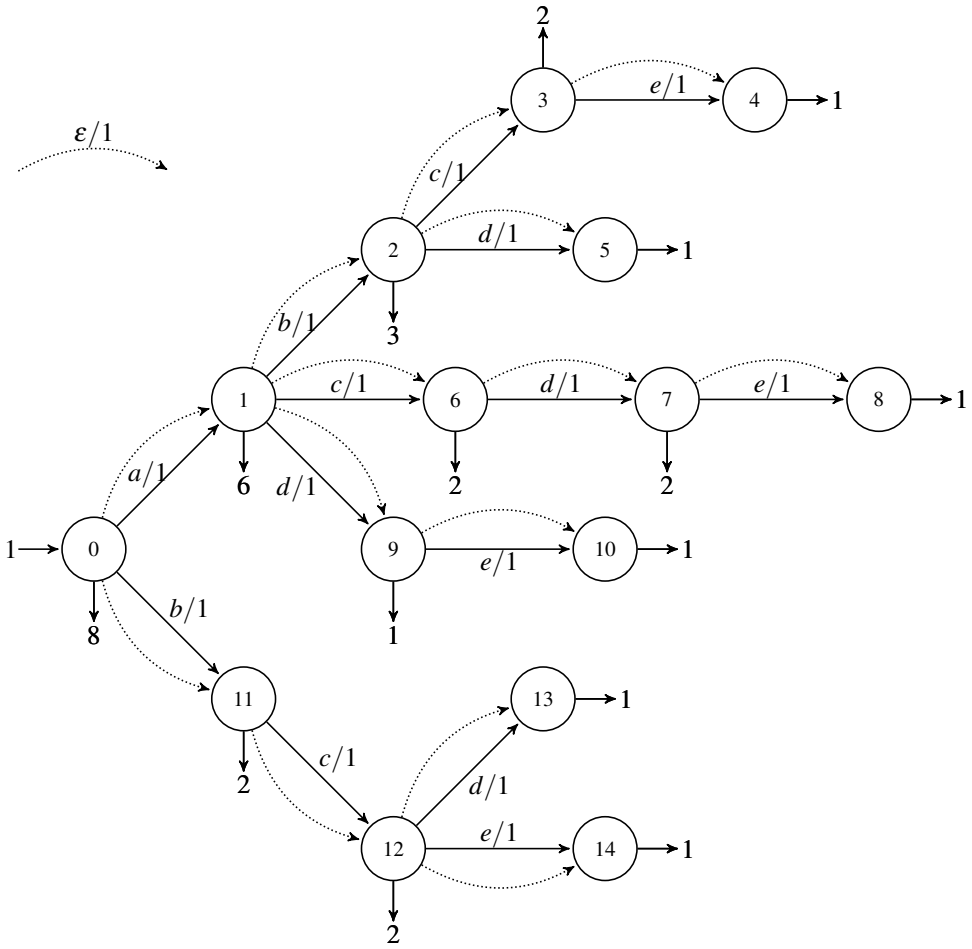


FIGURE 4.4 – L'AMS associé à l'exemple de référence ; les arcs en pointillé sont des ε-transitions

Théorème 4.4.1. *Soit $D = \{t_1, \ldots, t_n\}$ une base de données, $\mathscr{S}_D$ son AMS associé. Les deux problèmes suivants, conditionnés par le critère du seuil de support s, sont équivalents :*

- *La fouille de motifs fréquents de D,*
- *La déterminisation de l'AMS associé à D .*

Démonstration. Les automates à multiplicité ne sont pas déterminisables dans le cas général. Notons d'emblée que les automates de notre classe particulière admettent toujours des équivalents déterministes comme ils sont acycliques (Mohri, 2009).

Nous pouvons simplement constater que le processus de déterminisation de l'AMS $\mathscr{S}_D$ produit à chaque étape un chemin unique pour chaque motif représenté par le mot w. La multiplicité, dans l'automate déterministe, du motif w est alors la somme des multiplicités de tous les chemins étiquetés w dans l'automate original non déterministe réduit ici aux multiplicités de sorties, étant donné que le comportement de l'automate original est déterminé par le vecteur de sortie γ comme les deux autres (μ et λ) sont crisps. Par conséquent, la valeur retournée coïncide exactement avec le support du motif dénoté par w. De manière analogue à la construction des sous ensembles dans les automates classiques (Hopcroft et al., 2001), nous retenons ici seulement les nouveaux états (ensemble d'états) accessibles générés s'ils vérifient aussi la condition du support minimal. C'est à dire, lorsque la somme des poids de sorties des éléments de ces états dépasse le seuil minimal du support s. □

Le théorème précédent nous induit à deux autres propriétés intéressantes. La première concerne la nature de l'Algorithme 7, introduit dans la section précédente, qu'additionnellement d'être un algorithme de fouille de motifs est également un algorithme de déterminisation. La deuxième est une borne sur la taille de l'automate déterministe résultant. D'où le corollaire et la proposition ci-dessous :

Corollaire 4.4.2. *L'Algorithme 7 est un algorithme de déterminisation d'automates à multiplicité.*

Proposition 4.4.3. *Pour une base de données D, la taille de l'automate déterministe $det(\mathscr{S}_D)$ est bornée par le cardinal de l'ensemble $\mathscr{F}$ de motifs fréquents dans D :*

$$|det(\mathscr{S}_D)| \leq |\mathscr{F}| \tag{4.17}$$

Démonstration. Par contradiction, supposons que la propriété est fausse ; autrement dit, $|det(\mathscr{S}_D)| > |\mathscr{F}|$. Par conséquent, selon la procédure de déterminisation, il existe au moins un état accessible, soit q, dans l'automate $det(\mathscr{S}_D)$ où le chemin entre l'état initial et q encode un motif infréquent $w_q \notin \mathscr{F}$. Mais, et pour les même raisons, l'algorithme de déterminisation exige que la multiplicité du chemin w_q soit supérieure au seuil de support s pour qu'il sera maintenu dans l'automate $det(\mathscr{S}_D)$. Ceci implique que le poids de w_q est supérieur ou égal à s. Donc, w_q est fréquent et appartient bien à $\mathscr{F}$, i.e., $w_q \in \mathscr{F}$, Ce qui contredit notre proposition de départ. □

De la table 4.1, ne pouvons simplement vérifier, pour notre exemple de référence, qu'il y a treize motifs fréquents. Ce nombre est exactement le nombre d'états

de l'automate déterministe $det(\mathscr{S}_D)$ indiqués en gras dans la table 4.2 et la table 4.3. L'automate déterministe intégral $det(\mathscr{S}_D)$ par rapport à une valeur minimale de support égale à 3 est illustré dans la figure 4.5.

La proposition 4.4.3 donne une borne supérieure à la taille de l'automate déterministe ; elle n'avance rien, cependant, en ce qui concerne la borne inférieur, cela est laissé intentionnellement car l'automate minimal dans le cas pondéré n'existe pas toujours (Eisner, 2003). Néanmoins, pour montrer le fait qu'on peut, concernant $|det(\mathscr{S}_D)|$, descendre en dessous de $|\mathscr{F}|$, nous pouvons imaginer une base de données triviale contenant une seule transaction *ab* avec un seuil de support égal à 1. Il est évident de voir dans ce cas que $|\mathscr{F}|$ est égal à 4, tandis que $det(\mathscr{S}_D)$ compte seulement 3 états.

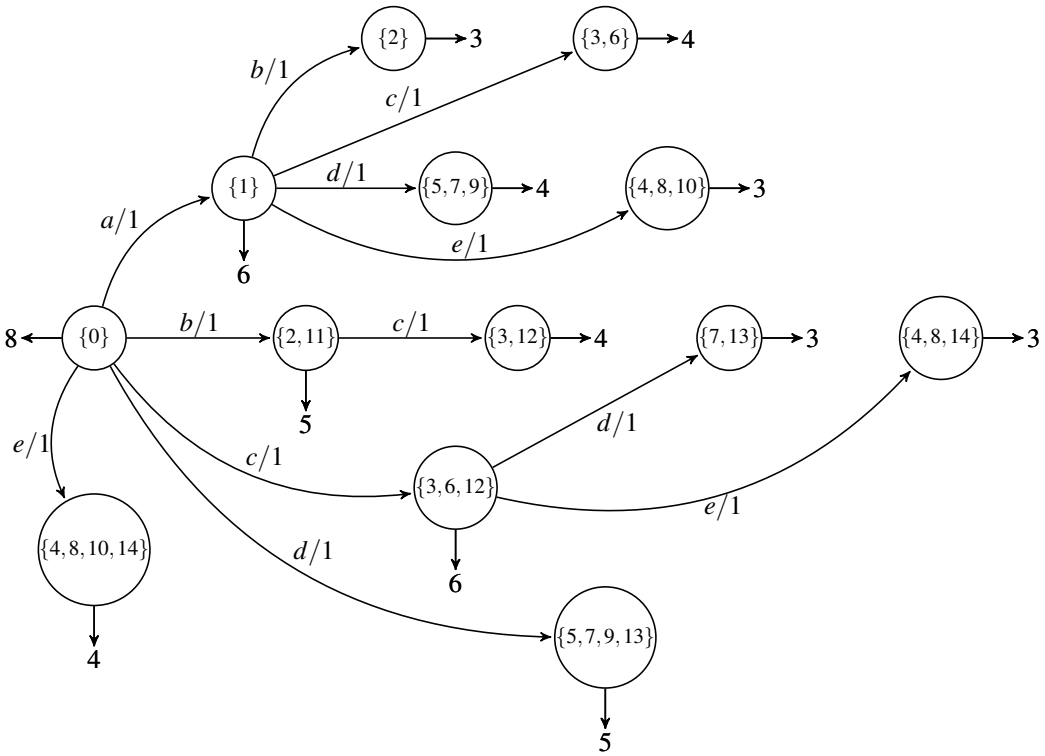


FIGURE 4.5 – L'automate déterministe de l'exemple de référence par rapport au seuil minimal du support

4.4.2 Approches par niveau et déterminisation en largeur sur le semi anneau de comptage

Un algorithme du style Apriori (Agrawal and Srikant, 1994) procède niveau par niveau. Premièrement, il calcule les singletons fréquents et forme ensuite de cette liste une collection de paires candidates. Après avoir déterminé les doublets fréquents par comptage, il continue de la même manière pour générer les triplets fréquents, et ainsi de suite. Ce processus est réitéré jusqu'aucun nouveau motif fréquent ne peut être généré. En dépit de ses limites connues : la génération d'un nombre excessif de candidats et les nombreux balayages répétés de la base de données, cet algorithme reste un des algorithmes les plus cités de la communauté fouille de données (Wu et al., 2008). Notre modèle peut être adapté pour se conformer au même principe si nous effectuons une déterminisation en largeur d'abord (Bread First Determinization (BFD)) de l'AMS associé à la base à fouiller. Nous commençons ainsi la déterminisation de l'état initial $\{q_0\}$, et pour chaque item a de l'alphabet nous calculons les états fréquents accessibles correspondants [4]. Nous fusionnons donc tous les transitions étiquetés a et atteignables de l'état initial $\{q_0\}$ en utilisant des transitions explicites ou implicites (via des mouvements en ε). Ce mécanisme est renouvelé de proche en proche et selon un schéma par niveau, jusqu'aucun nouvel état fréquent accessible ne peut être généré.

Ce calcul est montré, pour l'exemple de référence et un seuil de support égal à 3, dans la table 4.2, que nous renseignons ligne par ligne. Le coefficient (ou le poids de sortie) de chaque état accessible est exhibé en texte encerclé. Mentionnons que cette adaptation au paradigme d'Apriori permet de construire un algorithme plus efficace, étant donné que d'un coté la génération de candidats est plus contrôlés et d'un autre les balayages de la base ne sont plus nécessaires.

Corollaire 4.4.4. *Une déterminisation en largeur d'abord de l'AMS sur le semi-anneau de comptage de la base de données D par rapport au seuil de support s est équivalente aux approches par niveaux de fouille de motifs (l'algorithme Apriori en particulier).*

4. Dans notre modèle, un état accessible fréquent est un état atteignable de l'état initial, via ou non des ε-transitions, pour lequel le coefficient du chemin correspondant est supérieur au seuil minimal du support.

TABLE 4.2 – Une déterminisation en largeur de l'AMS de l'exemple de référence

Items ⟶ / Etats ↓	*a*	*b*	*c*	*d*	*e*
{0}	**{1} ❻**	**{2,11} ❺**	**{3,6,12} ❻**	**{5,7,9,13} ❺**	**{4,8,10,14} ❹**
{1}		**{2} ❸**	**{3,6} ❹**	**{5,7,9} ❹**	**{4,8,10} ❸**
{2,11}			**{3,12} ❹**	{5,13} ②	{4,14} ②
{3,6,12}				**{7,13} ❸**	**{4,8,14} ❸**
{5,7,9,13}					{8,10} ②
{2}			{3} ②	{5,} ①	{4} ①
{3,6}				{7} ②	{4,8} ②
{5,7,9}					{8,10} ②
{3,12}				{13} ①	{4,14} ②
{7,13}					{8} ①

4.4.3 Approches verticales et déterminisation en profondeur sur le semi-anneau des sous ensembles

L'atout principal des approches verticales (Zaki, 2000) contre les approches horizontales est l'accélération dans le calcul du support réalisé par des intersections d'ensembles. Mais à l'opposé, leur point faible, comme déjà noté dans le chapitre précédent et aussi par les auteurs eux-même dans une version plus améliorée (Zaki and Gouda, 2003), est les exigences en mémoire qu'elles requièrent lorsque les résultats intermédiaires deviennent trop larges où ces approches affichent alors des exceptions de dépassement de capacité mémoire. Notre méthode est basée en revanche sur de simples opérations de lecture des multiplicités de sorties ou leur additions sans d'autre exigences en mémoire additionnelle.

L'approche verticale peut être considérée comme le calcul d'une série formelle sur le semi-anneau des sous ensembles (ensemble puissance) de l'ensemble T de transactions $(2^T, \cup, \cap, \emptyset, T)$, avec les opérations $\oplus$ et $\otimes$ réalisées par l'union et l'intersection ensemblistes. Le poids d'une transition représente la tidlist du motif formé le long du chemin de la racine $\emptyset$ au nœud considéré calculé moyennant l'intersection des tidlists précédents dans le chemin. En effet, si $\mathbb{T}$ dénote la série formelle de

l'ensemble puissance de l'ensemble de transactions T, nous définissons dans cette approche le coefficient d'un éléments w comme suit :

$$\begin{cases} \langle \mathbb{T}, \varepsilon \rangle = T \\ \langle \mathbb{T}, a \rangle = \text{tid-list}(a) \quad \text{pour chaque } a \in A \\ \langle \mathbb{T}, wa \rangle = \langle \mathbb{T}, w \rangle \cap \langle \mathbb{T}, a \rangle \end{cases} \tag{4.18}$$

Le calcul du coefficient d'un élément w peut encore être optimisé et dérivé de façon équivalente en intersectant n'importe quels deux éléments w_1 et w_2 satisfont $w = w_1 w_2$, i.e., nous pouvons réécrire l'équation en haut avec les mêmes cas de base comme suit :

$$\langle \mathbb{T}, w \rangle = \langle \mathbb{T}, w_1 \rangle \cap \langle \mathbb{T}, w_2 \rangle \quad \text{pour tout } w_1, w_2, \text{ tel que } w = w_1 w_2 \tag{4.19}$$

Pour obtenir le résultat final (les supports) nous passons par la cardinalité des ensembles trouvés. En d'autre termes, si nous dénotons par $\mathbb{Z}$ la série de l'approche verticale de (Zaki, 2000), celle-ci peut être statué ainsi :

$$\langle \mathbb{Z}, w \rangle = |\langle \mathbb{T}, w \rangle| \tag{4.20}$$

À titre d'exemple, considérons dans la base de référence du tableau 4.1 le calcul du coefficient du motif bc. Nous pouvons voir que le coefficient de b est l'ensemble tid-list$(b) = \{1,2,3,7,8\}$, celui de c est tid-list$(c) = \{1,2,4,5,7,8\}$. Par conséquent, nous obtenons le coefficient de bc comme l'intersection des deux tid-lists : tid-list$(bc) = \{1,2,3,7,8\} \cap \{1,2,4,5,7,8\} = \{1,2,7,8\}$. Le support final du motif bc est donné par le cardinal du tidlist obtenu qui vaut ici 4.

