Un cadre générique de découverte de motifs sous contraintes fondées sur des primitives

Arnaud Soulet

Université de Caen / Basse-Normandie

Département d'Informatique
U.F.R. Sciences

École doctorale SIMEM

Un cadre générique de découverte de motifs sous contraintes fondées sur des primitives

THÈSE

présentée et soutenue publiquement le 13 novembre 2006

pour l'obtention du

Doctorat de l'Université de Caen

Spécialité Informatique

(Arrêté du 7 août 2006)

par

Arnaud Soulet

Composition du jury

Rapporteurs :	Amedeo Napoli	Directeur de recherche CNRS	LORIA de Nancy, UHP
	Osmar R. Zaïane	Professeur	Université d'Alberta (Canada)
Examinateurs :	Anne Doucet	Professeur	Université de Paris 6
	Dominique Laurent	Professeur	Université de Cergy-Pontoise
	Nicolas Spyratos	Professeur	Université de Paris-Sud 11
Directeur :	Bruno Crémilleux	Professeur	Université de Caen Basse-Normandie

Groupe de Recherche en Informatique, Image, Automatique et Instrumentation de Caen — CNRS UMR 6072

Mis en page avec la classe thloria.

Remerciements

En premier lieu, je suis profondément reconnaissant à Bruno Crémilleux d'avoir dirigé mes travaux de recherche durant les trois années de ma thèse. Sans son aide, le travail contenu dans ce mémoire n'existerait pas. Il s'est toujours rendu disponible pour me donner ses conseils et ses analyses qui ont enrichis mes réflexions. Au-delà des activités de recherche, il a su me guider dans le monde universitaire et me rendre autonome. Nos discussions m'ont donc beaucoup appris, mais j'en retiens aussi leur caractère détendu et chaleureux.

Amedeo Napoli et Osmar R. Zaïane m'ont fait l'immense honneur d'être les rapporteurs de ma thèse. Je les remercie d'avoir relu et rapporté ce manuscrit avec patience. Je suis également très honoré de la présence dans mon jury d'Anne Doucet, de Dominique Laurent et de Nicolas Spyratos.

Au cours de cette thèse, j'ai étroitement collaboré avec Jiří Kléma, Loïck Lhote et François Rioult. Dès le début de ma thèse, les discussions enthousiastes avec François ont été fructueuses. Loïck en se plongeant dans la fouille de données m'a aussi offert un éclairage complémentaire sur l'extraction de motifs. Enfin, Jiří en faisant l'effort de s'immerger dans mon cadre, a beaucoup apporté à cette thèse. Je n'oublie évidemment pas les travaux réalisés avec d'autres membres du GREYC dont Nicolas Durand, Céline Hébert et Nadine Lucas. Toutes ces collaborations, durant lesquelles j'ai pris beaucoup de plaisir, seront je l'espère, prolongées dans l'avenir.

Mon cadre de travail stimulant et amical a amplement contribué à cette thèse et j'en remercie tout spécialement les principaux acteurs : Céline, François, Loïck, Nicolas et Pierre. Je n'oublierai pas leur bonne humeur quotidienne. En particulier, Loïck m'a accueilli il y a trois ans pour ma plus grande joie entre les murs du bureau S3-352 dans lequel nous avons passé d'innombrables heures.

Finalement, je tiens à remercier mon entourage pour leur soutien. Je pense bien sûr à ma famille et surtout, à mes parents et à Stéphane qui sont toujours présents à mes côtés d'une façon ou d'une autre. Enfin, sans Arno, Denis, Did, Flav, Glenn, Loïck et Orel, ces années de thèse auraient été beaucoup plus ternes.

Table des matières

Table des figures

Liste des tableaux

Introduction

Contexte

L'Extraction de Connaissances dans les Bases de Données (ECBD) (aussi désignée par l'expression plus restrictive de "fouille de données") est une discipline récente qui recoupe les domaines des bases de données, des statistiques, de l'intelligence artificielle et de l'interface homme/machine. À partir d'une base de données, son objectif est de découvrir automatiquement des informations généralisables en connaissances nouvelles sous le contrôle des experts des données. Cela nécessite la conception et la mise au point de méthodes pour extraire les informations et les transformer en connaissance apportant une plus-value aux experts.

Un processus complet d'ECBD met en jeu, de manière interactive et itérative, de multiples méthodes pour la préparation des données, leur exploration, la visualisation et l'interprétation [Fayyad *et al.*, 1996]. Les méthodes de fouille de données proposent des solutions aux problèmes de recherche des règles d'association, de classification supervisée et non supervisée. Une étape centrale de ces processus est la découverte de motifs locaux telles que les régularités. Ces derniers capturent des informations spécifiques de la base de données et sont le fondement de la découverte de la connaissance utile aux experts. Compte tenu des tailles des bases de données (qui comprennent des milliers d'instances décrites par des milliers de descripteurs), il s'agit de problèmes algorithmiquement ardus nécessitant la conception de méthodes efficaces pour parcourir l'espace de recherche.

Le paradigme des motifs contraints, tout en palliant cette difficulté algorithmique, cherche à améliorer la qualité des motifs locaux extraits [Ng *et al.*, 1998]. Une contrainte permet de cibler la recherche de l'information à extraire suivant les centres d'intérêt de l'utilisateur. Dans ce mémoire, nous nous focalisons sur l'extraction de motifs locaux sous contraintes.

Motivations

L'essor applicatif des motifs locaux a encouragé la mise au point de méthodes pour leur obtention. Cependant, devant la complexité algorithmique, la plupart des méthodes se limitent aux seuls motifs fréquents [Agrawal et Srikant, 1994] ou à leur représentation [Pasquier *et al.*, 1999]. Cette quête de vitesse a occulté un objectif primordial, à savoir la découverte d'informations significatives et nouvelles portant sur d'autres propriétés. La masse des motifs fréquents, trop importante, ne peut être exploitée directement et noie les motifs les plus pertinents pour l'utilisateur parmi ceux trop généraux ou triviaux. D'autre part, l'usage des motifs fréquents est limité. Ils ne permettent pas, par exemple, de découvrir des exceptions ou des contrastes entre plusieurs classes.

La découverte de motifs sous contraintes a pour but de sélectionner les motifs locaux d'une base de données, qui satisfont un prédicat spécifié par l'utilisateur, appelé *contrainte*.

Celle-ci évalue l'intérêt des motifs pour ne sélectionner que ceux qui la satisfont. Depuis plusieurs années, la littérature foisonne de travaux ad hoc dédiés à une ou plusieurs contraintes spécifiques [Srikant *et al.*, 1997]. Plusieurs travaux ont aussi proposé de rassembler des contraintes partageant une même propriété formelle, en classes [Mannila et Toivonen, 1997, Ng *et al.*, 1998, Pei *et al.*, 2001a]. Il est alors possible d'utiliser un algorithme dédié à cette classe pour l'extraction des motifs.

Malheureusement, de nombreuses contraintes utiles en combinant plusieurs contraintes dont les propriétés formelles sont incompatibles entre elles, ne relèvent d'aucune classe. Des nouvelles méthodes doivent alors traiter ces combinaisons de classes [Bucila *et al.*, 2002], mais leurs possibilités restent limitées. Les architectures actuelles les plus développées reposent sur un dictionnaire de contraintes que l'utilisateur peut éventuellement combiner [Bonchi et Lucchese, 2005]. Clairement le dictionnaire de contraintes ne peut pas répondre à tous les besoins de l'utilisateur et limite à la fois ses possibilités et son imagination. L'interactivité du processus de fouille est donc réduit car l'utilisateur ne peut pas formuler au mieux ses attentes à travers la contrainte. Par ailleurs, l'usage de ces méthodes requiert aussi de solides connaissances techniques pour choisir les solveurs adéquats.

Contributions

Dans ce travail, nous optons pour une démarche différente en proposant un cadre fondé sur les primitives (*primitive-based framework*, PBF) : plutôt qu'un dictionnaire de contraintes, nous préférons mettre à la disposition de l'utilisateur un dictionnaire de primitives. Celles-ci sont des fonctions d'un fin niveau de granularité qui peuvent être combinées les unes avec les autres pour former une large panoplie de contraintes. L'utilisateur a ainsi la possibilité de construire des contraintes relevant des classes usuelles, mais surtout, il dispose de la faculté d'en proposer de nouvelles. L'originalité et l'expressivité de ces contraintes fondées sur les primitives (*primitive-based constraints*, PBC) découlent de la combinaison des sémantiques individuelles de chaque primitive.

Malgré la diversité des contraintes fondées sur les primitives, nous verrons que nous les manipulons uniformément et automatiquement grâce à des opérateurs formels. En particulier, les opérateurs de minoration et de majoration permettent d'approximer (en donnant des conditions suffisantes) le comportement d'une contrainte. Cet aspect se différencie radicalement des approches classiques de l'extraction de motifs contraints qui privilégient des conditions nécessaires et suffisantes pour une efficacité accrue au détriment de la généricité des contraintes traitées. De plus, nous verrons que les approximations que nous déduisons fonctionnent plutôt bien en pratique pour les différentes méthodes d'extraction de motifs proposées.

Quel que soit le langage des motifs (i.e., motifs ensemblistes ou séquentiels), nous étendons l'usage des solveurs usuels aux contraintes fondées sur les primitives grâce à des méthodes de relaxation. Cette approche approxime la contrainte considérée par d'autres qui possèdent de bonnes propriétés de monotonie. Les motifs satisfaisant ces dernières peuvent facilement être extraits, puis filtrés pour retrouver les motifs satisfaisant la contrainte originelle. Nous traitons aussi des contraintes globales comme la recherche des k motifs maximisant une mesure d'intérêt, qui présentent un vif intérêt pour l'utilisateur. Leur extraction se révèle problématique car leur évaluation nécessite de connaître par avance toutes les informations locales sur l'ensemble de la base de données. Nous proposons alors une approche de relaxation évolutive, appelée Approximer-et-Pousser, pour les extraire.

Pour le langage des motifs ensemblistes, nous donnons un algorithme d'extraction de motifs

satisfaisant une contrainte fondée sur des primitives, spécifique et bien adapté aux données corrélées. Ce dernier, nommé MUSIC-DFS, a l'originalité d'effectuer d'élagages efficaces sur des intervalles à la place des seules généralisations ou spécialisations. Il produit une représentation condensée d'intervalles consistante avec la contrainte considérée. Par ailleurs, aussi bien pour optimiser MUSIC-DFS que pour être utilisé de manière indépendante, nous définissons un nouvel opérateur de fermeture adéquat à certaines fonctions. Cet opérateur généralise la fermeture classique [Pasquier *et al.*, 1999]. La valeur d'un motif pour ces fonctions peut être inférée grâce aux représentations condensées issues de cette nouvelle fermeture.

Enfin, les différentes méthodes d'extraction proposées dans ce mémoire ont été utilisées dans divers problèmes applicatifs réels. Outre la validation de nos méthodes d'extraction avec plusieurs jeux de données, l'usage des contraintes basées sur des primitives et étendues à des jeux de données issues de ressources textuelles, s'est révélé particulièrement fécond dans un cas d'étude génomique.

Organisation du mémoire

La première partie de ce mémoire situe la problématique de l'extraction de motifs contraints et dresse un état de l'art des méthodes classiques en s'attachant à en montrer leurs atouts et leurs limites.

Le chapitre 1 illustre la portée des motifs locaux contraints en expliquant la difficulté de leur extraction. L'intérêt de ces motifs pour l'utilisateur est mis en lumière à travers la diversité des contraintes et des langages. La grande variété des motifs contraints renforce la possibilité de construire des motifs globaux et modèles. Enfin, nous positionnons brièvement l'extraction de motifs par rapport à des domaines connexes.

À partir du cadre unificateur de Mannila et Toivonen, le chapitre 2 explique la nécessité de classes de contraintes et définit formellement cette notion. Ce chapitre présente ensuite les classes les plus usuelles de la littérature en précisant leurs principales caractéristiques. Une synthèse compare ces différentes classes.

Le chapitre 3 s'intéresse à l'extraction des motifs relevant de plusieurs classes de contraintes. En particulier, nous décrivons le principe des architectures d'extraction à travers les bases de données inductives. Ces dernières reposent sur les représentations condensées de motifs comme les bordures, motifs fermés et motifs libres. Enfin, nous concluons sur la faiblesse de cette approche dont le maniement est encore peu aisé pour l'utilisateur.

La deuxième partie présente l'ensemble de nos contributions relatives à l'extraction de motifs contraints. Les chapitres 4 et 5 introduisent le cadre formel utilisé dans l'ensemble de cette partie. Les chapitres 6 et 7 proposent des méthodes d'extractions par relaxation pour tout langage, tandis que les chapitres 8 et 9 dédiés aux motifs ensemblistes décrivent des méthodes d'extraction directe.

Nous introduisons le cadre fondé sur les primitives au chapitre 4 en définissant les notions de primitives, primitives de haut niveau et les contraintes fondées sur les primitives. Ces dernières sont variées et expressives. Puis, nous montrons que ces contraintes englobent les classes de contraintes usuelles.

Les opérateurs de minoration et majoration sont introduits au chapitre 5. D'abord, nous présentons le principe général de ces opérateurs et en montrons la mise en œuvre pratique. Nous définissons alors ces deux opérateurs et illustrons leur utilisation pour la détection de contraintes monotones et anti-monotones.

Le chapitre 6 décrit notre méthode de relaxation pour l'extraction de motifs satisfaisant une contrainte basée sur des primitives. Pour cela, nous définissons les motifs virtuels le plus général et le plus spécifique. Nous montrons que, combinés avec l'opérateur de majoration, il est possible de construire des opérateurs de relaxation monotone et anti-monotone. Ces opérateurs sont évalués avec l'extraction de motifs ensemblistes et séquentiels contraints.

Le chapitre 7 se focalise sur l'extraction de contraintes globales. Après avoir introduit cette notion, nous décrivons les deux étapes majeures de l'approche Approximer-et-Pousser. Cette approche est ensuite appliquée à l'extraction des top-k motifs. Nous réalisons alors plusieurs expériences qui valident l'approche.

Le chapitre 8 présente MUSIC-DFS, un nouvel algorithme en profondeur qui extrait les contraintes du PBF pour les données ensemblistes. Il est particulièrement performant sur les large jeux de données et les données corrélées. Nous introduisons l'opérateur d'élagage et la nouvelle fermeture par préfixe sur lesquels repose l'algorithme. Après une description détaillée de l'algorithme MUSIC-DFS, une étude expérimentale confirme son efficacité.

Nous étendons les représentations condensées aux fonctions conservées dans le chapitre 9. Pour cela, nous généralisons l'opérateur de fermeture usuel pour le mettre en adéquation avec la fonction choisie. Nous décrivons alors l'algorithme MICMAC qui extrait ces représentations condensées adéquates. Ensuite, nous définissons la propriété de mesure forte et montrons qu'elle permet d'isoler des motifs de grande qualité au regard des mesures d'intérêts.

La dernière partie de ce travail est dédiée à nos résultats applicatifs obtenus avec les motifs contraints dans les domaines industriel, médical et biologique.

Le chapitre 10 traite de l'identification d'équipements défectueux au sein d'une chaîne de production de plaques de silicium. À cette fin, nous extrayons et analysons des motifs émergents forts.

Le chapitre 11 décrit différentes expérimentations effectuées sur des données relatives à la maladie de l'athérosclérose et de la fibrose du foie. Plus précisément, nous caractériserons les patients sains et ceux atteints par une pathologie liée à l'athérosclérose. Enfin, nous proposons une méthode de catégorisation contrainte pour caractériser les différents stades de la fibrose du foie.

Nous présentons au chapitre 12 une étude de données d'expression de gènes. Une description fine des données et du pré-traitement est proposée car les PBC exploitent les connaissances du domaine pour découvrir des gènes significatifs. Un motif jugé pertinent par les biologistes est isolé et interprété.

Le dernier chapitre conclut sur l'ensemble de notre travail. Nous rappelons alors les résultats obtenus en les discutant. Nous proposons aussi plusieurs prolongements.

Première partie

Découverte de motifs sous contraintes : état de l'art

Introduction

Cette partie présente l'enjeu et les principales approches de l'extraction de motifs sous contraintes. Ce domaine se distingue des statistiques et de l'apprentissage automatique. En effet, les méthodes de ces domaines produisent des modèles globaux, c'est-à-dire une synthèse structurée des données en vue d'un certain objectif. Par exemple, un arbre de décision résume un jeu de données sous forme arborescente par divisions successives en séparant au mieux les exemples suivant une valeur de classe [Quinlan, 1986]. Cependant, cette démarche conduit rarement à découvrir une information nouvelle et surprenante, ce qui est l'essence même d'un processus ECBD. Au contraire, celle-ci a tendance à faire ressortir les connaissances les plus générales du domaine.

La recherche de motifs locaux est une tâche centrale en ECBD. Ces motifs peuvent correspondre à des sous-parties des données, éventuellement de faible taille ou impliquant peu d'attributs mais qui ont un fort intérêt parce qu'ils traduisent un comportement qui s'écarte des connaissances générales sur les données. La recherche de motifs locaux est au cœur de l'extraction sous contraintes. Une contrainte permet à l'utilisateur de focaliser la recherche de l'information à extraire suivant ses centres d'intérêts. D'un point de vue algorithmique, lorsque la contrainte peut être poussée lors de la phase d'extraction des motifs, c'est aussi un moyen d'améliorer les performances d'extraction et dans certains cas de rendre le processus faisable. D'autre part, un champ actuel de recherche en plein essor est l'élaboration de motifs globaux et de modèles à partir de motifs locaux [Morik *et al.*, 2005]. C'est un moyen de revisiter les méthodes de construction de modèles descriptifs (groupes ou partitions) ou prédictifs (classifieurs).

Le chapitre 1 introduit la problématique de l'extraction de motifs locaux à partir d'exemples issus de besoins d'utilisateurs ainsi que les contraintes caractérisant les motifs. Nous nous attachons à montrer que les données peuvent être plus complexes que les classiques données ensemblistes [Agrawal et Srikant, 1994]. Nous montrons aussi qu'il est nécessaire de pouvoir définir une large variété de contraintes. Une brève conclusion positionne ces méthodes d'extraction par rapport aux domaines connexes.

Le chapitre 2 donne un panorama des différentes classes de contraintes existantes dans la littérature. Nous présentons celles-ci à partir du cadre unifiant de Mannila et Toivonen. Nous pensons que cette démarche permet de mieux les comparer. Par ailleurs, ce chapitre fait ressortir les points clés des algorithmes d'extractions en s'appuyant sur leur grande variété.

Le chapitre 3 présente le cadre des bases de données inductives mettant en relation les différentes classes de contraintes usuelles. Ce cadre fait interagir de multiples extractions de motifs pour permettre des combinaisons de contraintes et faciliter la répétition d'un processus de découverte. Nous verrons que la notion de représentation condensée devient alors centrale. Finalement, nous dégageons plusieurs limites de ces méthodes usuelles et nous en profitons pour ébaucher notre stratégie à venir.

Chapitre 1

Problématique de l'extraction de motifs sous contraintes

Sommaire

Ce chapitre introduit la problématique de l'extraction de motifs locaux sous contraintes à partir de quelques exemples. Ceux-ci montrent la nécessité d'offrir à l'utilisateur le moyen d'exprimer de façon simple une grande variété de contraintes. Les motifs peuvent être de natures diverses, et pas uniquement les traditionnels motifs ensemblistes [Agrawal et Srikant, 1994]. Puis, nous précisons l'intérêt et la portée des usages des motifs locaux. Enfin, nous situons l'extraction de motifs par rapport à plusieurs domaines connexes.

1.1 Extraction de motifs locaux sous contraintes

1.1.1 Motifs ensemblistes fréquents

Considérons une étude médicale portant sur la maladie de l'athérosclérose et dont le but est d'identifier des facteurs pathogènes[1]. Supposons qu'on dispose de données comme celles indiquées dans le tableau 1.1. Celles-ci résument la situation médicale de malades au début de l'étude. Chacun des patients est caractérisé par des descripteurs physiques ou comportementaux. Le contexte `malades` proposé par le tableau 1.1 représente 6 patients identifiés par $P_1, \ldots, P_6$ et décrits par les 6 descripteurs étiquetés de A à F. La première ligne signifie que les 4 descripteurs A, B, E et F sont présents pour le patient P_1. Par exemple, le descripteur A correspond à une forte consommation de tabac ; le descripteur B, à des antécédents familiaux ; le descripteur C, à une taille supérieure à 1m80 ; etc.

[1] Ces données ont été utilisées lors de plusieurs ECML/PKDD Discovery Challenges (cf. chapitre 11).

`malades`

Patient	Descripteurs					
P_1	A	B			E	F
P_2	A				E	
P_3	A	B	C	D		
P_4	A	B	C	D	E	
P_5				D	E	
P_6			C			F

TAB. 1.1 – Descripteurs médicaux caractérisant un groupe pathologique.

Les médecins sont intéressés par les combinaisons de descripteurs présents auprès de nombreux patients car ceux-ci sont de potentiels facteurs de risque. De telles régularités sont appelées *motifs fréquents*. Plus précisément, un motif est dit fréquent si son nombre de répétitions (ici, le nombre de patients qu'il caractérise) excède un seuil fixé. La fréquence de $\{A, B\}$, dénotée $freq(\{A, B\})$, est 3 car A et B apparaissent simultanément chez les 3 patients P_1, P_3 et P_4. De cette manière, si le seuil minimal retenu est 3, le motif $\{A, B\}$ (i.e., "une forte consommation de tabac accompagnée d'antécédents familiaux") sera extrait. Ce dernier n'est qu'un exemple de la collection des 7 motifs du contexte satisfaisant la *contrainte* $freq(X) \geq 3$, à savoir $\{A\}$, $\{B\}$, $\{C\}$, $\{D\}$, $\{E\}$, $\{A, B\}$ et $\{A, E\}$. Dans la suite, afin d'alléger les notations, les motifs ensemblistes seront notés sous forme de chaînes (e.g., AB désignera $\{A, B\}$).

Plus généralement, pour les motifs ensemblistes, les objets d'études formant la base de données sont appelés *transactions* et leurs descripteurs, *items*. Cette terminologie est issue de la tâche originelle de l'analyse du "panier du consommateur" [Agrawal *et al.*, 1993]. Dans notre exemple, les transactions modélisent donc les patients et les items, les descripteurs.

1.1.2 Contraintes d'émergence et d'aire minimale

La section précédente montre que les régularités d'une base de données découlent facilement des motifs fréquents. Mais la contrainte de fréquence minimale n'est pas adaptée à toutes les applications.

Considérons un groupe de patients sains en complément des données relatives aux patients malades. La recherche de facteurs de risque peut alors bénéficier de ce second groupe en le comparant au premier. On peut maintenant rechercher les motifs qui, en plus d'être fréquents parmi les patients malades, décrivent un minimum de patients sains (voire aucun). Cette discrimination peut, par exemple, se formuler en terme de contrainte par $freq(X, \texttt{malades}) \geq n \times freq(X, \texttt{sains})$ (où $freq(X, \texttt{malades})$ est la fréquence du motif X parmi les patients malades). Cette dernière sélectionne les motifs n fois plus présents dans `malades` que dans `sains`. Ils sont appelés *motifs émergents* [Dong et Li, 1999]. D'autres tâches exploitent ces motifs pour révéler des contrastes entre plusieurs parties de la base de données. Par exemple, la comparaison de lots défectueux aux lots normaux permet de pointer des équipements mal réglés (cf. le chapitre 10).

Par ailleurs, la fréquence n'est pas le seul critère significatif pour construire des motifs intéressant l'utilisateur. Supposons que celui-ci recherche des motifs recouvrant largement le contexte transactionnel (i.e., suffisamment fréquents et longs). Les motifs désirés peuvent être soit très fréquents et courts, soit peu fréquents mais longs. Un tel compromis entre la fréquence et la longueur du motif s'exprime par la *mesure d'aire* : $freq(X) \times count(X)$ (où $count(X)$ est la cardinalité de l'ensemble X). En particulier, nous verrons que la contrainte d'aire mi-

nimale est utile pour l'étude du transcriptome (cf. le chapitre 12) et est un cas de contrainte particulièrement difficile à mettre en oeuvre (cf. le chapitre 3).

1.1.3 Motifs séquentiels

Les motifs présentés dans les sections précédentes portent sur des données ensemblistes. Ce type de données (bien que le plus étudié) est insuffisant pour rendre compte de formes structurelles comme la séquentialité présentée maintenant.

Reprenons le problème introduit dans la section 1.1.1 et supposons maintenant que pour chaque patient, l'historique de ses examens est conservé. Chaque examen est un groupe de descripteurs (i.e., un ensemble). Le contexte `suivi` du tableau 1.2 présente les séquences d'examens de 5 patients identifiés par $P_1, \ldots, P_5$ et constituées des 6 descripteurs étiquetés de A à F. La seconde ligne indique que les descripteurs A et B sont présents pour le premier examen du patient P_2, puis le descripteur C est présent pour ce même patient lors d'un examen ultérieur, etc. Remarquons qu'un descripteur peut être répété pour un même patient (par exemple, A pour le patient P_2).

`suivi`

Patient	Séquence
P_1	$\langle(C)(A)\rangle$
P_2	$\langle(AB)(C)(ADF)\rangle$
P_3	$\langle(ACE)\rangle$
P_4	$\langle(C)(AD)(A)\rangle$
P_5	$\langle(B)(A)\rangle$

TAB. 1.2 – Exemple d'une base de données séquentielles.

Il est naturel d'introduire la séquentialité dans l'étude des facteurs de risque avec le contexte du tableau 1.2. Un motif séquentiel résume ici l'évolution des patients au cours du temps en pointant les ensembles de descripteurs vérifiés à des visites successives. Par exemple, le *motif séquentiel* $\langle(C)(AD)\rangle$ indique que le descripteur C a été observé lors d'un examen, puis que les descripteurs A et D ont été observés conjointement lors d'un examen ultérieur. La fréquence d'un motif séquentiel X est le nombre de patients contenant X (e.g., $freq(\langle(C)(AD)\rangle) = 2$ correspondant aux patients P_2 et P_4). De cette manière, la tâche de l'extraction des motifs ensemblistes fréquents est naturellement étendue aux motifs séquentiels.

1.1.4 Bilan

Au-delà de la sélection du type de motifs à extraire (e.g., régularités, exceptions, contrastes), nous verrons que la contrainte d'extraction cristallise les attentes et les connaissances de l'utilisateur. Elle constitue donc une dimension essentielle de l'extraction de motifs. Transversalement à la contrainte, le type de motifs potentiellement intéressant à extraire forme le *langage* (par exemple, dans la section 1.1.1, le langage est constitué de tous les sous-ensembles de $\{A, B, \ldots, F\}$). Nous verrons que la souplesse de définition du langage utilisé autorise des traitements raffinés de bases de données aux formes atypiques.

Bien que la formulation du problème de l'extraction de motifs contraints soit simple, sa résolution est difficile. L'extraction de motifs sous contraintes est une tâche complexe car l'espace de recherche des motifs est gigantesque. La taille de cet espace est évidemment liée à la

cardinalité du langage. Ainsi, l'espace de recherche augmente exponentiellement avec le nombre d'items pour la recherche des motifs ensemblistes. Un langage issu de 200 items, classique pour les problèmes réels, comporte 2^{200} motifs au pire à tester soit beaucoup plus que les environ 10^{80} atomes de l'univers. Dans ces conditions, il est vain de penser que les progrès techniques pallieront les faiblesses algorithmiques (d'autant que les quantités de données à analyser augmentent également). Par ailleurs, un des axiomes usuels de la fouille de données est de définir l'extraction des motifs comme une tâche correcte et complète par rapport à une contrainte, ce qui exclut le recours à une solution approchée. La justesse garantit que chacun des motifs extraits satisfait la contrainte d'extraction. L'utilisateur peut donc exploiter en toute confiance les résultats obtenus sans validation supplémentaire. De plus, la complétude assure que tous les motifs satisfaisant la contrainte de la base de données sont retournés. De cette manière, aucune information jugée pertinente pour l'utilisateur (i.e., satisfaisant sa contrainte) n'est omise.

1.2 Intérêts de l'extraction de motifs

L'extraction sous contrainte centre la recherche d'informations suivant les souhaits de l'utilisateur en portant sur des langages divers. Cette section montre la nécessité d'exprimer des contraintes variées. Elle indique aussi que les motifs locaux ont un grand rôle dans la construction de motifs globaux ou de modèles, plus aptes à faire ressortir la quintessence même des données.

1.2.1 Une richesse sémantique des contraintes

L'expression de la contrainte résume les attentes de l'utilisateur. Bien qu'il soit quasiment impossible de dresser la liste complète des contraintes utilisées dans la littérature [Agrawal et Srikant, 1994, Ng *et al.*, 1998, Kiefer *et al.*, 2003, Bonchi et Lucchese, 2005], on distingue plusieurs catégories majeures [Pei et Han, 2002] que nous indiquons maintenant :

Contraintes d'agrégat

Une *contrainte d'agrégat* évalue la qualité d'un motif au regard d'une mesure d'intérêt. Introduite dans [Agrawal et Srikant, 1994], la plus utilisée est certainement la contrainte de fréquence minimale que nous avons présentée à la section 1.1.1. La forme caractéristique des contraintes d'agrégat est $m(X)\theta seuil$ où m est une fonction d'agrégat et $\theta \in \{<, \leq, =, \geq, >\}$. Plusieurs travaux portant sur les motifs ensemblistes considèrent un motif comme un agrégat et les fonctions d'agrégat sont alors proches de celles utilisées en algèbre relationnelle [Codd, 1970]. Certaines d'entre elles sont directement issues de `SQL` et, de manière similaire, nécessitent des valeurs numériques associées aux items. Par exemple, lorsque les items désignent les articles d'un magasin, un prix peut être associé à chaque article pour rechercher les motifs dont la moyenne des prix des articles le composant soit inférieure à un seuil. Les contraintes basées sur *min*, *max*, *sum* et *avg* sont étudiées dans [Ng *et al.*, 1998] et celles basées sur la variance, dans [Kiefer *et al.*, 2003]. Les contraintes d'agrégat désignent l'archétype des motifs à extraire tels que les régularités, les exceptions ou les contrastes d'une base de données. Souvent le réglage du seuil modifie la sélectivité de la contrainte et influence la qualité des motifs associés. Par exemple, lorsque le seuil de fréquence minimale *minfr* croît avec la contrainte $freq(X) \geq minfr$, le nombre de motifs extraits diminue pour ne conserver que les plus représentés au sein de la base de données. Les contraintes d'émergence ou d'aire minimale (proposées

dans la section 1.1.2) sont également des contraintes d'agrégats. De même que la contrainte d'émergence résulte du taux de croissance (une mesure d'intérêt voisine de celle proposée par Sebag-Schoenauer [Sebag et Schoenauer, 1988]), de nombreuses autres contraintes d'agrégat (cf. la section 9.4) bénéficient de mesures d'intérêts définies pour évaluer des règles (comme la confiance [Agrawal *et al.*, 1993], le lift [International Business Machines, 1996], etc).