De point de vu automate, le calcul dans cette approche peut être vu similairement comme une déterminisation en profondeur (Depth First Determinization (DFD)) de l'AMS associé à la base en question. Le corollaire suivant donne cette conclusion que nous pouvons constater dans le tableau 4.3.

Corollaire 4.4.5. *Une déterminisation en profondeur d'abord de l'AMS sur le semi-anneau des sous ensembles associé à la base de données D par rapport au seuil de support s est équivalente aux approches verticales de fouille de motifs (l'algorithme Eclat en particulier).*

TABLE 4.3 – Une déterminisation en profondeur de l'AMS de l'exemple de référence

Items ⟶ / Etats ↓	*a*	*b*	*c*	*d*	*e*
{0}	**{1} ❻**	**{2,11} ❺**	**{3,6,12} ❻**	**{5,7,9,13} ❺**	**{4,8,10,14} ❹**
{1}		**{2} ❸**	**{3,6} ❹**	**{5,7,9} ❹**	**{4,8,10} ❸**
{2}			{3} ②	{5,} ①	{4} ①
{3,6}				{7} ②	{4,8} ②
{5,7,9}					{8,10} ②
{2,11}			**{3,12} ❹**	{5,13} ②	{4,14} ②
{3,12}				{13} ①	{4,14} ②
{3,6,12}				**{7,13} ❸**	**{4,8,14} ❸**
{7,13}					{8} ①
{5,7,9,13}					{8,10} ②

4.4.4 Approches projectives et les séries dérivées

Au premier coup d'œil, il peut paraître que notre automate est un FPTree (Han et al., 2000) vu d'une autre manière. Nous devons souligner d'ores et déjà que la ressemblance à FPTree ou à tout autre concept introduit dans les travaux antérieurs ne serait en principe pas surprenante et devait être reçue comme un point positif et non l'inverse, étant donné que notre objectif annoncé dès le départ est la définition d'un modèle unificateur.

De plus nous sommes persuadés que cela n'est pas assez correct. Primo, notre automate n'est pas une structure de données, mais plutôt un modèle de calcul qui peut être implémenté de différentes manières. Secundo, les algorithmes de fouille sont significativement différents. En effet, tandis que FPGrowth (Han et al., 2000) exige une mémoire intermédiaire et un temps de calcul importants lors des opérations répétées de tris et de (re)constructions de bases conditionnelles, notre algorithme ne requière aucune extra-mémoire autre que celle nécessaire à l'automate, qui contrairement à un FPTree ne contient ni de pointeurs ni de table entête. En outre, et à l'opposé de FPGrowth, nous estimons à priori (à vérifier lors des expérimentations) que l'ordre ouvert retenu dans notre modèle offre des gains en temps aussi bien dans

la phase de construction (car cela nécessite seulement un seul scan de la base de transaction), que celle de fouille puisque les opérations de projection et de tri ne sont guère indispensables.

Pour fins d'unification, cette approche peut être perçue comme une séquence de dérivations à droite (ou à gauche [5]) par rapport aux éléments de l'alphabet A des items de notre polynôme de sous-séquences de la base $\mathbb{S}_D$. En fait, la dérivée à droite du polynôme $\mathbb{S}_D$ par rapport à un item a donne la représentation polynomiale de la base conditionnelle de l'item a dans FPGrowth (Han et al., 2000). Le support d'un motif w dans la base de données est spécifié par la proposition suivante, qui stipule que la valeur du support d'un motif est le terme constant qui correspond au coefficient du motif nul dans la série dérivée de $\mathbb{S}_D$ au regards du dit motif.

Proposition 4.4.6. *Soit $D = \{t_1, \ldots, t_n\}$ une base transactionnelle, et $\mathbb{S}_D$ son PSB associé. Pour tout motif w l'égalité suivante est vérifiée :*

$$\mathrm{sprt}(w, D) = \langle \mathbb{S}\, w^{-1}, \varepsilon \rangle \tag{4.21}$$

Démonstration. Immédiate de l'équation (1.9) et la proposition 4.1.1 □

C'est-à-dire, que pour calculer le support d'un motif $w = w_1 \ldots w_k$ dans FP-Growth, nous devons effectuer une suite de projections graduelles par rapport à ses items constituants w_i. Le même travail équivalent peut être obtenu via une suite de dérivation de notre polynôme de sous-séquences de la base $\mathbb{S}_D$. Prenons un exemple illustratif. Afin d'aboutir au support de ce, réalisons les deux dérivés à droite sur PSB et sur l'automate ASB ; nous obtenons :

$$\begin{aligned} \mathbb{S}_D\, e^{-1} &= \mathbf{4} + 3a + 2b + 3c + 2d + ab + 2ac + 2ad + 2bc + cd + abc + acd \\ (\mathbb{S}_D\, e^{-1})c^{-1} &= \mathbb{S}_D(ce)^{-1} = \mathbf{3} + 2a + 2b + ab \end{aligned}$$

De ces dérivées, nous aurons les supports de e (4) et celui de ce (3). La figure 4.6 montre ces résultats sur l'automate.

4.4.5 Bi-monoïde tropical

Pour finir, nous mentionnons qu'une modélisation équivalente de notre approche peut être obtenue en travaillant sur le bi-monoïde tropical $(\mathbb{N} \cup$

5. Le sens de la dérivation n'est pas important et conduit toujours au même coefficients finaux. Toutefois, le nombre d'étapes dépend de l'ordre adopté sur les items et la base de données à fouiller.

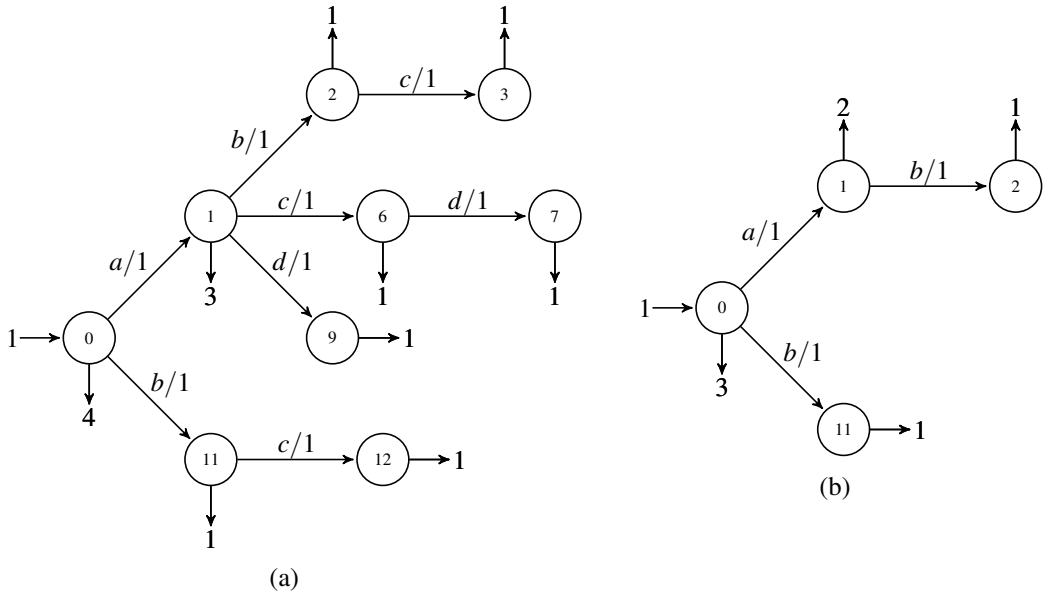


FIGURE 4.6 – Des AMP associés aux dérivés $\mathbb{S}_D\, e^{-1}$ (a) et $\mathbb{S}_D(ce)^{-1}$ (b)

$\{\infty\}, +, min, 0, \infty$) (Droste et al., 2010). Le calcul dans ce contexte est légèrement à remanier : dans l'automate, les transitions cette fois ci au lieu d'être crisp elles transmettent les poids de sorties. De cette façon, les nouveaux poids de sortie de tous les états sont à l'infini.

Dans ce modèle, la multiplicité d'un chemin est calculée en prenant le minimum (opération *min*) des multiplicités des transitions qui le constituent. Le poids d'un motif w dans l'automate est alors la somme (opération +) de tous les chemins étiquetés w. La figure 4.7 exhibe l'automate à multiplicité associé à l'exemple de référence sur le bi-monoïde tropical. Il est aisé de remarquer que ce modèle bien qu'équivalent à le notre, il est relativement coûteux (opération *min*) par rapport à celui retenu qui consiste à de simples opérations de lecture des poids de sorties des différents états.

4.5 Conclusion

Dans ce chapitre, nous avons modélisé le problème de la fouille de motifs par des polynômes. Les principales propriétés de ce modèle ont été établies et analysées. Par ailleurs, une catégorie particulière d'automates servant de structure permettant de

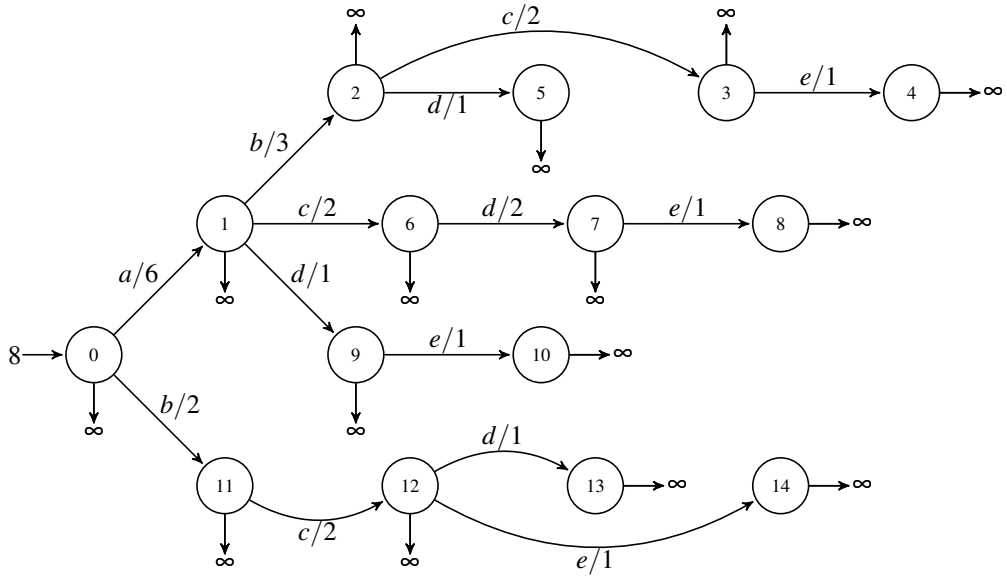


FIGURE 4.7 – Un AMP sur le bi-monoïde tropical $(\mathbb{N} \cup \{\infty\}, +, min, 0, \infty)$ associé à l'exemple de référence

passer à une implémentation a fait l'objet de définition et étude. Ce travail de modélisation accompli est jugé raisonnablement répondant à l'objectif primaire préalablement tracé à savoir l'élaboration d'un modèle unificateur. Ainsi, nous avons expliqué comment les principales approches de fouille de motifs de la littérature peuvent être dérivées de la notre comme instances particulières. Dans le chapitre suivant, nous allons aborder une réalisation suivi d'une expérimentation et une évaluation. Nous voulons souligner d'emblée que l'objectif dans le chapitre qui suit est de montrer la faisabilité de l'approche et non pas surtout la création de rivalité et la concurrence des principaux algorithmes de la littérature.

CHAPITRE 5

Implémentation, expérimentation et évaluation

> Ce qui est affirmé sans preuve peut être nié sans preuve.
>
> Euclide

Sommaire

Notre objectif dans cette thèse est, comme déjà mentionné, l'élaboration d'un nouveau formalisme générique pour le problème de la fouille de motifs fréquents. Autrement dit, définir une autre façon claire et unifiée pour réfléchir autour du sujet, et cela indépendamment de tout algorithme ou technique de programmation et les détails d'implémentation y afférents.

Dans ce chapitre, nous décrivons une implémentation préliminaire de notre modèle et algorithmes introduits dans la section 4.3 du chapitre précédent vus de l'angle de déterminisation d'automates et discutons quelques heuristiques permettant d'améliorer son efficacité. Nous effectuons à l'issue également une évaluation élémentaire de notre approche en terme de métriques standards : le temps d'exécution et l'usage de la mémoire, en le rapprochant avec quelques algorithmes représentatifs et récents. Ce volet constitue une autre brique additionnelle de notre contribution dans cette thèse.

5.1 Construction initiale

L'automate que nous avons défini est implémenté comme un graphe acyclique orienté (Directed Acyclic Graph (DAG)) en utilisant la représentaton par liste d'adjacence, puisqu'il est sparse. En effet, pour toute base de données de transactions le nombre de transitions $|E|$ de l'automate correspondant est en effet égal au nombre d'états $|Q|$ [1], qui est clairement nettement loin de $|Q|^2$. Par conséquent, l'occupation mémoire de notre modèle est $O(|Q|)$, ce qui justifie l'utilisation de $|Q|$ pour faire référence à un automate et dénoter plus particulièrement sa taille. A cet effet, et suivant la définition 4.2.3, chaque état consigne initialement : un item, une valeur de fréquence et une liste ordonnée de successeurs. L'ordre de la liste des successeurs est basé sur la valeur de l'item, que nous l'avons adopté afin d'accélérer le module de chargement via une stratégie de recherche binaire.

1. Précisément, en retranchant l'état initial : $|E| = |Q| - 1$, qui peut s'apparenter à un arbre.

5.2 Affinement

Nous pouvons améliorer la première version de l'algorithme de fouille selon plusieurs idées. L'actuelle version tire profit des trois astuces ci-dessous bien connues dans la sphère des algorithmes de fouille de motifs :

— Une version itérative serait hautement appréciée. En fait, pour les ensembles de données trop larges, une version récursive peut se coincer lors de l'exploration des graphes associés souvent de tailles gigantesques.

— Étendre efficacement un motif par l'indexation des états, ceci permet l'accès direct à un état particulier (ou un ensemble d'entre eux) et d'éviter les scans linéaires et répétés de l'automate.

— Omission des extensions à equisupports (Schmidt-Thieme, 2004; Deng and Lv, 2015), connue également sous le terme élagage des extensions parfaites (Borgelt, 2012). Cette dernière est rencontrée si le support d'un motif x est égal au support de l'une de ses extensions xa ; ce cas, permet de déduire que le support de tout motif xy est égal à celui de xya, pour y disjoint de x et n'incluant pas l'item a. Cette optimisation est exploitée en ignorant tous les sous-espaces de l'arborescence du problème ayant un support égal au support de leur parents.