Contraintes syntaxiques

Une *contrainte syntaxique* réduit le langage de recherche en spécifiant la forme (ou patron) des motifs désirés. Typiquement, ces contraintes définissent l'appartenance ou non de certains items aux motifs [Srikant *et al.*, 1997], la longueur attendue des motifs, etc. Pour les motifs ensemblistes, les primitives classiques utilisées pour les décrire sont *count* (i.e., le nombre d'items composant le motif) et les opérateurs ensemblistes. Certaines de ces contraintes nécessitent d'enrichir la base de données avec des ensembles catégoriels voire des taxonomies [Srikant et Agrawal, 1995]. Les contraintes syntaxiques formalisent principalement la connaissance de l'expert sur les données [Perng *et al.*, 2002] pour que les motifs extraits soient compatibles avec ses connaissances. Si une étude a prouvé que le café n'a aucun impact sur la maladie de l'athérosclérose, le médecin pourra d'emblée exclure ce descripteur via la contrainte. Par ailleurs, intégrer la connaissance de l'expert focalise la fouille sur des informations inattendues (en excluant les motifs triviaux) et facilite ainsi la découverte de connaissances nouvelles [Wang *et al.*, 2003].

Les contraintes syntaxiques sont moins étudiées dans la littérature que les contraintes d'agrégats. En effet, contrairement à ces dernières, les contraintes syntaxiques dépendent rarement de la base de données. Leur extraction se limite souvent à une méthode ad hoc en amont ou en aval de l'extraction de motifs satisfaisant une contrainte d'agrégat [Boulicaut et Jeudy, 2000]. Par exemple, plutôt que d'exclure un descripteur grâce à la contrainte, une étape de prétraitement retire ce descripteur de la base de données.

Combinaisons de contraintes

Jusqu'à présent les contraintes proposées sont définies par un critère de sélection unique. Le terme *contrainte atomique* est alors privilégié pour les désigner. Dans la suite, le terme *contrainte* désigne une combinaison de contraintes atomiques. Les combinaisons sont importantes pour l'utilisateur car elles enrichissent encore l'expressivité des motifs extraits. Si une contrainte s'avère insuffisante pour exprimer la nature des motifs recherchés, l'utilisateur peut alors la compléter par un ou plusieurs autres critères afin d'affiner ses attentes. Une combinaison de contraintes atomiques permet ainsi d'associer leur sémantique respective. En particulier, une conjonction de contraintes extrait des motifs satisfaisant la sémantique individuelle de chaque contrainte. Par exemple, la contrainte $freq(X, \texttt{malades}) \geq minfr \wedge freq(X, \texttt{sains}) \leq maxfr$ sélectionne des motifs souvent présents dans le contexte `malades` et rarement dans le contexte `sains`. Cette alternative aux motifs émergents (moins complète, cf. la section 3.2.2) pointe également des motifs caractéristiques du groupe pathologique. En plus de cibler des informations intéressantes, cette conjonction de contraintes réduit le nombre de motifs extraits et ainsi, facilite leur analyse ultérieure. En effet, parmi les motifs fréquents de `malades`, l'ajout de $freq(X, \texttt{sains}) \leq maxfr$ élimine les motifs récurrents du contexte `sains`.

Le chapitre 4 montrera que notre travail considère à la fois les contraintes d'agrégats, les contraintes syntaxiques et leurs combinaisons.

1.2.2 Des motifs de natures diverses

Comme l'a illustrée la section 1.1.3, une base de données peut recueillir des informations sous des formes très variées (e.g., ensembles, séquences). Selon la nature de ces données, les motifs potentiellement intéressants rassemblés au sein du langage sont différents. Il est important de proposer à l'utilisateur des méthodes capables de traiter plusieurs types de langages car cette souplesse offre des méthodes d'analyses pour les données complexes et inhabituelles que peu d'approches peuvent traiter. Par exemple, la recherche de répétitions d'épisodes dans une séquence est utilisée pour la détection d'événements [Mannila *et al.*, 1995]. Plus récemment, l'extraction de fragments moléculaires a permis d'étudier des propriétés chimiques dans [Kramer *et al.*, 2001, Raedt et Kramer, 2001]. L'extraction d'arbres ou de graphes [Kuramochi et Karypis, 2001] est appliquée à la recherche d'informations dans des fichiers XML [Termier *et al.*, 2004] ou des traces d'utilisateurs web.

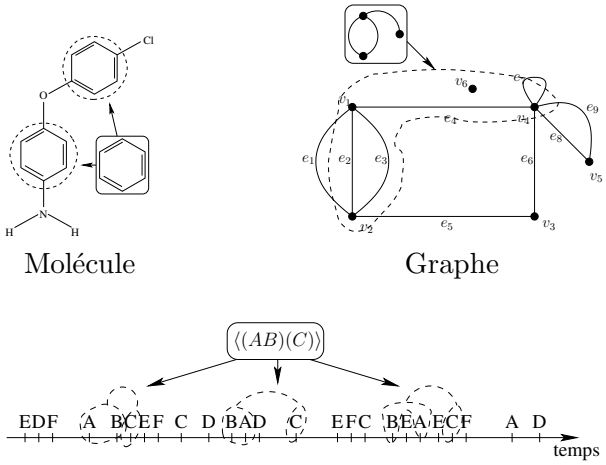


FIG. 1.1 – Exemple de données complexes.

La figure 1.1 fournit des exemples de données traitées dans la littérature. Pour chacune des données, le langage est différent et, en illustration, un motif particulier est encadré. Pour certains langages, la reconnaissance du motif au sein des données soulève des difficultés (e.g., l'identification de fragments moléculaires ou de sous-graphes engendre des problèmes d'isomorphisme).

1.2.3 Usages multiples des motifs

L'extraction de motifs permet de répondre à des usages très divers. Les motifs obtenus peuvent soit être interprétés de manière brute (motif local), soit être combinés les uns avec les autres (motif global) ou encore être exploités pour créer un modèle (prédictif ou descriptif). Ces trois niveaux de granularité forment les emboîtements de la poupée russe décrite par la figure 1.2.

Motifs locaux

Les motifs locaux ne traduisent pas des comportements de l'ensemble de la base de données mais plutôt des situations précises au sein des données. En particulier, les informations extraites auraient pu échapper à des analyses statistiques plus classiques (e.g., analyse multi-variées) qui

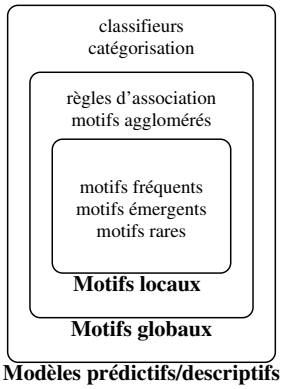


FIG. 1.2 – Différentes granularités dans les usages.

ont tendance à gommer les événements marginaux. Ils offrent donc des informations qualitatives et locales enrichies par la sémantique de la contrainte, qui se révèlent facilement analysables de manière indépendante. De plus, la contrainte est une façon efficace pour réduire le nombre de motifs produits (cf. le chapitre 12). Ils sont souvent complétés par une ou plusieurs mesures statistiques pour en faciliter l'analyse. Typiquement, à un motif fréquent, on associe la valeur de sa fréquence. En effet, un motif très au-dessus du seuil minimal de fréquence n'a pas les mêmes signification et impact qu'un motif le dépassant à peine.

Motifs globaux

Les motifs locaux extraits sont parfois combinés pour obtenir des motifs plus généraux. La dérivation de règles d'association [Agrawal *et al.*, 1993] est l'un des usages les plus courants des motifs fréquents. Une règle d'association basée sur un motif Z est une expression du type $X \rightarrow Y$ où $X \subset Z$ et $Y = X \backslash Z$. Plus généralement, la recherche de règles satisfaisant une mesure minimale (e.g., la confiance) ou des propriétés syntaxiques (e.g., taille, minimalité) découle de motifs contraints. D'autres travaux proposent de fusionner les motifs locaux pour en généraliser la portée. Typiquement les motifs les plus proches selon une distance sont groupés ensembles dans [Pensa et Boulicaut, 2005]. Tout en diminuant le nombre, l'objectif est alors d'effacer les perturbations issues de données bruitées. La construction de motifs globaux comme les bipartitions se poursuit dans le cadre du projet BINGO (ACI MD 46).

Les règles d'association sont parfois considérées comme des motifs locaux lorsque leur comportement dévie du comportement général [Bonchi et Giannotti, 2004]. La frontière entre les motifs locaux et globaux reste floue [Morik *et al.*, 2005]. De manière générale, les motifs globaux sont issus d'un post-traitement sur les motifs locaux.

Construction de modèles

Rappelons que la collection des motifs est consistante au regard de la contrainte qui en schématise l'intérêt. La construction de modèles issus de motifs locaux et globaux peut aussi tirer profit de la complétude des représentations (parfois condensées, cf. le chapitre 3) provenant de l'extraction de motifs. Pour obtenir une véritable connaissance sur le domaine étudié, la construction de modèles nécessite l'apport de méthodes d'apprentissage pour leur généralisation :

– **Classification :** les motifs fréquents (ou règles de classification associées) ont été exploités

par les classifieurs CBA [Liu *et al.*, 1998] et CMAR [Li *et al.*, 2001]. En fait, seuls les motifs significatifs sont conservés [Liu *et al.*, 2000] et cette pertinence s'exprime en terme de contraintes pour leur sélection. CMAR exploite par exemple une mesure statistique basée sur le χ^2. De même, le classifieur CAEP [Dong *et al.*, 1999] est construit à l'aide de motifs émergents. Plus récemment, le classifieur HARMONY sélectionne les règles de classification en utilisant l'instance à classer [Wang et Karypis, 2005]. La littérature foisonne d'autres travaux [Zhang *et al.*, 2000].

– **Catégorisation (ou *clustering*) :** De nombreuses méthodes de catégorisation sont basées sur des motifs fréquents telles que TrK-Means [Giannotti *et al.*, 2002], ARHP [Han *et al.*, 1997], Ping-Pong [Oyanagi *et al.*, 2001] ou Ecclat [Durand et Crémilleux, 2002]. Certaines méthodes utilisent directement la mesure de fréquence comme ARHP, d'autres, une mesure de similarité à partir de la fréquence [Ronkainen, 1998]. Ecclat sélectionne les clusters parmi les motifs fréquents fermés maximisant deux mesures (l'homogénéité et la concentration).

Bien évidemment, l'extraction de motifs n'est pas une étape obligée pour la classification ou la catégorisation. Mais la variété du langage des motifs permet d'obtenir de tels modèles dans des données atypiques où certaines méthodes statistiques sont moins efficaces (e.g., un classifieur pour des données XML [Zaki et Aggarwal, 2003]). Par ailleurs, la sémantique de la contrainte offre des possibilités originales comme la construction de partitions de clusters contraints (cf. chapitre 11).

1.3 Positionnement de l'extraction de motifs

Notre travail s'inscrit dans la lignée de l'extraction de motifs originellement proposée par Agrawal et al. [Agrawal *et al.*, 1993, Agrawal et Srikant, 1994] et qui a été largement illustrée dans ce premier chapitre. L'extraction de motifs est un champ de recherche pour lequel il existe des connexions avec de nombreux autres domaines. L'extraction de motifs sous contraintes peut être vue comme une classe particulière de problèmes de satisfaction de contraintes[2]. Les motifs recherchés satisfont une contrainte unaire dont le domaine est le langage. Les liens sont aussi forts avec la communauté des treillis de Galois et de l'analyse de concepts formels. En effet, les motifs fermés (cf. la section 3.3.1) sont les objets d'étude de cette communauté [Ganter, 1984] (mais la notion de contrainte est absente). Des travaux proches [Dehaspe, 1998] sont aussi menés en programmation logique inductive[3] avec des volumes de données plus restreints organisés sous forme de prédicats logiques. La reconnaissance de motifs étudie la présence de motifs pour des sources variées. De façon générale, la fouille de données se distingue de ces travaux en autorisant un langage de motif souple et en le reliant à de très larges volumes de données par le biais de la contrainte.

Enfin, notons que l'extraction de motifs (en particulier, ensemblistes) diffère de l'algèbre relationnelle par la forme des motifs recherchés qui ne correspondent pas à une liste d'attributs prédéfinie. Seule la contrainte est fixée et le processus détermine les combinaisons d'attributs qui la satisfont. Pour cette raison, plusieurs opérateurs ont été proposés afin d'étendre le langage `SQL` à l'extraction de motifs dont `CUBE BY` [Gray *et al.*, 1996, Gray *et al.*, 1997]. Ces derniers travaux ont développé une terminologie propre reprise dans le tableau 1.3. Celui-ci indique dans sa première colonne le vocabulaire relatif au "panier du consommateur", puis de l'analyse de concepts formels et enfin, de l'algèbre relationnelle. Dans la suite, nous ne discute-

[2]En anglais : *Constraint Satisfaction Problems* abrégé *CSP*
[3]En anglais : *Inductive Logive Programming* abrégé *ILP*

rons pas de l'intégration de l'extraction de motifs au sein des systèmes de gestion de bases de données [Meo *et al.*, 1996] bien que cela constitue une suite naturelle à notre travail. De même, nous ne discuterons pas de la problématique des traitements en ligne incontournables pour les flux de données (*data stream*) [Gehrke et Hellerstein, 2004].

Contexte transactionnel	Analyse de Concepts Formels	Algèbre relationnelle
contexte transactionnel	contexte formel	relation
item	attribut	dimension
transaction	objet	nuplet
contrainte	-	requête
motif ensembliste	motif d'attributs	group-by

TAB. 1.3 – Equivalences terminologiques.

Chapitre 2

Les classes de contraintes

Sommaire

Ce chapitre présente les difficultés algorithmiques de l'extraction sous contraintes et les classes de contraintes qui en découlent. Pour cela, il s'appuie sur le cadre de Mannila et Toivonen [Mannila et Toivonen, 1997] pour présenter de façon unifiée les différents types d'extraction quelque soit la contrainte ou le langage. La section 2.1 montre l'importance de la structuration du langage pour l'extraction de motifs et explique comment la structuration est exploitée algorithmiquement. À partir de ces observations, une étude typologique des classes usuelles de contraintes (i.e., monotones, succinctes et convertibles) est menée dans la section 2.2. Enfin, la section 2.3 compare brièvement ces différentes classes et synthétise leurs limites.

2.1 Cadre de Mannila et Toivonen

2.1.1 Théorie d'une base de données

Comme indiqué au chapitre précédent, les extractions de motifs sont effectuées dans des contextes très divers dépendant à la fois du langage (e.g., ensembles, séquences) et de la contrainte (e.g., contrainte de fréquence minimale, contrainte d'aire minimale). Bien évidemment, elles sont aussi liées à la base de données. Nous formalisons à présent ces trois notions essentielles que sont le langage, la base de données et la contrainte :

Définition 1 (Langage) *Un langage $\mathcal{L}$ est un ensemble de motifs.*

Rappelons qu'un motif traduit une propriété ou un extrait de la base de données (comme c'est le cas pour les motifs ensemblistes). Il décrit un comportement ou rend compte d'un phénomène.

Le langage des motifs ensemblistes $\mathcal{L}_\mathcal{I}$ correspond exactement à toutes les sous-ensembles non vides[4] de $\mathcal{I}$ i.e., $\mathcal{L}_\mathcal{I} = 2^\mathcal{I} \backslash \{\emptyset\}$. Le langage $\mathcal{L}$ peut être infini dans certains cas, comme pour les séquences. En effet, pour un ensemble d'items spécifiés $\mathcal{I}$, le langage des séquences $\mathcal{L}_\mathcal{S}$ regroupe tous les multi-ensembles possibles de $\mathcal{L}_\mathcal{I}$. La section 2.1.2 complétera le langage avec une structure en la munissant d'une relation de spécialisation.

Définition 2 (Base de données) *Une base de données* $\mathbf{r}$ *regroupe l'ensemble des données à disposition de l'extraction.*

Aucune forme particulière n'est imposée à la base de données et cette dernière peut ne pas avoir de lien avec le langage $\mathcal{L}$. Par exemple, pour le langage $\mathcal{L}_\mathcal{S}$, le tableau 1.2 décrit la base de données comme un multi-ensemble de $\mathcal{L}_\mathcal{S}$. Dans [Mannila *et al.*, 1995], ce multi-ensemble est remplacé par une séquence unique (cf. la figure 1.1, page 14). En pratique, la base de données contient souvent un contexte transactionnel qui est un multi-ensemble de $\mathcal{L}$ (cf. les tableaux 1.1 et 1.2). Ce contexte est alors complété suivant les contraintes utilisées par des tables de valeurs, des taxonomies, etc (cf. la figure 2.2).

Nous définissons maintenant la notion de contrainte :

Définition 3 (Contrainte) *Une contrainte* q *est un prédicat booléen défini sur un langage.*

Une contrainte évalue si un motif φ[5] est intéressant ou non. Elle est aussi appelée prédicat ou requête. Le plus souvent la contrainte dépend de la base de données $\mathbf{r}$ (e.g., la contrainte de fréquence minimale) même si elle n'y fait pas référence explicitement. Abusivement, on écrit $q(\varphi)$ à la place de $q(\mathbf{r}, \varphi)$. Cette notation met en exergue le lien fort que la contrainte établit entre le langage et la base de données. La définition 3 n'exige aucune propriété particulière sur la contrainte (la section 2.1.3 introduit la notion de propriété sur les contraintes dont découlent les classes).

L'extraction de motifs d'une base de données $\mathbf{r}$ est la sélection des motifs d'un langage $\mathcal{L}$ intéressant au regard d'une contrainte q. Plus formellement, il s'agit de déterminer la théorie correspondante :

Définition 4 (Théorie) *Pour un langage donné* $\mathcal{L}$*, une base de données* $\mathbf{r}$ *et une contrainte* q*, la théorie* $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ *est l'ensemble des motifs de* $\mathcal{L}$ *satisfaisant la contrainte* q *dans* $\mathbf{r}$*.*

Le cadre de Mannila et Toivonen [Mannila et Toivonen, 1997] ne spécifie pas que le motif doit appartenir à la base de données. Dans notre contexte d'extraction de motifs (cf. les chapitres 6 et 8), nous imposerons cette condition supplémentaire. Par exemple, les motifs fréquents d'un langage $\mathcal{L}$ correspondent exactement à la théorie $\mathcal{T}h(\mathcal{L}, \mathbf{r}, freq(\varphi) \geq minfr)$. En particulier, la théorie $\mathcal{T}h(\mathcal{L}_\mathcal{I}, \mathbf{r}, freq(X) \geq 3)$ donne $\{A, B, C, D, E, AB, AE\}$ avec le contexte `malades` (cf. le tableau 1.1).

Le cadre de Mannila et Toivonen est aussi utilisé pour décrire d'autres problèmes algorithmiques comme la découverte de toutes les dépendances fonctionnelles [Mannila et Toivonen, 1997] ou de la recherche de toutes les traverses minimales [Hébert *et al.*, 2007].

[4] Nous excluons l'ensemble vide car il est rarement porteur de sens.

[5] Pour un langage quelconque $\mathcal{L}$, les lettres grecques φ, γ ou θ désignent un motif. Pour le langage des ensembles $\mathcal{L}_\mathcal{I}$ ou des séquences $\mathcal{L}_\mathcal{S}$, les lettres du début de l'alphabet désignent les items et les lettres de la fin, les motifs.

2.1.2 Structuration du langage

Nous introduisons maintenant une relation de spécialisation/généralisation, comme proposée par Mitchell dans [Mitchell, 1982]. Une telle relation structure le langage $\mathcal{L}$ et est utile pour localiser les motifs potentiels à extraire et parcourir le moins possible de motifs du langage. L'ordre lexicographique d'un dictionnaire exploite ce même principe. Il sert à trouver plus rapidement un mot recherché en évitant d'avoir à passer en revue tous les mots du dictionnaire.

Relation de spécialisation

Une *relation de spécialisation* $\preceq$ est un ordre partiel défini sur les motifs de $\mathcal{L}$. φ est dit *plus général* (resp. *plus spécifique*) que γ, si et seulement si on a $\varphi \preceq \gamma$ (resp. $\gamma \preceq \varphi$). Quand $\varphi \preceq \gamma$ et $\varphi \neq \gamma$, φ est strictement plus général que γ et on note $\varphi \prec \gamma$. Pour un motif φ et une relation de spécialisation $\preceq$, un motif γ tel que $\gamma \preceq \varphi$ est une *généralisation* de φ. À l'inverse, un motif γ tel que $\varphi \preceq \gamma$ est une *spécialisation* de φ. Par exemple, les langages des motifs ensemblistes et séquentiels sont tous deux munis d'une relation de spécialisation. Pour les ensembles d'items, l'inclusion $\subseteq$ constitue une relation de spécialisation. Par exemple, comme $A \subseteq AB$, A est plus général que AB et AB est une des spécialisations de A. Similairement, pour les séquences, $X = \langle x_1 x_2 \dots x_n \rangle$ est plus général que $Y = \langle y_1 y_2 \dots y_m \rangle$ (dénoté par $X \preceq_{\mathcal{S}} Y$) si il existe des entiers $i_1 < i_2 < \dots < i_n$ tels que $x_1 \subseteq y_{i_1}$, $x_2 \subseteq y_{i_2}$,..., $x_n \subseteq y_{i_n}$.

Une relation de spécialisation peut toujours être définie sur un langage. Ainsi, en imposer une ne constitue pas une limitation à ce cadre. En revanche, certaines approches nécessitent des structures sur le langage plus complexes et moins génériques (comme un langage algébrique [Bucila *et al.*, 2002]).

Impact sur l'espace de recherche

L'espace de recherche dépend intimement du langage des motifs à extraire et son organisation découle de la relation de spécialisation du langage. La figure 2.1 illustre des espaces de recherche associés aux langages des motifs ensemblistes et séquentiels.

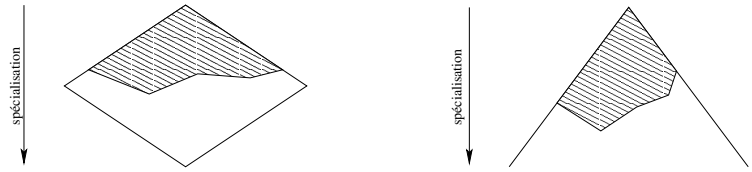


FIG. 2.1 – Espaces de recherche des motifs ensemblistes et séquentiels.

Sur cette figure, les motifs les plus généraux (resp. spécifiques) sont situés en haut (resp. en bas). L'espace de recherche des motifs ensemblistes constitue un treillis dont la forme en losange traduit la répartition des motifs en fonction de la spécialisation. La répartition des motifs séquentiels constitue un triangle ouvert car le langage est infini. Les formes hachurées schématisent les motifs présents dans la base de données. Contrairement à $\mathcal{L}_{\mathcal{S}}$, l'espace de recherche des séquences *présentes* dans une base de données est fini.

L'objectif des algorithmes d'extraction de motifs est de localiser au mieux les motifs désirés à travers le vaste espace de recherche afin d'en parcourir le minimum. La figure 2.1 montre que les motifs présents dans une base de données sont les plus généraux du langage $\mathcal{L}$ et les classes de contraintes (cf. la section 2.2) utilisent ce fait.

2.1.3 Classe de contraintes

La méthode naïve de l'énumération de tous les motifs puis de la vérification de la contrainte sur chacun d'entre eux, n'est pas envisageable en pratique. Il est alors nécessaire de tirer profit de certaines propriétés de la contrainte pour améliorer l'approche. On parle de *pousser* la contrainte au cœur de l'extraction[6]. Dans la pratique, l'extraction de motifs se limite donc aux contraintes que l'on peut pousser. Au lieu de pousser une contrainte particulière, les solveurs sont souvent dédiés à un ensemble de contraintes. Un tel ensemble de contraintes est appelé *classe de contraintes*. De façon assez surprenante, la notion de classe n'est pas définie dans la littérature. Nous en proposons la définition suivante :

Définition 5 (Classe de contrainte) *Une classe de contraintes est un ensemble infini de contraintes atomiques liées par une sémantique.*

La sémantique (parfois, une simple propriété formelle) justifie le regroupement des différentes contraintes en une classe. Notons qu'implicitement, on considère que deux contraintes équivalentes aux seuils près ne constituent qu'une seule contrainte. Ainsi, la contrainte de fréquence minimale ne donne pas lieu à une classe de contraintes car si son unique paramètre (i.e., le seuil minimal de fréquence) est fixé, on obtient une seule contrainte atomique. Un ensemble fini de contraintes atomiques même combinées mutuellement avec les opérateurs booléens ne constitue pas davantage une classe de contraintes car il ne peut être enrichi avec de nouvelles contraintes atomiques.

Les contraintes d'agrégats ou les contraintes syntaxiques décrites dans le chapitre précédent forment en revanche des classes de contraintes. Ces deux classes distinctes sont intéressantes pour l'utilisateur. Malheureusement, celles-ci sont mal prises en compte par les solveurs. En effet, devant la difficulté de la tâche d'extraction de motifs, les classes usuelles de contraintes sont définies de façon pragmatique selon les propriétés d'élagage (i.e., de réduction) de l'espace de recherche et en reléguant en arrière plan les besoins réels de l'utilisateur.

Plus formellement, une *condition d'élagage* (ou un *critère d'élagage*) est un prédicat booléen dont la vérification pour un motif assure qu'une partie de l'espace de recherche ne contient pas de motifs satisfaisant la contrainte. Il n'est donc pas nécessaire de parcourir cet espace. On parle alors d'élagage *négatif*. De manière duale, l'élagage *positif* élude une partie de l'espace de recherche où tous les motifs satisfont la contrainte. Le plus souvent, les motifs élagués correspondent aux généralisations ou aux spécialisations du motif satisfaisant la condition d'élagage. Les conditions d'élagage traduisent en fait le comportement de la contrainte par rapport à la relation de spécialisation. Nous verrons dans le chapitre 5 que l'étude des variations de la contrainte (i.e., sa croissance ou sa décroissance suivant la relation de spécialisation) modélise son évolution.

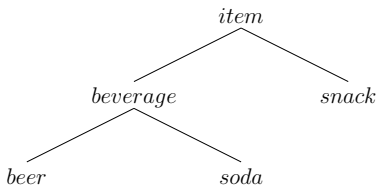
2.2 Typologie orientée élagage

Cette section présente les classes de contraintes présentes dans la littérature. Le tableau 2.1 donne quelques exemples de contraintes appartenant à ces différentes classes. Ces contraintes s'appuient sur la base de données de la figure 2.2. Ce tableau montre que les contraintes d'agrégats et syntaxiques se répartissent dans les diverses classes. Cependant, la généricité de la définition par rapport au langage, la tolérance sur les contraintes et la possibilité ou non de les combiner, donnent des points de comparaisons entre les classes.

[6]Des méthodes de pré-traitements bénéficient aussi de la contrainte pour améliorer l'extraction ultérieure comme ExAnte [Bonchi *et al.*, 2003].

Contrainte	$\mathcal{Q}_{AM}$	$\mathcal{Q}_{M}$	$\mathcal{Q}_{S}$	$\mathcal{Q}_{CAM}$	$\mathcal{Q}_{CM}$	$\mathcal{Q}_{LAM}$
$q_1 \equiv min(X.Qty) \geq 500$	×		×	×	×	×
$q_2 \equiv max(X.Price) \geq 30$		×	×	×	×	×
$q_3 \equiv X.Type \supseteq \{snack, soda\}$		×	×		×	
$q_4 \equiv q_1 \wedge q_2$			×			×
$q_5 \equiv q_2 \vee q_3$		×	×		×	×
$q_6 \equiv q_1 \vee (\neg q_2 \wedge q_3)$			×			
$q_7 \equiv X.Price = 25$	×		×	×		×
$q_8 \equiv X.Type \subseteq \{beer, snack\}$	×		×	×		×
$q_9 \equiv soda \in X.Type$		×	×		×	×
$q_{10} \equiv max(X.Price)/avg(X.Price) \leq 7$				×	×	×

TAB. 2.1 – Exemples de contraintes (provenant de [Leung *et al.*, 2002]) définies sur $\mathcal{L}_{\mathcal{I}}$ nécessitant une base de données comme celle proposée par la figure 2.2.



Trans.	Contents
t_1	$A\ B\ C\ D$
t_2	$B\ D\ F$
t_3	$A\ B\ D\ E$
t_4	$A\ B\ C\ E\ G$
t_5	$C\ E\ G$

Ìtem	A	B	C	D	E	F	G
Qty	600	200	300	500	700	400	100
$Price$	40	10	25	30	20	35	15
$Type$	*beer*	*snack*	*soda*	*beer*	*beer*	*beer*	*beer*

FIG. 2.2 – Exemple d'une base de données associée aux contraintes du tableau 2.1 contenant un contexte transactionnel, une table de valeurs et une taxonomie.

2.2.1 Les contraintes monotones et anti-monotones

La littérature abonde de travaux concernant les contraintes anti-monotones par rapport à la spécialisation. La plus populaire est la contrainte de fréquence minimale qui se révèle essentielle dans de nombreuses applications. Mais le succès de cette classe est essentiellement due à la simplicité et à l'efficacité de sa condition d'élagage.