5.2.1 Stratégie et étiquetage des états

L'examen du module d'extension dans l'Algorithme 8 montre qu'il calcule ce qui appelé la clôture en epsilon (ou la fermeture transitive) d'un ensemble d'états, mais de manière simple par une exploration séquentielle de l'automate. Il est claire, vue la nature combinatoire du problème que cela coûtera un temps considérable particulièrement dans les ensembles de données denses et massifs. Aussi, le recours à certaines techniques d'optimisation s'avère nécessaire afin de réduire le temps de réponse du module d'extension (Schmidt-Thieme, 2004; Borgelt, 2012).

L'indexation est considérée comme une des heuristiques les plus importantes dans ce problème, puisqu'elle permet d'éviter de traiter la totalité de la structure de données de façon linéaire. La première utilisation de l'indexation dans ce contexte est due à Han et al. (Han et al., 2000) lorsqu'ils introduisent la structure de données FPTree. De notre part, nous estimons que la formalisation de l'indexation comme un problème d'accessibilité à des nœuds d'un graphe (Cohen et al., 2002) constituera

un bon artifice d'optimisation. Le principe est d'attacher des étiquettes (labels) à chaque état p puis d'utiliser seulement ces informations dans toutes les requêtes (accessibilité, distance, etc.) impliquants l'état p.

À ce titre, les travaux dans (Deng and Wang, 2010; Deng et al., 2012; Deng and Lv, 2015) combinent les avantages de l'approche verticale (Zaki, 2000) et celle projective fondée sur l'algorithme FPgrowth (Han et al., 2000). Les auteurs étendent la structure du nœud d'un FPTree avec deux codes et proposent un temps linéaire pour le calcul de l'intersection de deux listes de nœuds de deux $(k-1)$-motifs afin de former un nouveau k-motif étendu. Pour ce faire, chaque nœud de l'arbre est augmenté par deux nombres qui représentent les codes pré-ordre et post-ordre du nœud en question selon des parcours pré-ordre ou post-ordre de l'arbre. Dans notre implémentation, nous utiliserons une version légèrement adaptée de la dernière idée comme implémentée dans les l'algorithmes PPV (Deng and Wang, 2010) et PrePost (Deng et al., 2012). La présente implémentation est une version itérative équivalente de la version de l'algorithme 7, ceci a été retenu afin d'éviter tout blocage dû au dépassement de la capacité de la pile dans le cas de très longues récursions rencontrées pour des instances du problème ayant des graphes colossaux comme espace de recherche.

Pour atteindre cet objectif, et comme mentionné en haut, nous suivons l'idée de (Cormen et al., 2009) et (Deng and Wang, 2010; Deng et al., 2012), et augmentons chaque état avec deux estampilles temporelles : *start* et *end*. Ces deux attributs représentent respectivement le temps de début et de fin où un état est rencontré dans un parcours en profondeur de l'automate. Au commencement du processus, une estampille temporelle globale *Time* est créé et initialisée à zéro.

Tandis que (Deng and Wang, 2010; Deng et al., 2012) utilisent deux encodages différents pour ces deux champs et effectuent deux balayages de la structure de données par deux parcours séparés (pré-ordre et post-ordre), nous utilisons ici un code de référence global unique et un seul parcours en profondeur simple afin d'initialiser les différents états de l'automate. Notons que l'exigence en mémoire demeure de l'ordre de $O(|Q|)$, étant donné que les fragments de mémoire ajoutés sont constants pour chaque état. L'invocation de la procédure récursive de l'algorithme 9 assure l'initialisation des estampilles des états de l'automate, l'état initial est fourni comme paramètre d'entrée.

Grâce au théorème des parenthèses (Cormen et al., 2009), nous pouvons recon-

Algorithme 9 INITTIMESTAMPS

Entrée : un état p.
Sortie : p et tous les états accessibles à partir de p augmentés par les estampilles temporelles.

```
procedure INITTIMESTAMPS(p)
    p.start ← ++Time
    pour tout q ∈ p.Next faire
        INITTIMESTAMPS(q)
    fin pour
    p.end ← ++Time
fin procedure
```

naître la relation de descendance reliant deux états qui sera utilisée dans la phase de fouille pour réaliser les extensions. Ci-après une version adaptée de ce théorème.

Théorème 5.2.1 (Théorème des parenthèses (Cormen et al., 2009))**.** *Dans tout parcours en profondeur d'un graphe, et pour toute paire de nœuds p et q, Exactement une des trois conditions suivantes est réalisée :*

- *L'intervalle* $[p.start, p.end]$ *et* $[q.start, q.end]$ *sont complètement disjoints, aucun des deux nœuds (ni p ni q) n'est descendant de l'autre,*
- *L'intervalle* $[q.start, q.end]$ *est entièrement contenu dans l'intervalle* $[p.start, p.end]$*, et q est descendant de p, ou*
- *L'intervalle* $[p.start, p.end]$ *est entièrement contenu dans l'intervalle* $[q.start, q.end]$*, et p est descendant de q.*

Démonstration. Se référer à (Cormen et al., 2009) □

Un point important noté par (Cormen et al., 2009; Deng and Wang, 2010) est que les entiers *start* et *end*, qui s'étalent le long de l'intervalle $[0, 2|Q| - 1]$, définissent une correspondance bijective avec l'ensemble des états de l'automate. Ainsi, nous utilisons dans notre implémentation des structures de données basées sur les tables de hachage, qui vont nous permettre d'exploiter des accès directs aux différents états via leurs codes temporels *start* ou *end*. Plus intéressant, la relation de descendance utilisée plus tard dans le module d'extension peut être exprimée seulement par un seul de ces codes, chose que nous utilisons dans notre implémentation. Le corollaire suivant exprime cette idée.

Algorithme 10 DISCOVER-FI

Entrée : un AMP de D, le seuil du support s.
Sortie : l'ensemble $\mathscr{F}$ de tous les motifs fréquents dans D.

```
1: 𝓛 ← {les SStates des singletons fréquents}
2: tant que 𝓛 ≠ ∅ faire
3:     P ← DEQUEUE(𝓛)
4:     i ← 0
5:     tant que i < |𝓛| et PREFIXWITHOUTTAIL(P) == PREFIXWITHOUTTAIL(𝓛(i)) faire
6:         Q ← 𝓛(i)
7:         R ← EXTEND-MOTIF(SSET(P), TAIL(Q))
8:         si CurrentSupport ≥ s alors
9:             si CurrentSupport == SUPPORT(P) alors
10:                ADD(eqsList,TAIL(Q))
11:            sinon
12:                x ←SSTATES(R,PREFIX(P)+TAIL(Q),EQUISUPPORT(P),CurrentSupport)
13:                ADD(newStateList,x)
14:                NbFI ← NbFI + 1
15:            fin si
16:        fin si
17:        i ← i + 1
18:    fin tant que
19:    si newStateList ≠ ∅ alors
20:        pour tout e ∈ newStateList faire
21:            si eqsList ≠ ∅ alors
22:                AUGMENTEQS(e,eqsList)
23:            fin si
24:            ENQUEUE(𝓛,e)
25:        fin pour
26:        newStateList ← ∅
27:    fin si
28:    si eqsList ≠ ∅ alors
29:        AUGMENTEQS(P,eqsList)
30:        eqsList ← ∅
31:    fin si
32:    si EQS(P) ≠ ∅ alors
33:        GENERATEALLSUBSETS(PREFIX(P),EQS(P),SUPPORT(P))
34:    fin si
35: fin tant que
```

Corollaire 5.2.2. *Soient p et q deux états de notre automate. q est accessible à partir*

(descendant) de p si et seulement si :

$$p.start < q.start < p.end \tag{5.1}$$

5.2.2 WAFI

Dans ce qui suit, nous introduisons WAFI : une version itérative de notre algorithme initial décrit dans la section 4.3. Nous exploitons une file $\mathscr{L}$ pour gérer les états de notre automate durant la déterminisation. Chaque nouvel état donne naissance à un nouveau motif fréquent.

Initialement, la file $\mathscr{L}$ contient les états correspondants aux singletons fréquents (items fréquents). Cette file est consignée dans une table de hachage et sera utilisée pour étendre les nouveaux états qui représentent les k-motifs nouvellement découverts.

Étant donné que l'algorithme 10 est une procédure de déterminisation, il traite ses éléments dans la file essentiellement comme un ensemble d'états avec d'autre informations utiles. Dans la version actuelle de l'algorithme, nous avons défini un nouveau type nommé SStates pour modéliser un ensemble d'états. Une instance de ce type (classe) inclut dans sa partie statique en plus de la valeur du support (nous avons préféré stocker le support afin d'éviter des temps perdus dans son calcul) les trois ensembles suivants :

1. **SSet** : un ensemble ordonné d'entiers qui symbolise l'ensemble des états correspondant ; chaque état est codifié par son *start* code.
2. **Prefix** : un ensemble ordonné d'entiers qui représente le motif enregistré par l'état.
3. **EquiSupport (EQS)** : un ensemble d'items qui définie l'extension parfaite de l'instance, le cas échéant.
4. **Support** : le support de l'ensemble des états, qui est la somme des supports de ses composants (les membres de **SSet**).

Les principales étapes de l'algorithme sont décrites comme suit. Dans chaque itération, un état P (ensemble d'états) qui représente un k-motif fréquent est défilé. La boucle tant que interne (lignes 5-18) tente d'étendre le k-motif fréquent associé

Algorithme 11 EXTEND-MOTIF

Entrée : Q : le SSet du motif, et a l'item d'extension.
Sortie : R, le SSet de l'extension.

```
1:  function EXTEND-MOTIF( Q, a)
2:      R← ∅
3:      T ← FREQUENTITEMSSET(a)
4:      CurrentSupport ← 0
5:      i ← 0
6:      j ← 0
7:      tant que i < |Q| et j < |T|  faire
8:          si T(j) < Q(i) alors
9:              j++
10:         sinon si T(j) < END(Q(i)) alors
11:             CurrentSupport ← CurrentSupport +SUPPORT(T(j))
12:             si END(T(j))-T(j)-1 > 0 alors
13:                 ADD(R,T(j))
14:             fin si
15:             j++
16:         sinon
17:             i++
18:         fin si
19:     fin tant que
20:     retourner R
21: fin function
```

par les symboles (items) suivants selon l'ordre prédéfini sur les items. Il est clair, que cette manière de procéder est équivalente à la stratégie de (Zaki, 2000) consistant à parcourir tous les états Q contenant le même k-préfixe avec l'état P.

En utilisant l'inégalité (5.1) du corollaire 5.2.2, nous adoptons une extension similaire à l'intersection de paires de listes proposée par (Deng and Wang, 2010), mais considérée du point de vue extension au lieu de fusion. Nous estimons que cette astuce permet de tirer profit d'au moins deux optimisations. D'une part, nous pouvons éliminer du résultat de l'extension tous les états feuilles (états sans successeurs), étant donné que ces états ne peuvent participer à aucune extension future, chose que nous vérifions à la ligne numéro 12 de la fonction EXTEND-MOTIF de l'Algorithme 11. D'autre part, nous conjecturons que si nous réussissions à exprimer la condition d'accessibilité (descendance entre états) en fonction du niveau (la profondeur) d'un état, cela offre plus de possibilité d'élagage des résultats de l'extension. Cette dernière idée n'est pas encore implémentée dans la présente version

de l'algorithme, et fera l'objet d'investigation future.

Dans la cas d'une extension réussie, l'algorithme maintient deux listes temporaires : *eqsList* qui consigne les extensions parfaites dans les lignes (8-10) et *newStateList* des états nouvellement découverts créé en utilisant le dernier ensemble d'états retourné par la fonction d'extension comme est montré dans la ligne 12. Dans les lignes 19-31, nous examinons les extensions parfaites possibles. Si un de ces derniers cas se présente, nous augmentons alors les attributs $EquiSupport$ de l'état P et ses descendants dans la liste temporaire *newStateList*, qui sont par la suite enfilés dans la structure $\mathscr{L}$. Finalement, nous générons comme nouveaux motifs, dans les lignes 32-33, tous les sous ensembles de la liste courante des extensions parfaites (Schmidt-Thieme, 2004) *eqsList* avec le préfixe et le support de l'état P en cours de traitement.

Un dernier point important à mentionner relativement à la généricité et l'aspect unificateur de WAFI est que l'Algorithme 10, comme une implémentation de notre formalisme, est fondé sur des objets mathématiques qui sont abstraits et génériques. Afin de tirer avantages de cet aspect, nous devons d'abord choisir un semi-anneau adéquat $\mathbb{K}$ comme domaine de travail et bien définir les opérations binaires $\oplus$ et $\otimes$ selon les items de base, éléments de l'alphabet A. Par la suite, le choix de la stratégie ou la discipline de la liste $\mathscr{L}$ est à déterminer. Ainsi, une stratégie de type FIFO/LIFO de $\mathscr{L}$ donne une exploration de l'espace de recherche en largeur/profondeur, ce qui rend une implémentation par niveau vs verticale. Par ailleurs, comme déjà noté, dans le présent travail, nous avons choisi une modélisation de l'univers du problème par le monoïde libre A^*, ou chaque symbole représente un item. Le même framework peut être retenu, moyennant des adaptations, pour fouiller des univers plus complexes comme des séquences, des arbres, et ainsi de suite.

5.3 Expérimentation

Dans l'objectif d'évaluer notre travail, nous avons mené de multiples expérimentations sur aussi bien des données réelles que synthétiques. Nous avons exploité, à cet effet, les ensembles de données communs et les plus populaires (Fimdr, 2003; Lichman, 2013; Fournier-Viger et al., 2016). Ces ensembles de données sont décrits brièvement dans le tableau 5.1.

TABLE 5.1 – Ensembles de données objets de l'expérimentation

Ensembles de données	Taille	Catégorie	Contenu
mushrooms	8 416	Réel et dense	Des attributs de différents espèces de champignons
retail	88 162	Réel et dense	Paniers d'achat d'un magasin Belge
kddcup99	1 000 000	Réel et dense	Des données tcpdump d'environ 7 semaines de trafic réseau
kosarak	990 002	Réel, énorme, et sparse	Des données de clicks d'un portail Hongrois
BMSWebView1	59 602	Réel	Une base de flux de clicks d'un site de e-commerce
BMSWebView2	77 512	Réel	Une base de flux de clicks d'un site de e-commerce
T10I4D100K	100 000	Synthétique, large et sparse	Fichier généré par l'outil d'IBM Almaden Quest research group
I16T100K	100 000	Synthétique, large et sparse	Fichier généré via SPMF generator

5.3.1 Environnement

Les codes de notre implémentation sont écrits en Java. Ils sont comparés également aux trois leaders et principaux algorithmes : FPGrowth (Han et al., 2000) de l'approche projective, dEclat (Zaki and Gouda, 2003) de celle verticale, et Pre-Post (Deng et al., 2012) un algorithme hybride, comme implémentés dans la bibliothèque de fouille de données open source SPMF (Fournier-Viger et al., 2016).

Nos expérimentations sont réalisées sur une machine Dell Insprinon 7000 series, basée sur un processeur Intel i7-2 GHz, équipée de 16 Go de mémoire RAM exécutant Windows 10.

Dans nos expérimentations, nous avons adopté le protocole suivant. Nous lançons chaque algorithme plusieurs fois en fournissant les paramètres nécessaires tels que le fichier d'entrée, celui de destination, et le seuil du support. Afin d'analyser le comportement des algorithmes, nous les testons sur une étendue large de valeurs de support ; ainsi nous balayons la quasi-totalité des valeurs relatives de support allant (en descendant) de 0.9 à 0.01. Après l'obtention des résultats, nous prenons

leurs moyennes afin d'atténuer relativement les biais et aléas liés à l'environnement [2] (hardware, Java Virtual Machine, etc.).

5.3.2 Résultats et interprétation

Dans le reste du chapitre, nous rapportons quelques graphiques des expérimentations suivis de brèves interprétations. Comme notre objectif dans ce travail est principalement théorique, nous nous étalons pas sur les détails dans l'analyse des résultats, mais plutôt nous nous contenterons des aspects les plus remarquables. La discussion portera, comme mentionner en début de ce chapitre, sur les critères usuelles de temps d'exécution et la consommation de la mémoire.

Premièrement, nous pouvons observé dans la figure 5.1 que tous les algorithmes exhibent le même ordre de temps de calcul sur l'ensemble de données mushrooms, qui, rappelons le, est une base de données réelle et dense. La même remarque peut être observée sur la figure 5.2, où les différences en temps sont insignifiantes. Cependant, en terme de consommation mémoire, WAFI se comporte mieux que Prepost et dEclat, mais pas assez bien comparé à FPgrowth. Ceci est dû par l'extra mémoire causée par les informations relative à l'étiquetage des états, et celle nécessaire à la gestion de la file.

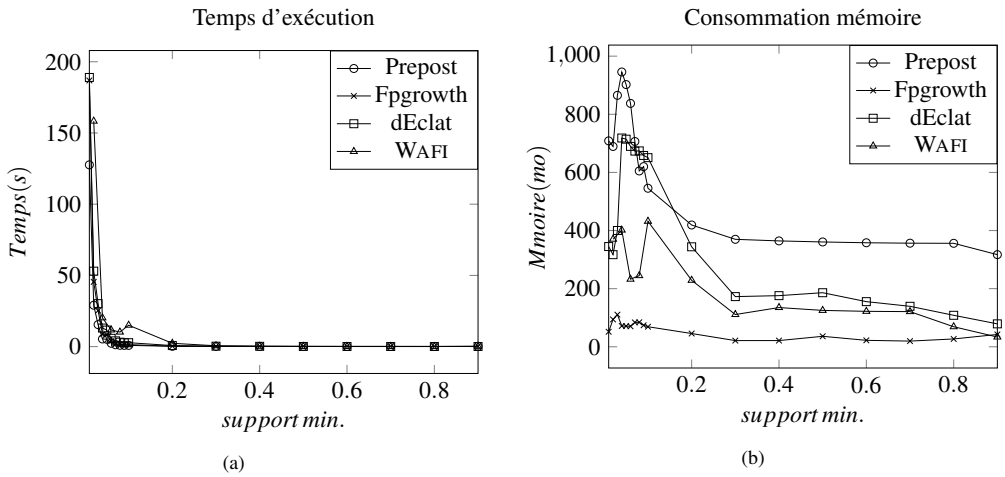


FIGURE 5.1 – Performances sur mushroom. (a) temps (b) mémoire

2. Nous avons simplifié la tâche car, en principe, mesurer la complexité en moyenne exige de disposer d'une distribution de probabilité sur les entrées

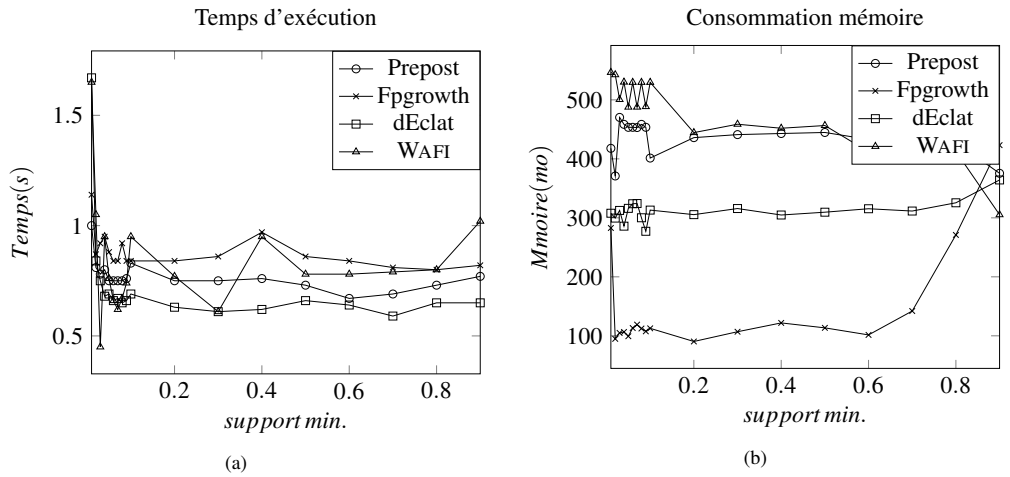


FIGURE 5.2 – Performances sur retail. (a) temps (b) mémoire

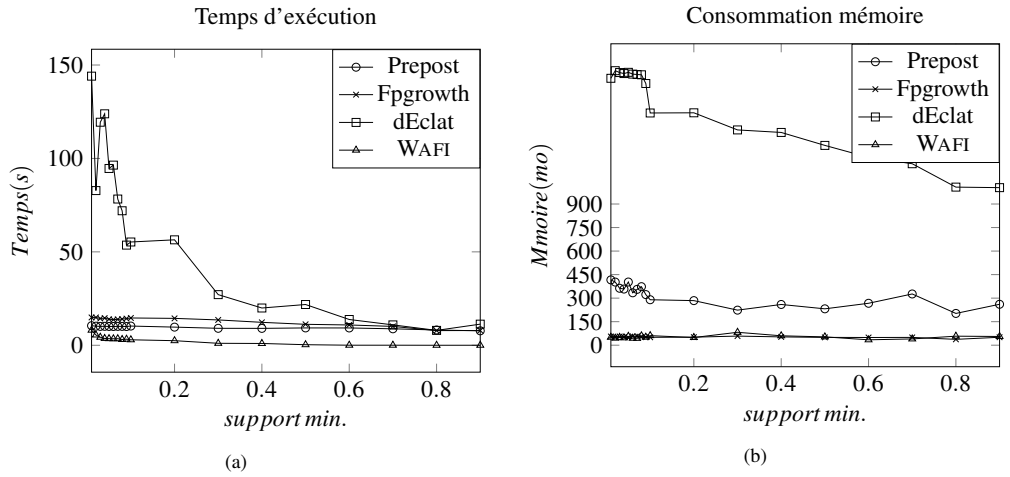


FIGURE 5.3 – Performances sur kddcup99. (a) temps (b) mémoire

Par ailleurs, la figure 5.3 illustre le cas où WAFI se comporte beaucoup mieux que ses pairs et les dépasse en ce qui concerne les deux métriques temps et espace devant la base de données la plus importante. Le même effet peut être constaté avec la figure 5.4 lorsque nous sélectionnons la base de données BMSWebView1. Néanmoins, pour le deuxième ensemble de données BMSWebView2, nous pouvons aisément remarqué sur la figure 5.5 la présence d'une différence insignifiante en terme de temps d'exécution, mais un écart évident en mémoire. En effet, dans ce cas, WAFI est dépassé par dEclat et FPGrowth relativement à la mémoire requise.

Un résultat négatif n'est pas toujours à négliger et peut véhiculer des enseigne-

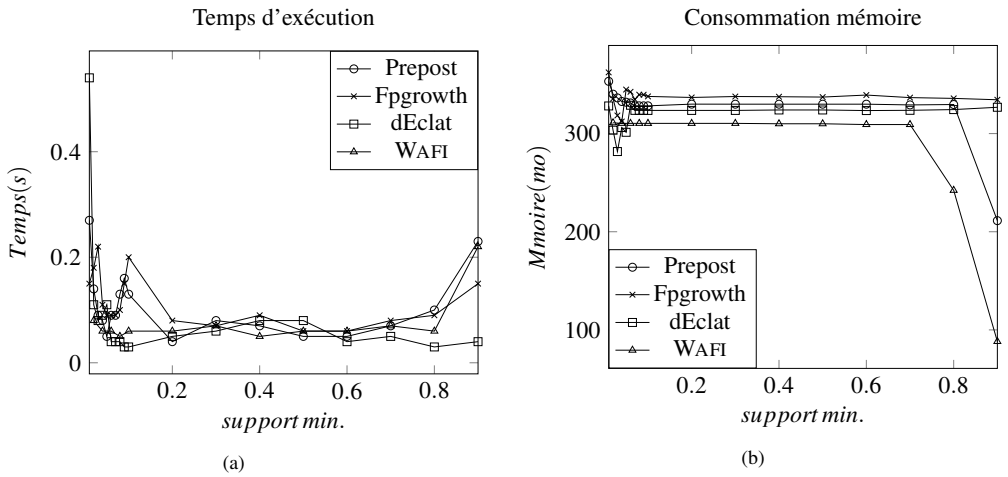


FIGURE 5.4 – Performances sur BMSWebView1. (a) temps (b) mémoire

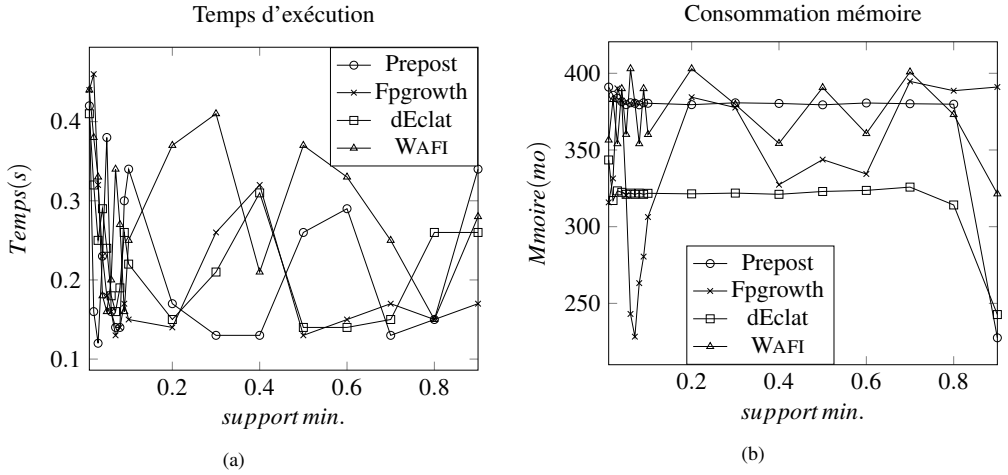


FIGURE 5.5 – Performances sur BMSWebView2. (a) temps (b) mémoire

ments ; les résultats médiocres de notre algorithme sont enregistrés sur la figure 5.6 lorsque appliqué sur la base de transactions kosarak. En dépit que ce dernier important ensemble de données est sparce, le comportement rapporté n'est pas clair pour nous en ce moment. Il existe diverses raisons dernière la hausse du temps et particulièrement de la mémoire exigée dans le graphique. Nous explorons ce cas et ses différents motifs possibles dans nos travaux futurs.

Tous les algorithmes agissent de manière similaire en terme de temps de réponse devant la base de données synthétique T10I4D100K comme est clairement montré dans la figure 5.7. Toutefois, WAFI est mieux, relativement à la mémoire, que l'algo-

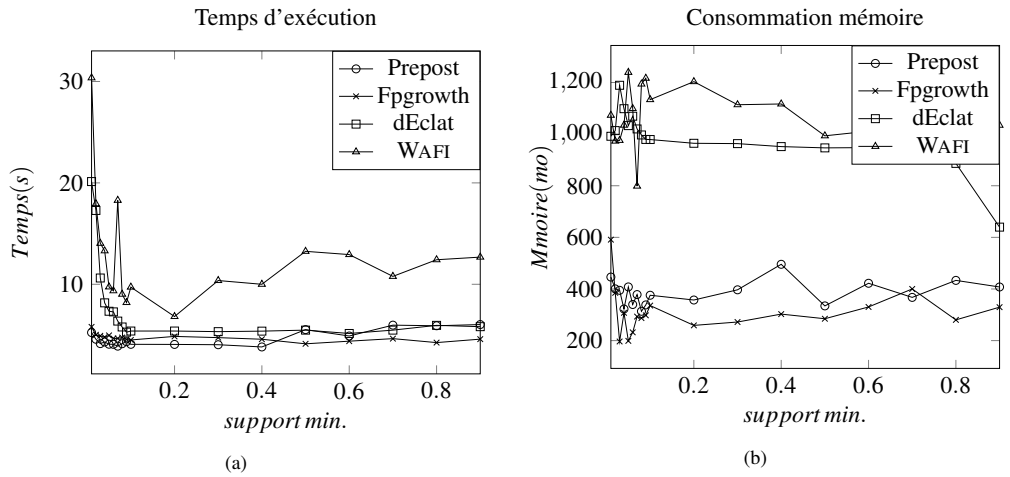


FIGURE 5.6 – Performances sur kosarak. (a) temps (b) mémoire

rithme Prepost. Mais, il exige plus que FPgrowth et dEclat.

Comme illustre la courbe de la figure 5.8, lorsque nous traitons notre dernier ensemble de données synthétique, WAFI affiche des indicateurs positifs relativement aux temps et espace. De plus, pour les valeurs trop faibles du support, il est dépassé par FPGrowth et Prepost. Ce résultat n'est pas surprenant, étant donné que ces derniers algorithmes adoptent l'ordre décroissant du support sur les items, qui donne toujours une implémentation plus rapides (Han et al., 2000; Zaki and Gouda, 2003; Schmidt-Thieme, 2004). Tandis que pour WAFI, nous utilisons le simple ordre des items codés par des entiers naturels.