Caractéristiques

Suite à l'introduction de la contrainte de fréquence minimale [Agrawal et Srikant, 1994], la classe des contraintes monotones et anti-monotones par rapport à la spécialisation a été formellement définie dans [Mannila et Toivonen, 1997] :

Définition 6 (Contrainte monotone ou anti-monotone) *Une contrainte q est monotone (resp. anti-monotone) suivant la relation de spécialisation $\preceq$ si et seulement si pour tout motif satisfaisant q, ses spécialisations (resp. généralisations) satisfont également la contrainte q.*

Une contrainte monotone (ou anti-monotone) par rapport à la spécialisation[7] est en fait une simple fonction croissante (ou décroissante) par rapport à la spécialisation. Par la suite, $\mathcal{Q}_M$ et $\mathcal{Q}_{AM}$ désignent respectivement l'ensemble des contraintes monotones et anti-monotones. Ces deux ensembles recouvrent une partie des contraintes les plus classiques (cf. le tableau 2.1) dont bien sûr la contrainte de fréquence.

Les contraintes monotones/anti-monotones sont bien adaptées à l'extraction de motifs. Tout d'abord, elles peuvent être aisément combinées par conjonction ou disjonction. La classe des contraintes anti-monotones (ou monotones) est stable pour ces opérations. En revanche, la négation d'une contrainte monotone (resp. anti-monotone) est une contrainte anti-monotone (resp. monotone). La conjonction d'une contrainte monotone et d'une contrainte anti-monotone n'est ni monotone, ni anti-monotone (cf. la section 3.1.2).

La négation d'une contrainte monotone ou anti-monotone donne directement sa condition d'élagage :

Condition d'élagage 1 *Si un motif φ ne satisfait pas la contrainte monotone (resp. anti-monotone) q, alors toutes les généralisations (resp. spécialisations) de φ ne satisfont pas la contrainte q.*

Cette condition d'élagage bénéficie de la variation de la contrainte pour garantir que soit toutes les généralisations, soit toutes les spécialisations ne vérifieront plus la contrainte. Ainsi, pour une contrainte anti-monotone, tous les motifs satisfaisant la contrainte forment un espace convexe contenant les motifs les plus généraux. Ce phénomène s'observe sur la figure 2.1 où l'espace de recherche correspond à la contrainte anti-monotone $freq(X) \geq 1$. La condition d'élagage d'une contrainte anti-monotone se traduit en pratique par la vérification des deux critères suivants :

1. Si un motif ne vérifie pas une contrainte anti-monotone, aucune de ses spécialisations ne la vérifie.
2. Si une généralisation d'un motif ne vérifie pas une contrainte anti-monotone, ce motif et ses spécialisations ne vérifient pas la contrainte anti-monotone.

Malheureusement, ces deux classes ne permettent pas de traiter de nombreuses contraintes utiles dont celles basées sur la moyenne, la variance, l'aire, etc.

[7] Cette précision de la relation de spécialisation (i.e., "par rapport à la spécialisation") est souvent omise dans la suite de ce mémoire lorsqu'il n'y a pas d'ambiguïté.

Algorithmes

Les algorithmes extrayant les motifs satisfaisant une contrainte anti-monotone sont légion et abordent des langages variés. Nous les présentons brièvement ci-dessous. De nombreux algorithmes s'intéressent plus particulièrement aux motifs fréquents en exploitant l'anti-monotonie de la contrainte de fréquence minimale [Goethals, 2003b]. La plupart d'entre eux sont facilement adaptables à toute contrainte anti-monotone. En revanche, les algorithmes dédiés aux contraintes monotones sont plus rares (e.g., [de Knijf et Feelders, 2005] extrait les arbres fréquents satisfaisant une contrainte monotone). Certains proposent de ne pas les pousser (i.e., ne pas les exploiter) [Boulicaut et Jeudy, 2000]. En plus des algorithmes présentés ci-dessous, d'autres algorithmes peuvent traiter simultanément une contrainte monotone et une contrainte anti-monotone ou exploitent des bordures (cf. le chapitre 3).

L'algorithme le plus connu, Apriori, se focalise sur la contrainte de fréquence pour les ensembles d'items [Agrawal et Srikant, 1994]. Il a été rapidement adapté aux séquences fréquentes [Agrawal et Srikant, 1995] et aux épisodes fréquents [Mannila *et al.*, 1995]. Sa généralisation, l'algorithme par niveaux, traite toute contrainte anti-monotone (ou monotone) pour un langage quelconque [Mannila et Toivonen, 1997]. Son principe est de générer une partie des candidats avant de les tester simultanément pour parcourir le moins de fois possible la base de données. On parle de méthode *generate-and-test*. Tout d'abord, on vérifie la contrainte sur les motifs du premier niveau (i.e., les plus généraux). Seuls ceux vérifiant la contrainte sont conservés. Ensuite, le second niveau considère toutes leurs spécialisations immédiates dont chaque généralisation satisfait la contrainte (étape de génération des candidats, vérification du critère 2, page 24). On teste alors ces candidats pour exclure ceux ne satisfaisant pas la contrainte (étape de test des candidats, vérification du critère 1). On réitère le processus jusqu'à épuisement des candidats. Ainsi, niveau par niveau, l'intégralité de l'espace de recherche est parcouru. L'efficacité d'Apriori repose souvent sur des structures de données particulières (hash-tree [Agrawal *et al.*, 1996] ou trie [Borgelt et Kruse, 2002]). De plus, de nombreuses optimisations ont été proposées dont AprioriTid, AprioriHybrid ou DHP (Direct Hashing and Pruning) détaillés dans [Goethals, 2003b].

Il existe aussi des algorithmes ayant une approche en profondeur dont l'un des premiers fut Buc [Beyer et Ramakrishnan, 1999]. L'algorithme Eclat [Zaki, 2000b] dédié à la recherche de motifs ensemblistes fréquents parcours l'espace de recherche en profondeur. L'originalité de son approche est de calculer la fréquence d'un motif en faisant l'intersection des ensembles des transactions contenant ses spécialisations. En revanche, une telle méthode ne permet pas de bénéficier pleinement des capacités de la condition d'élagage (le second critère énoncé ci-avant n'est plus utilisé). Cette approche a ensuite été généralisée aux arbres [Zaki, 2002] puis à tout langage [Zaki *et al.*, 2005]. Les algorithmes du type *pattern-growth* exploitent quant à eux une structure de données particulière appelée FP-tree (Frequent-Pattern tree) [Han *et al.*, 2000]. L'idée est de construire un arbre résumant la base de données et de le parcourir en profondeur afin de générer tous les motifs fréquents (ou satisfaisant une autre contrainte anti-monotone). PrefixSpan applique le même principe pour la recherche de séquences fréquentes [Pei *et al.*, 2001b].

Des approches dites hybrides ou adaptatives utilisent un parcours en profondeur sur les premiers niveaux, puis basculent en un parcours en profondeur. La difficulté majeure est de définir une heuristique pour déterminer à partir de quel niveau le changement de parcours doit opérer. Hybrid [Hipp *et al.*, 2000] commence l'extraction en utilisant Apriori avant de basculer sur Eclat. Ce changement intervient à un niveau spécifié par l'utilisateur.

2.2.2 Les contraintes succinctes

Caractéristiques

Introduite dans [Ng *et al.*, 1998], la classe des contraintes succinctes, restreinte aux motifs ensemblistes, est la première à avoir traité des contraintes sans propriété de monotonie.

La définition originale nécessite deux autres définitions intermédiaires dans [Ng *et al.*, 1998]. Nous l'avons ici reformulée en une seule :

Définition 7 (Contrainte succincte) *Une contrainte q est dite succincte ssi il existe $I_1 \subseteq \mathcal{I}, \ldots, I_n \subseteq \mathcal{I}$ tels que la théorie de q soit exprimable en terme d'unions et de différences des langages $\mathcal{L}_{I_1}, \ldots, \mathcal{L}_{I_n}$*[8].

Cette définition est assez peu intuitive et nous pensons qu'elle est discutable. En effet, la théorie associée à une contrainte q peut toujours se décomposer en opérations ensemblistes sur des sous-langages (car la théorie est finie). En fait, la définition même si elle ne l'impose pas nécessite une décomposition explicite de la théorie. Dans la suite, l'ensemble des contraintes succinctes est dénoté par $\mathcal{Q}_S$. De nombreuses contraintes utiles sont succinctes (cf. tableau 2.1). De par sa définition basée sur des sous-langages, les contraintes succinctes englobent majoritairement des contraintes syntaxiques telle que q_3 (et de rares contraintes d'agrégats comme q_1 ou q_2). En revanche, la contrainte minimale de fréquence n'est pas une contrainte succincte (comme toutes les contraintes basées sur des mesures de fréquences telle que le taux de croissance). Néanmoins, les algorithmes dédiés aux contraintes succinctes acceptent la contrainte de fréquence minimale comme paramètre additionnel à la contrainte succincte.

Contrairement aux autres classes, la classe des contraintes succinctes est close pour n'importe quelle combinaison booléenne de contraintes succinctes. Si un solveur implémente toutes les contraintes succinctes, ce solveur peut donc rechercher des motifs satisfaisant des formules booléennes complexes de contraintes succinctes.

Les contraintes succinctes bénéficient souvent de l'anti-monotonie de certaines contraintes succinctes et de la contrainte de fréquence minimale. Néanmoins, l'ajout d'une nouvelle condition d'élagage a permis d'extraire des motifs satisfaisant des contraintes originales. Cette condition d'élagage se base sur une fonction de génération de membres associée à la contrainte succincte. Un candidat n'est généré que si une de ses spécialisations peut satisfaire la contrainte.

Certaines contraintes ne sont que "faiblement" succinctes (comme par exemple $count(X) \leq \gamma$) car son explicitation nécessite une fonction de génération de membres particulière.

Algorithmes

Peu d'algorithmes exploitent cette classe de contrainte et de fait, le concept n'a pas été étendu à d'autres langages comme pour les monotones ou les convertibles présentées dans la section suivante. Les stratégies pour tirer profit des contraintes succinctes sont multiples. Si la contrainte est également anti-monotone, on applique la même condition d'élagage que dans la section précédente. Sinon, le second critère d'élagage n'est pas appliqué. En revanche, chaque contrainte succincte possède une fonction de génération de membres distincte qui permet d'éliminer des motifs dont aucune spécialisation ne satisfait la contrainte. À nouveau, cette classe exploite donc un élagage négatif suivant la spécialisation des motifs.

[8] Rappelons que $\mathcal{I}$ est l'ensemble des items et que pour chaque ensemble $I_i \subseteq \mathcal{I}$, le langage $\mathcal{L}_{I_i}$ correspond à 2^{I_i}.

L'algorithme originel CAP (Constrained APriori) présenté dans [Ng *et al.*, 1998] est un algorithme par niveaux où seul diffère la génération des nouveaux candidats par rapport à APRIORI. [Grahne *et al.*, 2000] propose 4 algorithmes traitant des contraintes monotones et succinctes en ajoutant une contrainte supplémentaire de corrélation. L'algorithme FPS (FP-tree and Succinct) utilise les FP-trees pour extraire des contraintes succinctes [Leung *et al.*, 2002]. De nouveau, deux stratégies sont nécessaires pour pousser les contraintes "purement" succinctes et les contraintes succinctes anti-monotones. Plus récemment, l'algorithme DCF (pour Dynamic Constrained Frequent-set) offre une méthode pour pouvoir dynamiquement changer la contrainte d'extraction (comme le seuil de fréquence minimale) [Lakshmanan *et al.*, 2003].

2.2.3 Les contraintes convertibles

Introduite dans [Pei et Han, 2000], la convertibilité transforme une contrainte sans bonne propriété de monotonie en une contrainte monotone ou anti-monotone. Pour cela, elle nécessite une notion de préfixe et se limite donc à certains langages comme les ensembles ou les séquences.

Caractéristiques

La convertibilité se décline de deux façons tout comme la monotonie :

Définition 8 (contrainte convertible (anti-)monotone [Pei *et al.*, 2001a]) *Une contrainte est convertible anti-monotone (resp. monotone) ssi il existe un ordre R sur les items $\mathcal{I}$ tel que la contrainte soit anti-monotone (resp. monotone) sur les préfixes.*

Avec l'ordre lexicographique (i.e., $A < B < C < \ldots$), les préfixes de ABD sont A et AB. En particulier, AD n'est pas un préfixe de ABD. Parfois, la notion de convertibilité est définie de manière symétrique sur les suffixes [Pei et Han, 2000]. L'ensemble des contraintes convertibles monotones (resp. anti-monotones) est noté $\mathcal{Q}_{CM}$ (resp. $\mathcal{Q}_{CAM}$). Une contrainte à la fois convertible anti-monotone et convertible monotone est dite *fortement* convertible. Le tableau 2.1 donne plusieurs exemples de contraintes convertibles.

En fait, la notion de convertibilité adapte la structure du langage des ensembles à la contrainte considérée. Cette relation de spécialisation $\preceq_R$ se base sur l'ordre R de convertibilité et on a $X \preceq_R Y$ ssi X est un préfixe de Y (en ordonnant les ensembles X et Y avec l'ordre R). Ainsi, la convertibilité s'avère être une forme particulière de monotonie. La figure 2.3 montre que le remplacement de la relation de spécialisation $\subseteq$ par $\preceq_R$ rend convexe l'espace de recherche (parties hachurées) et ainsi, les variations de la contrainte deviennent prévisibles. La relation de spécialisation $\preceq_R$ étant plus lâche que la relation de spécialisation $\subseteq$, les contraintes convertibles monotones (resp. anti-monotones) forment un sur-ensemble des contraintes monotones (resp. anti-monotones) pour les motifs ensemblistes ou séquentiels (même si dans [Pei et Han, 2000], les auteurs prennent étrangement le soin d'exclure les contraintes anti-monotones et monotones des contraintes convertibles). En revanche, la classe des contraintes convertibles n'est pas un sur ensemble de la classe des contraintes succinctes (comme le montre la contrainte q_3 du tableau 2.1).

Contrairement aux contraintes monotones ou succinctes, les contraintes convertibles n'ont aucune bonne propriété au niveau des combinaisons booléennes. En particulier, la conjonction de deux contraintes convertibles n'est pas toujours convertible. Cela s'explique par l'incompatibilité entre les relations de spécialisations issues des deux relations d'ordre de contraintes convertibles atomiques [Leung *et al.*, 2002].

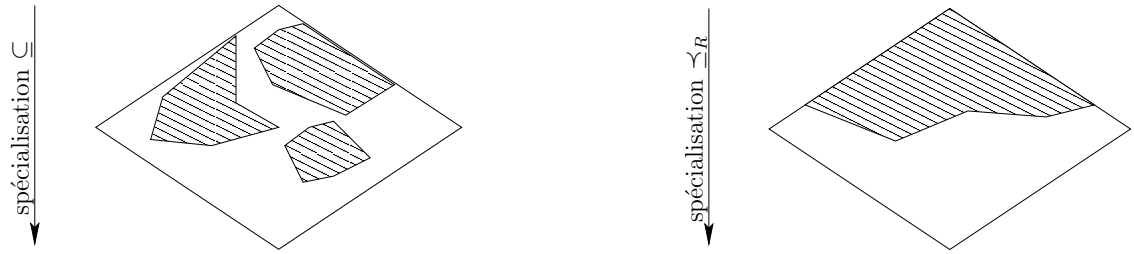


FIG. 2.3 – Impact du changement de relation de spécialisation sur l'organisation de l'espace de recherche.

La définition de la convertibilité s'étend aisément aux séquences [Pei *et al.*, 2002]. Les auteurs parlent alors de monotonie préfixée. La même stratégie a également été utilisée dans le contexte particulier des *data cubes* [Han *et al.*, 2001] (où la structure H-Cubing remplace celle de FP-tree).

Algorithmes

Les algorithmes qui extraient les motifs satisfaisant une contrainte convertible, utilisent des conditions d'élagage similaires à celles utilisées pour les contraintes monotones. La condition d'élagage 1 (cf. page 24) s'adapte naturellement pour donner la condition d'élagage 2 :

Condition d'élagage 2 *Si un motif X ne satisfait pas la contrainte convertible monotone (resp. anti-monotone) q, alors toutes les généralisations (resp. spécialisations) de X (selon $\preceq_R$) ne satisfont pas la contrainte q.*

L'algorithme fondateur fut la méthode CFG (Constrained Frequent pattern Growth) [Pei et Han, 2000] qui extrait des motifs ensemblistes satisfaisant une contrainte convertible en s'inspirant de [Han *et al.*, 2000]. Par ailleurs, les algorithmes $\mathcal{FIC}$ (Frequent Itemsets with Convertible) [Pei *et al.*, 2001a, Pei *et al.*, 2004] se déclinent en $\mathcal{FIC}^{\mathcal{M}}$ de $\mathcal{FIC}^{\mathcal{A}}$ qui permettent d'extraire respectivement avec des contraintes convertibles monotones et convertibles anti-monotones. Notons l'originalité de $\mathcal{FIC}^{\mathcal{M}}$ qui utilise un élagage positif suivant la spécialisation. Pour les séquences, PREFIX-GROWTH [Pei *et al.*, 2002] étend l'algorithme PREFIXSPAN [Pei *et al.*, 2001b].

Tous les algorithmes proposés dans la littérature pour extraire des motifs sous une contrainte convertible sont basés sur une approche en profondeur du type "pattern-growth view" [Pei et Han, 2002]. Bien que l'algorithme général par niveaux [Mannila et Toivonen, 1997] soit applicable, son intérêt est moindre car le treillis avec la relation de spécialisation $\preceq_R$ forme exactement un arbre. De cette manière, on perd le bénéfice d'une approche par niveaux qui vérifie l'existence de toutes les généralisations d'un motif avant de le générer.

2.2.4 Les autres classes

Cette section mentionne deux classes de contraintes plus récentes et peu étudiées.

Les contraintes séparables

Dans [Wang *et al.*, 2005], les auteurs proposent une approche pour un sous-ensemble des contraintes d'agrégats. Une contrainte $m(X)\theta seuil$ (avec $\theta \in \{<, \leq, \geq, >\}$) est *séparable* ssi la

fonction m s'écrit comme une somme ou un produit des fonctions monotones m_i. L'algorithme proposé pour extraire les motifs vérifiant de telles contraintes, énumère les motifs ensemblistes et s'arrête lorsque toute spécialisation a une mesure trop petite ou trop grande pour pouvoir satisfaire la contrainte. Plutôt que d'utiliser un ordre comme pour les contraintes convertibles, une approximation est faite pour évaluer le comportement de m (grâce aux fonctions m_i). Malheureusement, cette approche ne permet pas de traiter les contraintes syntaxiques ou les combinaisons booléennes de contraintes atomiques.

Les contraintes anti-monotones relâchées

L'*anti-monotonie relâchée* (traduction de "loose anti-monotone") a été introduite dans [Bonchi et Lucchese, 2005]. Dans l'esprit, il s'agit seulement d'assouplir le "pour tout sous-ensemble" (des contraintes anti-monotones) à "il existe un sous-ensemble" :

Définition 9 (Contrainte anti-monotone relâchée) *Une contrainte q est anti-monotone relâchée si pour tout motif de cardinalité supérieure à 2 satisfaisant q, l'un de ses sous-ensembles immédiats satisfait aussi q.*

Alors que pour une contrainte anti-monotone, toutes les spécialisations d'un motif doivent satisfaire la contrainte pour éventuellement la satisfaire également, une seule spécialisation suffit pour une contrainte anti-monotone relâchée. L'ensemble des contraintes anti-monotones relâchées est dénoté par $\mathcal{Q}_{LAM}$. Ainsi, les contraintes d'agrégats basées sur la variance qui ne sont ni succinctes, ni convertibles, sont des exemples de contraintes anti-monotones relâchées. La définition relativement souple de cette classe en fait un sur-ensemble des contraintes convertibles anti-monotones (et donc des contraintes anti-monotones). En revanche, cette classe n'est pas un sur-ensemble des succinctes ou des monotones. De plus, la négation d'une contrainte anti-monotone relâchée n'est plus une contrainte anti-monotone relâchée. À ce jour une seule méthode extrait sous contrainte anti-monotone relâchée. *ExAMiner*$^{\mathcal{LAM}}$ [Bonchi et Lucchese, 2005] fournit les motifs satisfaisant une contrainte anti-monotone relâchée par une méthode de réduction de données inspirée de ExAnte.

2.3 Synthèse sur les classes de contraintes

Dressons une rapide synthèse des classes de contraintes que nous avons présentées dans ce chapitre. Du point de vue du langage utilisé, seules les classes des contraintes monotones et anti-monotones peuvent être définies pour tout langage et disposent d'algorithmes génériques. En contrepartie, elles couvrent moins de contraintes que d'autres classes dont de nombreuses sont pourtant très utiles (cf. le tableau 2.1). Afin d'augmenter la portée de ces classes, des travaux cherchent à combiner celles-ci, ce point fait l'objet du prochain chapitre.

Les classes de contraintes les plus larges (i.e., convertibles et anti-monotones relâchées) sont quant à elles restreintes aux motifs ensemblistes. Mais même pour ce langage, elles n'admettent pas de nombreuses contraintes dont la contrainte d'émergence ou d'aire minimale. La figure 2.4 compare les différentes classes de contraintes pour le langage des motifs ensemblistes.

Dans la pratique, les techniques d'élagage utilisées par les solveurs sont, à de rares exceptions, basées sur l'élagage négatif suivant la spécialisation (ou la généralisation pour les contraintes monotones). En effet, les seules parties de l'espace de recherche éliminées sont celles où aucun motif ne satisfait la contrainte. Dans le chapitre suivant, certaines méthodes proposent aussi de ne pas parcourir des espaces où tous les motifs satisfont la contrainte (élagage positif). Enfin,

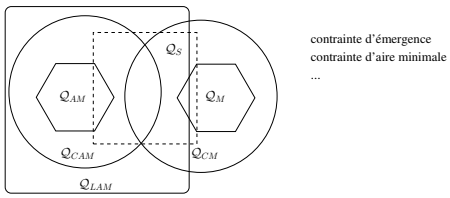


FIG. 2.4 – Comparaison des classes usuelles pour les motifs ensemblistes.

plutôt que d'éliminer soit toutes les généralisations, soit toutes les spécialisations, des méthodes proposent de faire les deux simultanément ou de se restreindre à des sous-algèbres.

D'autre part, l'utilisation pratique des classes que nous avons introduites pose de nombreuses difficultés. En effet, les caractéristiques qui les définissent, correspondent à des propriétés d'élagage et non pas aux besoins et à la sémantique souhaités par les utilisateurs (excepté les contraintes séparables). D'un point vue théorique ou pratique, elles limitent la possibilités de définir des contraintes atomiques nouvelles utiles pour des applications variées. À contre-pied, dans la seconde partie de ce mémoire, nous souhaitons proposer des contraintes définies en terme de besoins et non de faisabilité.

Chapitre 3

Bases de données inductives : méthodes d'extraction sous plusieurs contraintes

Sommaire

Dans le chapitre précédent, nous avons vu que chaque méthode d'extraction se focalise sur une classe particulière de contraintes. Les bases de données inductives [Imielinski et Mannila, 1996] étendent ces méthodes pour traiter des contraintes plus complexes. Une base de données inductive se caractérise par le fait de contenir, outre les données originelles, des modèles sur ces données. Dans notre contexte, il s'agit de résultats d'extractions de motifs contraints. Le stockage et l'interrogation de ces modèles facilitent les extractions multiples.

La section 3.1 présente les deux étapes fondamentales des bases de données inductives en soulignant l'intérêt des représentations condensées dans ces processus. Parmi ces représentations, la section 3.2 présente les bordures qui résument succinctement les motifs satisfaisant une formule booléenne de contraintes monotones. La section 3.3 s'intéresse plus particulièrement aux représentations des motifs fréquents. Enfin, la dernière section dégage plusieurs limites de ces représentations condensées et montre plus généralement qu'en l'état, elles ne constituent pas une réponse satisfaisante à la problématique de l'extraction de motifs.

3.1 Processus d'extraction

3.1.1 Description

Les processus inspirés des bases de données inductives se décomposent en une phase d'extraction et une phase d'inférence. Leur double avantage est de pouvoir traiter des contraintes plus complexes en recoupant plusieurs extractions et d'améliorer l'itérativité du processus en réexploitant les extractions précédentes.

Plusieurs architectures d'extractions se basent sur ce principe. Dans [Ng *et al.*, 1998], une première étape extrait des motifs fréquents satisfaisant une contrainte anti-monotone ou succincte. Une seconde étape permet alors de sélectionner des motifs ou construire des règles en leur associant des métriques diverses. Dans [Bayardo, 2005], le même type de schéma fait ressortir une étape d'extraction de motifs et une étape de sélection de motifs. La figure 3.1 décrit avec plus de précisions ces deux architectures (en haut [Ng *et al.*, 1998] et en bas [Bayardo, 2005]).

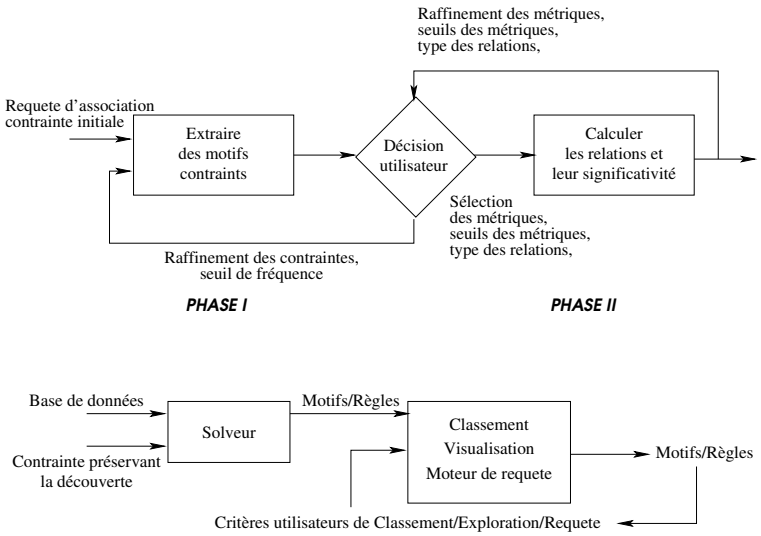


FIG. 3.1 – Architectures d'extraction (en haut [Ng *et al.*, 1998] et en bas [Bayardo, 2005]).

Ces architectures distinguent l'étape complexe d'extraction de motifs (phase 1) de celle de sélection de motifs ou de construction de règles (phase 2). L'objectif est de permettre à l'utilisateur d'améliorer itérativement la production de motifs/règles par raffinements successifs sans avoir à recommencer l'extraction. Se pose alors le problème de l'extraction originelle : quelle contrainte d'extraction choisir ? Dans [Bayardo, 2005], l'auteur propose que la contrainte "préserve la découverte", c'est-à-dire n'élimine pas de motifs potentiellement intéressants lors de la seconde phase. Ce choix se révèle particulièrement difficile car le résultat d'un processus d'ECBD est par nature large et il est difficile de sélectionner a priori un sur-ensemble des motifs recherchés. Ainsi, la première architecture autorise la correction de la contrainte initiale si nécessaire.

Pour faciliter les extractions dans ce type d'architectures, les bases de données inductives reposent sur des *représentations condensées* et leurs manipulations [Mannila, 1997]. Plutôt que d'extraire tous les motifs satisfaisant une contrainte, il est souvent judicieux de n'en extraire qu'une représentation formalisée comme suit :

Définition 10 (Représentation condensée adéquate) *Une représentation condensée adéquate à une fonction $f : \mathcal{L} \rightarrow E$ est une collection de motifs $\mathcal{R}$ tel que pour tout motif φ, la valeur $f(\varphi)$ puisse être déduite de un ou plusieurs motifs de $\mathcal{R}$.*

Par exemple, la section 3.3, à travers les motifs fermés et les motifs libres, présente des représentations condensées adéquates à la fréquence. Ces représentations ont l'avantage d'être à la fois plus concises et plus aisées à extraire. La concision améliore l'intelligibilité des résultats produits en facilitant l'analyse de l'expert ou les manipulations ultérieures. De cette manière, les représentations condensées sont souvent utilisées comme étape préliminaire d'obtention d'autres motifs locaux [Jeudy, 2002], de motifs globaux [Morik *et al.*, 2005] ou de modèles [Li *et al.*, 2001]. Les sections 3.2 et 3.3 dégagent les principales représentations condensées de la littérature.

D'autres représentations résument la base de données plutôt que les motifs extraits, par compression avec une méthode MDL (Minimum Description Length) [Siebes *et al.*, 2006] ou échantillonnage en sélectionnant un sous-ensemble de la base de données [Mielikäinen, 2004]. Ces représentations ne sont pas pertinentes pour l'extraction de motifs contraints car elles ne garantissent ni la complétude, ni la consistance.

3.1.2 Extractions de motifs sous contraintes relevant de plusieurs classes

Ces méthodes, à l'instar des bases de données inductives, ont pour objectif d'extraire des motifs satisfaisant des contraintes relevant de plusieurs classes usuelles.

Conjonction d'une contrainte monotone et d'une contrainte anti-monotone

L'extraction de motifs satisfaisant à la fois une contrainte monotone q_M et une contrainte anti-monotone q_{AM} se révèle nécessaire dans de nombreux problèmes. En effet, la contrainte q_M élimine les motifs peu intéressants car trop généraux au regard d'un certain critère tandis que la contrainte q_{AM} rejette les motifs trop spécifiques. L'ensemble des motifs satisfaisant cette conjonction de contraintes forme un espace convexe. Dans le domaine de l'apprentissage, un tel espace est appelé *espace des versions* [Mitchell, 1982]. La figure 3.2 montre l'espace des versions résultant de l'intersection des théories d'une contrainte monotone et d'une contrainte anti-monotone.

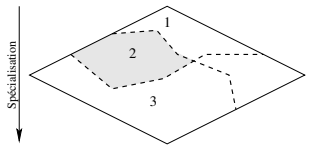


FIG. 3.2 – Espace des versions (zone 2) associé à une contrainte monotone (zones 2 et 3) et une contrainte anti-monotone (zones 1 et 2).