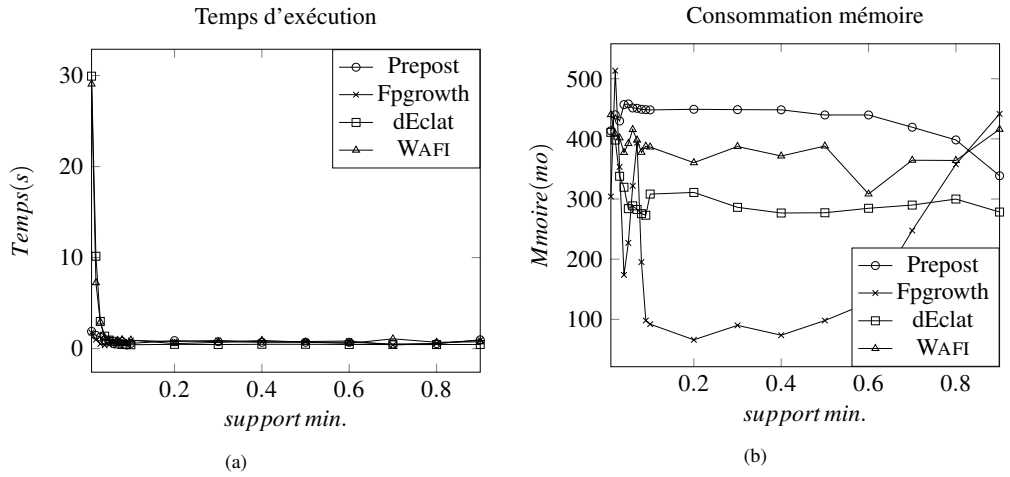


FIGURE 5.7 – Performances sur T10I4D100K. (a) temps (b) mémoire

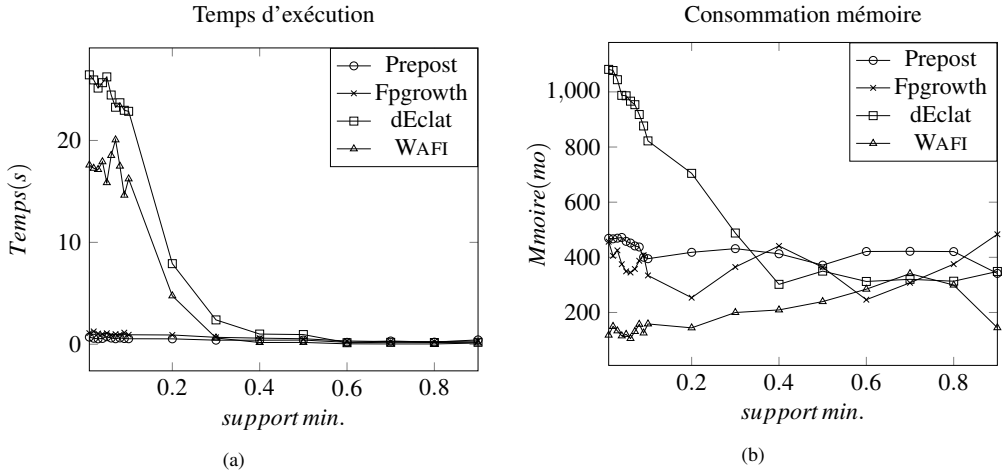


FIGURE 5.8 – Performances sur I16T100K. (a) temps (b) mémoire

En outre, un fait dûment établi est l'implication de plusieurs autre paramètres qui influent substantiellement sur les résultats obtenus. Par exemple, nous avons élaboré un algorithme basé sur une liste, où a priori toute stratégie est envisageable et fonctionnelle. Cependant, il est évident que les stratégies de gestion d'une liste et ses possibles implémentations nécessitent différentes temps et espace. Finalement, l'ordre des items affecte, comme noté auparavant, la taille de la structure de données et par conséquent l'efficacité des divers modules et algorithmes.

5.4 Conclusion

Dans ce chapitre, une implémentation de notre formalisme a été discutée. Nous avons montré la réalisabilité des concepts et idées introduits dans le chapitre précédent. D'une part, les résultats obtenus confirment le caractère unificateur de notre approche. WAFI, comme une implémentation de notre modèle, jouit d'une généricité permettant ainsi une conception flexible et ouverte. C'est ainsi, que les principales approches de la littérature peuvent être dérivées de la notre comme cas spéciaux. Par ailleurs, l'analyse expérimentale des performances de WAFI a, sans prétention, révélé, d'autre part, des métriques et indices concurrentiels notamment en ce qui concerne le temps de réponse et la mémoire requise. La présente implémentation est, bien entendu, objet à plusieurs extensions et améliorations que nous allons investiguer ultérieurement dans des travaux subséquents.

Quatrième partie

Épilogue

Conclusion générale

Le danger qui menace les chercheurs aujourd'hui
serait de conclure qu'il n'y a plus rien à découvrir.

Pierre Joliot-Curie

Devant le nombre impressionnant d'algorithmes introduits pour résoudre le problème de la fouille de motifs fréquents, nombreux sont les chercheurs qui ont conclu que l'état de la recherche dans ce domaine requière des contributions théoriques et des analyses profondes que simplement la production de nouveaux algorithmes qui ont prouvé d'être difficilement séparables. Notre travail est un essai dans cette direction.

Ainsi, nous avons montré que le modèle des séries formelles ou les langages pondérés ouvre de nouvelles perspectives pour la prise en charge de la question élémentaire de la fouille de motifs fréquents et des sujets liés ou étendus. Les propriétés clefs de notre modèle sont multiple : premièrement, il est simple, intuitif et complet. Deuxièmement, il combine de manière intrinsèque à la fois les aspects quantitatifs que qualitatifs du problème de base que nous avons à solutionner. En outre, il a été prouvé formellement mais aussi à travers des expérimentations extensives que l'implémentation du modèle proposé offre une efficacité compétitive.

Pour récapituler, nous avons tout au long de cette thèse, après l'introduction du bagage requis et avoir fait le tour des principales approches de la littérature, exprimé le problème de la fouille de motifs comme une série formelle qui fait correspondre aux mots du monoïde libre A^* modélisant les motifs des coefficients pris dans le semi-anneau de comptage $(\mathbb{N}, +, \times, 0, 1)$. Autrement dit, une série formelle dont l'étendue représente les motifs et les coefficients sont leurs supports associés. Cette façon de voir le problème constitue un formalisme général et unifié qui prouve être capable de traduire les travaux entrepris dans les approches prédominantes de l'état de l'art. Comme escompté, toutes ces approches calculent la même série formelle $\mathbb{S}_D$ que nous avons définie et baptisée le polynôme de sous-séquences de l'ensemble de données. En outre, nous avons expliqué que notre contribution ne se limite pas à une

vision théorique pure, mais peut prétendre à une implémentation. Aussi, nous avons passé à une réalisation de nos séries formelles à travers des automates à multiplicité.

L'accent est mis sur le fait que divers choix du semi-anneau de travail où encore les schémas de pondération des transitions/états conduisent à des calculs différents mais équivalents de l'ensemble de motifs fréquents. Quelle que soit la configuration retenue de l'automate, nous avons prouvé que sa taille est toujours bornée par la taille de l'ensemble de données, et que la complexité de l'algorithme de fouille associé est sensible à la taille de la sortie. Nous atteignons une complexité spatiale en $O(|Q|)$ et temporelle en $O(|\mathscr{F}||Q|)$. Dans le même contexte, notre formalisation a révélé, au passage, une bonne équivalence entre le problème étudié et celui de la déterminisation d'automates à multiplicité acycliques augmentés par des ε-transitions en guise de représentation des bases de données à fouiller. Par ailleurs, nous estimons que notre modèle peut être généralisé pour la fouille de motifs plus complexes tels que les motifs pondérés, les séquences, les arbres, voire même les graphes, vu que toutes ces classes d'objets sont des structures rationnelles.

L'imperfection est humaine, et le travail entrepris dans cette thèse ne fait pas l'exception. Par conséquent, il peut donc être étendu sur plus d'un axe. Nous retenons les points essentiels ci-dessous énumérés.

— La généralisation, comme nous l'avons conjecturé à la fouille d'objets plus élaborés comme les motifs pondérés, les séquences, les arbres, etc,

— Pour suivre la philosophie des représentations compactes, il serait intéressant d'étudier aussi l'adaptation du modèle introduit ici à la fouille de motifs fréquents fermés et fréquents maximaux,

— Quoi que non triviale, l'étude de la minimisabilité de l'automate défini constitue une autre piste convoitée à explorer,

— Avec le Big data era, faire porter l'implémentation actuelle aux plateformes parallèles et/ou distribués comme MapReduce peut constitué également un créneau digne de découverte,

— Le présent travail concerne l'analyse d'association, nous estimons qu'étendre l'effort de formalisation et d'unification pour d'abord les autres tâches de fouille, puis le pousser aux autres phases capitales du processus d'extraction de connaissances telles que la préparation de données, l'analyse et la présentation des résultats constitue des questions fondamentales qui nécessitent des investigations futures,

— Hors les aspects fondamentaux, nous avons traduit le modèle proposé en un algorithme de déterminisation d'automates à multiplicité acyclique, que nous avons implémenté et évalué. Bien que cela n'était pas notre premier objectif, nous avons réussi à atteindre une version itérative préliminaire. Comme est clairement détaillé dans le chapitre sur les expérimentations, l'implémentation actuelle n'est pas complètement optimisée. Toutefois, nous avons identifié déjà quelque axes d'amélioration, qui feront l'objet de travaux postérieurs.

En fin, nous souhaitons que ce travail motivera d'autres travaux subséquents pour attaquer non seulement ce problème basique, mais aussi ses nombreuses ramifications par un formalisme fascinant, soit de l'angle et la sémantique des automates à multiplicité ou du point de vu algébrique.

Pour des raisons de simplicité, précision, et modération de la longueur du document et aussi par insuffisance du bagage requis, le dernier aspect n'est pas développé dans cette thèse. Nous estimons que la vision algébrique des séries formelles mérite également une exploration et peut mener à d'autres résultats fondamentaux ou pratiques.

ANNEXE A

Annexe

Preuve de la proposition 4.2.2

Démonstration. Nous construisons l'automate $\mathscr{C}$ par déterminisation des deux automates $\mathscr{A}$ et $\mathscr{B}$ vus comme un seul ayant deux états initiaux. Autrement dit, déterminiser l'automate $\mathscr{A} \cup \mathscr{B}$.
Soit $\mathscr{A} = (Q_A, A_1, \mu_A, \lambda_A, \gamma_A)$ un AMP isomorphe, via h_A, à l'automate $\mathscr{P}_X = (Q_X, A_X, \mu_X, \lambda_X, \gamma_X)$ qui réalise le polynôme $\mathbb{P}_X$ associé à l'ensemble de données X, et $\mathscr{B} = (Q_B, A_2, \mu_B, \lambda_B, \gamma_B)$ celui isomorphe, via h_B, à l'automate $\mathscr{P}_Y = (Q_Y, A_Y, \mu_Y, \lambda_Y, \gamma_Y)$ qui réalise le polynôme $\mathbb{P}_Y$ associé à l'ensemble de données Y.
Nous donnons ci-dessous premièrement la construction de l'automate à multiplicité préfixiel $\mathscr{C} = (Q_C, A_1 \cup A_2, \mu_C, \lambda_C, \gamma_C)$ puis une application h de l'ensemble d'états Q_C à l'ensemble d'états de l'automate $\mathscr{P}_{X \cup Y}$ qui est l'étendu du polynôme $\mathbb{P}_{X \cup Y}$ (q_A et q_B sont des éléments de Q_A et Q_B respectivement).

1. $Q_C = T \cup W \cup Z$, où
   - $T = \{\{q_A, q_B\} \mid h_A(p) = h_B(q)\}$
   - $W = \{\{q_A\} \mid h_A(q_A) \in h_A(Q_A) \setminus h_B(Q_B)\}$
   - $Z = \{\{q_B\} \mid h_B(q_B) \in h_B(Q_B) \setminus h_A(Q_A)\}$
2. $\mu_C(\{q_A, q_B\}) = \begin{cases} 1 & \text{pour l'état initial} (q_{A0}, q_{B0}) \text{ en jumelant ceux de } \mathscr{A} \text{ et } \mathscr{B} \\ 0 & \text{ailleurs} \end{cases}$
3. $\lambda_C(q, a, q\prime)$ est une matrice binaire, pour q et $q\prime \in Q_C$ et $a \in A_1 \cup A_2$. C-à-d, le poids de chaque transition est soit 0 ou 1. Nous avons cinq cas selon les sous-ensembles T, W ou Z auxquels appartiennent q et $q\prime$:
   - $q \in T$ et $q\prime \in T$: $\lambda_C(\{q_A, q_B\}, a, \{q\prime_A, q\prime_B\}) = 1$, si $\lambda_A(q_A, a, q\prime_A)$ et $\lambda_B(q_B, a, q\prime_B)$ sont définies tous les deux,

— $q \in T$ et $q\prime \in W$: $\lambda_C(\{q_A, q_B\}, a, \{q\prime_A\}) = 1$, si seulement $\lambda_A(q_A, a, q\prime_A)$ est définie,

— $q \in T$ et $q\prime \in Z$: $\lambda_C(\{q_A, q_B\}, a, \{q\prime_B\}) = 1$, si seulement $\lambda_B(q_B, a, q\prime_B)$ est définie,

— $q \in W$: si définie $q\prime$ peut uniquement appartenir à W (fermeture des ensembles préfixiels) :
$\lambda_C(\{q_A\}, a, \{q\prime_A\}) = 1$ si $\lambda_A(q_A, a, q\prime_A)$ existe,

— $q \in Z$: pour la même raison, si définie $q\prime$ peut seulement appartenir à Z :
$\lambda_C(\{q_B\}, a, \{q\prime_B\}) = 1$ si $\lambda_B(q_B, a, q\prime_B)$ existe,

Afin de simplifier la preuve, notons par Δ_C la fonction λ_C, de même δ_A, δ_B, δ_X et δ_Y dénotent les fonctions λ_A, λ_B, λ_X et λ_Y, nous pouvons récapituler la fonction λ_C dans les cas suivants :

$$\Delta_C(q,a) = \begin{cases} \{q\prime_A, q\prime_B\} & \text{si} \quad q = \{q_A, q_B\} \in T \wedge \delta_A(q_A,a) = q\prime_A \wedge \delta_B(q_B,a) = q\prime_B \\ \{q\prime_A\} & \text{si} \quad \begin{cases} q = \{q_A, q_B\} \in T \wedge \delta_A(q_A,a) = q\prime_A \wedge \delta_B(q_B,a) = \emptyset \\ q = \{q_A\} \in W \wedge \delta_A(q_A,a) = q\prime_A \end{cases} \\ \{q\prime_B\} & \text{si} \quad \begin{cases} q = \{q_A, q_B\} \in T \wedge \delta_B(q_B,a) = q\prime_B \wedge \delta_A(q_A,a) = \emptyset \\ q = \{q_B\} \in Z \wedge \delta_B(q_B,a) = q\prime_B \end{cases} \end{cases}$$

$$\gamma_C(I) = \begin{cases} \gamma_A(q_A) + \gamma_B(q_B) & \text{if} \quad I = \{q_A, q_B\} \in T \\ \gamma_A(q_A) & \text{if} \quad I = \{q_A\} \in W \\ \gamma_B(q_B) & \text{if} \quad I = \{q_B\} \in Z \end{cases}$$

Définissons également l'application h de Q_C dans Étendue($\mathbb{P}_{X \cup Y}$) comme suit :

4. $h(I) = \begin{cases} h_A(\{q_A\}) & \text{si} \quad I = \{q_A, q_B\} \in T, \text{ ou } I = \{q_A\} \in W \\ h_B(\{q_B\}) & \text{si} \quad I = \{q_B\} \in Z \end{cases}$

Maintenant, nous devons vérifier que h représente bien un isomorphisme d'automates.