Plusieurs algorithmes extraient tous les motifs satisfaisant $q \equiv q_M \wedge q_{AM}$. Leur principe est d'exploiter simultanément l'élagage négatif suivant la généralisation issue de q_M et l'élagage négatif suivant la spécialisation issue de q_{AM}. Pour les motifs ensemblistes, une approche originale par pré-traitement ExAnte [Bonchi *et al.*, 2003] réduit un contexte transactionnel (et donc l'espace de recherche) sans éliminer de motifs satisfaisant q. Il élimine alternativement

les items ne satisfaisant pas q_{AM}, puis les transactions ne satisfaisant pas q_M. Malheureusement, une telle approche est inefficace lorsque l'interaction entre les deux contraintes se limite à une seule itération. Par exemple, si $q_{AM}(X) \equiv count(X) \leq 10$ et $q_M(X) \equiv count(X) \geq 5$, la seule réduction effectuée est le retrait des transactions du contexte transactionnel de longueur inférieure à 4 (car aucun des motifs de cette transaction ne peut satisfaire q_M). Le gain de cette approche est résiduel car la contrainte q_{AM} n'est pas exploitée. Une méthode similaire de réduction de données a été adaptée à la réduction de FP-tree dans [Bonchi et Goethals, 2004]. Dans [El-Hajj *et al.*, 2005], les auteurs reprennent partiellement ce principe et présentent un nouvel algorithme basé sur les COFI-tree.

Les témoins

Le cadre des *témoins* est une approche unificatrice des extractions de motifs [Kiefer *et al.*, 2003]. Il propose d'extraire les motifs satisfaisant plusieurs contraintes atomiques en combinant leurs élagages suivant la spécialisation. D'autre part, ces différents élagages peuvent être soit négatif, soit positif (ces types d'élagages ont été présentés dans la section 2.1.3). Un témoin est un représentant d'un espace de recherche qui peut être élagué. Il constitue alors un point d'arrêt lors du parcours de l'espace de recherche. En particulier, les auteurs décrivent l'obtention des témoins pour les contraintes basées sur la variance. Cependant, étant donnée une contrainte quelconque, aucune méthode générale de calcul des témoins n'est proposée. Par ailleurs, comme pour la convertibilité, il semble difficile de concilier tous les élagages entre eux (car les relations de spécialisation sont parfois incompatibles).

Le chapitre 6 propose, en déduisant des conditions d'élagage suivant la spécialisation, une méthode possible pour obtenir des témoins à partir d'une contrainte.

3.2 Représentations condensées de la monotonie

Cette section se concentre sur les représentations condensées de motifs satisfaisant une combinaison de contraintes monotones et anti-monotones.

3.2.1 Notion de bordure

Introduite dans le domaine de l'apprentissage [Mitchell, 1982], la notion de bordure est reprise dans [Mannila et Toivonen, 1997]. Les bordures permettent de représenter sans perte d'information la théorie d'une contrainte monotone ou anti-monotone. Leur principe est de séparer le langage en deux parties : d'un côté tous les motifs satisfont la contrainte, de l'autre, aucun motif ne satisfait la contrainte. L'obtention de telles frontières se base sur un principe de borne. Comme les contraintes monotones et anti-monotones sont croissantes ou décroissantes, le dernier motif satisfaisant la contrainte ou le premier ne la satisfaisant pas constitue une borne entre ses généralisations et ses spécialisations.

Bordure maximale

Nous définissons maintenant la notion de bordure maximale (tout comme celle de bordure minimale, cf. la définition 12) indépendamment de la notion de contrainte. Nous les utiliserons dans le chapitre 6 pour résumer les motifs présents dans une base de données et ainsi, relaxer des contraintes.

La bordure des motifs maximaux regroupe les spécialisations les plus fortes :

Définition 11 (Bordure des motifs maximaux) *La bordure des motifs maximaux de E (selon $\preceq$) est l'ensemble des motifs $S(E) = \{\varphi \in E |$ il n'existe pas $\gamma \in E$ tel que $\varphi \prec \gamma\}$.*

La définition 11 est majoritairement utilisée avec un espace E convexe. La bordure maximale retient les motifs les plus spécifiques selon $\preceq$. En particulier, la bordure des motifs maximaux de la théorie d'une contrainte anti-monotone décrit l'ensemble des motifs satisfaisant la contrainte. On parle de *bordure positive*. Toutes les généralisations (et aucune spécialisation) des motifs inclus dans cette bordure satisfont la contrainte anti-monotone. La bordure maximale de $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q_{AM})$ est donc une représentation condensée adéquate de la contrainte anti-monotone q_{AM} (cf. la définition 10, page 32). Plus précisément, la représentation condensée $S(\mathcal{T}h(\mathcal{L}, \mathbf{r}, q_{AM}))$ est adéquate à q.

Par exemple, avec le contexte `malades` (cf. page 10), la bordure des motifs maximaux de $\mathcal{T}h(\mathcal{L}_{\mathcal{I}}, \mathbf{r}, freq(X) \geq 3) = \{A, B, C, D, E, AB, AE\}$ est restreinte à seulement 3 motifs : $\{D, AB, AE\}$. On vérifie bien que chaque motif de la théorie est inclus dans au moins un motif de cette bordure positive de $freq(X) \geq 3$.

Bordure minimale

La bordure des motifs minimaux se définit de manière analogue :

Définition 12 (Bordure des motifs minimaux) *La bordure des motifs minimaux de E (selon $\preceq$) est l'ensemble des motifs $G(E) = \{\varphi \in E |$ il n'existe pas $\gamma \in E$ tel que $\gamma \prec \varphi\}$.*

La bordure des motifs minimaux conserve les motifs les plus généraux (selon $\preceq$) de E. La bordure des motifs minimaux de la théorie d'une contrainte monotone représente tous les motifs satisfaisant cette contrainte. Seuls les motifs plus spécifiques que l'un des motifs de cette bordure satisfont la contrainte monotone. La figure 3.3 donne les deux bordures associées à un ensemble E.

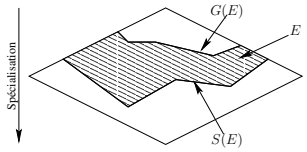


FIG. 3.3 – Illustration des bordures.

Remarquons que la bordure des motifs minimaux de $\mathcal{L} \backslash \mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ où q est anti-monotone forme la *bordure négative* de q. Pour chaque motif X de cette bordure, aucune des spécialisations de X ne satisfait la contrainte q.

Extraction de bordures

Les bordures sont des représentations souvent très concises. Naturellement, des algorithmes cherchent à les calculer pour synthétiser la théorie d'une contrainte monotone et anti-monotone. Pour n'importe quel langage, l'algorithme par niveaux [Mannila et Toivonen, 1997] peut facilement être adapté pour ne conserver que la bordure (des motifs maximaux ou minimaux). Néanmoins, d'autres méthodes bénéficient des bordures pour ne pas parcourir l'intégralité de la théorie. Pour les motifs ensemblistes, les motifs les plus spécifiques peuvent être extraits avec

MaxMiner [Bayardo, 1998], Mafia [Burdick *et al.*, 2001] ou GenMax [Gouda et Zaki, 2005]. Grâce à une méthode de dualisation, [Gunopulos *et al.*, 1997] parcourt en profondeur l'espace de recherche jusqu'à arriver sur un motif ne satisfaisant pas la contrainte anti-monotone. À ce stade, en tirant parti de la bordure des motifs déjà extraits (via les traverses minimales), son parcours évite de passer par des motifs dont on est sûr qu'ils satisfont la contrainte anti-monotone. ABS (Adaptative Border Search) [Flouvat *et al.*, 2004] exploite une approche par niveaux au début, puis bascule sur une méthode de dualisation similaire.

Bien qu'exacte, une représentation sous la forme d'une bordure ne permet pas de retrouver toutes les informations pourtant nécessaires à certains usages. Par exemple, la bordure des motifs fréquents $\{D, AB, AE\}$ (même en retenant la fréquence de chaque motif) est insuffisante pour déduire la fréquence d'un motif donné (e.g., $freq(AB) = 3$ et $freq(AE) = 3$ ne donnent aucune indication pour A de fréquence 4). Ainsi, si on souhaite augmenter le seuil minimal de fréquence, il faut soit recalculer une nouvelle bordure, soit corriger l'ancienne avec un algorithme type Guess-&-Correct [Mannila et Toivonen, 1997]. Dans le cas spécifique de la fréquence, d'autres représentations pallient ces manques comme nous le verrons dans la section 3.3.

3.2.2 Combinaisons de bordures

Dans la section 3.1.2, les espaces des versions sont issus d'une contrainte monotone et d'une contrainte anti-monotone. Les espaces peuvent donc se représenter par deux bordures, chacune correspondant à une contrainte (cf. la figure 3.3). Bien évidemment, les algorithmes présentés ci-dessus peuvent être employés pour les extraire individuellement. On recherche alors les deux bordures positives correspondant à chacune des contraintes. Dans [Fischer, 2003], une méthode, adaptée à tout langage, les obtient simultanément en se basant sur un algorithme probabiliste. Dédié aux motifs ensemblistes, DualMiner [Bucila *et al.*, 2002] utilise le principe de dualité entre les ensembles pour restreindre l'espace de recherche. Par exemple, pour un motif X, si son motif dual $Y = \mathcal{I} \backslash X$ ne satisfait pas q_M, toutes les généralisations de Y peuvent être éliminées de l'espace de recherche. Plutôt que de bordures, cet algorithme fournit en fait des sous-algèbres, i.e., des intervalles.

De manière plus générale, l'extraction de motifs satisfaisant une contrainte complexe peut s'effectuer en manipulant des bordures. Les étapes clés sont (1) la décomposition de la contrainte en une formule booléenne de contraintes monotones, (2) les extractions séparées et (3) le recoupement de ces extractions. En partant directement d'une requête construite avec des contraintes atomiques monotones et anti-monotones, la première étape est souvent éludée. L'étape 2 se base sur une réécriture de la requête pour effectuer le moins possible d'extractions [Lee et Raedt, 2003]. Par ailleurs, ces dernières portent souvent sur l'extraction de bordures. Enfin, l'étape 3 s'appuie sur des structures particulières comme les VST (Version Space Tree) [Lee et Raedt, 2004]. Une étude poussée dans [Giacometti *et al.*, 2002] montre comment retrouver certaines métriques avec de telles manipulations en tenant compte des propriétés de monotonie des métriques.

Illustrons la manipulation de bordures pour identifier les motifs émergents présentés à la section 1.1.2. La contrainte d'émergence se décompose en $\bigvee_i (freq(X, \texttt{malades}) \geq i \times n \wedge freq(X, \texttt{sains}) \leq i)$ [Dong et Li, 1999]. Plutôt que de tenir compte de chaque i, une valeur particulière est privilégiée pour simplifier la contrainte [Kramer *et al.*, 2001]. De cette manière, la contrainte devient une simple conjonction $freq(X, \texttt{malades}) \geq i \times n \wedge freq(X, \texttt{sains}) \leq i$. On recherche alors la bordure des motifs fréquents dans `malades` et la bordure des motifs non fréquents dans `sains`. Ensuite, la différence entre ces deux bordures constitue un sous-

ensemble des motifs émergents (cette différence peut être effectuée avec l'algorithme BORDER-DIFF [Dong et Li, 1999]). Au final, cette méthode est donc incomplète car elle devrait être répétée pour chaque i (ce qui reviendrait à énumérer tous les motifs !).

Décomposition universelle. D'un point de vue théorique, nous pouvons montrer que toute contrainte est décomposable en une formule booléenne de contraintes monotones. En effet, pour une contrainte donnée q, la disjonction de conjonctions d'une contrainte monotone et d'une contrainte anti-monotone $q'(X) \equiv \bigvee_{Y \in \mathcal{T}h(\mathcal{L},\mathbf{r},q)}(Y \subseteq X \wedge X \subseteq Y)$ est équivalente à q. Par exemple, avec $\mathcal{T}h(\mathcal{L}_{\mathcal{I}}, \mathbf{r}, q) = \{AB, AD, ADE\}$, q' devient $(AB \subseteq X \wedge X \subseteq AB) \vee (AD \subseteq X \wedge X \subseteq AD) \vee (ADE \subseteq X \wedge X \subseteq ADE)$. En pratique, cette décomposition a peu d'intérêt puisqu'il est nécessaire de disposer de la théorie pour l'expliciter. Par ailleurs, les décompositions même explicites peuvent devenir trop complexes (comme pour les motifs émergents) pour être utilisées.

3.3 Représentations condensées des motifs fréquents

Cette section dépeint les deux représentations condensées de motifs fréquents les plus courantes [Calders *et al.*, 2004]. Elles permettent d'inférer la fréquence de n'importe quel motif.

3.3.1 Motifs fermés

Pour raffiner le principe des bordures, la maximalité (au sens de l'inclusion) est cette fois appliquée sur des sous-ensembles du langage où les motifs possèdent la même fréquence. Pour le contexte `malades` (cf. page 10), la figure 3.4 montre ces regroupements de motifs selon leur fréquence. Au sein de chaque ensemble, tous les motifs possèdent la même fréquence tout en étant liés par la relation de spécialisation. Ces ensembles sont appelés classes d'équivalence de fréquence.

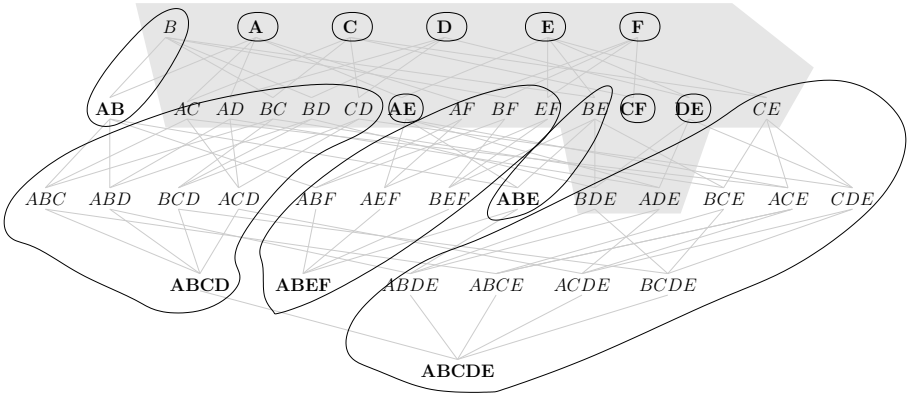


FIG. 3.4 – Classes d'équivalence de fréquence au sein du treillis correspondant au contexte `malades`.

À l'intérieur de chaque classe d'équivalence, on focalise sur deux types de motifs particuliers mis en exergue par la figure 3.4 : ceux correspondant à la zone grisée (décrits à la section suivante) et ceux en gras. Ces derniers, dits *fermés*, sont en fait les éléments maximaux des classes d'équivalence et peuvent être définis de la manière suivante :

Définition 13 (motif fermé) *Un motif φ est fermé ssi toutes ses spécialisations strictes ont une fréquence strictement inférieure à celle de φ.*

Cette définition est commune à différents langages dont les ensembles [Pasquier *et al.*, 1999], les séquences [Yan *et al.*, 2003], les arbres [Chi *et al.*, 2005]. Les motifs fermés sont indifféremment appelés motifs *clos*. Pour les motifs ensemblistes, la *fermeture* d'un motif X est le motif fermé minimal incluant X et on la note par $h(X)$. L'opérateur h est bien un opérateur de fermeture car il possède les propriétés suivantes [Stadler et Stadler, 2002] :

Propriété 1 (Opérateur de fermeture)
- *Extensivité :* $\forall X \in \mathcal{L}_{\mathcal{I}}$, *on a* $X \subseteq h(X)$
- *Idempotence :* $\forall X \in \mathcal{L}_{\mathcal{I}}$, *on a* $h(h(X)) = h(X)$
- *Isotonie :* $\forall X, Y \in \mathcal{L}_{\mathcal{I}}$, *on a* $X \subseteq Y \Rightarrow h(X) \subseteq h(Y)$

Tous les motifs ayant la même fermeture ont la même fréquence. Ainsi, la fermeture forme des classes d'équivalence de fréquence. Une définition alternative repose sur la définition d'une connexion de Galois [Birkhoff, 1967].

La représentation condensée des motifs fermés conserve alors tous les motifs fermés avec leur fréquence. Ainsi, la fréquence d'un motif quelconque peut être déduite grâce à celle de sa fermeture contenue dans cette représentation (e.g., $freq(AC) = freq(h(AC)) = freq(ABCD) = 2$). En ne retenant qu'un représentant par classe d'équivalence, cette représentation conserve peu de motifs et est dite condensée. En pratique, cette concision est souvent très forte. Si on ne s'intéresse qu'aux motifs fréquents, on peut se contenter de ne conserver que les motifs fermés fréquents. Pour le contexte `malades`, on dénombre seulement 13 motifs fermés décrivant 40 motifs. Par rapport à la définition 10, l'ensemble des fermés forme une représentation condensée adéquate à *freq*.

Remarquons que jusqu'ici seule la croissance et la décroissance par rapport à $\preceq$ étaient utilisées soit pour l'élagage, soit pour la représentation (e.g., bordure). La notion de motif fermé repose quant à elle, sur la constance locale de la fréquence au sein des classes d'équivalence. Dans la section suivante, la notion duale de motif libre se base sur ce même principe.

De nombreux algorithmes permettent de calculer les motifs fermés avec divers langages. Pour les motifs ensemblistes, Close [Pasquier *et al.*, 1999] extrait les motifs fermés par un parcours par niveaux. Inspiré d'Eclat, Charm [Zaki et Hsiao, 1999] procède par une approche en profondeur. D'autres privilégient des structures de données facilitant l'obtention des motifs fermés comme Closet [Pei *et al.*, 2000] avec les FP-tree ou [El-Hajj et Zaïane, 2005] avec les COFI-tree. Pour les motifs séquentiels, CloSpan [Yan *et al.*, 2003] ou BIDE [Wang et Han, 2004] énumèrent les séquences fermées. Enfin, CMTreeMiner [Chi *et al.*, 2005] extrait des arbres fermés. Tous les motifs fermés ne sont pas issus d'une connexion de Galois (e.g., lorsque plusieurs motifs fermés appartiennent à la même classe d'équivalence). Seule une connexion de Galois est proposée de manière générale dans [Casas-Garriga, 2003] pour tous les langages basés sur des attributs.

L'extraction de motifs fermés se révèle d'une importance capitale dans les données très corrélées où la plupart des algorithmes présentés à la section 2.2.1 sont inefficaces voire inopérants (cf. le site du FIMI[9]).

[9] `fimi.cs.helsinki.fi`

3.3.2 Motifs libres

Les motifs libres[10] [Boulicaut et Jeudy, 2000] (aussi appelés motifs *générateurs* ou *clés* [Pasquier *et al.*, 1999]) exploitent quant à eux la minimalité des classes d'équivalence de fréquence :

Définition 14 (motif libre) *Un motif φ est libre ssi toutes ses généralisations strictes ont une fréquence strictement supérieure à la sienne.*

Sur la figure 3.4, les motifs libres correspondent exactement aux 21 motifs de la zone grisée. On constate bien que le ou les minimaux de chaque classe d'équivalence est libre.

La réunion de tous les motifs libres fréquents (avec leur fréquence respective) et de la bordure négative forme une représentation condensée des motifs fréquents. Pour un motif donné, la bordure négative permet de vérifier si le motif est fréquent. Si ce dernier n'est pas exclu par ce test, il possède la fréquence minimale de celles des motifs libres qu'il contient (e.g., $freq(ABC) = freq(AC)$ ou $freq(BC) = 2$). Cette représentation bien que condensée est de plus grande cardinalité que celle des motifs fermés car plusieurs libres peuvent appartenir à une même classe d'équivalence. L'un des usages courants de la représentation condensée des motifs libres est la génération des règles d'association en conjonction avec les motifs fermés. En effet, les libres constituent les prémisses minimales tandis que les fermés donnent les conclusions maximales [Bastide *et al.*, 2000] (à support et confiance constantes).

Les motifs libres sont relativement simples à extraire puisque la liberté (i.e., "être libre") est une contrainte anti-monotone [Boulicaut et Bykowski, 2000]. D'ailleurs, les motifs libres sont parfois à la base de l'extraction de motifs fermés [Pasquier *et al.*, 1999, Hamrouni *et al.*, 2005]. D'autres algorithmes extraient des motifs libres satisfaisant aussi une contrainte anti-monotone [Boulicaut et Jeudy, 2001].

Liberté et langage. À notre connaissance, le concept de liberté n'a pas été exploité pour d'autres langages que celui des motifs ensemblistes. Cette faible popularité s'explique peut-être par une représentation moins concise que celle des motifs fermés. Pour les motifs ensemblistes, les motifs libres possèdent en revanche de nombreuses généralisations dont les motifs δ-libres [Boulicaut *et al.*, 2000], les motifs non dérivables [Calders et Goethals, 2002], et les motifs k-libres [Calders et Goethals, 2003].

3.4 Limites de l'extraction de motifs contraints

Malgré l'apport incontestable des bases de données inductives, l'extraction de motifs contraints reste un problème ouvert. La première section montre que la maturité algorithmique n'offre malheureusement pas un processus d'ECBD souple et simple d'utilisation.

3.4.1 Interactivité et itérativité

La littérature insiste sur le fait qu'un processus d'ECBD de qualité requiert une interactivité et une itérativité fortes avec l'utilisateur/analyste [Brachman et Anand, 1996].

L'interactivité du processus doit mettre en avant l'utilisateur au sein de l'extraction. L'utilisateur doit avoir une totale liberté dans l'expression de ses attentes sans contrepartie technique :

[10] Le terme "ouvert" (par opposition à "fermé" pour les motifs maximaux) aurait probablement été plus heureux pour désigner ces motifs minimaux.

1. Le processus doit pouvoir accepter des contraintes variées afin de couvrir la richesse sémantique décrite dans la section 1.2.1 (contraintes d'agrégat ou syntaxique, combinaisons de contraintes). Par ailleurs, il semble important de lui laisser la liberté de proposer de nouvelles contraintes atomiques (différentes de celles imaginées par les informaticiens).
2. L'utilisation du processus d'extraction ne doit pas requérir de connaissances formelles spécifiques à l'extraction sous contraintes. Typiquement, l'utilisateur ne peut pas choisir le solveur selon la contrainte considérée ou calculer une heuristique d'optimisation lui-même.

Malheureusement, la pratique montre que ces deux points sont souvent contradictoires. Le premier favorise une souplesse importante sur la contrainte d'extraction. Pourtant, sans propriétés formelles sur cette contrainte, il est difficile d'avoir une extraction rapide (voire faisable). Les différentes approches proposées dans le chapitre 2 en privilégiant les propriétés formelles, négligent l'interactivité. D'une part, la sémantique des contraintes est restreinte. D'autre part, le choix du solveur adéquat parmi l'importante diversité, requiert de solides connaissances. Même si les bases de données inductives pallient ce problème pour les formules booléennes de contraintes monotones, aucun processus automatique n'est proposé en toute généralité.

L'itérativité se traduit quant à elle par la nécessité de pouvoir répéter le processus. De cette manière, l'utilisateur peut ajuster les paramètres du processus de fouille. Par exemple, il doit pouvoir modifier ou compléter la contrainte (e.g., diminuer le seuil minimal de fréquence). Cela nécessite d'avoir un processus d'extraction suffisamment rapide. Les deux solutions envisagées séparément ou conjointement afin de maîtriser l'itérativité sont des algorithmes d'extraction efficaces (cf. chapitre 2) et des principes d'inférence basés sur les représentations condensées[11]. Longtemps l'itérativité s'est focalisée sur l'obtention des seuls motifs fréquents [Goethals, 2003a] au détriment des motifs satisfaisant d'autres contraintes très utiles pour l'utilisateur. Typiquement, les représentations condensées les plus sophistiquées sont limitées à la fréquence (cf. la section 3.3) exceptée l'approche itérative [Diop, 2003] qui étend le stockage à d'autres mesures. Seul un algorithme propose d'extraire les motifs maximaux de classes d'équivalences adaptées à une contrainte monotone [Bonchi et Lucchese, 2004]. Ainsi, ces nombreux efforts n'ont pas permis de réaliser des méthodes efficaces pour rechercher les motifs satisfaisant la contrainte d'aire minimale ou d'émergence. De manière plus générale, l'interactivité a souffert de cette quête de la vitesse.

Nous souhaitons remettre l'utilisateur au centre du processus d'extraction. Pour cela, nous proposons de raffiner la granularité des primitives jusqu'ici limitées aux contraintes atomiques. En les combinant entre elles, l'utilisateur peut ainsi définir les contraintes usuelles mais surtout en imaginer de nouvelles. À partir de ces contraintes flexibles, notre objectif est de proposer des méthodes efficaces d'extractions nécessitant le moins possible de connaissances techniques pour l'utilisateur. Cette tâche est périlleuse car la nature de ces contraintes diverge radicalement de celles traitées dans la littérature.

3.4.2 Faisabilité des extractions

Plus les méthodes tolèrent des contraintes variées, moins elles sont génériques au niveau du langage. Réciproquement, les méthodes les plus générales pour les langages s'appliquent à un ensemble de contraintes plus restreint. Ce phénomène est illustré par la figure 3.5. Schématiquement, les méthodes générales d'extractions échouent au-dessus de la ligne en pointillés lorsqu'on considère des contraintes et des langages variés. Au final, seules les formules

[11] À un autre niveau, des solutions de parallélisation d'algorithmes et de calcul distribué sont aussi discutées dans la littérature [Zaki, 1999].

booléennes de contraintes monotones peuvent être extraites pour un langage quelconque (et représentées grâce aux bordures, cf. la section 3.2). Mais des contraintes utiles comme celles d'aire minimale, de moyenne minimale ou d'émergence ne peuvent pas être traitées de la sorte.

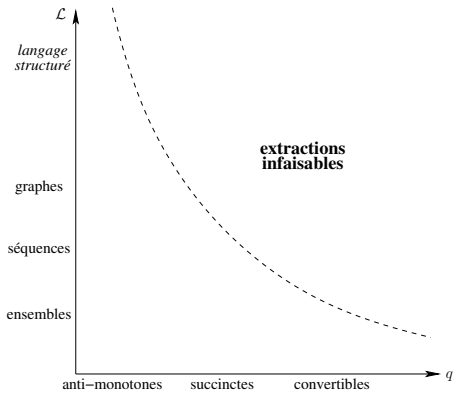


FIG. 3.5 – Limites de l'extraction de motifs.

Le traitement des contraintes flexibles décrites ci-avant (e.g., contrainte d'aire minimale) dépassent allègrement les limites de faisabilités actuelles. Dans la partie suivante, nous allons concevoir et développer de nouvelles méthodes fondées sur l'ensemble des élagages proposés dans la littérature. Ainsi, nous exploiterons des élagages négatifs et positifs qui interviendront sur les généralisations, les spécialisations ou des intervalles.

Conclusion

Le premier chapitre a souligné l'intérêt des contraintes et de langages variés pour la tâche d'extraction de motifs. Ces deux dimensions ouvrent aux motifs un vaste champ applicatif à travers la découverte de motifs locaux, de motifs globaux et la construction de modèles.

Malheureusement les classes de contraintes usuelles (e.g., monotones ou convertibles) ont une richesse sémantique limitée et ne s'appliquent pas à tous les langages. Les propriétés formelles définissant ces classes sont plus souvent issues de conditions d'élagage requises pour diminuer la complexité d'extraction. Ces classes ne recouvrent pas tous les besoins de l'utilisateur. Les combinaisons des contraintes de ces classes et les principes d'inférence décrits au chapitre 3 donnent plus de liberté à l'utilisateur sans toutefois lui permettre de formuler certaines contraintes. Comme ce dernier n'a pas toujours les connaissances requises pour établir une chaîne de traitements adaptée à sa contrainte, il se contente de contraintes moins originales et surtout, moins expressives.

La partie suivante propose un cadre où l'utilisateur peut formuler librement des contraintes très diverses sans s'interroger sur l'adéquation de sa contrainte au solveur. Pour pallier la difficulté algorithmique de l'extraction de motifs, nous revisitons certaines approches d'extractions et nous en proposons des nouvelles.

Deuxième partie

Un cadre générique de motifs contraints basé sur les primitives

Introduction

La partie précédente a mis en lumière l'inadéquation entre les méthodes d'extraction définies en terme de propriété d'élagage et les besoins de l'utilisateur. À contre-pied, nous proposons maintenant un cadre fondé sur des primitives (PBF) qui privilégie des contraintes orientées utilisateur. Des fonctions telles que la fréquence ou la longueur d'un motif constituent le centre d'intérêt de l'utilisateur car elles décrivent qualitativement et quantitativement les motifs désirés. Nous en faisons les primitives à partir desquelles l'utilisateur peut librement construire sa contrainte (PBC) [Soulet et Crémilleux, 2005a]. La richesse combinatoire de cette classe englobe les classes de contraintes les plus usuelles dont les contraintes monotones et anti-monotones [Soulet et Crémilleux, 2005c]. Nous verrons que la généricité de cette approche ne s'oppose pas à la possibilité de pousser les PBC au cœur de l'extraction. En particulier, les opérateurs de majoration et minoration, importants dans nos méthodes d'extraction, bornent une contrainte sur un intervalle de manière automatique. Ils permettent aussi de détecter les PBC satisfaisant un critère de monotonie. Par ailleurs, la partie théorique du PBF (i.e., définition des PBC et les opérateurs) est translangagière.

Les chapitres 6 à 9, présentent chacun une contribution nouvelle à l'extraction de motifs contraints. Ces méthodes complémentaires peuvent être utilisées individuellement ou simultanément. Nous proposons une première méthode d'extraction qui approxime la PBC par des contraintes monotone et anti-monotone appelées *relaxations*. Ces relaxations peuvent alors être utilisées par les algorithmes classiques d'extraction de contraintes monotones et anti-monotones pour n'importe quel langage. Nous les obtenons grâce à un artifice de calcul reposant sur deux motifs, dits *virtuels*, qui résument un espace des versions [Soulet et Crémilleux, 2005b]. Deuxièmement, nous étendons alors cette approche aux *contraintes globales* dont la vérification nécessite des comparaisons entre plusieurs motifs. La méthode proposée par *Approximer-et-Pousser* sera utilisée pour la découverte des k motifs maximisant une mesure basée sur des primitives.

Les applications de notre travail nous ont confronté à des contextes transactionnels difficiles où le nombre de transactions est réduit mais la taille de chaque transaction excède plusieurs milliers d'items. Ces contextes applicatifs nous ont suggéré de concevoir un solveur spécifique dédié à n'importe quelle PBC. L'efficacité de celui-ci, appelé MUSIC-DFS, repose sur un puissant élagage sur les intervalles combiné avec un parcours en profondeur de l'espace de recherche [Soulet *et al.*, 2006]. Enfin, nous généralisons la notion d'opérateur de fermeture afin d'obtenir des *représentations condensées adéquates* à d'autres fonctions que celle classique de fréquence. Ces représentations condensées adéquates, fournies par l'algorithme MICMAC, sont très concises. Dans le cas particulier des mesures d'intérêt basées sur la fréquence (e.g., la confiance ou le taux de croissance), une nouvelle représentation condensée de motifs forts est définie [Soulet *et al.*, 2004b]. Ces motifs forts forment une couverture de la base de données maximisant les mesures de fréquences. Les nouveaux opérateurs de fermeture peuvent être repris par certaines approches fondées sur la fermeture de Galois (e.g., les règles d'association). En

particulier, MUSIC et MUSIC-DFS peuvent en bénéficier pour optimiser le parcours de l'espace de recherche en fonction de la contrainte.