Soit I un état de Q_C :

1. Il est clair de la définition du vecteur μ_C que l'état initial de $\mathscr{C}$ est lié par h à l'état initial de Étendue($\mathbb{P}_{X \cup Y}$) qui est le couple $(\varepsilon, \varepsilon)$

2. $h(\Delta_C(q,a)) =$

$$\begin{cases} h(\{q\prime_A, q\prime_B\}) & \text{si} \quad q = \{q_A, q_B\} \in T \wedge \delta_A(q_A,a) = q\prime_A \wedge \delta_B(q_B,a) = q\prime_B \\ h(\{q\prime_A\}) & \text{si} \begin{cases} q = \{q_A, q_B\} \in T \wedge \delta_A(q_A,a) = q\prime_A \wedge \delta_B(q_B,a) = \emptyset \\ q = \{q_A\} \in W \wedge \delta_A(q_A,a) = q\prime_A \end{cases} \\ h(\{q\prime_B\}) & \text{si} \begin{cases} q = \{q_A, q_B\} \in T \wedge \delta_B(q_B,a) = q\prime_B \wedge \delta_A(q_A,a) = \emptyset \\ q = \{q_B\} \in Z \wedge \delta_B(q_B,a) = q\prime_B \end{cases} \end{cases}$$

Selon la définition de l'application h, nous obtenons :

$h(\Delta_C(q,a)) =$

$$\begin{cases} h_A(\{q\prime_A\}) & \text{si} \quad q = \{q_A, q_B\} \in T \wedge \delta_A(q_A,a) = q\prime_A \wedge \delta_B(q_B,a) = q\prime_B \\ h_A(\{q\prime_A\}) & \text{si} \begin{cases} q = \{q_A, q_B\} \in T \wedge \delta_A(q_A,a) = q\prime_A \wedge \delta_B(q_B,a) = \emptyset \\ q = \{q_A\} \in W \wedge \delta_A(q_A,a) = q\prime_A \end{cases} \\ h_B(\{q\prime_B\}) & \text{si} \begin{cases} q = \{q_A, q_B\} \in T \wedge \delta_B(q_B,a) = q\prime_B \wedge \delta_A(q_A,a) = \emptyset \\ q = \{q_B\} \in Z \wedge \delta_B(q_B,a) = q\prime_B 1 \end{cases} \end{cases}$$

$=$

$$\begin{cases} h_A(\delta_A(q_A,a)) & \text{si} \quad q = \{q_A, q_B\} \in T \wedge \delta_A(q_A,a) = q\prime_A \wedge \delta_B(q_B,a) = q\prime_B \\ h_A(\delta_A(q_A,a)) & \text{si} \begin{cases} q = \{q_A, q_B\} \in T \wedge \delta_A(q_A,a) = q\prime_A \wedge \delta_B(q_B,a) = \emptyset \\ q = \{q_A\} \in W \wedge \delta_A(q_A,a) = q\prime_A \end{cases} \\ h_B(\delta_B(q_B,a)) & \text{si} \begin{cases} q = \{q_A, q_B\} \in T \wedge \delta_B(q_B,a) = q\prime_B \wedge \delta_A(q_A,a) = \emptyset \\ q = \{q_B\} \in Z \wedge \delta_B(q_B,a) = q\prime_B \end{cases} \end{cases}$$

Comme les deux applications h_A et h_B sont des isomorphismes d'automates à multiplicité :

$h(\Delta_C(q,a)) =$

$$\begin{cases} \delta_X(h_A(\{q_A\}),a) & \text{si} \quad q = \{q_A, q_B\} \in T \wedge \delta_A(q_A,a) = q\prime_A \wedge \delta_B(q_B,a) = q\prime_B \\ \delta_X(h_A(\{q_A\}),a) & \text{si} \begin{cases} q = \{q_A, q_B\} \in T \wedge \delta_A(q_A,a) = q\prime_A \wedge \delta_B(q_B,a) = \emptyset \\ q = \{q_A\} \in W \wedge \delta_A(q_A,a) = q\prime_A \end{cases} \\ \delta_Y(h_B(\{q_B\}),a) & \text{si} \begin{cases} q = \{q_A, q_B\} \in T \wedge \delta_B(q_B,a) = q\prime_B \wedge \delta_A(q_A,a) = \emptyset \\ q = \{q_B\} \in Z \wedge \delta_B(q_B,a) = q\prime_B \end{cases} \end{cases}$$

Selon notre application définie h, et comme toutes nos fonctions de poids sont binaires, nous obtenons pour chaque cas : $h(\Delta_C(q,a)) = \delta_{X \cup Y}(h(q),a)$

Maintenant, nous considérons la fonction de poids de sortie pour seulement le premier cas (les autres cas sont simples à prouver). Soit $I \in Q_C$. Quand $I \in T$ nous

avons :

$$\begin{aligned}
\gamma_C(I) &= \gamma_C(\{q_A, q_B\}) \\
&= \gamma_A(q_A) + \gamma_B(q_B) \\
&= \gamma_X(h_A(\{q_A\})) + \gamma_Y(h_B(\{q_B\})) \text{ comme } h_A, \text{ et } h_B \text{ sont des isomorphismes} \\
&= \langle \mathbb{P}_X, h_A(\{q_A\})\rangle + \langle \mathbb{P}_Y, h_B(\{q_B\})\rangle : \text{par la définition de la fonction} \gamma_X \text{ et } \gamma_Y \\
&= \langle \mathbb{P}_X, h_A(\{q_A\})\rangle + \langle \mathbb{P}_Y, h_A(\{q_A\})\rangle : \text{car } I \in T \\
&= \langle \mathbb{P}_X + \mathbb{P}_Y, h_A(\{q_A\})\rangle \\
&= \langle \mathbb{P}_{X\cup Y}, h_A(\{q_A\})\rangle \\
&= \langle \mathbb{P}_{X\cup Y}, h(\{q_A, q_B\})\rangle \\
&= \langle \mathbb{P}_{X\cup Y}, h(I)\rangle \\
&= \gamma_{X\cup Y}(h(I))
\end{aligned}$$

Finalement, il est aisé de montrer que h est bijective, étant donné qu'elle est dérivée de deux isomorphismes d'automates à multiplicité. La définition de h implique h_A ou h_B qui sont toutes les deux bijectives.

□

Bibliographie

Achar, A., Laxman, S., and Sastry, P. (2012). A unified view of the apriori-based algorithms for frequent episode discovery. *Knowl. Inf. Syst.*, 31(2) :223–250. (Cité en page 104.)

Achtert, E., Kriegel, H., and Zimek, A. (2008). ELKI : A software system for evaluation of subspace clustering algorithms. In *Scientific and Statistical Database Management, 20th International Conference, SSDBM 2008, Hong Kong, China, July 9-11, 2008, Proceedings*, pages 580–585. (Cité en page 48.)

Agarwal, R. C., Aggarwal, C. C., and Prasad, V. V. V. (2000). Depth first generation of long patterns. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, MA, USA, August 20-23, 2000*, pages 108–118. (Cité en page 74.)

Aggarwal, C. C. (2013). *Outlier Analysis*. Springer. (Cité en page 39.)

Aggarwal, C. C. (2015). *Data Mining - The Textbook*. Springer. (Cité en page 52.)

Aggarwal, C. C., Bhuiyan, M., and Hasan, M. A. (2014). Frequent pattern mining algorithms : A survey. In *Frequent Pattern Mining*, pages 19–64. (Cité en pages 7, 35 et 52.)

Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA.*, pages 94–105. (Cité en page 39.)

Agrawal, R., Imielinski, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proc of The 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington DC, USA. (Cité en pages 6, 27, 29, 30, 36 et 60.)

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., and Verkamo, A. I. (1996). Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI/MIT Press. (Cité en page 60.)

Agrawal, R. and Shafer, J. C. (1996). Parallel mining of association rules. *IEEE Trans. Knowl. Data Eng.*, 8(6) :962–969. (Cité en page 83.)

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499. (Cité en pages 6, 7, 29, 30, 56, 60, 63, 72, 93 et 109.)

Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, ICDE '95, pages 3–14, Washington, DC, USA. IEEE Computer Society. (Cité en page 82.)

Altalhi, A. H., Luna, J. M., Vallejo, M., and Ventura, S. (2017). Evaluation and comparison of open source software suites for data mining and knowledge discovery. *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, 7(3). (Cité en pages 44, 45, 46, 48 et 52.)

Angiulli, F., Ianni, G., and Palopoli, L. (2001). On the complexity of mining association rules. In Celentano, A., Tanca, L., and Tiberio, P., editors, *Nono Convegno Nazionale Sistemi Evoluti per Basi di Dati, SEBD 2001, Venezia, Italy, 27-29 Giugno 2001*, pages 177–184. (Cité en page 62.)

Atluri, G., Gupta, R., Fang, G., Pandey, G., Steinbach, M., and Kumar, V. (2009). Association analysis techniques for bioinformatics problems. In *Bioinformatics and Computational Biology, First International Conference, BICoB 2009, New Orleans, LA, USA, April 8-10, 2009. Proceedings*, pages 1–13. (Cité en page 40.)

Barbara, D. (2002). *Applications of Data Mining in Computer Security*. Kluwer Academic Publishers, Norwell, MA, USA. (Cité en page 43.)

Barbut, M. and Monjardet, B. (1970). *Ordre et classification : algèbre et combinatoire*. Classiques Hachette. Hachette. (Cité en pages 8, 12, 65, 78 et 104.)

Bayardo, R. (1998). Efficiently mining long patterns from databases. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA.*, pages 85–93. (Cité en pages 8, 74 et 75.)

Bayardo, R., Goethals, B., and Zaki, M. J., editors (2004). *FIMI '04, Frequent Itemset Mining Implementations, Proceedings of the ICDM 2004 Workshop on Frequent Itemset Mining Implementations, 01 November 2004, Brighton, UK*. (Cité en pages 44, 52 et 80.)

Ben-Gal, I. (2010). Outlier detection. In *Data Mining and Knowledge Discovery Handbook*, pages 117–130. Springer. (Cité en page 39.)

Bergson, H. (1907). *L'Évolution créatrice*. Collection Bibliothèque de philosophie contemporaine. Presses universitaires de France. (Cité en page 5.)

Berrar D., Dubitzky W., G. M. and Eils, R. (2001). Analysis of gene expression and drug activity data by knowledge-based association mining. In *Proceedings of Critical Assessment of Microarray Data Analysis Techniques (CAMDA' 01)*. (Cité en page 41.)

Berstel, J. and Reutenauer, C. (1988). *Rational series and their languages*. EATCS monographs on theoretical computer science. Springer-Verlag. (Cité en pages 12 et 94.)

Berthold, M. R., Cebron, N., Dill, F., Gabriel, T. R., Kötter, T., Meinl, T., Ohl, P., Thiel, K., and Wiswedel, B. (2009). KNIME - the konstanz information miner : version 2.0 and beyond. *SIGKDD Explorations*, 11(1) :26–31. (Cité en page 46.)

Bodon, F. (2003). A fast APRIORI implementation. In Goethals, B. and Zaki, M. J., editors, *FIMI '03, Frequent Itemset Mining Implementations, Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations, 19 December 2003, Melbourne, Florida, USA*, volume 90 of *CEUR Workshop Proceedings*. CEUR-WS.org. (Cité en page 63.)

Bodon, F. (2006). A survey on frequent itemset mining. Technical report, Budapest University of Technology and Economics. (Cité en page 29.)

Bondy, J. A. (1976). *Graph Theory With Applications*. Elsevier Science Ltd., Oxford, UK. (Cité en page 12.)

Borgelt, C. (2003). Efficient implementations of apriori and eclat. In *Proc. 1st IEEE ICDM Workshop on Frequent Item Set Mining Implementations (FIMI 2003, Melbourne, FL). CEUR Workshop Proceedings 90*, page 90. (Cité en page 63.)

Borgelt, C. (2012). Frequent item set mining. *Wiley Interdisc. Rew. : Data Mining and Knowledge Discovery*, 2(6) :437–456. (Cité en pages x, 7, 29, 52, 70, 73 et 118.)

Brin, S., Motwani, R., Ullman, J. D., and Tsur, S. (1997). Dynamic itemset counting and implication rules for market basket data. In Peckham, J., editor, *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA.*, pages 255–264. ACM Press. (Cité en page 64.)

Brown, M.P, G. W. L. D. C. N. S. C. F. T. A. M. and D., H. (2000). Knowledge-based analysis of microarray gene expression data by using support vector machines. In *In Proceedings of National Academy of Sciences, Vol. 97 N1*. (Cité en page 41.)

Burdick, D., Calimlim, M., and Gehrke, J. (2001). MAFIA : A maximal frequent itemset algorithm for transactional databases. In *Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany*, pages 443–452. (Cité en page 74.)

Calders, T., Rigotti, C., and Boulicaut, J. (2004). A survey on condensed representations for frequent sets. In *Constraint-Based Mining and Inductive Databases, European Workshop on Inductive Databases and Constraint Based Mining, Hinterzarten, Germany, March 11-13, 2004, Revised Selected Papers*, pages 64–80. (Cité en page 73.)

Caspard, N., Leclerc, B., and Monjardet, B. (2007). Ensembles ordonnés finis : concepts, résultats, usages. Université Paris1 Panthéon-Sorbonne (Post-Print and Working Papers) halshs-00197128, HAL. (Cité en page 12.)

Ceglar, A. and Roddick, J. F. (2006). Association mining. *ACM Comput. Surv.*, 38(2) :5. (Cité en page 52.)

Chaoji, V., Hasan, M. A., Salem, S., and Zaki, M. J. (2008). An integrated, generic approach to pattern mining : data mining template library. *Data Min. Knowl. Discov.*, 17(3) :457–495. (Cité en pages 48 et 104.)

Chen, C., Yeh, J., and Ke, H. (2010). Plagiarism detection using ROUGE and wordnet. *CoRR*, abs/1003.4065. (Cité en page 42.)

Chen, X., Ye, Y., Williams, G. J., and Xu, X. (2007). A survey of open source data mining systems. In *Emerging Technologies in Knowledge Discovery and Data Mining, PAKDD 2007, International Workshops, Nanjing, China, May 22-25, 2007, Revised Selected Papers*, pages 3–14. (Cité en page 44.)

Cheng, H., Yan, X., Han, J., and Hsu, C. (2007). Discriminative frequent pattern analysis for effective classification. In *ICDE*, pages 716–725. IEEE Computer Society. (Cité en page 36.)

Cheung, D. W., Han, J., Ng, V. T. Y., and Wong, C. Y. (1996). Maintenance of discovered association rules in large databases : An incremental updating technique. In *Proceedings of the Twelfth International Conference on Data Engineering, February 26 - March 1, 1996, New Orleans, Louisiana*, pages 106–114. (Cité en pages 8 et 81.)

Cheung, D. W.-L., Lee, S. D., and Kao, B. (1997). A general incremental technique for maintaining discovered association rules. In *Proceedings of the Fifth International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 185–194. World Scientific Press. (Cité en page 81.)

Cheung, W. and Zaïane, O. R. (2003). Incremental mining of frequent patterns without candidate generation or support constraint. In *7th International Database Engineering and Applications Symposium (IDEAS 2003), 16-18 July 2003, Hong Kong, China*, pages 111–116. (Cité en pages 93, 94 et 99.)

Cliche, A. (2012). *La Grande Unification en Science : Une approche conceptuelle*. Cybair Publishing Ltee. Canada. (Cité en page 5.)

Cohen, E., Halperin, E., Kaplan, H., and Zwick, U. (2002). Reachability and distance queries via 2-hop labels. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '02, pages 937–946, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics. (Cité en page 118.)

Cooley, R., Mobasher, B., and Srivastava, J. (1997). Web mining : Information and pattern discovery on the world wide web. In *9th International Conference on Tools with Artificial Intelligence, ICTAI '97, Newport Beach, CA, USA, November 3-8, 1997*, pages 558–567. IEEE Computer Society. (Cité en page 44.)

Copeland, G. P. and Khoshafian, S. (1985). A decomposition storage model. In Navathe, S. B., editor, *Proceedings of the 1985 ACM SIGMOD International Conference on Management of Data, Austin, Texas, May 28-31, 1985.*, pages 268–279. ACM Press. (Cité en page 65.)

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition. (Cité en pages 119 et 120.)

Davey, B. A. and Priestley, H. A. (1990). *Introduction to lattices and order*. Cambridge University Press, Cambridge. (Cité en pages 8, 78 et 104.)

Dean, J. and Ghemawat, S. (2008). Mapreduce : simplified data processing on large clusters. *Commun. ACM*, 51(1) :107–113. (Cité en pages 47 et 83.)

Demšar, J., Curk, T., Erjavec, A., Črt Gorup, Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., Štajdohar, M., Umek, L., Žagar, L., Žbontar, J., Žitnik, M., and Zupan, B. (2013). Orange : Data mining toolbox in python. *Journal of Machine Learning Research*, 14 :2349–2353. (Cité en page 46.)