Chapitre 4

Les contraintes basées sur des primitives

Sommaire

Dans ce chapitre, nous proposons le *primitive-based framework* (PBF) qui est un cadre générique pour définir de façon flexible les contraintes. Nous montrons en quoi ce cadre pallie les insuffisances des classes de contraintes présentées dans la partie précédente. Ce cadre est fondé sur des primitives permettant d'exprimer des contraintes avec un fin niveau de granularité. Un tel niveau de granularité permet d'évaluer avec précision la qualité d'un motif au regard de multiples critères. Combinées entre elles, ces primitives permettent alors d'exprimer une large panoplie de contraintes. Clairement la définition de ces contraintes basées sur des primitives (PBC) a un pouvoir d'expression plus élevé que celles des classes usuelles.

La première section définit la notion de contraintes basées sur des primitives. La section 4.2 en montre l'intérêt qualitatif de par la portée sémantique et la flexibilité. Enfin, la section 4.3 montre la généralité de la classe des PBC en la comparant avec les classes usuelles de la littérature.

4.1 Des primitives aux contraintes

4.1.1 Primitives

La construction des contraintes présentées au chapitre 2 repose sur des fonctions fines évaluant la qualité d'un motif au regard de différents critères. Par exemple, la fréquence est une fonction estimant la présence du motif au sein de la base de données et la longueur donne sa taille. Ces différentes fonctions sont très intuitives à manipuler car elles rendent compte de propriétés dont la sémantique est immédiate. Il est alors aisé de les combiner entre elles de façon

à décrire au mieux les motifs intéressants. Typiquement, la mesure d'aire traduit un compromis entre les deux mesures d'intérêts que sont la fréquence et la longueur. Si l'une de ses deux mesures augmente, l'intérêt global du motif augmente également.

Notre démarche première est de mettre ces fonctions au cœur de notre cadre en en faisant les éléments primitifs qui permettent de construire les contraintes. Pour cette raison, nous parlerons de *cadre fondé sur les primitives* (abrégé PBF pour *primitive-based framework*). Nous introduisons maintenant ces primitives de manière plus formelle :

Définition 15 (Primitive) *Une primitive est une fonction monotone suivant chacune de ses variables (lorsque les autres restent constantes).*

La monotonie suivant chacune des variables imposée aux primitives, est nécessaire pour calculer automatiquement des bornes dans le chapitre 5. Cette notion de monotonie impose que les ensembles de départ et d'arrivée des primitives sont partiellement ou totalement ordonnés. L'annexe A présente des primitives du PBF pour le langage des motifs ensemblistes et séquentiels. Par exemple, la multiplication vérifie cette définition pour les réels positifs car elle croît suivant chacune de ses deux variables si l'autre est fixée. De manière similaire, pour les motifs ensemblistes et séquentiels, les fonctions *freq* et *count* sont des primitives de notre cadre (la première étant décroissante et la seconde, croissante). Ces exemples soulignent qu'en général une primitive dépend du langage et/ou de la base de données. Par exemple, la multiplication n'est pas une primitive du PBF pour tous les réels positifs *et* négatifs. La fonction *freq* varie évidemment selon la base de données sur laquelle le motif est évalué. Par la suite, l'ensemble des primitives est dénoté par $\mathcal{P}$. L'ensemble $\mathcal{P}$ contient les fonctions constantes qui sont évidemment des primitives du cadre. Enfin, parmi cet ensemble, nous distinguons les primitives dites *terminales* qui évaluent un motif (e.g., *freq* ou *count*), de celles dites *non-terminales* qui permettent seulement de les combiner (e.g., $\times$).

Certaines fonctions ne peuvent pas (ou pas immédiatement) être considérées comme des primitives du PBF. Par exemple, la fonction *sum* avec des valeurs positives et négatives n'est pas monotone (en considérant $\mathcal{L}_{\mathcal{I}}$ muni de la relation de spécialisation $\subseteq$). De même, la fonction sinus n'est ni croissante, ni décroissante pour tous les réels. La section suivante montre comment lever cette limite.

4.1.2 Combinaisons de primitives

Grâce à un principe de décomposition, l'utilisateur peut exprimer des contraintes portant sur des propriétés ne correspondant pas à des primitives monotones. Le plus souvent une primitive non-monotone est décomposable en plusieurs primitives monotones. Par exemple, la primitive *sum* peut facilement être étendue à $\Re$ en définissant sum^+ (restreinte aux réels positifs) et sum^- (restreinte aux réels négatifs), et en observant que $sum = sum^+ + sum^-$. De plus, toutes les primitives de notre cadre sont amenées à être combinées pour définir d'autres fonctions plus évoluées telle que la mesure d'aire (i.e., $area(X) = freq(X) \times count(X)$). Ces combinaisons peuvent alors être vues comme des primitives de haut niveau du PBF :

Définition 16 (Primitive de haut niveau) *Une primitive de haut niveau définie sur un langage $\mathcal{L}$ est une composition de primitives.*

Soit $\mathcal{H}$ l'ensemble des primitives de haut niveau. Le plus grand nombre de combinaisons d'une primitive h de $\mathcal{H}$ est appelé *degré*. Il détermine un certain niveau de complexité. Plus précisément, les primitives de haut niveau de degré 0 sont exactement les primitives terminales.

Ensuite, chaque primitive de haut niveau de degré n résulte de la composition d'une primitive p avec k primitives de haut niveau h_i de degré inférieur tel qu'on ait $h = p(h_1, \ldots, h_k)$ (au moins une primitive de haut niveau doit être de degré $n-1$). La forme $p(h_1, \ldots, h_k)$ est appelée la décomposition de h. Le degré de h correspond à $1 + \deg\{h_i | i \in \{1, \ldots, k\}\}$ où $p(h_1, \ldots, h_k)$ est la décomposition de h. Par exemple, comme *freq* et *count* sont des primitives terminales (de $\mathcal{L}_\mathcal{I}$ ou $\mathcal{L}_\mathcal{S}$ vers $\Re^+$), leur degré est 0. Ainsi, le degré de l'aire est 1 puisque sa décomposition préfixée correspond à $\times(freq, count)$.

En réalité, une même primitive de haut niveau est décrite par une multitude d'expressions. Même si $1 \times freq(X)$ est une expression différente de $freq(X)$, les deux fonctions associées (i.e., $X \mapsto 1 \times freq(X)$ et $X \mapsto freq(X)$) sont équivalentes. Dans la suite, nous assimilons les notions d'expression et de fonction (bien qu'une primitive de haut niveau et son expression ne se réfèrent pas au même objet)[12].

4.1.3 Contraintes fondées sur des primitives

Les primitives de haut niveau constituent donc une grammaire correspondant aux fonctions définies sur $\mathcal{L}$ (et composées de fonctions monotones). En observant que la comparaison $\geq$ appartient aux primitives $\mathcal{P}$, on constate que la contrainte $freq(X) \times count(X) \geq \rho$ est aussi une primitive de haut niveau. Finalement, une *contrainte fondée sur des primitives* est une primitive de haut niveau à valeur booléenne $\mathfrak{B} = \{false, true\}$:

Définition 17 (Contrainte fondée sur les primitives) *Une primitive de haut niveau définie sur un langage $\mathcal{L}$ à valeur booléenne est une contrainte fondée sur des primitives.*

Par la suite, l'acronyme anglais PBC issu de *primitive-based constraint* désigne une contrainte basée sur des primitives. L'ensemble des contraintes fondées sur des primitives est noté $\mathcal{Q}_\mathcal{L}$ et est un sous-ensemble de $\mathcal{H}$ vérifiant la relation suivante : $\mathcal{Q}_\mathcal{L} = \{h \in \mathcal{H} | h : \mathcal{L} \rightarrow \mathfrak{B}\}$. Lorsqu'il n'y a pas d'ambiguïté sur le langage, $\mathcal{Q}_\mathcal{L}$ est noté $\mathcal{Q}$.

Les contraintes présentées dans le tableau 2.1 (page 23) sont toutes des contraintes basées sur des primitives pour le langage des motifs ensemblistes. Un contre-exemple de contrainte n'appartenant pas à $\mathcal{Q}$ nécessite d'utiliser des primitives non-monotones sur leur domaine de définition. Typiquement, la fonction $sin(freq(X)) \geq 0$ n'est pas une contrainte basée sur des primitives de $\mathcal{Q}$ puisque la fonction *sin* n'est pas une primitive de notre cadre sur $\Re^+$.

Le tableau 4.1 décrit récursivement les PBC correspondant aux primitives $\{\wedge, \vee, \neg, <, \leq, \subset, \subseteq, +, -, \times, /, count, freq, sum, max, min, \cup, \cap, \setminus\} \subseteq \mathcal{P}$ pour le langage des motifs ensemblistes (cf. annexe A). Cet ensemble correspond à une partie de la grammaire de contraintes reconnues par le solveur MUSIC (cf. le chapitre 8). Notons que f est la fonction d'intension i.e. $f(T)$ est l'ensemble maximal d'items contenu dans chacune des transaction de T et g est la fonction d'extension i.e. $g(X)$ est l'ensemble maximal de transaction contenant le motif ensembliste X.

Les contraintes basées sur des primitives constituent ensembles une classe de contraintes. En effet, comme le prouve le tableau 4.1, une infinité de contraintes atomiques (à paramètres constants) satisfont la définition d'une contrainte basée sur des primitives. Pour cette raison, les méthodes et prototypes exposés dans les chapitres suivants s'appliquent à des contraintes diverses et variées. Le reste de ce chapitre va désormais montrer l'intérêt qualitatif et quantitatif de ces contraintes pour l'utilisateur.

[12] Cette pratique est courante en matière de grammaires. Par exemple, une expression régulière est souvent confondue avec le langage rationnel qu'elle décrit.

Contrainte $q \in \mathcal{Q}$	Primitive(s)	Opérande(s)
$q_1 \theta q_2$	$\theta \in \{\wedge, \vee\}$	$(q_1, q_2) \in \mathcal{Q}^2$
θq_1	$\theta \in \{\neg\}$	$q_1 \in \mathcal{Q}$
$e_1 \theta e_2$	$\theta \in \{<, \leq\}$	$(e_1, e_2) \in \mathcal{E}^2$
$s_1 \theta s_2$	$\theta \in \{\subset, \subseteq\}$	$(s_1, s_2) \in \mathcal{S}^2$
constante $b \in \mathfrak{B}$	-	-
Expression d'agrégat $e \in \mathcal{E}$	**Primitive(s)**	**Opérande(s)**
$e_1 \theta e_2$	$\theta \in \{+, -, \times, /\}$	$(e_1, e_2) \in \mathcal{E}^2$
$\theta(s)$	$\theta \in \{freq, count\}$	$s \in \mathcal{S}$
$\theta(s.val)$	$\theta \in \{sum, max, min\}$	$s \in \mathcal{S}$
constante $r \in \Re^+$	-	-
Expression syntaxique $s \in \mathcal{S}$	**Primitive(s)**	**Opérande(s)**
$s_1 \theta s_2$	$\theta \in \{\cup, \cap, \setminus\}$	$(s_1, s_2) \in \mathcal{S}^2$
$\theta(s_1)$	$\theta \in \{f, g\}$	$s_1 \in \mathcal{S}$
variable $X \in \mathcal{L}_\mathcal{I}$	-	-
constante $l \in \mathcal{L}_\mathcal{I}$	-	-

TAB. 4.1 – Un sous-ensemble de contraintes de $\mathcal{Q}_{\mathcal{L}_\mathcal{I}}$.

4.2 Des contraintes flexibles

La richesse sémantique de notre classe découle de sa richesse combinatoire. La finesse du grain primitif utilisé pour définir les contraintes associée à la possibilité de les combiner à outrance permet d'exprimer un large spectre de contraintes avec chacune de ses nuances. Les contraintes (même atomiques) sont obtenues à partir de la notion de primitive. Alors que les méthodes usuelles des bases de données inductives combinent seulement les contraintes atomiques pour grossir l'expressivité des contraintes, les PBC combinent des primitives plus petites pour produire des contraintes atomiques variées. De fait, en combinant une dizaine de primitives, nous obtenons beaucoup plus de contraintes qu'en combinant une dizaine de contraintes atomiques.

Or combiner des primitives, c'est avant tout combiner leurs sens. De cette manière, à partir de la sémantique individuelle et simple de chaque primitive, la subtilité et la variété des combinaisons construisent une sémantique très riche de la contrainte dans sa globalité. La variété des primitives terminales et des primitives pour les combiner issue de la souplesse de la définition 15 offre au final une classe à la sémantique couvrant les besoins de l'utilisateur. En particulier, les primitives peuvent relier des informations issues de données variées (taxonomies, matrices de similarités, etc) comme en témoigne l'annexe A. Ainsi, avec seulement quelques dizaines de primitives, les contraintes de MUSIC sont multiples et variées, et répondent à des problèmes pourtant très divers.

La variété des exemples pratiques de contraintes listés dans la section 2.2 et appartenant à $\mathcal{Q}$ illustre cette richesse sémantique. Elle atteste du bienfondé de cette classe de contraintes et donc, des méthodes d'extraction présentées dans les chapitres à venir. En particulier, les PBC sont transverses aux deux types de contraintes présentés dans la section 1.2.1 à savoir les contraintes d'agrégats et les contraintes syntaxiques. En effet, le type de contrainte repose essentiellement sur le type de primitives utilisées. Or la définition 15 couvre à la fois les primitives d'agrégats (e.g., *count*, *sum*) et les primitives syntaxiques (e.g., les opérateurs ensemblistes).

Enfin, celles-ci peuvent aussi être utilisées conjointement dans une même requête en les com-

binant avec des opérateurs booléens (e.g, la requête $freq(X, \texttt{malades}) \geq minfr \wedge freq(X, \texttt{sains}) \leq maxfr$ de la page 13). Ce point illustre à nouveau la puissance générée par les combinaisons.

L'une des principales limitations des classes usuelles de contraintes est l'impossibilité de les combiner pour améliorer la finesse de description des motifs souhaités. Par exemple, la conjonction de deux contraintes convertibles n'est pas toujours une contrainte convertible, il est alors nécessaire de mettre en œuvre des méthodes spécifiques pour traiter de telles combinaisons de contraintes atomiques. À l'inverse, notre classe de contrainte autorise les combinaisons de contraintes :

Propriété 2 *Les contraintes basées sur des primitives sont stables par opérations booléennes.*

Preuve. Soient q_1 et q_2 deux contraintes de $\mathcal{Q}$. Comme $\{\wedge, \vee, \neg\} \subset \mathcal{P}$, les contraintes $q_1 \wedge q_2$, $q_1 \vee q_2$ et $\neg q_1$ sont des contraintes basées sur des primitives. □

D'un point de vue théorique, les contraintes basées sur des primitives forment une algèbre de Boole. Cette propriété permet à l'utilisateur de combiner différentes contraintes atomiques pour affiner ses souhaits. En pratique, un solveur pour toutes les PBC pourra affiner une requête extrayant de trop nombreux motifs en la complétant avec une autre contrainte atomique.

4.3 Comparaisons avec les autres classes

Il est naturel de chercher à comparer les PBC avec les autres classes de contraintes (cf. le chapitre 2). Cette comparaison n'est pas simple à mener, car les PBC ne sont pas définies à partir d'une propriété globale.

4.3.1 Langage quelconque : contraintes monotones et anti-monotones

Seule la monotonie et les contraintes associées sont définies et utilisées pour n'importe quel langage partiellement ordonné. En premier lieu, il est donc naturel de confronter notre classe de contraintes $\mathcal{Q}$ aux contraintes monotones et anti-monotones. Les contraintes basées sur des primitives sont un sur-ensemble des contraintes monotones et anti-monotones :

Propriété 3 (Sur-ensemble des contraintes (anti-)monotones) *Une contrainte monotone ou anti-monotone est une contrainte basée sur des primitives.*

Preuve. Soit q une contrainte monotone, on a $\forall \varphi \preceq \gamma \wedge q(\varphi) \Rightarrow q(\gamma)$. En considérant que $faux < vrai$, on obtient que $q(\varphi) \leq q(\gamma)$ pour tout $\varphi \preceq \gamma$ i.e. q est une fonction croissante à valeurs booléennes. Ainsi, q est une primitive du PBF et donc, une contrainte de $\mathcal{Q}$. La même démonstration montre qu'une contrainte anti-monotone est une contrainte basée sur des primitives. □

Ainsi, les PBC recouvrent entièrement les contraintes monotones et anti-monotones. De nombreuses contraintes usuelles ayant des critères de monotonie sont donc directement utilisables au sein du PBF. Cette propriété en conjonction avec la propriété 2 implique même que les requêtes inductives basées sur la monotonie sont des contraintes fondées sur des primitives. Le PBF englobe donc la plupart des contraintes usuellement manipulées par les travaux sur les bases de données inductives.

Certaines contraintes monotones ou anti-monotones peuvent se décomposer à l'aide de primitives monotones comme c'est le cas pour la contrainte de fréquence minimale. En revanche, d'autres contraintes comme la liberté ne peuvent pas toujours être exprimées facilement avec des

primitives usuelles. De telles contraintes non-décomposables peuvent néanmoins être directement implémentées au sein d'un solveur.

4.3.2 Langage des motifs ensemblistes : contraintes succinctes et convertibles

Des comparaisons sur des langages plus spécifiques peuvent être effectuées avec d'autres classes de contraintes. En particulier, le langage des motifs ensemblistes possède plusieurs classes de contraintes originales (cf. section 2.2) avec lesquelles il est intéressant de comparer les PBC. Les contraintes succinctes qui sont définies avec des unions et des différences de sous-langages peuvent se traduire en terme de primitives. De plus, comme la convertibilité est une forme particulière de monotonie, on s'attend à retrouver la propriété 3. Les propriétés 4 et 5 traduisent ces observations :

Propriété 4 (Sur-ensemble des contraintes succinctes) *Une contrainte succincte est aussi une contrainte basée sur des primitives.*

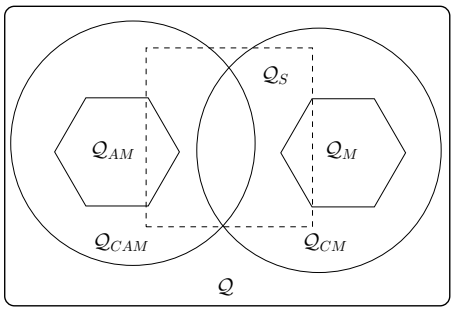
Preuve. Comme l'union $\cup$ est croissante suivant chacune de ses variables (i.e., pour un motif A, on a $\forall X \subseteq Y \Rightarrow A \cup X \subseteq A \cup Y$), elle est une primitive de $\mathcal{P}$ tout comme la différence $\setminus$ (croissante suivant sa première variable et décroissante suivant la seconde). Une contrainte succincte étant définie avec ces seules primitives (définition 7) et des sous-langages qui sont des constantes, elle est aussi une contrainte basée sur des primitives. □

Propriété 5 (Sur-ensemble des contraintes convertibles) *Une contrainte convertible est aussi une contrainte basée sur des primitives.*

Preuve. La preuve est similaire à celle de la propriété 3 en considérant $\preceq_R$ au lieu de $\preceq$, où R est la relation d'ordre pour convertir la contrainte convertible en contrainte (anti-)monotone suivant les préfixes. □

Les propriétés 4 et 5 montrent que les contraintes succinctes ou convertibles sont intégrées à notre cadre théorique. En particulier, toutes les contraintes appartenant à ces deux classes et présentées dans les travaux [Ng *et al.*, 1998, Pei *et al.*, 2001a, Leung *et al.*, 2002] pourront être manipulées avec nos méthodes. Tout comme pour la monotonie, les différentes formes de contraintes convertibles (i.e., monotone, anti-monotone ou forte) sont toutes contenues dans notre cadre. De plus, le résultat de la propriété 5 est extensible aux motifs séquentiels (qui rappelons-le ont aussi une forme de contraintes convertibles). Si on exploite des relations de spécialisations basées sur des ordres distincts, le même problème d'incompatibilité entre plusieurs contraintes convertibles perdure (cf. la section 2.2.3). Néanmoins, toutes les contraintes convertibles exposées dans [Pei *et al.*, 2001a] peuvent être exprimées en utilisant des primitives monotones basées sur la relation $\subseteq$ et leurs combinaisons deviennent alors possibles. Par exemple, la contrainte $avg(X.val) \geq \rho$ se décompose en $sum(X.val)/count(X) \geq \rho$.

La figure 4.1 résume ces différentes comparaisons. La comparaison des contraintes basées sur des primitives avec les contraintes anti-monotones relâchées [Bonchi et Lucchese, 2005] ou les contraintes séparables [Wang *et al.*, 2005] reste un problème ouvert et n'apparaît donc pas sur la figure. Néanmoins, toutes les spécimens de contraintes séparables ou anti-monotones relâchées données en exemple dans ces travaux [Wang *et al.*, 2005, Bonchi et Lucchese, 2005] sont des PBC.



$\mathcal{Q}$: contraintes basées sur les primitives
$\mathcal{Q}_{AM}$: contraintes anti-monotones
$\mathcal{Q}_M$: contraintes monotones
$\mathcal{Q}_S$: contraintes succinctes
$\mathcal{Q}_{CAM}$: contraintes convertibles anti-monotones
$\mathcal{Q}_{CM}$: contraintes convertibles monotones

FIG. 4.1 – Comparaisons des différentes classes de contraintes.

Chapitre 5

Opérateurs de bornes et détection de la monotonie

Sommaire

Dans ce chapitre, nous montrons comment les bonnes propriétés des contraintes du PBF permettent de définir des opérateurs de bornes qui sont au cœur de nombreux résultats. La majoration et la minoration de n'importe quelle PBC conduisent à la détection de la monotonie des contraintes à la section 5.3, à l'obtention de relaxations monotones et anti-monotones au chapitre 6 et à un opérateur d'élagage sur les intervalles, fondement de l'algorithme MUSIC au chapitre 8. L'idée centrale des opérateurs du PBF est de combiner récursivement les propriétés individuelles de chaque primitive afin d'aboutir à une propriété globale sur la contrainte. Malgré la grande diversité des PBC, cette méthodologie permet de les manier automatiquement et uniformément de façon aussi bien théorique que pratique.

La section 5.1 introduit la problématique de manipulation des PBC et donne les principes généraux en les illustrant pratiquement sur des tâches élémentaires. Ensuite, la section 5.2 décrit le processus pour minorer et majorer une contrainte via des opérateurs. Ces derniers sont directement exploités pour détecter des contraintes monotones et anti-monotones à la section 5.3.

5.1 Principes des opérateurs

5.1.1 Principes

Le chapitre précédent a montré que le PBF définit un large ensemble de contraintes aux formes variées, englobant par exemple les contraintes monotones, anti-monotones, succinctes, etc.

Cette diversité des contraintes du PBF rend difficile la réutilisation des méthodes d'extraction de motifs développés pour chacune de ces classes : il serait alors nécessaire de proposer des solutions nouvelles pour les PBC ne relevant pas de ces méthodes et de combiner toutes ces stratégies (i.e., automatiser le choix de la stratégie adéquate suivant la PBC). Une telle approche garantirait difficilement l'exhaustivité pour traiter toutes les PBC (et tout langage) et la multiplication des implémentations serait lourde en pratique. Plutôt que de développer des approches différentes et de chercher à les associer ad hoc, nous proposons de traiter n'importe quelle contrainte avec la même stratégie automatique.

Pour n'importe quelle contrainte, l'objectif est de trouver une propriété sur l'ensemble de la contrainte en s'appuyant sur les propriétés individuelles de chaque primitive et en les combinant entre elles. Plus précisément, les caractéristiques issues de la monotonie des primitives terminales servent de conditions initiales. Ces dernières sont alors combinées récursivement suivant les primitives qui construisent la contrainte à analyser. Outre un traitement générique pour toutes les PBC, les manipulations basées sur des primitives ont l'avantage de s'effectuer indépendamment du langage $\mathcal{L}$ ou de la base de données $\mathbf{r}$. Elles peuvent donc être employées indifféremment avec les motifs ensemblistes, les séquences, les arbres... De plus, la section suivante explique que leur mise en œuvre pratique est aisée.

Nous verrons dans la section 5.2 que la monotonicité des primitives reflète bien leur comportement mais ne procure pas toujours une exactitude suffisante. Par exemple, la fonction *freq* est décroissante, mais la décroissance entre deux motifs ne peut pas être quantifiée en se basant uniquement sur la contrainte. Ainsi, les informations globales obtenues s'avèrent souvent être des approximations (i.e., des conditions suffisantes pas forcément nécessaires). Cet aspect s'oppose radicalement aux approches usuelles de l'extraction de motifs contraints qui privilégient des conditions nécessaires et suffisantes pour une efficacité accrue au détriment de la généricité. Nous verrons cependant que les approximations que nous déduisons fonctionnent plutôt bien en pratique.

5.1.2 Illustrations pratiques

Cette section illustre le paradigme de manipulation de contraintes sur deux tâches : vérifier qu'une contrainte peut être traitée par un solveur et calculer le degré d'une contrainte. La mise en œuvre pratique de manipulations est immédiate. À partir d'une contrainte formulée par l'utilisateur, les implémentations des traitements automatiques (e.g., l'extraction de motifs) se fondent sur deux points clés : (1) disposer d'un dictionnaire de primitives avec leurs différentes caractéristiques comme la monotonie suivant chacune des variables, et (2) les combiner récursivement.

Avant de détailler les deux exemples simples de traitements, précisons que l'utilisateur spécifie la PBC en utilisant un langage déclaratif. Ce dernier découle de la grammaire des primitives de haut niveau décrite dans la section 4.1.2. Par exemple, la figure 5.1 représente l'arbre syntaxique de l'expression $\mathit{freq}(X) \times \mathit{count}(X) \geq 6$. Les feuilles correspondent exactement aux primitives terminales (i.e., *freq*, *count* et 6). De la même manière, $\times$ et $\geq$ qui constituent en quelque sorte des règles de composition, sont les nœuds internes. La profondeur de l'arbre (ici 2) donne le degré de la contrainte.

À partir de l'arbre syntaxique d'une PBC, illustrons le calcul du degré ou la vérification de l'appartenance de la contrainte au langage du prototype.

Vérification d'une contrainte Dans la pratique, un solveur manipulant les PBC (e.g., MUSIC) contient nécessairement un nombre défini de primitives implémentant un sous-ensemble P

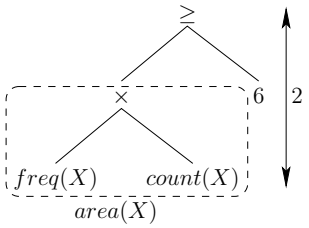


FIG. 5.1 – Arbre syntaxique de la contrainte d'aire minimale.

de $\mathcal{P}$. La vérification d'une contrainte q spécifiée par l'utilisateur consiste à tester son appartenance à la grammaire de contraintes issue de P. Pour cela, l'arbre syntaxique de la contrainte est parcouru pour vérifier que chaque nœud est une primitive de P et que chaque feuille est une primitive terminale de P.

Calcul du degré d'une contrainte Le calcul du degré d'une contrainte (et plus généralement d'une primitive de haut niveau) correspond à la traduction de la formule récursive proposée dans la section 4.1.2. Chaque primitive terminale est caractérisée par un degré nul. Le degré d'une primitive de haut niveau résulte alors de l'incrémentation du degré maximal des différentes sous-primitives de haut niveau.

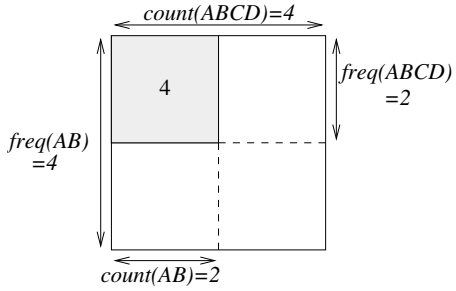
Contrairement aux manipulations ultérieures, ces deux exemples traitent de manière identique la croissance et la décroissance d'une primitive selon une variable. Le second exemple ne distingue pas non plus le traitement à effectuer entre les primitives terminales et les autres. Les chapitres à venir traitent des tâches plus complexes comme la détection de la monotonie et l'extraction de motifs. Elles nécessitent parfois plusieurs manipulations distinctes pour une même contrainte, mais elles restent automatisables. La section suivante se concentre sur la minoration et la majoration de contraintes qui s'avère être une étape préliminaire cruciale à de nombreuses tâches.

5.2 Minoration et majoration de contraintes sur un intervalle

5.2.1 Intuitions clés

L'état de l'art, notamment dans la section 2.1.3, a montré la nécessité de prévoir le comportement de la contrainte afin de réduire l'espace de recherche. Les motifs satisfaisant les conditions d'élagage apparaissent alors comme des bornes au-delà desquelles plus aucun motif ne satisfait la contrainte. De manière similaire, nous proposons de déduire des minorations et des majorations afin de connaître l'évolution de la contrainte. Plutôt que de se limiter aux généralisations ou aux spécialisations comme il est d'usage, ces bornes seront calculées sur des intervalles quelconques.

À partir de la mesure d'aire, nous allons maintenant montrer comment prendre en compte les caractéristiques des primitives pour calculer des bornes sur un intervalle. Tout d'abord, notons que toutes les spécialisations de X ont une fréquence inférieure à celle de X et que toutes les généralisations de X ont une longueur inférieure à celle de X car *freq* est décroissante et *count* est croissante. Ainsi en considérant les motifs AB et $ABCD$ du contexte $\mathcal{D}$ (cf. la figure 5.2), on obtient le minorant $freq(ABCD) \times count(AB)$ $(= 2 \times 2 = 4)$ de l'aire des motifs compris entre AB et $ABCD$ à savoir les motifs AB, ABC, ABD, $ABCD$. En admettant que la contrainte



D

Trans.	Items					
t_1	A	B			E	F
t_2	A	B			E	
t_3	A	B	C	D		
t_4	A	B	C	D	E	
t_5				D	E	
t_6			C			F

FIG. 5.2 – Illustration d'un minorant pour l'aire avec le contexte $\mathcal{D}$.

soit $freq(X) \times count(X) \geq 4$, il n'est pas nécessaire de vérifier la contrainte sur ces 4 motifs. La figure 5.2 schématise ce résultat où les motifs compris entre AB et $ABCD$ ont une aire plus grande que celle du carré grisé. Similairement, quand $freq(X) \times count(Y)$ est strictement inférieur à 4, l'aire du motif Z (tel que $X \subseteq Z \subseteq Y$) est inévitablement inférieure à 4. Au final, l'aire du motif Z est donc bornée par $freq(Y) \times count(X) \leq freq(Z) \times count(Z) \leq freq(X) \times count(Y)$. Ce principe sera particulièrement développé dans le chapitre 8 à propos de l'élagage sur les intervalles.

5.2.2 Opérateurs de minoration et de majoration

Nous formalisons maintenant les intuitions exposées ci-dessus par l'intermédiaire de deux opérateurs. Ceux-ci déduisent les bornes sur un intervalle pour chacune des primitives terminales et les combinent judicieusement pour obtenir les valeurs extrêmes d'une primitive de haut niveau. Les minorations et les majorations sont obtenues à partir de n'importe quelle primitive de haut niveau. En particulier, elles s'appliquent directement aux contraintes en considérant que $false < true$. Notons que l'intervalle $[\varphi, \gamma]$ désigne l'ensemble des motifs $\{\theta \in \mathcal{L} | \varphi \preceq \theta \preceq \gamma\}$[13].

Définition 18 (Opérateurs de bornes) *Soit h une primitive de haut niveau et $[\varphi, \gamma]$ un intervalle, $\lfloor h \rfloor\langle\varphi, \gamma\rangle$ et $\lceil h \rceil\langle\varphi, \gamma\rangle$ sont définies de la manière suivante*[14] *:*

- *si* $\deg h = 0$ *: $\lfloor h \rfloor\langle\varphi, \gamma\rangle = h(\varphi)$ et $\lceil h \rceil\langle\varphi, \gamma\rangle = h(\gamma)$ ssi h est une fonction croissante. Sinon h décroît, $\lfloor h \rfloor\langle\varphi, \gamma\rangle = h(\gamma)$ et $\lceil h \rceil\langle\varphi, \gamma\rangle = h(\varphi)$.*
- *si* $\deg h \geq 1$ *: $\lfloor h \rfloor\langle\varphi, \gamma\rangle = p(h'_1, \ldots, h'_k)$ et $\lceil h \rceil\langle\varphi, \gamma\rangle = p(H'_1, \ldots, H'_k)$ où $p(h_1, \ldots, h_k)$ est la décomposition de h et pour chaque variable $i \in \{1, \ldots, k\}$:*

$$\begin{cases} h'_i = \lfloor h_i \rfloor\langle\varphi, \gamma\rangle \text{ et } H'_i = \lceil h_i \rceil\langle\varphi, \gamma\rangle & \text{si } p \text{ croît avec la } i^{ème} \text{ variable} \\ h'_i = \lceil h_i \rceil\langle\varphi, \gamma\rangle \text{ et } H'_i = \lfloor h_i \rfloor\langle\varphi, \gamma\rangle & \text{sinon} \end{cases}$$

Le théorème 1 ci-après justifiera le nom des opérateurs. À partir d'une primitive de haut niveau h et d'un intervalle $[\varphi, \gamma]$, ces opérateurs sont récursivement appliqués dans le but d'obtenir une minoration et une majoration de la valeur de h sur $[X, Y]$. Le tableau 5.1 donne la description des deux opérateurs de bornes correspondant à l'ensemble des primitives de $\mathcal{P}$ données dans le tableau 4.1 (page 52). Dans ce tableau 5.1, la notation générale E_i désigne un espace

[13] Ces intervalles correspondent à des sous-algèbres dans le cas particulier des motifs ensemblistes [Bucila *et al.*, 2002]. Dans [Kryszkiewicz, 2002], cette même structure est utilisée pour représenter les motifs fréquents.

[14] Pour alléger les notations, on remplace $\lfloor h \rfloor([\varphi, \gamma])$ et $\lceil h \rceil([\varphi, \gamma])$ par $\lfloor h \rfloor\langle\varphi, \gamma\rangle$ et $\lceil h \rceil\langle\varphi, \gamma\rangle$.

parmi $\mathfrak{B}$, $\Re^+$ ou $\mathcal{L}_\mathcal{I}$, et $\mathcal{E}_i$ les expressions associées (par exemple, l'ensemble des contraintes $\mathcal{Q}$ pour les booléens $\mathfrak{B}$). Notons que conformément à la définition 18 les fonctions du tableau sont regroupées suivant la monotonie de leurs variables.

$e \in \mathcal{E}_i$	Primitive(s)	$\lfloor e \rfloor\langle X, Y\rangle$	$\lceil e \rceil\langle X, Y\rangle$
$e_1 \theta e_2$	$\theta \in \{\wedge, \vee, +, \times, \cup, \cap\}$	$\lfloor e_1 \rfloor\langle X, Y\rangle \theta \lfloor e_2 \rfloor\langle X, Y\rangle$	$\lceil e_1 \rceil\langle X, Y\rangle \theta \lceil e_2 \rceil\langle X, Y\rangle$
$e_1 \theta e_2$	$\theta \in \{>, \geq, \supset, \supseteq, -, /, \setminus\}$	$\lfloor e_1 \rfloor\langle X, Y\rangle \theta \lceil e_2 \rceil\langle X, Y\rangle$	$\lceil e_1 \rceil\langle X, Y\rangle \theta \lfloor e_2 \rfloor\langle X, Y\rangle$
θe_1	$\theta \in \{\neg, freq, f, g\}$	$\theta \lceil e_1 \rceil\langle X, Y\rangle$	$\theta \lfloor e_1 \rfloor\langle X, Y\rangle$
$\theta(e_1.val)$	$\theta \in \{min\}$	$\theta(\lceil e_1 \rceil\langle X, Y\rangle.val)$	$\theta(\lfloor e_1 \rfloor\langle X, Y\rangle.val)$
$\theta(e_1)$	$\theta \in \{count\}$	$\theta \lfloor e_1 \rfloor\langle X, Y\rangle$	$\theta \lceil e_1 \rceil\langle X, Y\rangle$
$\theta(e_1.val)$	$\theta \in \{sum, max\}$	$\theta(\lfloor e_1 \rfloor\langle X, Y\rangle.val)$	$\theta(\lceil e_1 \rceil\langle X, Y\rangle.val)$
$c \in E_i$	-	c	c
$X \in \mathcal{L}_\mathcal{I}$	-	X	Y

TAB. 5.1 – La définition de $\lfloor . \rfloor$ et $\lceil . \rceil$ restreinte à un ensemble particulier de primitives.

Illustrons l'application de $\lfloor . \rfloor$ et $\lceil . \rceil$ sur la contrainte d'aire : comme $\geq$ croît dans $\mathfrak{B}$ selon sa première variable et décroît selon la seconde, nous avons $\lfloor area(X) \geq 6 \rfloor\langle X, Y\rangle = \lfloor area(X) \rfloor\langle X, Y\rangle \geq \lceil 6 \rceil\langle X, Y\rangle$ (cela correspond à la seconde ligne du tableau 5.1). Comme 6 est une constante et $\times$ croît suivant chacune de ses variables, nous obtenons respectivement que $\lceil 6 \rceil\langle X, Y\rangle = 6$ (ligne 7) et $\lfloor area(X) \rfloor\langle X, Y\rangle = \lfloor freq(X) \rfloor\langle X, Y\rangle \times \lfloor count(X) \rfloor\langle X, Y\rangle$ (ligne 1). Finalement, $\lfloor area(X) \geq 6 \rfloor\langle X, Y\rangle$ est égal à $freq(Y) \times count(X) \geq 6$ car *freq* décroît (ligne 3) et *count* croît (ligne 5). De la même manière, $\lceil area(X) \geq 6 \rceil\langle X, Y\rangle$ est égal à $freq(X) \times count(Y) \geq 6$.

Maintenant, nous présentons le résultat le plus important de ce chapitre qui sera fortement utilisé par la suite :

Théorème 1 (Bornes sur un intervalle) *$\lfloor h \rfloor$ et $\lceil h \rceil$ sont respectivement un minorant et un majorant de la primitive de haut niveau h. Étant donné un intervalle $[\varphi, \gamma]$ et un motif $\theta \in [\varphi, \gamma]$, on a $\lfloor h \rfloor\langle \varphi, \gamma\rangle \leq h(\theta) \leq \lceil h \rceil\langle \varphi, \gamma\rangle$.*

Nous commençons par définir le lemme 1 qui facilite la démonstration de ce théorème. Le raffinement des intervalles indiqué par ce lemme améliore la qualité du minorant et du majorant (i.e., le minorant croît et le majorant décroît).

Lemme 1 *Soient $h \in \mathcal{H}$ et $[\varphi_1, \gamma_1] \subseteq [\varphi_2, \gamma_2]$, on a $\lfloor h \rfloor\langle \varphi_1, \gamma_1\rangle \geq \lfloor h \rfloor\langle \varphi_2, \gamma_2\rangle$ et $\lceil h \rceil\langle \varphi_1, \gamma_1\rangle \leq \lceil h \rceil\langle \varphi_2, \gamma_2\rangle$.*

Preuve. Soit $h \in \mathcal{H}$ et soient $[X_1, Y_1] \subseteq [X_2, Y_2]$ deux intervalles. Tout d'abord, si $\deg h = 0$, on peut distinguer deux cas. Si h est une fonction croissante, comme on a $\lfloor h \rfloor\langle X, Y\rangle = h(X)$ et $\lceil h \rceil\langle X, Y\rangle = h(Y)$, on vérifie bien que $h(X_1) \geq h(X_2)$ et $h(Y_1) \leq h(Y_2)$. Avec une fonction décroissante, on peut conclure que l'hypothèse est également vraie. Deuxièmement, si $\deg h = n$, on fixe la décomposition de h à $p(h_1, \ldots, h_k)$. Supposons que pour tout h' telle que $\deg h' < n$, nous avons $\lfloor h' \rfloor\langle X_1, Y_1\rangle \geq \lfloor h' \rfloor\langle X_2, Y_2\rangle$ et $\lceil h' \rceil\langle X_1, Y_1\rangle \leq \lceil h' \rceil\langle X_2, Y_2\rangle$. Si p est une fonction croissante avec la $i^{\text{ème}}$ variable, la définition 18 assure que l'opérateur de minoration est encore appliqué sur la $i^{\text{ème}}$ opérande dans le but de calculer $\lfloor h \rfloor$. Comme on a $\lfloor h_i \rfloor\langle X_1, Y_1\rangle \geq \lfloor h_i \rfloor\langle X_2, Y_2\rangle$ par hypothèse, p est plus grand sur $[X_1, Y_1]$ que sur $[X_2, Y_2]$. Dans le cas contraire, quand p décroît avec la $i^{\text{ème}}$ variable, l'opérateur de majoration est appliqué pour

calculer $\lfloor h \rfloor$. Ainsi, p est plus grand sur $[X_1, Y_1]$ que sur $[X_2, Y_2]$ car $\lceil h_i \rceil \langle X_1, Y_1 \rangle \leq \lceil h_i \rceil \langle X_2, Y_2 \rangle$. Finalement, en appliquant cette approche sur toutes les variables $i \in \{1, \ldots, k\}$, nous obtenons que $\lfloor h \rfloor \langle X_1, Y_1 \rangle \geq \lfloor h \rfloor \langle X_2, Y_2 \rangle$. Dualement, on a aussi $\lceil h \rceil \langle X_1, Y_1 \rangle \leq \lceil h \rceil \langle X_2, Y_2 \rangle$. Ainsi, par récurrence, on conclut que le lemme 1 est correct. □

Pour le langage regroupant tous les intervalles (i.e., $\mathcal{L} \times \mathcal{L}$) et la relation de spécialisation $\subseteq$, le lemme 1 montre que le minorant $\lfloor q \rfloor$ et le majorant $\lceil q \rceil$ de q sont respectivement anti-monotone et monotone. Nous pouvons désormais prouver le théorème 1 :

Preuve. Soient q une PBC et θ un motif de $[\varphi, \gamma]$. Comme on a $[\theta, \theta] \subseteq [\varphi, \gamma]$, le lemme 1 donne que $\lfloor h \rfloor \langle \theta, \theta \rangle \geq \lfloor h \rfloor \langle \varphi, \gamma \rangle$ et $\lceil h \rceil \langle \theta, \theta \rangle \leq \lceil h \rceil \langle \varphi, \gamma \rangle$. Évidemment $\lfloor h \rfloor \langle \theta, \theta \rangle = \lceil h \rceil \langle \theta, \theta \rangle = h(\theta)$ et alors, on déduit que $\lfloor h \rfloor \langle \varphi, \gamma \rangle \leq h(\theta) \leq \lceil h \rceil \langle \varphi, \gamma \rangle$. Nous concluons que le théorème 1 est juste. □

Les opérateurs $\lfloor \cdot \rfloor$ et $\lceil \cdot \rceil$ donnent un minorant et un majorant pour n'importe quelle primitive de haut niveau. En particulier, le théorème 1 s'applique aux PBC. Par exemple, avec $Z \in [X, Y]$, l'évaluation de la contrainte d'aire pour le motif Z est comprise entre celle de $freq(Y) \times count(X) \geq 6$ et celle de $freq(X) \times count(Y) \geq 6$ (cf. ci-dessus pour le détail du calcul des bornes).

Ces bornes ne sont pas toujours atteintes, mais s'avèrent souvent efficaces pour approximer une contrainte (cf. la section 5.3.2). Typiquement, avec la mesure d'aire et l'intervalle $[AB, ABCD]$, on a $\lceil area(X) \rceil \langle AB, ABCD \rangle = freq(AB) \times count(ABCD) = 4 \times 4 = 16$ alors qu'aucun motif entre AB et $ABCD$ n'a une aire égale à 16.

Remarquons que la notion de monotonie sur les primitives (définition 15) est centrale pour l'obtention de ces bornes.

Dans la suite, les manipulations de contraintes sont majoritairement fondées sur ces deux opérateurs. Dans la section 5.3, ils sont utilisés pour identifier des contraintes monotones ou anti-monotones. Le chapitre 6 montre comment les utiliser pour déduire des contraintes monotones et anti-monotones depuis une PBC quelconque. Dans le chapitre 8, ils permettent d'obtenir des conditions d'élagages sur un intervalle.

5.2.3 Propriétés des opérateurs de bornes

En pratique, les bornes $\lfloor q \rfloor$ et $\lceil q \rceil$ ne sont pas coûteuses à calculer. Pour le calcul de l'une d'entre elles avec h, il suffit de parcourir l'intégralité de l'arbre de son expression (cf. la section 5.1.2). Au pire si ce dernier est complet, on a donc $2^{\deg h + 1} - 1$ appels récursifs. En outre, ce calcul peut être effectué une unique fois pour être ensuite appliqué à différents intervalles.

Les opérateurs de minoration et de majoration satisfont des propriétés de linéarité et de dualité par rapport aux opérateurs booléens :

Propriété 6 (Linéarité) *Les opérateurs de bornes sont linéaires par rapport aux opérateurs booléens $\wedge$ et $\vee$ i.e., pour deux PBC q_1 et q_2 avec $\theta \in \{\wedge, \vee\}$, on a :*

$$\begin{cases} \lfloor q_1 \theta q_2 \rfloor \equiv \lfloor q_1 \rfloor \theta \lfloor q_2 \rfloor \\ \lceil q_1 \theta q_2 \rceil \equiv \lceil q_1 \rceil \theta \lceil q_2 \rceil \end{cases}$$

Preuve. La preuve est immédiate en considérant que $\wedge$ et $\vee$ sont des fonctions croissantes suivant chacune de leurs variables et la définition 18. □

Cette linéarité implique que l'utilisation des bornes sur une PBC prend en considération chacune de ses sous-contraintes atomiques. Dans la suite, les traitements prennent donc en

compte naturellement les combinaisons de contraintes et évite d'avoir à déployer des stratégies particulières comme c'est le cas pour la plupart des méthodes des bases de données inductives. En fait, les opérateurs ont un comportement linéaire avec toutes les fonctions croissantes suivant chacune de leurs variables.

Par ailleurs, les définitions des opérateurs $\lfloor . \rfloor$ et $\lceil . \rceil$ sont intimement liées :

Propriété 7 (Dualité) *Les opérateurs de bornes sont duaux i.e., pour toute PBC q, on a :*

$$\neg\lfloor q \rfloor = \lceil \neg q \rceil$$

Preuve. La preuve est immédiate en considérant que $\neg$ est une fonction décroissante et la définition 18. □

La dualité des opérateurs explique la symétrie entre $\lfloor q \rfloor\langle\varphi, \gamma\rangle$ et $\lceil q \rceil\langle\varphi, \gamma\rangle$: il suffit d'intervertir le φ et le γ pour passer de l'un à l'autre. Cela se vérifie bien avec le calcul des bornes de la contrainte d'aire où $\lceil area(X) \geq 6 \rceil\langle X, Y\rangle \equiv freq(X) \times count(Y) \geq 6$ s'obtient aussi en calculant $\lfloor area(X) \geq 6 \rfloor\langle Y, X\rangle$. La dualité des opérateurs facilite de nombreuses démonstrations.

5.3 Détection de la monotonie d'une contrainte

5.3.1 Problématique

La découverte de propriétés de monotonie pour une contrainte n'est pas une tâche abordée par la littérature. En fait, la plupart des travaux se base sur un dictionnaire de contraintes atomiques dont on connaît déjà les propriétés [Ng *et al.*, 1998, Bonchi et Lucchese, 2005]. Dans le PBF, où les contraintes sont formulées plus librement, cette tâche a toute son importance. De manière naïve, si une contrainte s'avère monotone ou anti-monotone, il est alors possible d'utiliser un solveur usuel. Dans une première approche [Soulet et Crémilleux, 2005c], nous avons proposé de déduire des propriétés de monotonie sur l'ensemble d'une formule booléenne de contraintes atomiques. Néanmoins, cette approche bénéficie seulement des propriétés de monotonies individuelles de chaque contrainte atomique. Nous lui préférerons donc la méthode de relaxation du chapitre 6 bien plus générale.

Notre principal intérêt est en fait d'identifier des contraintes qui nous le verrons ultérieurement possèdent de bonnes propriétés. En effet, les bornes issues des opérateurs de minoration et majoration sur les contraintes monotones et anti-monotones détectées offrent des conditions nécessaires et suffisantes. Cela provient ainsi d'une première utilisation des opérateurs de bornes.

5.3.2 Opérateurs de détection de la monotonie

Nous avions souligné à la section 2.2.1 que les contraintes anti-monotones et monotones étaient respectivement des fonctions décroissantes et croissantes. Cela se vérifie à la fois pour les contraintes d'agrégat et les contraintes syntaxiques. Notre objectif est donc de déduire l'éventuelle croissance ou décroissance d'une contrainte. Remarquons que sur l'intervalle $[\gamma, \varphi]$, la valeur d'une contrainte anti-monotone q_{AM} est minorée par $q_{AM}(\varphi)$ puisque q_{AM} est décroissante. Nous formalisons cette approche en définissant deux opérateurs qui testent la croissance ou la décroissance d'une contrainte :

Définition 19 (Opérateurs de détection de l'(anti-)monotonie) *Les opérateurs de détection de la monotonie et de l'anti-monotonie respectivement dénotés par $\lceil . \rceil^M$ et $\lfloor . \rfloor^M$, sont définis pour toute PBC q comme suit :*

$$\lfloor q \rfloor^M = \begin{cases} true, & si\ \lfloor q \rfloor \langle \gamma, \varphi \rangle\ est\ équivalent\ à\ q(\varphi) \\ false, & sinon \end{cases}$$

$$\lceil q \rceil^M = \begin{cases} true, & si\ \lceil q \rceil \langle \gamma, \varphi \rangle\ est\ équivalent\ à\ q(\varphi) \\ false, & sinon \end{cases}$$

Ces opérateurs sont donc deux prédicats. Lorsque leur réponse est positive, la contrainte est monotone ou anti-monotone :

Théorème 2 (Détection de la monotonie) *Une contrainte basée sur des primitives satisfaisant $\lfloor q \rfloor^M$ (resp. $\lceil q \rceil^M$) est anti-monotone (resp. monotone).*

Preuve. Soit q une contrainte telle que $\lfloor q \rfloor \langle \gamma, \varphi \rangle \equiv q(\varphi)$. Soient γ et φ tels que $\gamma \preceq \varphi$ et $q(\varphi) = true$. Comme on a $\lfloor q \rfloor \langle \gamma, \varphi \rangle = q(\varphi)$ et que $q(\varphi) = true$, tous les motifs de $[\gamma, \varphi]$ satisfont la contrainte donc $q(\gamma) = true$ et q est anti-monotone. Par dualité des opérateurs (propriété 7), on conclut que le théorème 2 est correct. □

Par exemple, comme $\lfloor freq(X) \geq minfr \rfloor^M$ est vrai, la contrainte de fréquence minimale est bien détectée comme une contrainte anti-monotone. Observons que $\lfloor 2 \times count(X) - count(X) \leq \rho \rfloor^M$ est faux même si la contrainte $2 \times count(X) - count(X) \leq \rho$ est anti-monotone. Cela montre que la réciproque du théorème 2 est fausse car les bornes issues des opérateurs $\lfloor . \rfloor$ et $\lceil . \rceil$ sont approximatives.

Dans la suite, l'ensemble des contraintes anti-monotones détectables $\lfloor \mathcal{Q} \rfloor^M$ (resp. monotones détectables $\lceil \mathcal{Q} \rceil^M$) regroupe toutes les contraintes satisfaisant $\lfloor . \rfloor^M$ (resp. $\lceil . \rceil^M$). Ces deux ensembles partagent certaines caractéristiques avec les contraintes monotones. La conjonction de deux contraintes (anti-)monotones détectables est encore (anti-)monotone détectable. La négation d'une contrainte anti-monotone détectable est monotone détectable et réciproquement. Ces propriétés découlent directement de la linéarité (propriété 6) et de la dualité (propriété 7) des opérateurs de bornes.

Remarquons que la minoration et la majoration issues des opérateurs sont exactes avec les contraintes (anti-)monotones détectables :

Propriété 8 *Soient $q \in \lfloor \mathcal{Q} \rfloor^M \cup \lceil \mathcal{Q} \rceil^M$ et $[\varphi, \gamma]$ un intervalle, les valeurs $\lfloor q \rfloor \langle \varphi, \gamma \rangle$ et $\lceil q \rceil \langle \varphi, \gamma \rangle$ sont atteintes sur cet intervalle.*

Preuve. Soient $q \in \lfloor \mathcal{Q} \rfloor^M$ et $[\varphi, \gamma]$ un intervalle, on a $\lfloor q \rfloor \langle \varphi, \gamma \rangle \equiv q(\gamma)$ (cf. définition 19). La propriété de dualité (propriété 7) donne que $\lceil q \rceil \langle \varphi, \gamma \rangle = \lfloor q \rfloor \langle \gamma, \varphi \rangle$. Or $\lfloor q \rfloor \langle \gamma, \varphi \rangle = q(\varphi)$, on déduit que $\lceil q \rceil \langle \varphi, \gamma \rangle = q(\varphi)$. De la même manière, on montre que les bornes sont bien atteintes pour une contrainte monotone détectable. □

En d'autres termes, si q est une contrainte (anti-)monotone détectable, alors on a θ_1 et θ_2 appartenant à l'intervalle $[\varphi, \gamma]$ tels que $\lfloor q \rfloor \langle \varphi, \gamma \rangle = q(\theta_1)$ et $\lceil q \rceil \langle \varphi, \gamma \rangle = q(\theta_2)$. Typiquement, cela se vérifie pour la contrainte minimale de fréquence car $\lfloor freq(X) \geq minfr \rfloor \langle X, Y \rangle = q(Y)$ et $\lceil freq(X) \geq minfr \rceil \langle X, Y \rangle = q(X)$.

Nous reviendrons sur ce résultat car il assure l'optimalité de plusieurs méthodes d'extraction pour les contraintes (anti-)monotones détectables (cf. les sections 6.3.3 et 8.1.1).

Chapitre 6

Extraction de motifs par relaxation

Sommaire

L'objectif de ce chapitre est de proposer une méthode d'extraction de tous les motifs vérifiant une PBC et présents dans une base de données et ce, pour un langage quelconque. Vu le spectre large des contraintes du PBF et le fait que celles-ci dépassent les classes de contraintes usuelles, il n'est pas possible d'utiliser directement des solveurs de ces classes. Une approche "naïve" d'extraction serait d'énumérer tous les motifs du langage $\mathcal{L}$ apparaissant au moins une fois dans $\mathbf{r}$ et de conserver uniquement ceux qui satisfont la contrainte désirée. Comme nous l'avons déjà mentionné à la section 2.1.3, cette approche échoue dès que le jeu de données est conséquent. Plutôt que de chercher à concevoir un solveur dédié, nous proposons dans ce chapitre de réutiliser les nombreux algorithmes existants spécialisés dans l'extraction de contraintes monotones ou anti-monotones. Pour cela, nous suggérons d'approximer la contrainte originale d'extraction q par une contrainte q' de sorte que (1) aucun motif satisfaisant q ne satisfasse pas q' et (2) q' soit monotone ou anti-monotone. Puis dans un deuxième temps, les motifs vérifiant q' sont filtrés pour ne conserver que ceux satisfaisant q.

La section 6.1 détaille l'approche de relaxation et précise nos deux points forts : généricité et automaticité du processus. Elle donne brièvement les intuitions de notre approche de relaxation. L'idée clé est de tirer profit des spécificités de la base de données $\mathbf{r}$ résumées en deux motifs virtuels présentés à la section 6.2. Puis, la section 6.3 relie ces motifs virtuels au PBF afin d'obtenir automatiquement une relaxation monotone et une relaxation anti-monotone.

6.1 Extraction de motifs : approche de la relaxation de contraintes

6.1.1 Problématique de la relaxation

Nous souhaitons approximer la théorie de la contrainte originale q par une collection de motifs plus large correspondant à la théorie d'une contrainte plus lâche $q' : \mathcal{T}h(\mathcal{L}, \mathbf{r}, q) \subseteq \mathcal{T}h(\mathcal{L}, \mathbf{r}, q')$. La contrainte moins restrictive q' induite de q, est appelée une *relaxation* et satisfait l'implication $q \Rightarrow q'$. La figure 6.1 schématise la théorie de q (i.e., la forme grise) et sa relaxation q' (i.e., la forme hachurée).

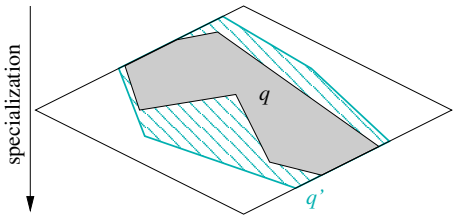


FIG. 6.1 – Une relaxation de la contrainte q.

L'idée clé est d'obtenir une relaxation vérifiant une propriété de monotonie dans le but de pouvoir réutiliser les algorithmes usuels comme celui par niveaux. Plus précisément, **étant donné un langage $\mathcal{L}$, une base de données $\mathbf{r}$ et une contrainte q, nous souhaitons obtenir automatiquement une relaxation monotone et une relaxation anti-monotone de q**. En effet, des algorithmes efficaces existent pour de telles contraintes (cf. les chapitres 2 et 3). Ils permettent alors d'obtenir les théories associées aux relaxations anti-monotone q_{AM} et monotone q_M de la contrainte q. Ces théories peuvent être extraites car elles sont souvent très restreintes par rapport à l'ensemble de tous les motifs présents dans le jeu de données. A partir de ces théories, un simple filtrage sélectionne alors les motifs satisfaisant q. Une telle approche est une méthode d'optimisation qui préserve la découverte [Bayardo, 2005] puisque l'élagage issu de la relaxation ne rejette pas de motifs satisfaisant q.

$\mathcal{D}$

Trans.	Items					
t_1	A	B			E	F
t_2	A				E	
t_3	A	B	C	D		
t_4	A	B	C	D	E	
t_5				D	E	
t_6			C			F

Item	A	B	C	D	E	F
val	50	30	75	10	30	15

TAB. 6.1 – Une base de données $\mathbf{r}$ constituée d'un contexte transactionnel $\mathcal{D}$ et d'une table de valeurs.

La figure 6.2 illustre l'efficacité de la relaxation en revenant sur notre exemple de la contrainte d'aire minimale. Admettons que nous soyons intéressés par tous les motifs ensemblistes présents dans le jeu de données $\mathcal{D}$ (cf. le tableau 6.1) dont l'aire excède 6 (motifs en gras). L'approche naïve requiert d'extraire $\mathcal{T}h(\mathcal{L}_\mathcal{I}, \mathbf{r}, freq(X) \geq 1)$ soit 40 motifs correspondant à l'intégralité du treillis puis de les filtrer. Nous verrons (cf. la section 6.3) que la contrainte anti-monotone

$freq(X) \geq 2$ et la contrainte monotone $count(X) \geq 2$ sont des relaxations de $area(X) \geq 6$. Ces relaxations offrent trois stratégies d'extraction différentes : $\mathcal{T}h(\mathcal{L}_\mathcal{I}, \mathbf{r}, freq(X) \geq 2)$ (21 motifs, à gauche), $\mathcal{T}h(\mathcal{L}_\mathcal{I}, \mathbf{r}, count(X) \geq 2)$ (34 motifs, à droite) et $\mathcal{T}h(\mathcal{L}_\mathcal{I}, \mathbf{r}, freq(X) \geq 2 \wedge count(X) \geq 2)$ (15 motifs). La théorie de $area(X) \geq 6$ a en fait 8 motifs à savoir AB, AE, ABC, ABD, ABE, ACD, BCD et $ABCD$. La meilleure stratégie basée sur la conjonction des deux relaxations, décrite dans ce chapitre, conduit à seulement 7 motifs superflus.