Deng, Z. and Lv, S. (2015). Prepost$^+$: An efficient n-lists-based algorithm for mining frequent itemsets via children-parent equivalence pruning. *Expert Syst. Appl.*, 42(13) :5424–5432. (Cité en pages 72, 118 et 119.)

Deng, Z. and Wang, Z. (2010). A new fast vertical method for mining frequent patterns. *Int. J. Computational Intelligence Systems*, 3(6) :733–744. (Cité en pages 119, 120 et 123.)

Deng, Z., Wang, Z., and Jiang, J. (2012). A new algorithm for fast mining frequent itemsets using n-lists. *SCIENCE CHINA Information Sciences*, 55(9) :2008–2030. (Cité en pages 119 et 125.)

Dexters, N., Purdom, P., and Van Gucht, D. (2006). Analysis of candidate-based frequent itemset algorithms. In *Proc. ACM Symposium on Applied Computing, Data Mining Track.* (Cité en page 71.)

Dhaenens, C. and Jourdan, L. (2016). *Metaheuristics for Big Data*. Wiley. (Cité en page 84.)

Djenouri, Y., Drias, H., Habbas, Z., and Mosteghanemi, H. (2012). Bees swarm optimization for web association rule mining. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, Macau, China, December 4-7, 2012*, pages 142–146. (Cité en page 84.)

Droste, M., Stüber, T., and Vogler, H. (2010). Weighted finite automata over strong bimonoids. *Inf. Sci.*, 180(1) :156–166. (Cité en pages 95, 96 et 114.)

Eisen, M.B., S.-P. B. P. B. D. (1998). Cluster analysis and display of genome-wide expression patterns. In *Proceedings of National Academy of Sciences ;95(25) :14863-8.* (Cité en page 41.)

Eisner, J. (2003). Simpler and more general minimization for weighted finite-state automata. In Hearst, M. A. and Ostendorf, M., editors, *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2003, Edmonton, Canada, May 27 - June 1, 2003*. The Association for Computational Linguistics. (Cité en page 108.)

El-Hajj, M. and Zaïane, O. R. (2005). YAFIMA : Yet Another Frequent Itemset Mining Algorithm. *JDIM*, 3(4) :244–249. (Cité en pages 71 et 72.)

El-matarawy, A., El-ramly, M., and Bahgat, R. (2013). Plagiarism detection using sequential pattern mining. *International Journal of Applied Information Systems*, 5(2) :24–29. Published by Foundation of Computer Science, New York, USA. (Cité en page 43.)

Etzioni, O. (1996). The world-wide web : Quagmire or gold mine ? *Commun. ACM*, 39(11) :65–68. (Cité en page 43.)

Fayyad, U. M., Piatetsky-Shapiro, G., and Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine*, 17(3) :37–54. (Cité en pages 4 et 6.)

Fernando, B., Fromont, É., and Tuytelaars, T. (2012). Effective use of frequent itemset mining for image classification. In *ECCV (1)*, volume 7572 of *Lecture Notes in Computer Science*, pages 214–227. Springer. (Cité en page 37.)

Fimdr (2003). Fimi repository for frequent itemset mining, implementations and datasets. http ://http ://fimi.ua.ac.be/data/. (Cité en page 124.)

Flouvat, F., Marchi, F. D., and Petit, J. (2008). izi : A new toolkit for pattern mining problems. In *Foundations of Intelligent Systems, 17th International Symposium, ISMIS 2008, Toronto, Canada, May 20-23, 2008, Proceedings*, pages 131–136. (Cité en page 49.)

Fournier-Viger, P., Lin, J. C., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., and Lam, H. T. (2016). The SPMF open-source data mining library version 2. In *Proc. 19th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2016)*, pages 36–40. (Cité en pages 45, 124 et 125.)

Fournier-Viger, P., Lin, J. C., Vo, B., Truong, T. C., Zhang, J., and Le, H. B. (2017). A survey of itemset mining. *Wiley Interdisc. Rew. : Data Mining and Knowledge Discovery*, 7(4). (Cité en pages 7, 29, 35, 44, 45, 48 et 52.)

Franck, R. (1999). La pluralité des disciplines, l'unité du savoir et les connaissances ordinaires. *Sociologie et sociétés*, 31(1) :129–142. (Cité en page 5.)

Frawley, W. J., Piatetsky-Shapiro, G., and Matheus, C. J. (1991). Knowledge discovery in databases : An overview. In *Knowledge Discovery in Databases*, pages 1–30. AAAI/MIT Press. (Cité en page 4.)

Frécon, L. (2002). *Eléments de mathématiques discrètes*. Collection des sciences appliquées de l'INSA de Lyon. Presses polytechniques et universitaires romandes. (Cité en page 12.)

Freitas, A. A. (2013). *Data mining and knowledge discovery with evolutionary algorithms*. Springer Science & Business Media. (Cité en page 84.)

Fung, B. C. M., Wang, K., and Ester, M. (2003). Hierarchical document clustering using frequent itemsets. In *Proceedings of the Third SIAM International Conference on Data Mining, San Francisco, CA, USA, May 1-3, 2003*, pages 59–70. (Cité en page 39.)

Ghosh, A. and Nath, B. (2004). Multi-objective rule mining using genetic algorithms. *Inf. Sci.*, 163(1-3) :123–133. (Cité en page 84.)

Giacometti, A., Li, D. H., Marcel, P., and Soulet, A. (2013). 20 years of pattern mining : a bibliometric survey. *SIGKDD Explorations*, 15(1) :41–50. (Cité en page 27.)

Godin, R., Missaoui, R., and Alaoui, H. (1995). Incremental concept formation algorithms based on galois (concept) lattices. *Computational Intelligence*, 11 :246–267. (Cité en pages 8 et 104.)

Goethals, B. (2003). Survey on frequent pattern mining. *Univ. of Helsinki*, 19 :840–852. (Cité en pages 7, 52 et 71.)

Goethals, B. (2004). Memory issues in frequent itemset mining. In *Proceedings of the 2004 ACM Symposium on Applied Computing (SAC), Nicosia, Cyprus, March 14-17, 2004*, pages 530–534. (Cité en page 93.)

Goethals, B. and Zaki, M. J., editors (2003). *FIMI '03, Frequent Itemset Mining Implementations, Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations, 19 December 2003, Melbourne, Florida, USA*, volume 90 of *CEUR Workshop Proceedings*. CEUR-WS.org. (Cité en pages 7, 9, 44 et 52.)

Gouda, K. and Zaki, M. J. (2005). Genmax : An efficient algorithm for mining maximal frequent itemsets. *Data Min. Knowl. Discov.*, 11(3) :223–242. (Cité en pages 74 et 76.)

Grahne, G. and Zhu, J. (2003). High performance mining of maximal frequent itemsets. In *6th International Workshop on High Performance Data Mining*. (Cité en pages 74 et 76.)

Guichard, D. (2017). *An Introduction to Combinatorics and Graph Theory*. Whitman College-Creative Commons. (Cité en pages 33, 63 et 98.)

Gunopulos, D., Khardon, R., Mannila, H., Saluja, S., Toivonen, H., and Sharm, R. S. (2003). Discovering all most specific sentences. *ACM Trans. Database Syst.*, 28(2) :140–174. (Cité en page 62.)

Hahsler, M., Chelluboina, S., Hornik, K., and Buchta, C. (2011). The arules r-package ecosystem : Analyzing interesting patterns from large transaction data sets. *Journal of Machine Learning Research*, 12 :2021–2025. (Cité en page 47.)

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software : An update. *SIGKDD Explor. Newsl.*, 11(1) :10–18. (Cité en page 45.)

Han, J., Cheng, H., Xin, D., and Yan, X. (2007). Frequent pattern mining : Current status and future directions. *Data Mining and Knowledge Discovery*, 15(1) :55–86. (Cité en pages 7, 35, 52 et 82.)

Han, J., Kamber, M., and Pei, J. (2011). *Data Mining : Concepts and Techniques, 3rd edition*. Morgan Kaufmann. (Cité en pages 39, 52 et 82.)

Han, J., Pei, J., and Yin, Y. (2000). Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA.*, pages 1–12. (Cité en pages 7, 9, 68, 94, 112, 113, 118, 119, 125 et 129.)

He, Z., Xu, X., Huang, J. Z., and Deng, S. (2005). Fp-outlier : Frequent pattern based outlier detection. *Comput. Sci. Inf. Syst.*, 2(1) :103–118. (Cité en page 40.)

Heraguemi, K. E., Kamel, N., and Drias, H. (2016). Multi-swarm bat algorithm for association rule mining using multiple cooperative strategies. *Appl. Intell.*, 45(4) :1021–1033. (Cité en page 84.)

Hipp, J., Güntzer, U., and Nakhaeizadeh, G. (2000a). Algorithms for association rule mining - A general survey and comparison. *SIGKDD Explorations*, 2(1) :58–64. (Cité en pages 7, 9 et 52.)

Hipp, J., Güntzer, U., and Nakhaeizadeh, G. (2000b). *Mining Association Rules : Deriving a Superior Algorithm by Analyzing Today's Approaches*, pages 159–168. Springer Berlin Heidelberg, Berlin, Heidelberg. (Cité en page 72.)

Hoare, C. A. R. (1995). Unification of theories : A challenge for computing science. In *Recent Trends in Data Type Specification, 11th Workshop on Specification of Abstract Data Types Joint with the 8th COMPASS Workshop, Oslo, Norway, September 19-23, 1995, Selected Papers*, pages 49–57. (Cité en pages 5 et 6.)

Hopcroft, J. E., Motwani, R., and Ullman, J. D. (2001). *Introduction to automata theory, languages, and computation - (2. ed.)*. Addison-Wesley series in computer science. Addison-Wesley-Longman. (Cité en pages 18, 91, 97 et 107.)

Inokuchi, A., Washio, T., and Motoda, H. (2000). An apriori-based algorithm for mining frequent substructures from graph data. In *Principles of Data Mining and Knowledge Discovery, 4th European Conference, PKDD 2000, Lyon, France, September 13-16, 2000, Proceedings*, pages 13–23. (Cité en page 82.)

Jain, A. K. and Dubes, R. C. (1988). *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA. (Cité en page 37.)

Jean-Pierre, Mével, C. J. A. M. (1997). *Grand dictionnaire Hachette encyclopédique illustré*. Hachette. (Cité en page 5.)

Jiang, M., Tseng, S., and Su, C. (2001). Two-phase clustering process for outliers detection. *Pattern Recognition Letters*, 22(6/7) :691–700. (Cité en page 40.)

Kaya, M. (2006). Multi-objective genetic algorithm based approaches for mining optimized fuzzy association rules. *Soft Comput.*, 10(7) :578–586. (Cité en page 84.)

Khabzaoui, M., Dhaenens, C., and Talbi, E. (2008). Combining evolutionary algorithms and exact approaches for multi-objective knowledge discovery. *RAIRO - Operations Research*, 42(1) :69–83. (Cité en page 84.)

Kosala, R. and Blockeel, H. (2000). Web mining research : A survey. *SIGKDD Explorations*, 2(1) :1–15. (Cité en page 44.)

Kosters, W. A. and Pijls, W. (2003). Apriori, A depth first implementation. In *FIMI '03, Frequent Itemset Mining Implementations, Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations, 19 December 2003, Melbourne, Florida, USA*. (Cité en page 63.)

Kotala, P., Perera, A., Zhou, K., Perrizo, W., and Deckard, E. (2001). Gene expression profiling of dna microarray data using peano count trees (p-trees). In *Proceedings of the First Virtual Conference on Genomics and Bioinformatics*. (Cité en page 41.)

Kuok, C. M., Fu, A. W., and Wong, M. H. (1998). Mining fuzzy association rules in databases. *SIGMOD Record*, 27(1) :41–46. (Cité en page 84.)

Kurra, G., Niu, W., and Bhatnagar, R. (2001). Mining microarray expression data for classifier gene-cores. In *Proceedings of the 1st International Conference on Data Mining in Bioinformatics*, BIOKDD'01, pages 8–14, London, UK, UK. Springer-Verlag. (Cité en page 41.)

Lee, W. and Fan, W. (2001). Mining system audit data : Opportunities and challenges. *SIGMOD Record*, 30(4) :35–44. (Cité en page 43.)

Leskovec, J., Rajaraman, A., and Ullman, J. D. (2014). *Mining of Massive Datasets, 2nd Ed*. Cambridge University Press. (Cité en pages 29, 35, 42 et 52.)

Li, H., Wang, Y., Zhang, D., Zhang, M., and Chang, E. Y. (2008). Pfp : parallel fp-growth for query recommendation. In *Proceedings of the 2008 ACM Conference on Recommender Systems, RecSys 2008, Lausanne, Switzerland, October 23-25, 2008*, pages 107–114. (Cité en pages 48 et 83.)

Li, J., Li, H., Soh, D., and Wong, L. (2005). A correspondence between maximal complete bipartite subgraphs and closed patterns. In Jorge, A., Torgo, L., Brazdil, P., Camacho, R., and Gama, J., editors, *Knowledge Discovery in Databases : PKDD 2005, 9th European Conference on Principles and Practice of Knowledge Discovery in Databases, Porto, Portugal, October 3-7, 2005, Proceedings*, volume 3721 of *Lecture Notes in Computer Science*, pages 146–156. Springer. (Cité en page 79.)

Li, Z. and Zhou, Y. (2005). Pr-miner : automatically extracting implicit programming rules and detecting violations in large software code. In *Proceedings of the 10th European Software Engineering Conference held jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2005, Lisbon, Portugal, September 5-9, 2005*, pages 306–315. (Cité en page 43.)

Lichman, M. (2013). UCI machine learning repository. http ://archive.ics.uci.edu/ml. (Cité en page 124.)

Lin, D. and Kedem, Z. M. (1998). Pincer-search : A new algorithm for discovering the maximum frequent set. In *Advances in Database Technology - EDBT'98, 6th International Conference on Extending Database Technology, Valencia, Spain, March 23-27, 1998, Proceedings*, pages 105–119. (Cité en page 74.)

Lin, F., Le, W., and Bo, J. (2010). Research on maximal frequent pattern outlier factor for online high dimensional time-series outlier detection. *Journal of convergence information technology*, 5(10) :66–71. (Cité en page 40.)

Liu, B., Hsu, W., and Ma, Y. (1998). Integrating classification and association rule mining. In *KDD*, pages 80–86. AAAI Press. (Cité en page 36.)

Mannila, H. and Toivonen, H. (1997). Levelwise search and borders of theories in knowledge discovery. *Data Min. Knowl. Discov.*, 1(3) :241–258. (Cité en pages 56, 61 et 62.)

Mannila, H., Toivonen, H., and Verkamo, A. I. (1994). Efficient algorithms for discovering association rules. In *Knowledge Discovery in Databases : Papers from the 1994 AAAI Workshop, Seattle, Washington, July 1994. Technical Report WS-94-03*, pages 181–192. (Cité en pages 56 et 60.)

Mannila, H., Toivonen, H., and Verkamo, A. I. (1997). Discovery of frequent episodes in event sequences. *Data Min. Knowl. Discov.*, 1(3) :259–289. (Cité en page 105.)

Martín-Bautista, M. J., Sánchez, D., Chamorro-Martínez, J., Serrano, J., and Vila, M. A. (2004). Mining web documents to find additional query terms using fuzzy association rules. *Fuzzy Sets and Systems*, 148(1) :85–104. (Cité en pages ix, 41 et 42.)