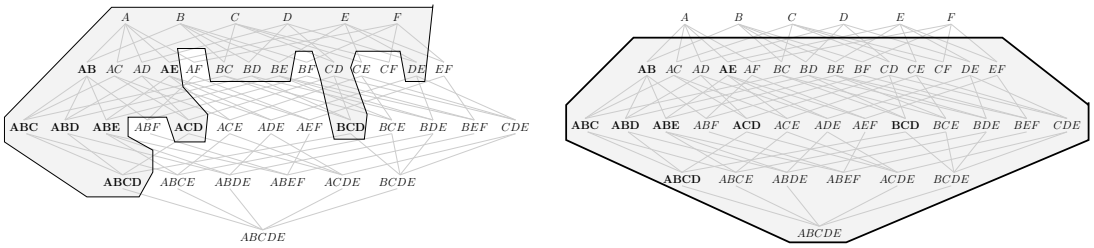


FIG. 6.2 – Une relaxation anti-monotone (à gauche) et une relaxation monotone (à droite) de la contrainte $area(X) \geq 6$ (les motifs en gras satisfont cette contrainte).

6.1.2 Méthodes de relaxation existantes

Dans notre contexte, la relaxation est une méthode d'optimisation pour rendre faisable certaines extractions de contraintes complexes et peut être vue comme une forme de pré-traitement. Dans la littérature, la relaxation est aussi utilisée pour découvrir des motifs plus inattendus [Antunes et Oliveira, 2004] ou introduire de la souplesse afin d'éviter une sélection trop binaire (effet "crisp") [Bistarelli et Bonchi, 2005]. Par ailleurs, le concept de relaxation est astucieusement exploité pour extraire des motifs ensemblistes contraints dans de larges jeux de données grâce à la transposition [Jeudy et Rioult, 2004].

À travers la littérature, le difficile problème de la relaxation est partiellement résolu dans des cas spécifiques. Les contraintes basées sur des expressions régulières sont relaxées en des contraintes anti-monotones pour extraire des séquences [Garofalakis *et al.*, 1999]. Deux contraintes basées sur le χ^2 et la corrélation sont relaxées pour trouver des ensembles [Morishita et Sese, 2000]. Dans le domaine des ensembles, une large collection de formules booléennes de contraintes monotones [Soulet et Crémilleux, 2005c] et de contraintes d'agrégats [Wang *et al.*, 2005] peuvent être relaxées. Comparée à toutes ces approches, ce chapitre apporte une réponse plus générique en relaxant n'importe quelle PBC et pour n'importe quel langage. Par ailleurs, notre méthode est complètement automatisable.

6.1.3 Intuitions clés

Illustrons les intuitions de notre approche avec le cas particulier de la contrainte d'aire minimale $freq(X) \times count(X) \geq 6$. L'idée clé est d'observer le comportement de l'aire avec les motifs les plus courts et les motifs les plus longs. Etant donnée une constante l, on remarque que $freq(X) \times l \geq 6$ est anti-monotone. La difficulté est de fixer l de sorte que la contrainte d'aire minimale implique la satisfaction de $freq(X) \times l \geq 6$. Pour cela, il est nécessaire de choisir l de sorte que $count(X) \leq l$ pour tous les motifs X. Comme les motifs extraits doivent être présents au moins une fois dans le jeu de données $\mathcal{D}$, la taille du plus long motif est celle de la plus longue transaction. Ainsi avec le contexte $\mathcal{D}$, l peut être fixé à 5 et on obtient la relaxation anti-monotone

$freq(X) \geq 6/5$. Similairement, comme tous les motifs de $\mathcal{D}$ ont une fréquence inférieure à 4, la contrainte $count(X) \geq 6/4$ est une relaxation monotone. Les valeurs significatives (5 pour la relaxation anti-monotone et 4 pour la relaxation monotone) sont seulement déduites des spécificités extrêmes de $\mathcal{D}$. Dans la section suivante, nous montrons comment les rassembler au sein de deux motifs virtuels. La section 6.3 généralise alors ces intuitions à toute PBC.

6.2 Motifs virtuels d'un espace des versions

6.2.1 Définition

Cette section introduit deux motifs artificiels qui résument un espace des versions. Ces motifs sont dits virtuels car ils ont des propriétés inattendues.

Rappelons qu'un espace des versions [Mitchell, 1982] est une collection convexe de motifs (cf. la section 3.1.2). Un tel espace correspond exactement à la théorie de la conjonction d'une contrainte monotone et d'une contrainte anti-monotone. Afin de mieux comprendre le rôle des motifs virtuels, nous rappelons les principales caractéristiques des espaces des versions introduits à la section 3.1. En terme d'intervalles, un ensemble $VS \subseteq \mathcal{L}$ est un espace des versions ssi pour tout $\varphi \in VS$ et $\gamma \in VS$ tels que $\varphi \preceq \gamma$, on a $[\varphi, \gamma] \subseteq VS$. Par exemple, la collection des motifs présents au moins une fois dans la base de données est un espace des versions (dès que cette collection est finie). En effet, si un motif φ et un motif γ sont présents dans la base de données $\mathbf{r}$, on peut vérifier qu'un motif θ compris entre φ et γ est aussi présent dans $\mathbf{r}$. Dans la suite, cette collection, dénotée par $\mathcal{C}$, est souvent utilisée.

Avant d'étendre la relation de spécialisation aux motifs virtuels, nous donnons leur définition :

Définition 20 (Motifs virtuels le plus général et le plus spécifique) *Soit VS un espace des versions, le motif virtuel le plus général $\bot(VS)$ et le motif virtuel le plus spécifique $\top(VS)$ sont définis comme suit pour chaque fonction $p : \mathcal{L} \rightarrow S$ (où S est un ensemble totalement ordonné) :*

$$p(\bot(VS)) = \begin{cases} \min_{\varphi \in VS} p(\varphi), & \textit{si } p \textit{ est une fonction croissante} \\ \max_{\varphi \in VS} p(\varphi), & \textit{si } p \textit{ est une fonction décroissante} \end{cases}$$

$$p(\top(VS)) = \begin{cases} \max_{\varphi \in VS} p(\varphi), & \textit{si } p \textit{ est une fonction croissante} \\ \min_{\varphi \in VS} p(\varphi), & \textit{si } p \textit{ est une fonction décroissante} \end{cases}$$

Les motifs virtuels synthétisent les spécificités de la base de données $\mathbf{r}$ au regard des différentes primitives. En effet, tout motif de l'espace des versions a une valeur pour une primitive p comprise entre les valeurs de p prises pour les motifs virtuels. De manière duale, le motif le plus général et le motif le plus spécifique sont aussi respectivement appelés le motif le moins spécifique et le motif le moins général.

Les motifs virtuels $\bot(VS)$ et $\top(VS)$ dépendent seulement de l'espace des versions VS et de la base de données $\mathbf{r}$. En particulier, les valeurs de sum, min et max pour chaque motif virtuel sont liées à celles des tables de valeurs. Par exemple, considérons les motifs virtuels pour les motifs ensemblistes présents dans le contexte $\mathcal{D}$ i.e., les motifs ensemblistes virtuels de la collection $\mathcal{C}$ (où $\mathcal{C} = \mathcal{T}h(\mathcal{L}_{\mathcal{I}}, \mathbf{r}, freq(X) \geq 1)$). Nous avons $count(\bot(\mathcal{C})) = \min_{X \in \mathcal{C}} count(X)$ car la longueur est une fonction croissante sur $\mathcal{L}_{\mathcal{I}}$. Comme la longueur des motifs les plus courts est égale à 1[15], nous obtenons que $count(\bot(\mathcal{C})) = 1$. Un raisonnement similaire conduit aux résultats présentés dans le tableau 6.2 à gauche.

[15] Le motif nul est exclu de $\mathcal{L}_{\mathcal{I}}$ (cf. la page 19).

Primitive p	$p(\bot(\mathcal{C}))$	$p(\top(\mathcal{C}))$
$freq$	4	1
$count$	1	5
sum	10	185
min	75	10
max	10	75
...	...	...

Primitive p	$p(\emptyset)$	$p(\mathcal{I})$
$freq$	6	0
$count$	0	6
sum	-	210
min	-	10
max	-	75
...	...	...

TAB. 6.2 – Motifs virtuels et motifs $\emptyset$ et $\mathcal{I}$ associés aux contexte $\mathcal{D}$.

Dans la suite, la *fermeture* d'un espace des versions VS, dénotée $\overline{VS}$, correspond à l'espace des versions VS complété avec ses motifs virtuels le plus général et le plus spécifique : $\overline{VS} = VS \cup \{\bot(VS), \top(VS)\}$. La relation de spécialisation $\preceq$ est alors étendue à $\overline{VS}$ en considérant que le motif virtuel le plus général $\bot(VS)$ est une généralisation de tous les motifs de $\overline{VS}$. Le motif virtuel le plus spécifique $\top(VS)$ est une spécialisation de tous les motifs de $\overline{VS}$. Tout motif de l'espace des versions VS est alors compris entre $\bot(VS)$ et $\top(VS)$ (i.e., $VS \subseteq [\bot(VS), \top(VS)]$). De même, notons que $\overline{VS} = [\bot(VS), \top(VS)]$.

Nous montrons maintenant les propriétés inattendues des motifs virtuels en prenant l'exemple des motifs ensemblistes. $\bot(\mathcal{C})$ est plus général que chaque item (e.g., $\bot(\mathcal{C}) \subseteq A$ et $\bot(\mathcal{C}) \subseteq B$). Comme $A \cap B = \emptyset$, on en déduit alors que $\bot(\mathcal{C})$ devrait être l'ensemble vide. Or la définition de $\bot(\mathcal{C})$ conduit à ce que sa longueur (i.e., sa cardinalité) soit égale à 1. De la même manière, chaque transaction de la base de données $\mathcal{D}$ est inclue dans $\top(\mathcal{C})$ i.e., $ABEF \subseteq \top(\mathcal{C})$, $AE \subseteq \top(\mathcal{C})$, $ABCD \subseteq \top(\mathcal{C})$, etc. Donc, on s'attend ainsi à ce que $\top(\mathcal{C})$ soit égal à $ABCDEF$, mais sa définition en fournit une longueur de 5. Les mêmes observations peuvent être faites sur les séquences. Pour la tâche de relaxation, nous verrons que ce sont ces propriétés inattendues qui permettent de cerner au mieux les motifs à représenter. En effet, pour le même contexte, les motifs $\emptyset$ et $\mathcal{I}$ ont des valeurs plus éloignées (voire indéfinies) pour les différentes primitives (cf. le tableau 6.2). Par exemple, la valeur $sum(\top(\mathcal{C}).val)$ (i.e., 185) est bien atteinte par le motif $ABCDE$, mais aucun motif de $\mathcal{C}$ n'atteint 210 correspondant à $sum(\mathcal{I}.val)$.

Dans [Raedt et Kramer, 2001], les auteurs introduisent aussi l'élément artificiel $\top$ pour trouver des fragments moléculaires, mais ce dernier a des propriétés attendues.

6.2.2 Propriétés des motifs virtuels

Cette section dégage deux propriétés importantes des motifs virtuels. En particulier, le calcul des motifs virtuels d'un espace des versions peut s'effectuer directement à partir des bordures de cet espace.

Un espace des versions peut être représenté par ses deux bordures G et S (cf. section 3.2.2). Tout motif d'un espace des versions est donc compris entre un élément de G et un élément de S. D'un point de vue abstrait, comme les deux bordures G et S résument l'espace des versions, les motifs virtuels de cet espace des versions doivent pouvoir découler des bordures. La propriété 9 établit ce lien :

Propriété 9 (Résumé des bordures) *Les motifs virtuels le plus général et le plus spécifique d'un espace des versions VS sont respectivement égaux au motif virtuel le plus général de la bordure $G(VS)$ et au motif virtuel le plus spécifique de la bordure $S(VS)$.*

Preuve. Soit $p : \mathcal{L} \rightarrow S$ une fonction croissante. Pour chaque motif $\varphi \in VS$, il y a $\gamma \in G(VS)$ tel que $\gamma \preceq \varphi$. Comme p croît, on obtient que $p(\gamma) \leq p(\varphi)$ et $\min_{\gamma \in G(VS)} p(\gamma) \leq \min_{\varphi \in VS} p(\varphi)$. Comme $G(VS) \subseteq VS$, nous concluons que $\min_{\varphi \in VS} p(\varphi) = \min_{\gamma \in G(VS)} p(\gamma)$. Les trois autres relations se prouvent avec un raisonnement similaire. □

La propriété 9 se traduit formellement par les deux relations suivantes : $\bot(VS) = \bot(G(VS))$ et $\top(VS) = \top(S(VS))$. Cette propriété assure que les motifs qui appartiennent aux bordures sont suffisants pour calculer les motifs virtuels. En pratique, au lieu d'énumérer tous les motifs d'un espace des versions VS, le calcul de $\bot(VS)$ et $\top(VS)$ s'effectue à partir des motifs contenus dans les bordures $G(VS)$ et $S(VS)$. Comme ces bordures sont réduites, le calcul des motifs virtuels est vraiment efficace. Par exemple, pour le contexte transactionnel $\mathcal{D}$, $\bot(\mathcal{C})$ est obtenu à partir des items (i.e., 6 éléments) et $\top(\mathcal{C})$, à partir des seules transactions (i.e., 6 éléments). D'autre part, ces relations soulignent que le motif virtuel le plus général (resp. spécifique) résume les connaissances les plus générales (resp. spécifiques) au sens de la relation $\preceq$. Les motifs virtuels forment une représentation d'un espace des versions plus condensée que celle issue des deux bordures. Bien que cette représentation perde certaines informations contenues par les bordures, elle en conserve l'essentiel pour le problème de relaxation comme nous le verrons.

Nous souhaitons maintenant montrer qu'on ne peut pas obtenir de meilleurs motifs virtuels avec la même approche (i.e., dont les spécificités encadrent mieux celles des motifs de VS) en en recalculant d'autres à partir de l'espace $\overline{VS}$. Pour cela, nous isolons le cas particulier où les motifs virtuels sont bien "réels" :

Propriété 10 *Si la bordure $G(VS)$ contient un seul motif φ, alors le motif virtuel le plus général de l'espace des versions VS est exactement φ. Similairement, si la bordure $S(VS)$ contient un seul motif φ, alors le motif virtuel le plus spécifique de l'espace des versions VS est exactement φ.*

Preuve. Soit $G(VS)$ égale à $\{\varphi\}$. Soit $p : \mathcal{L} \rightarrow S$ une fonction croissante. Comme $p(\bot(VS))$ est égal à $\min_{\varphi \in G(VS)} p(\varphi)$, on obtient $p(\bot(VS)) = p(\varphi)$. Le même résultat s'obtient avec une fonction décroissante p. De cette manière, $\bot(VS) = \varphi$. L'autre relation se prouve avec un raisonnement similaire. □

En d'autres termes, la propriété 10 signifie que si une bordure de VS est réduite à un unique élément, ce motif correspond à un motif virtuel de VS. Typiquement, si l'ensemble nul avait fait parti du langage des motifs ensemblistes, il aurait constitué à lui seul la bordure des motifs minimaux. Ainsi, on aurait eu $\bot(\mathcal{C}) = \emptyset$. Par ailleurs, comme les bordures de $G(\overline{VS})$ et $S(\overline{VS})$ correspondent respectivement à $\{\bot(VS)\}$ et $\{\top(VS)\}$, les motifs virtuels de la fermeture d'un espace des versions sont égaux à ceux de l'espace des versions i.e., on obtient que $\overline{\overline{VS}} = [\bot(\overline{VS}), \top(\overline{VS})] = \overline{VS}$. L'usage des motifs virtuels $\bot(\overline{VS})$ et $\top(\overline{VS})$ n'apporterait donc rien de plus que celui des motifs virtuels $\bot(VS)$ et $\top(VS)$ car ils sont identiques. Ainsi, dans la section 6.3.1, nous utilisons les motifs virtuels $\bot(VS)$ et $\top(VS)$.

6.2.3 Intégration des motifs virtuels au PBF

Cette section montre que les motifs virtuels le plus général et le plus spécifique peuvent naturellement être intégrés dans le PBF même si ces derniers ont été définis séparément. En d'autres termes, les sections suivantes utilisent des intervalles délimités par des motifs virtuels. L'utilisation de $\overline{VS}$ comme espace de recherche permet de retrouver la théorie des motifs dans l'espace des versions original VS.

Les sections suivantes manipulent les motifs virtuels comme un artifice de calcul dans $\overline{VS}$, mais leur but final est de trouver la théorie associée à l'espace des versions originel VS. La propriété suivante relie l'espace des versions à sa fermeture :

Propriété 11 *Une primitive monotone p sur VS est une primitive monotone sur $\overline{VS}$.*

Preuve. Soit $p : \mathcal{L} \rightarrow S$ une primitive croissante sur VS. Soit φ et γ deux motifs de $\overline{VS}$ tel que $\varphi \prec \gamma$. Premièrement, si φ et γ appartiennent à VS, $p(\varphi)$ est alors plus petit que $p(\gamma)$ car p croît sur VS. Deuxièmement, si φ appartient à VS, γ correspond à $\top(VS)$. Alors, comme $p(\top(VS)) = \max_{\theta \in VS} p(\theta)$, nous obtenons que $p(\gamma) \geq p(\varphi)$. Sinon, φ correspond à $\bot(VS)$, φ minimalise $p : p(\varphi) = \min_{\theta \in VS} p(\theta)$. De cette manière, comme γ est un simple motif de VS ou maximise p (en étant $\top(VS)$), on a $p(\gamma) \geq p(\varphi)$. On conclut que p est une fonction croissante sur $\overline{VS}$. Une preuve similaire montre que le résultat est également correct avec une primitive décroissante. □

Toute primitive p définie sur un espace des versions VS peut être étendue sur la fermeture $\overline{VS}$. En plus, par définition, l'évaluation de cette primitive p coïncide sur les deux espaces pour tout motif de VS. Ainsi, la théorie de la contrainte q dans l'espace des versions VS correspond exactement à la théorie de q dans la fermeture $\overline{VS}$ en excluant les deux motifs virtuels $\bot(VS)$ et $\top(VS)$: $\mathcal{T}h(VS, \mathbf{r}, q) = \mathcal{T}h(\overline{VS}, \mathbf{r}, q) \backslash \{\bot(VS), \top(VS)\}$.

6.3 Déduction de relaxations monotones et anti-monotones

6.3.1 Approche de relaxation

Cette section associe les motifs virtuels à l'opérateur de majoration (cf. le chapitre 5) afin d'obtenir des relaxations monotones ou anti-monotones de PBC qui serviront à l'élagage.

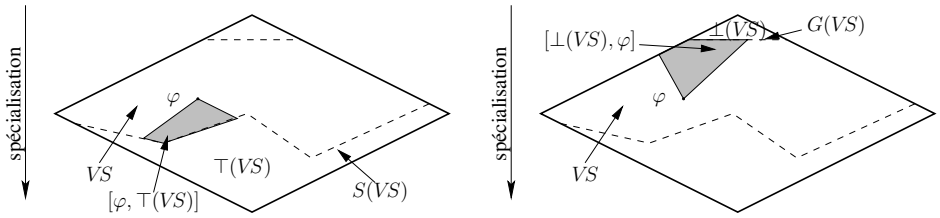


FIG. 6.3 – Représentations des intervalles $[\bot(VS), \varphi]$ et $[\varphi, \top(VS)]$ sur le treillis des motifs ensemblistes complété des motifs virtuels.

Le principe de la relaxation est fondé sur le résumé issu des motifs virtuels pour associer la contrainte à tous les motifs présents dans la base de données. Intuitivement, pour n'importe quel motif φ, nous supposons que toutes ses spécialisations possèdent les spécificités les plus favorables en vue de satisfaire la contrainte (comme nous l'avons effectué à la section 6.1.3). Dans de telles conditions, si aucun motif plus spécifique que φ ne peut satisfaire la contrainte, la relaxation anti-monotone retourne $false$. La relaxation monotone fonctionne sur un principe analogue, mais sur les généralisations.

Nous formalisons ces intuitions par le biais des motifs virtuels et de l'opérateur de majoration. Comme les motifs désirés doivent être présents dans la base de données, ils appartiennent tous à la collection $\mathcal{C}$. Nous rappelons qu'à chaque fois que cette collection contient un nombre fini

de motifs, ce dernier est un espace des versions. De cette manière, nous pouvons considérer sa fermeture $\overline{\mathcal{C}}$. En montrant que les motifs virtuels s'intègrent au PBF, la section précédente nous permet d'appliquer les opérateurs de minoration et majoration avec les intervalles de $\overline{\mathcal{C}}$, en particulier ceux délimités par un motif de $\mathcal{C}$ et un motif virtuel. Ainsi, toutes les généralisations (resp. spécialisations) d'un motif φ sont décrites par l'intervalle $[\bot(\mathcal{C}), \varphi]$ (resp. $[\varphi, \top(\mathcal{C})]$) comme le montre la figure 6.3. Alors, nous définissons les deux contraintes suivantes (remarquons que celles-ci sont des PBC) :

$$\begin{cases} \lceil q \rceil^{\bot}\langle\varphi\rangle \equiv \lceil q \rceil\langle\bot, \varphi\rangle \\ \lceil q \rceil^{\top}\langle\varphi\rangle \equiv \lceil q \rceil\langle\varphi, \top\rangle \end{cases}$$

Comme nous nous focalisons sur l'espace des versions $\mathcal{C}$, les motifs virtuels $\bot$ et $\top$ se réfèrent respectivement à $\bot(\mathcal{C})$ et $\top(\mathcal{C})$. Le théorème 3 justifie que les opérateurs $\lceil\bullet\rceil^{\bot}$ et $\lceil\bullet\rceil^{\top}$ sont nommés opérateurs de relaxation monotone et anti-monotone. En effet, il prouve que $\lceil q \rceil^{\bot}\langle\varphi\rangle$ et $\lceil q \rceil^{\top}\langle\varphi\rangle$ sont respectivement une relaxation monotone et une relaxation anti-monotone :

Théorème 3 (Relaxations monotone et anti-monotone) *Les contraintes basées sur les primitives $\lceil q \rceil^{\bot}$ et $\lceil q \rceil^{\top}$ sont respectivement une relaxation monotone et anti-monotone de q.*

Preuve. Tout d'abord, on prouve que $\lceil q \rceil^{\top}$ est anti-monotone. Soient $q \in \mathcal{Q}$ et φ un motif tel que $\lceil q \rceil^{\top}\langle\varphi\rangle$ soit *true*. Soit γ un motif tel que $\gamma \preceq \varphi$. Comme on a $\lceil q \rceil^{\top} = \lceil q \rceil\langle\varphi, \top\rangle$ et $[\varphi, \top] \subseteq [\gamma, \top]$, on obtient que $\lceil q \rceil\langle\varphi, \top\rangle \leq \lceil q \rceil\langle\gamma, \top\rangle = true$ (lemme 1, page 61). Ainsi $\lceil q \rceil^{\top}$ est anti-monotone. De plus, nous allons maintenant montrer que $\lceil q \rceil^{\top}$ est une relaxation de q. Supposons que $\lceil q \rceil^{\top}\langle\varphi\rangle$ est $false$ (i.e., $\lceil q \rceil\langle\varphi, \top\rangle = false$), le théorème 1 (cf. page 61) donne que pour n'importe quel $\theta \in [\varphi, \top]$ (i.e., $\varphi \preceq \theta$, on a $\lceil q \rceil\langle\varphi, \top\rangle \leq q(\theta) = false$. Finalement, $\neg\lceil q \rceil^{\top}$ implique que $\neg q$ et alors, nous concluons que la contrainte $\lceil q \rceil^{\top}$ est une relaxation anti-monotone de q. En appliquant le même raisonnement avec $\lceil q \rceil^{\bot}$, on prouve le théorème 3. □

Le théorème 3 assure que nous obtenons une relaxation monotone de q et une autre anti-monotone pour n'importe quelle PBC en lui appliquant simplement les opérateurs $\lceil\bullet\rceil^{\bot}$ et $\lceil\bullet\rceil^{\top}$. Ces relaxations héritent des bonnes propriétés issues des opérateurs de bornes. Comme ces opérateurs traitent les formules booléennes de contraintes, les relaxations peuvent tirer partie des spécificités de l'intégralité de la contrainte en combinant les relaxations des différentes contraintes atomiques. En plus, cette relaxation n'est calculée qu'une seule fois.

Revenons sur l'exemple de la contrainte d'aire pour lui appliquer l'opérateur $\lceil\bullet\rceil^{\top}$. Nous avons $\lceil area(X) \geq 6\rceil\langle X, Y\rangle = freq(X) \times count(\top) \geq 6$ car $\lceil area(X) \geq 6\rceil^{\top} = freq(X) \times count(Y) \geq 6$ (cf. la section 5.2.2) et $\lceil q \rceil^{\top}\langle X\rangle \equiv \lceil q \rceil\langle X, \top\rangle$. Comme $count(\top) = 5$, on obtient que $freq(X) \geq 6/5$ qui est une relaxation anti-monotone. Symétriquement, nous déduisons aussi la relaxation monotone $count(X) < 6/4$ donnée dans la section 6.1 issue de $\lceil area(X) \geq 6\rceil^{\bot}\langle\bot, X\rangle = freq(\bot) \times count(X) \geq 6 = 4 \times count(X) \geq 6$.

Finalement, pour n'importe quelle PBC, notre approche donne *automatiquement* une réponse au problème de la relaxation de contrainte défini à la section 6.1. De cette manière, la tâche d'extraction de motifs est optimisée avec de telles relaxations et devient faisable dans des contextes impossible sans (cf. la section 6.3.4).

Le tableau 6.3 donne les relaxations obtenues avec les opérateurs de relaxations en les appliquant à plusieurs exemples de contraintes. Tout d'abord, on remarque que la contrainte anti-monotone de fréquence minimale est sa propre relaxation anti-monotone. De même, la contrainte monotone $AE \subseteq X$ est sa propre relaxation monotone. Les opérateurs de relaxations traitent donc efficacement les contraintes (anti-)monotones (cf. la section 6.3.3). Bien sûr, l'utilisation d'une relaxation anti-monotone avec une contrainte monotone (ou vice-versa) conduit à

Contrainte q	Relaxation $\lceil q \rceil^{\perp}$	Relaxation $\lceil q \rceil^{\top}$
$freq(X) \times count(X) \geq 6$	$count(X) \geq 6/4$	$freq(X) \geq 6/5$
$(min(X.val) + max(X.val))/2 \leq 50$	$min(X.val) \leq 90$	$max(X.val) \leq 90$
$sum(X.val)/count(X) \geq 25$	$sum(X.val) \geq 25$	$185/count(X) \geq 25$
$AE \subseteq X$	$AE \subseteq X$	$true$
$freq(X) \geq 2$	$true$	$freq(X) \geq 2$

TAB. 6.3 – Relaxations monotones et anti-monotones d'exemples de contraintes.

la contrainte constante *true*. En d'autres termes, cette relaxation n'a aucun intérêt puisque tous les motifs du langage la satisfont. De plus, les opérateurs de relaxation ont surtout l'avantage de produire des contraintes monotones et anti-monotones pour des contraintes sans propriétés de monotonie connues au préalable. Les trois premières lignes du tableau 6.3 en donnent des exemples.

6.3.2 Autre espace des versions

Jusqu'ici, l'approche de relaxation considère seulement des motifs virtuels reposant sur l'espace des versions constitué des motifs présents au sein de la base de données. En particulier, l'espace des versions reste le même avec toutes les contraintes.

Dans certains cas, la contrainte q elle-même permet d'optimiser l'espace des versions à partir duquel les motifs virtuels sont calculés, par rapport à la collection des motifs présents. Les capacités d'élagage s'en trouvent alors améliorées. En effet, en admettant que nous cherchons une relaxation de contrainte composée de contraintes atomiques monotones, ces dernières peuvent être utilisées pour restreindre l'espace des versions $\mathcal{C}$. Les motifs virtuels découlant de ce nouvel espace des versions seront plus précis que ceux découlant de $\mathcal{C}$. En d'autres termes, les valeurs correspondant aux primitives terminales seront plus proches de celles des motifs à extraire. Ainsi, les relaxations obtenues avec ces motifs virtuels seront plus sélectives et donc, diminueront encore l'espace de recherche. Par exemple, en admettant que la longueur des motifs désirés n'excède pas l, nous pouvons choisir l'espace des versions délimité par $freq(X) \geq 1 \wedge count(X) \leq l$. Changer la collection $\mathcal{C}$ est aussi nécessaire quand elle est infinie pour pouvoir définir des motifs virtuels. Typiquement, extraire des épisodes dans une séquence infinie requiert de limiter la longueur de l'épisode maximal [Mannila *et al.*, 1995]. Sinon, il est impossible de définir le motif virtuel le plus spécifique.

6.3.3 Optimalité des relaxations

Cette section montre d'un point de vue théorique que notre approche de relaxation est très efficace pour les contraintes atomiques monotones et anti-monotones. Plus précisément, elle est optimale pour les contraintes monotones et anti-monotones détectables.

De nombreuses relaxations monotones ou anti-monotones peuvent être définies pour une même contrainte. Mais leur qualité diffère en fonction de la taille de leur théorie. Plus précisément, une relaxation est d'autant plus efficace que sa théorie est proche de celle de la contrainte originale. La relaxation (soit monotone, soit anti-monotone) qui approxime au mieux la contrainte originale, est dite *optimale*. La définition 21 précise de façon formelle cette notion d'optimalité :

Définition 21 (Relaxation optimale) *Une relaxation monotone (ou anti-monotone) q' de la contrainte q est optimale ssi pour n'importe quelle relaxation monotone (ou anti-monotone) q'' de q, q'' est aussi une relaxation de q'.*

Illustrons cette notion d'optimalité avec les relaxations issues de nos opérateurs. Comme le montre la figure 6.4, la relaxation $freq(X) \times count(\top) \geq 6$ issue de l'opérateur de relaxation anti-monotone (espace grisé), n'est pas optimale pour la contrainte d'aire minimale. L'espace délimité par la ligne grasse est la théorie correspondant à la relaxation anti-monotone optimale. Celle-ci élimine 2 motifs supplémentaires (à savoir F et DE) par rapport à la relaxation anti-monotone déduite des opérateurs de relaxation.