Martínez-Ballesteros, M., Troncoso, A., Martínez-Álvarez, F., and Riquelme, J. C. (2016). Obtaining optimal quality measures for quantitative association rules. *Neurocomputing*, 176 :36–47. (Cité en page 84.)

Maurer, H. A., Kappe, F., and Zaka, B. (2006). Plagiarism - A survey. *J. UCS*, 12(8) :1050–1084. (Cité en page 42.)

Mehta, D. P. and Sahni, S. (2004). *Handbook Of Data Structures And Applications (Chapman & Hall/Crc Computer and Information Science Series.)*. Chapman & Hall/CRC. (Cité en pages 12, 53 et 68.)

Mielikäinen, T. (2005). Transaction databases, frequent itemsets, and their condensed representations. In *Knowledge Discovery in Inductive Databases, 4th International Workshop, KDID 2005, Porto, Portugal, October 3, 2005, Revised Selected and Invited Papers*, pages 139–164. (Cité en page 73.)

Mohri, M. (2009). Weighted automata algorithms. In Droste, M., Kuich, W., and Vogler, H., editors, *Handbook of Weighted Automata*, Monographs in Theoretical Computer Science. An EATCS Series, pages 213–254. Springer Berlin Heidelberg. (Cité en page 106.)

Naulaerts, S., Meysman, P., Bittremieux, W., Vu, T., Berghe, W. V., Goethals, B., and Laukens, K. (2015). A primer to frequent itemset mining for bioinformatics. *Briefings in Bioinformatics*, 16(2) :216–231. (Cité en pages 40 et 44.)

Négrevergne, B., Termier, A., Méhaut, J., and Uno, T. (2010). Discovering closed frequent itemsets on multicore : Parallelizing computations and optimizing memory accesses. In *Proceedings of the 2010 International Conference on High Performance Computing & Simulation, HPCS 2010, June 28 - July 2, 2010, Caen, France*, pages 521–528. (Cité en page 83.)

Ono, S. S., Satou, K., Shibayama, G., Ono, T., Yamamura, Y., Furuichi, E., Kuhara, S., and Takagi, T. (1997). Finding association rules on heterogeneous genome

data. In *In Proceedings of the Second Pacific Symposium on Biocomputing (PSB*, pages 397–408. Pac.Symp.Biocomput. (Cité en page 41.)

Osman, I. H. and Laporte, G. (1996). Metaheuristics : A bibliography. *Annals of Operations Research*, 63(5) :511–623. (Cité en page 83.)

Otey, M. E., Ghoting, A., and Parthasarathy, S. (2006). Fast distributed outlier detection in mixed-attribute data sets. *Data Min. Knowl. Discov.*, 12(2-3) :203–228. (Cité en page 40.)

Otey, M. E., Parthasarathy, S., Wang, C., Veloso, A., and Jr., W. M. (2004). Parallel and distributed methods for incremental frequent itemset mining. *IEEE Trans. Systems, Man, and Cybernetics, Part B*, 34(6) :2439–2450. (Cité en page 83.)

Oulad-Naoui, S. and Cherroun, H. (2015). Mining frequent itemsets : a reduction to acyclic weighted automata determinisation. In *Proceedings of the Conference-School on Discrete Mathematics and Computer Science, Algeria, Sidi Bel Abbès, November 15-19, 2015.*, pages 49–59. (Cité en page 9.)

Oulad-Naoui, S., Cherroun, H., and Ziadi, D. (2015a). Mining frequent itemsets : a formal unification. *CoRR*, abs/1502.02642. (Cité en page 9.)

Oulad-Naoui, S., Cherroun, H., and Ziadi, D. (2015b). A unifying polynomial model for efficient discovery of frequent itemsets. In *DATA 2015 - Proceedings of 4th International Conference on Data Management Technologies and Applications, Colmar, Alsace, France, 20-22 July, 2015.*, pages 49–59. (Cité en page 9.)

Oulad-Naoui, S., Cherroun, H., and Ziadi, D. (2017). A formal series-based unification of the frequent itemset mining approaches. *Knowl. Inf. Syst.*, 53(2) :439–477. (Cité en page 9.)

Owen, S., Anil, R., Dunning, T., and Friedman, E. (2011). *Mahout in Action*. Manning Publications Co., Greenwich, CT, USA. (Cité en page 47.)

Papadimitriou, C. H. (1994). *Computational complexity*. Addison-Wesley. (Cité en pages 32 et 62.)

Park, J. S., Chen, M., and Yu, P. S. (1995a). An effective hash based algorithm for mining association rules. In *SIGMOD Conference*, pages 175–186. ACM Press. (Cité en pages 29 et 64.)

Park, J. S., Chen, M.-S., and Yu, P. S. (1995b). Efficient parallel data mining for association rules. In *Proceedings of the fourth international conference on Information and knowledge management*, pages 31–36. ACM. (Cité en page 83.)

Pasquier, N., Bastide, Y., Taouil, R., and Lakhal, L. (1998). Pruning closed itemset lattices for associations rules. In Bouzeghoub, M., editor, *14ème Journées Bases de Données Avancées, 26-30 octobre 1998, Hammamet, Tunisie (Informal Proceedings)*. (Cité en pages 8 et 78.)

Pasquier, N., Bastide, Y., Taouil, R., and Lakhal, L. (1999). Discovering frequent closed itemsets for association rules. In *Proceedings of the 7th International*

Conference on Database Theory, ICDT '99, pages 398–416, London, UK, UK. Springer-Verlag. (Cité en page 79.)

Pei, J., Han, J., Lu, H., Nishio, S., Tang, S., and Yang, D. (2001). H-mine : Hyper-structure mining of frequent patterns in large databases. In *Proceedings of the 2001 IEEE International Conference on Data Mining, 29 November - 2 December 2001, San Jose, California, USA*, pages 441–448. (Cité en pages 63 et 71.)

Pijls, W. and Kosters, W. A. (2010). Mining frequent itemsets : a perspective from operations research. *Statistica Neerlandica*, 64(4) :367–387. (Cité en page 104.)

Pin, J.-E. (1998). Tropical semirings. In Gunawardena, J., editor, *Idempotency*, pages 50–69. Cambridge University Press. (Cité en page 19.)

Poincaré, H. (1908). *Science et méthode*. Bibliothèque de philosophie scientifique. Flammarion. (Cité en page 5.)

R Core Team (2013). *R : A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0. (Cité en page 46.)

Rácz, B. (2004). nonordfp : An fp-growth variation without rebuilding the fp-tree. In *FIMI '04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Brighton, UK, November 1, 2004*. (Cité en page 72.)

Rakotomalala, R. (2005). TANAGRA : un logiciel gratuit pour l'enseignement et la recherche. In *Extraction et gestion des connaissances (EGC'2005), Actes des cinquièmes journées Extraction et Gestion des Connaissances, Paris, France, 18-21 janvier 2005, 2 Volumes*, pages 697–702. (Cité en page 47.)

Ramis, E., Deschamps, C., and Odoux, J. (1977). *Cours de mathématiques spéciales*. Number vol. 4 in Cours de mathématiques spéciales : classes préparatoires et enseignement supérieur. Masson. (Cité en page 12.)

Rijsbergen, C. J. V. (1979). *Information Retrieval*. Butterworth. (Cité en page 41.)

Rungsawang, A., Tangpong, A., Laohawee, P., and Khampachua, T. (1999). Novel query expansion technique using apriori algorithm. In *TREC*, volume Special Publication 500-246. National Institute of Standards and Technology (NIST). (Cité en page 41.)

Rymon, R. (1992). Search through systematic set enumeration. In Nebel, B., Rich, C., and Swartout, W. R., editors, *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92). Cambridge, MA, October 25-29, 1992.*, pages 539–550. Morgan Kaufmann. (Cité en page 75.)

Said, A. M., Dominic, D. D., and Samir, B. B. (2013). Outlier detection scoring measurements based on frequent pattern technique. *Research journel of applied sciences, Engineering and Technology, 6 (8)*, pages 1340–1347. (Cité en page 40.)

Salomaa, A., Soittola, M., Bauer, F., and Gries, D. (1978). *Automata-theoretic aspects of formal power series*. Texts and monographs in computer science. Springer-Verlag. (Cité en pages 9, 12 et 94.)

Savasere, A., Omiecinski, E., and Navathe, S. B. (1995). An efficient algorithm for mining association rules in large databases. In *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland.*, pages 432–444. (Cité en pages 29, 64 et 65.)

Schmidt-Thieme, L. (2004). Algorithmic features of eclat. In *FIMI '04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Brighton, UK, November 1, 2004*. (Cité en pages 118, 124 et 129.)

Song, M., Song, I., Hu, X., and Allen, R. B. (2007). Integration of association rules and ontologies for semantic query expansion. *Data Knowl. Eng.*, 63(1) :63–75. (Cité en page 42.)

Steinbach, M., Tan, P.-N., Xiong, H., and Kumar, V. (2007). Objective measures for association pattern analysis. *Contemporary Mathematics*, 443(2007) :205. (Cité en page 82.)

Sucahyo, Y. G. and Gopalan, R. P. (2004). CT-PRO : A bottom-up non recursive frequent itemset mining algorithm using compressed fp-tree data structure. In Jr., R. J. B., Goethals, B., and Zaki, M. J., editors, *FIMI '04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Brighton, UK, November 1, 2004*, volume 126 of *CEUR Workshop Proceedings*. CEUR-WS.org. (Cité en pages 71 et 72.)

Szathmary, L. (2006). *Symbolic Data Mining Methods with the Coron Platform. (Méthodes symboliques de fouille de données avec la plate-forme Coron)*. PhD thesis, Henri Poincaré University, Nancy, France. (Cité en page 74.)

Talbi, E. (2002). A taxonomy of hybrid metaheuristics. *J. Heuristics*, 8(5) :541–564. (Cité en page 83.)

Tamayo, P, S. D. M. J. Z. Q. K. S. D. E. L. E. G. T. (1999). Interpreting patterns of gene expression with self-organizing maps : Methods and applications to hematopoietic differentiation. In *In Proceedings of National Academy of Sciences, Vol. 96*. (Cité en page 41.)

Tan, P., Steinbach, M., and Kumar, V. (2005). *Introduction to Data Mining*. Addison-Wesley. (Cité en pages 33, 52 et 61.)

Toivonen, H. (1996). Sampling large databases for association rules. In *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India*, pages 134–145. (Cité en page 64.)

Totad, S. G., Geeta, R. B., and Reddy, P. V. G. D. P. (2012). Batch incremental processing for fp-tree construction using fp-growth algorithm. *Knowl. Inf. Syst.*, 33(2) :475–490. (Cité en pages 94 et 99.)

Tuzhilin, A. and Adomavicius, G. (2002). Handling very large numbers of association rules in the analysis of microarray data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 396–404, New York, NY, USA. ACM. (Cité en page 41.)

Uno, T., Kiyomi, M., and Arimura, H. (2004). LCM ver. 2 : Efficient mining algorithms for frequent/closed/maximal itemsets. In Jr., R. J. B., Goethals, B., and Zaki, M. J., editors, *FIMI '04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Brighton, UK, November 1, 2004*, volume 126 of *CEUR Workshop Proceedings*. CEUR-WS.org. (Cité en pages 74 et 80.)

Valtchev, P., Missaoui, R., Godin, R., and Meridji, M. (2002). Generating frequent itemsets incrementally : two novel approaches based on galois lattice theory. *J. Exp. Theor. Artif. Intell.*, 14(2-3) :115–142. (Cité en pages 8, 81 et 94.)

Veloso, A., Jr., W. M., de Carvalho, M., Pôssas, B., Parthasarathy, S., and Zaki, M. J. (2002). Mining frequent itemsets in evolving databases. In *Proceedings of the Second SIAM International Conference on Data Mining, Arlington, VA, USA, April 11-13, 2002*, pages 494–510. (Cité en pages 8, 81 et 83.)

Wang, W., Yang, J., and Yu, P. S. (2000). Efficient mining of weighted association rules (WAR). In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, MA, USA, August 20-23, 2000*, pages 270–274. (Cité en page 82.)

Webb, G. I. (1995). OPUS : an efficient admissible algorithm for unordered search. *J. Artif. Intell. Res.*, 3 :431–465. (Cité en page 31.)

Webb, G. I. (2000). Efficient search for association rules. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, MA, USA, August 20-23, 2000*, pages 99–107. (Cité en page 31.)

Wille, R. (1982). Restructuring lattice theory : An approach based on hierarchies of concepts. In Rival, I., editor, *Ordered Sets*, volume 83 of *NATO Advanced Study Institutes Series*, pages 445–470. Springer Netherlands. (Cité en pages 65 et 78.)

Williams, G. (2011). *Data Mining with Rattle and R : The art of excavating data for knowledge discovery*. Use R ! Springer Science+Business Media, LLC. (Cité en page 46.)

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A. F. M., Liu, B., Yu, P. S., Zhou, Z., Steinbach, M., Hand, D. J., and Steinberg, D. (2008). Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, 14(1) :1–37. (Cité en pages 36 et 109.)

Xu, R. and II, D. C. W. (2005). Survey of clustering algorithms. *IEEE Trans. Neural Networks*, 16(3) :645–678. (Cité en page 37.)

Yang, Q. and Wu, X. (2006). 10 challenging problems in data mining research. *International Journal of Information Technology and Decision Making*, 5(4) :597–604. (Cité en pages 4 et 6.)

Ye, Y., Wang, D., Li, T., Ye, D., and Jiang, Q. (2008). An intelligent pe-malware detection system based on association mining. *Journal in Computer Virology*, 4(4) :323–334. (Cité en page 43.)

Yiu, M. L. and Mamoulis, N. (2003). Frequent-pattern based iterative projected clustering. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), 19-22 December 2003, Melbourne, Florida, USA*, pages 689–692. (Cité en page 39.)

Yu, D., Sheikholeslami, G., and Zhang, A. (2002). Findout : Finding outliers in very large datasets. *Knowl. Inf. Syst.*, 4(4) :387–412. (Cité en page 40.)

Zaki, M. J. (2000). Scalable algorithms for association mining. *IEEE Transaction on Knowledge and Data Engineering*, 12(3) :372–390. (Cité en pages 7, 65, 94, 110, 111, 119 et 123.)

Zaki, M. J. (2002). Efficiently mining frequent trees in a forest. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pages 71–80. ACM. (Cité en page 82.)

Zaki, M. J. and Gouda, K. (2003). Fast vertical mining using diffsets. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*, pages 326–335. (Cité en pages 7, 63, 67, 110, 125 et 129.)

Zaki, M. J. and Hsiao, C. (2002). CHARM : an efficient algorithm for closed itemset mining. In *Proceedings of the Second SIAM International Conference on Data Mining, Arlington, VA, USA, April 11-13, 2002*, pages 457–473. (Cité en page 79.)

Zaki, M. J. and Ogihara, M. (1998). Theoretical foundations of association rules. In *3rd ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*. (Cité en pages 8, 30, 65, 78, 79 et 104.)

Zaki, M. J. and Wagner Meira, J. (2014). *Data Mining and Analysis : Fundamental Concepts and Algorithms*. Cambridge University Press. (Cité en pages 29, 31 et 52.)

Zimek, A., Assent, I., and Vreeken, J. (2014). Frequent pattern mining algorithms for data clustering. In *Frequent Pattern Mining*, pages 403–423. (Cité en pages ix et 38.)

Zimek, A. and Vreeken, J. (2015). The blind men and the elephant : on meeting the problem of multiple truths in data from clustering and pattern mining perspectives. *Machine Learning*, 98(1-2) :121–155. (Cité en page 37.)