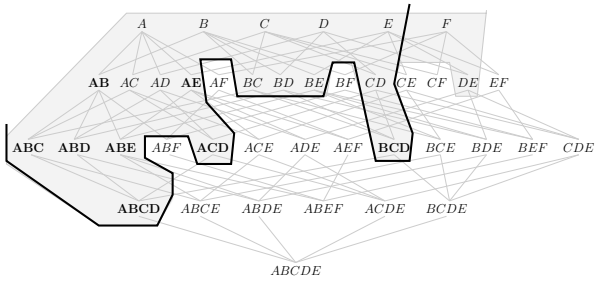


FIG. 6.4 – Optimalité de la relaxation anti-monotone pour la contrainte d'aire minimale.

La relaxation anti-monotone de la contrainte minimale de fréquence est optimale puisqu'on obtient la même contrainte (qui est déjà anti-monotone). Plus généralement, nous analysons maintenant l'optimalité des relaxations des contraintes monotones et anti-monotones détectables (définies à la section 5.3) issues de nos opérateurs :

Propriété 12 (Optimalité des relaxations) *Soit q une contrainte de $\lfloor \mathcal{Q} \rfloor^M \cup \lceil \mathcal{Q} \rceil^M$, les relaxations $\lceil q \rceil^\perp$ et $\lceil q \rceil^\top$ sont optimales.*

Preuve. Soit $q \in \lceil \mathcal{Q} \rceil^M$. Par définition des opérateurs de relaxation, on a donc $\lceil q \rceil^\top \langle \varphi \rangle \equiv \lceil q \rceil \langle \varphi, \top \rangle$ et $\lceil q \rceil^\perp \langle \varphi \rangle \equiv \lceil q \rceil \langle \perp, \varphi \rangle$. Or, la définition 19 donne que $\lceil q \rceil \langle \gamma, \varphi \rangle$ est équivalente à $q(\varphi)$. Donc, $\lceil q \rceil^\top \langle \varphi \rangle \equiv q(\top)$ qui est une constante (souvent égale à *true*). Ce résultat est optimal puisque q est monotone, elle n'a pas de meilleure relaxation anti-monotone. De plus, on a $\lceil q \rceil^\perp \equiv q$ i.e., la contrainte monotone détectable est sa propre relaxation monotone. Ce résultat est également optimal car une relaxation d'une contrainte q est au mieux égale à q. Par dualité (cf. la propriété 7 de la page 63), on conclut également que l'application des opérateurs de relaxation à une contrainte anti-monotone détectable permet d'obtenir les relaxations optimales. □

La propriété 12 garantit que toutes les contraintes monotones et anti-monotones détectables sont optimalement relaxées par nos opérateurs. Pour les autres contraintes, la découverte des relaxations optimales reste un problème ouvert. Néanmoins, l'étude expérimentale menée dans la section suivante montre que les relaxations obtenues même non-optimales restent performantes.

6.3.4 Expérimentations

L'objectif de ces expérimentations est de mesurer le gain apporté par les relaxations anti-monotones obtenues grâce aux opérateurs (et donc aux motifs virtuels). Les expérimentations

portent sur des motifs ensemblistes et séquentiels. Elles sont effectuées sur un ordinateur doté d'un processeur Xeon 2.2 GHz et de 3GB de mémoire RAM avec le système d'exploitation Linux.

Chaque expérimentation se déroule en trois étapes : calcul du motif virtuel le plus spécifique (en observant les valeurs limites des transactions), obtention automatique de la condition d'élagage (avec l'opérateur de relaxation anti-monotone) et exploitation de celle-ci avec un algorithme. Rappelons que n'importe quel algorithme dédié aux contraintes anti-monotones peut être utilisé.

Motifs ensemblistes

Pour les motifs ensemblistes, le jeu de données choisi est `mushroom` (cf. annexe B). La contrainte de moyenne minimale est appliquée avec les valeurs numériques générées aléatoirement entre [1,100]. La table 6.4 donne la définition des motifs virtuels $\bot$ et $\top$ qui résultent de cette base de données. Les expériences sont effectuées avec les deux algorithmes classiques d'extraction de motifs fréquents Apriori et Eclat implémentés par Borgelt[16].

Primitive p	$p(\bot)$	$p(\top)$
$freq$	8124	1
$count$	1	23
sum	0	1253
min	97	0
max	0	97

Tab. 6.4 – Les motifs virtuels le plus général et le plus spécifique associés à `mushroom`.

La figure 6.5 reporte les extractions de contraintes avec et sans les relaxations anti-monotones. Les courbes de gauche donnent le temps d'exécution pour l'extraction de tous les motifs satisfaisant la contrainte d'aire minimale en fonction du seuil d'aire minimale. Les courbes d'Apriori et d'Eclat sans la relaxation anti-monotone n'apparaissent pas car les extractions échouent. Sur la droite, les courbes correspondent aux temps d'extractions des motifs satisfaisant la contrainte de moyenne minimale suivant la variation du seuil de moyenne. Pour cette dernière, un seuil de fréquence minimal de 1% a été ajouté pour rendre faisable les extractions. Ce seuil minimal a été utilisé pour Apriori et Eclat.

Les courbes de la figure 6.5 montrent que dans tous les cas l'utilisation des motifs virtuels améliore les temps d'extraction. Cependant, les relaxations tirent d'autant mieux profit des spécificités du jeux de données que la contrainte est sélective (ici, lorsque les seuils sont élevés).

Motifs séquentiels

Nous étudions maintenant l'impact des relaxations anti-monotones sur les algorithmes d'extraction de séquences fréquentes PrefixSpan [Pei *et al.*, 2001b] et de séquences closes fréquentes CloSpan [Yan *et al.*, 2003] (avec les implémentations disponibles sur le site `http://illimine.cs.uiuc.edu/`). Le jeu de données utilisé $C100T2.5S10I2.5$ est décrit à l'annexe B. La figure 6.6 reporte les temps d'extraction de toutes les séquences satisfaisant la contrainte d'aire minimale suivant la variation de l'aire minimale. Comme pour les motifs ensemblistes, sans seuil de fréquence minimale, l'extraction est infaisable et l'approche sans relaxation est impossible.

[16] Ces implémentations sont disponibles sur le site du FIMI. Pour la contrainte de moyenne minimale, de légères adaptations du code source ont été nécessaires.

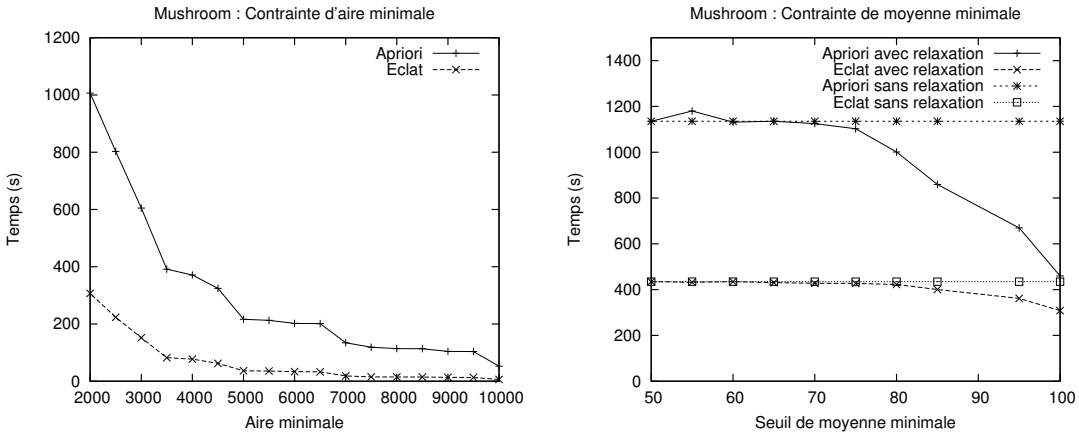


FIG. 6.5 – Temps d'exécution pour les motifs ensemblistes suivant la variation du seuil pour la contrainte d'aire minimale et de moyenne minimale (sur `mushroom`).

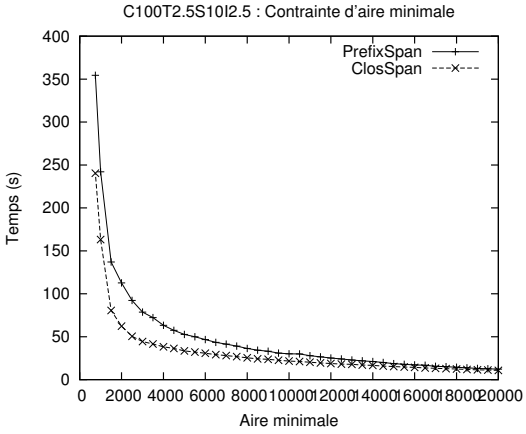


FIG. 6.6 – Temps d'exécution pour les séquences suivant la variation de l'aire minimale.

L'impact des relaxations anti-monotones sur l'extraction de séquences est similaire à celui observé pour les motifs ensemblistes. Lorsque la sélectivité augmente, l'efficacité de la relaxation anti-monotone augmente aussi.

6.4 Conclusion et discussion

À partir du langage des motifs ensemblistes et celui des motifs séquentiels, nous avons illustré notre approche de relaxation et montré son intérêt pratique (pousser une contrainte anti-monotone améliore toujours l'extraction). Par ailleurs, les relaxations sont complètement indépendantes du langage ou de l'algorithme utilisé. En particulier, l'approche en profondeur (e.g., ECLAT) ou par niveaux (e.g., APRIORI) sont toutes deux améliorées. Dès que l'extraction devient impossible pour certaines contraintes difficiles, l'utilisateur est tenté de choisir une contrainte anti-monotone arbitrairement pour faciliter l'extraction (e.g., une contrainte de fréquence minimale). Outre le choix de la bonne contrainte, sans cette approche de relaxation le seuil est soit fixé trop bas (on perd alors en efficacité voire l'extraction reste infaisable), soit il est fixé trop haut (et on perd alors des motifs qui satisfont la contrainte).

De manière plus générale, d'autres algorithmes (e.g., DUALMINER [Bucila *et al.*, 2002] ou EXANTE [Bonchi *et al.*, 2003]) peuvent aussi bénéficier des relaxations monotones obtenues à partir de l'opérateur de relaxation monotone. Dans [Soulet et Crémilleux, 2005b], nous définissons également les opérateurs $\lfloor q \rfloor^{\perp}\langle\varphi\rangle \equiv \lfloor q \rfloor\langle\perp, \varphi\rangle$ et $\lfloor q \rfloor^{\top}\langle\varphi\rangle \equiv \lfloor q \rfloor\langle\varphi, \top\rangle$. Ces derniers donnent des relaxations de la contrainte $\neg q$ grâce à la dualité entre $\lfloor \blacksquare \rfloor$ et $\lceil \blacksquare \rceil$. Ces relaxations permettent donc d'effectuer des élagages positifs et sont, par exemple, exploitables avec la méthode des témoins [Kiefer *et al.*, 2003].

Les relaxations anti-monotones issues des opérateurs de relaxation sont utilisées comme contrainte d'optimisation de MUSIC et MUSIC-DFS que nous présentons au chapitre 8. Dans le chapitre suivant, nous traitons des contraintes dont les primitives ne peuvent être évaluées directement et requièrent donc une approche spécifique. Cette dernière se fonde en partie sur les relaxations obtenues dans ce chapitre et en valide à nouveau l'efficacité.

Chapitre 7

Extraction de contraintes globales par Approximer-et-Pousser

Sommaire

Dans ce chapitre, nous introduisons la notion de contraintes globales. De telles contraintes soulèvent des problématiques d'extraction nouvelles que nous illustrons à partir de l'exemple de la contrainte des top-k motifs (celle-ci recherche les k motifs maximisant une mesure d'intérêt). Elle se révèle très utile pour trouver les motifs les plus significatifs au regard d'un critère choisi par l'utilisateur. Ce chapitre propose alors une méthode originale, appelée *Approximer-et-Pousser*, d'extraction de motifs satisfaisant une contrainte globale. L'idée fondamentale est de déduire une contrainte locale qui est affinée au cours de l'extraction. En particulier, cette approche permet d'obtenir efficacement les k motifs maximisant une mesure d'intérêt en tirant parti de la méthode de relaxation présentée au chapitre précédent. Contrairement aux approches usuelles, elle est applicable à toute mesure basée sur des primitives (et pas seulement à la fréquence) et ce pour tout langage.

La première section introduit la notion de contrainte globale en distinguant celle de couverture et celle d'optimisation. La section 7.2 présente la méthode générale d'extraction et en donne deux illustrations simples. Enfin, l'approche Approximer-et-Pousser extrayant les top-k motifs est décrite à la section 7.3.

7.1 Les contraintes globales

7.1.1 Définition

Les contraintes traitées dans le chapitre précédent se vérifient isolément sur chaque motif (même si elles nécessitent des accès à la base de données). Par exemple, la fréquence d'un motif X et sa longueur X permettent de déduire si le motif X satisfait la contrainte d'aire minimale (sans rien connaître sur les autres motifs de la base de données). Dans d'autres situations, la vérification d'une contrainte nécessite de comparer entre eux plusieurs motifs. Bien que de telles contraintes existent dans la littérature [Pasquier *et al.*, 1999, Fu *et al.*, 2000], aucune définition n'en est proposée. Nous introduisons donc la notion de *contrainte globale* à travers la définition suivante :

Définition 22 (Contrainte globale) *Une contrainte globale est une contrainte dont la vérification nécessite de comparer plusieurs motifs entre eux.*

Par opposition à cette définition, les contraintes usuelles présentées jusqu'ici sont dites *locales*. Les trois contraintes ci-dessous sont des exemples de contraintes globales car leur vérification sur un motif φ fait intervenir des tests avec les spécialisations ou les généralisations de φ :

$$Bd^+(\varphi) = \begin{cases} true & q_{AM}(\varphi) \wedge \forall \gamma \in \mathcal{L} \text{ tel que } \varphi \prec \gamma, \text{ on a } q_{AM}(\gamma) = false \\ false & sinon \end{cases}$$

$$fermé(\varphi) = \begin{cases} true & \forall \gamma \in \mathcal{L} \text{ tel que } \varphi \prec \gamma, \text{ on a } freq(\varphi) > freq(\gamma) \\ false & sinon \end{cases}$$

$$libre(\varphi) = \begin{cases} true & \forall \gamma \in \mathcal{L} \text{ tel que } \gamma \prec \varphi, \text{ on a } freq(\varphi) < freq(\gamma) \\ false & sinon \end{cases}$$

On remarquera que ces trois contraintes traduisent l'appartenance à une représentation condensée (cf. les sections 3.2 et 3.3). La bordure positive d'une contrainte anti-monotone q_{AM}, la représentation condensée des motifs fermés ou des libres correspondent respectivement à la théorie des contraintes Bd^+, *fermé*, *libre*.

En fait, les contraintes Bd^+, *fermé*, *libre* sélectionnent les motifs qui structurellement forment une couverture d'un autre ensemble des motifs. D'autres contraintes globales, appelées *contraintes d'optimisation*[17], conservent uniquement les motifs maximisant un critère donné. Par exemple, une bonne illustration est la contrainte extrayant les k motifs maximisant une mesure donnée (cf. la section 7.1.2). La figure 7.1 récapitule ces différentes formes de contraintes.

La notion de contrainte locale/globale est transverse à celle de contrainte syntaxique/d'agrégat (cf. la section 1.2.1). Par exemple, les bordures résultent d'une contrainte syntaxique globale liée à la maximalité au sens de l'inclusion.

7.1.2 Problématique de l'extraction : exemple des top-k motifs

Cette section présente un exemple de contrainte globale d'optimisation souvent utilisée dans la littérature [Fu *et al.*, 2000, Han *et al.*, 2002, Tzvetkov *et al.*, 2003]. Cette contrainte illustre parfaitement la problématique de l'extraction des contraintes globales.

[17] La notion d'optimisation se réfère ici à une mesure d'intérêt. Dans [Bayardo, 2005], les contraintes d'optimisation accélèrent le processus d'extraction indépendamment de la qualité des motifs extraits (il ne s'agit donc pas de sélectionner les meilleurs motifs par rapport à un critère) et le terme "optimisation" est à prendre dans un sens différent de celui de ce document.

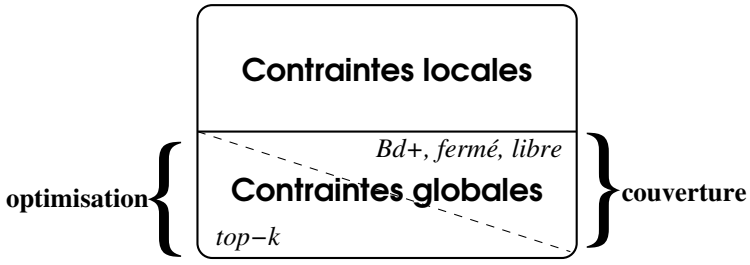


FIG. 7.1 – Distinction entre les contraintes.

Contrainte des top-k motifs selon une mesure

Le choix du seuil pour la contrainte minimale de fréquence ou d'aire (et plus généralement de $m(\varphi) \geq min$ où $m : \mathcal{L} \rightarrow \Re$ est une mesure) se révèle souvent difficile pour l'utilisateur. En effet, si ce seuil est trop élevé, trop peu de motifs sont extraits (au risque de n'obtenir que des informations triviales). À l'inverse, si min est trop bas, le nombre de motifs explose et les motifs les plus intéressants sont noyés dans la masse. Comme plusieurs tentatives d'extraction sont nécessaires pour estimer min, l'utilisateur préfère souvent fixer ce dernier relativement bas. Puis, parmi tous les motifs obtenus, il focalise son intérêt sur les premiers motifs maximisant sa mesure d'intérêt. La recherche de ces k meilleurs motifs est ainsi une tâche qui présente un vif intérêt et qui peut aussi se formuler sous forme d'une contrainte :

Définition 23 (Contrainte des top-k motifs) *Soient un entier $k > 0$ et une mesure $m : \mathcal{L} \rightarrow \Re$, la contrainte des top-k motifs selon m correspond à :*

$$top_{k,m}(\varphi) \equiv |\{\gamma \in \mathcal{L} | \gamma \neq \varphi \wedge m(\gamma) > m(\varphi)\}| < k$$

Cette contrainte compare les motifs entre eux pour ne conserver que ceux dont la mesure fait partie des k meilleures. Par exemple, dans le contexte $\mathcal{D}$ ci-dessous, les 3 motifs ensemblistes de plus grande aire correspondent exactement aux motifs satisfaisants $top_{3,area}$[18] : AB $(3 \times 2 = 6)$, AC $(3 \times 2 = 6)$ et ABC $(2 \times 3 = 6)$. Les motifs associés à la contrainte $top_{k,m}$ sont nommés les top-k motifs selon la mesure m. En fait, leur nombre est parfois supérieur à k (tous les motifs à partir du $k^{\text{ème}}$ ont alors la même mesure). Typiquement les top-3 motifs fréquents sont 4 à savoir A (5), C (4), B (3) et E (3), car la fréquence ne permet pas de distinguer les motifs B et E. Notons que les k motifs minimisant une mesure m satisfont la contrainte $top_{k,-m}$.

Extraction des top-k motifs

Naïvement, l'extraction des top-k motifs peut s'effectuer avec un post-traitement. Après l'extraction de tous les motifs dont la mesure m excède un seuil min, il suffit de sélectionner les k motifs maximisant m. Outre l'inefficacité algorithmique, la difficulté du choix du seuil minimal persiste. Si celui-ci est fixé trop haut, moins de k motifs peuvent être extraits. En revanche, si ce seuil est trop bas, des motifs inutiles sont extraits et ce processus ne profitant pas du paramètre k peut devenir très lent (voire infaisable). Pour résoudre ce problème, il est préférable de pousser la contrainte $top_{k,m}$ au sein de l'extraction de motifs.

Malheureusement, les contraintes globales (de couverture ou d'optimisation) sont encore plus complexes à pousser que les contraintes locales. La localisation des motifs les satisfaisant

[18] Dans cet exemple, les k motifs de plus grande aire ont la même aire, mais ce n'est pas toujours le cas.

$\mathcal{D}$

Trans.	Items					
t_1	A	B			E	F
t_2	A		C		E	
t_3	A	B	C	D		
t_4	A	B	C			
t_5	A			D		
t_6			C		E	

Item	A	B	C	D	E	F
val	50	30	75	10	30	15

TAB. 7.1 – Un contexte transactionnel $\mathcal{D}$ et une table de valeurs.

est souvent ardue car il s'agit de dégager une structure de la base de données. Par analogie, si une contrainte locale est une équation à une inconnue, une contrainte globale correspond à un système d'équations. Souvent la vérification immédiate d'une contrainte pour un motif donné devient impossible sans l'énumération de tous les autres motifs.

L'extraction des top-k motifs fréquents a été introduite dans [Fu *et al.*, 2000]. Les auteurs adaptent alors APRIORI pour ajuster le seuil de fréquence minimale au fur et à mesure de l'extraction. Dans [Hirate *et al.*, 2004], la structure FP-tree permet d'optimiser l'extraction des top-k motifs. Plus récemment, la structure COFI-tree a aussi été utilisée [Ngan *et al.*, 2005]. À notre connaissance, un seul travail étudie l'extraction des top-k motifs pour d'autres langages que celui des motifs ensemblistes, en recherchant des séquences [Tzvetkov *et al.*, 2003]. Plusieurs travaux extraient les top-k motifs contraints fréquents. Par exemple, les k motifs fermés les plus fréquents et de longueur minimale sont recherchés dans [Han *et al.*, 2002] en utilisant la structure FP-tree. D'autres recherchent les motifs les plus fréquents, fermés ou non, et de longueur minimale [Cong, 2001].

Tous ces travaux sont restreints à la mesure de fréquence comme mesure d'intérêt car la contrainte de fréquence minimale est anti-monotone. Remarquons aussi qu'ils se focalisent presque tous sur les motifs ensemblistes. Notre démarche se distingue donc en proposant une méthode adaptée à n'importe quelle mesure d'intérêt basée sur les primitives. Même si nos expériences sont dédiées aux motifs ensemblistes, l'approche Approximer-et-Pousser est applicable à d'autres langages.

7.2 Méthode Approximer-et-Pousser

7.2.1 Principes généraux

L'approche Approximer-et-Pousser permet d'extraire des motifs satisfaisant une contrainte globale. Brièvement, l'idée est de restreindre l'espace de recherche lors du parcours en affinant la localisation des motifs susceptibles de vérifier la contrainte globale. Pour cela, cette approche s'appuie sur la répétition de deux étapes majeures (et qui forment son nom) : (1) approximer la collection finale à extraire, (2) pousser des informations issues de cette approximation pour diminuer l'espace de recherche.

Approximer

La mise à jour de la collection de motifs candidats se décline en trois opérations : l'initialisation, l'ajout et la suppression. L'*initialisation* de la collection des motifs candidats doit

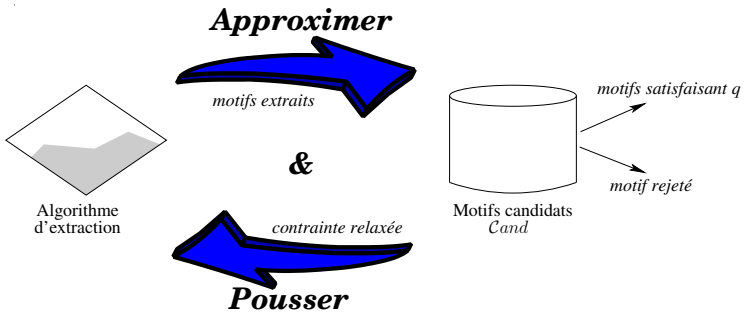


FIG. 7.2 – Illustration de la méthode Approximer-et-Pousser.

être choisie avec attention pour éviter de manquer un motif. Lorsque l'espace de recherche est parcouru dans son ensemble, la collection peut être initialisée à vide. Ensuite, l'*ajout* et la *suppression* des motifs interviennent à chaque nouvelle étape d'approximation i.e., un nouveau motif postule pour entrer dans la collection. Ce dernier est ajouté à cette collection si et seulement si au vu des motifs candidats déjà présents, il peut éventuellement satisfaire la contrainte globale. Enfin, un motif est supprimé de la collection s'il est exclu par un motif postulant (lors d'une étape de maintenance). Un motif peut être supprimé soit *positivement* (i.e., il est conservé car il satisfait la contrainte globale), soit *négativement* (sinon). Lorsqu'un motif est exclu par le motif postulant, cela n'implique pas toujours l'entrée de ce dernier.

Pousser

Par l'intermédiaire de la collection de motifs candidats, cette étape doit permettre de pousser la contrainte globale au cœur de l'extraction et ainsi, réduire l'espace de recherche. Dans un premier temps, cette étape déduit certaines informations de l'approximation (e.g., un calcul effectué sur les motifs candidats). Ces informations évoluent au gré de l'ajout et de la suppression des motifs. Ensuite, elles sont converties en une condition d'élagage afin d'éliminer des motifs de l'espace de recherche. Cette condition d'élagage peut par exemple être une contrainte locale adaptée à un algorithme d'extraction.

Plutôt que d'algorithmes Approximer-et-Pousser, nous préférons parler d'approches Approximer-et-Pousser car par la suite, notre approche délègue l'élagage de l'espace de recherche à un algorithme indépendant. La condition d'élagage lui est donnée sous forme d'une contrainte locale d'extraction qui est dynamiquement affinée à chaque itération. Une telle approche Approximer-et-Pousser peut être alors vue comme une relaxation évolutive de la contrainte globale en une contrainte locale. La figure 7.2 illustre l'approche Approximer-et-Pousser en décrivant l'interaction cyclique entre l'algorithme d'extraction et les opérations sur l'approximation (i.e., les motifs candidats).

7.2.2 Illustrations directes

Cette section donne deux approches Approximer-et-Pousser pour extraire les motifs satisfaisant la contrainte Bd^+, puis *libre* introduites à la section 7.1.1. Contrairement à la section 7.3, ces deux tâches sont trop simples pour mettre en valeur les atouts de l'approche Approximer-et-Pousser, mais elles montrent le caractère générique de cette méthode.

Extraction d'une bordure

Nous proposons d'extraire la bordure positive d'une contrainte anti-monotone q_{AM} en appliquant l'approche Approximer-et-Pousser avec l'algorithme par niveaux (cf. la section 2.2.1). Dans cet exemple, l'approximation contient donc la bordure temporaire.

Tout d'abord, on initialise la collection de motifs candidats à vide. À chaque étape d'approximation, le motif postulant issu de l'algorithme par niveaux et satisfaisant q_{AM} est ajouté à la collection des motifs candidats. Parallèlement, les motifs candidats appartenant à la collection et inclus dans le nouveau motif sont supprimés car ils ne pourront plus appartenir à la bordure finale. Dans cette approche Approximer-et-Pousser, l'étape "pousser" n'interagit pas avec l'étape "approximer". En effet, la contrainte locale permettant de réduire l'espace de recherche de l'algorithme par niveaux est q_{AM} et ne dépend donc pas de la bordure temporaire des motifs candidats. Finalement, lorsque l'algorithme par niveaux a fini l'extraction de tous les motifs satisfaisant q_{AM}, la collection des motifs candidats constitue exactement la bordure positive de la contrainte q_{AM}.

De manière similaire, on peut aussi envisager l'extraction de la bordure négative ou des bordures d'une contrainte monotone. Dans tous les cas, l'approche Approximer-et-Pousser s'apparente à un simple filtrage des motifs issus de l'algorithme par niveaux. En particulier, la bordure temporaire formée par les motifs candidats ne permet pas de réduire davantage l'espace de recherche.

Extraction des motifs libres

Nous décrivons maintenant une méthode d'extraction des motifs libres par Approximer-et-Pousser. Cette méthode nécessite à nouveau un algorithme par niveaux pour extraire les motifs présents dans le contexte transactionnel (i.e., de fréquence supérieure à 1) en leur associant leur fréquence. À un instant donné, les motifs candidats sont bien des libres, mais ils sont mis de côté pour tester la liberté des motifs postulant.

On initialise la collection de motifs candidats avec le motif vide et sa fréquence correspondant à la cardinalité de la base (puisqu'il est contenu dans chacune des transactions). À chaque étape "approximer", un nouveau motif de longueur l parvient de l'algorithme par niveaux avec sa fréquence. Ce dernier est ajouté aux motifs candidats si chacun de ses sous-motifs de longueur $l-1$ est contenu dans la collection et a une fréquence strictement supérieure à la sienne. Si le motif postulant est rejeté, l'étape "pousser" déduit qu'aucune de ses spécialisations ne pourra plus être libre et cette contrainte anti-monotone est poussée dans l'algorithme pour réduire l'espace de recherche. Lorsque l'espace de recherche a été épuisé par l'algorithme par niveaux, on retourne l'ensemble de la collection des motifs candidats.

Cette méthode Approximer-et-Pousser bénéficie cette fois d'une interaction entre l'étape d'approximation et l'étape d'extraction. Néanmoins, la tâche est grandement facilitée par l'anti-monotonie de la liberté et la possibilité de localiser simplement les motifs libres dans l'espace de recherche. Ainsi, à nouveau, cette approche Approximer-et-Pousser s'apparente à un simple algorithme par niveaux.

7.3 Application à l'extraction des top-k motifs selon une mesure

7.3.1 Aperçu de l'approche

Cette section donne un aperçu général de notre approche d'extraction des top-k motifs selon m en exploitant la méthode Approximer-et-Pousser.

Les deux illustrations de la section 7.2.2 montrent que la collection des motifs candidats sert à vérifier la contrainte globale en comparant les motifs entre eux. De cette manière, des motifs peuvent être rejetés ou au contraire conservés avec certitude. Dans la seconde application, la non-liberté d'un motif offre même une condition d'élagage immédiate pour éliminer toutes ses spécialisations. L'avantage des extractions présentées ci-dessus est de connaître la localisation des motifs satisfaisant la contrainte globale. Pour la bordure, il s'agit des motifs à l'extrémité de l'espace anti-monotone de la contrainte q_{AM}. Les motifs libres constituent quant à eux un espace regroupant les motifs les plus généraux.

L'extraction des top-k motifs selon une mesure m est plus épineuse car en général, on ne sait pas où se situeront dans l'espace de recherche les motifs vérifiant la contrainte. Par ailleurs, leur définition (cf. la page 81) ne permet pas directement d'obtenir une contrainte locale. Afin de pallier en partie ce dernier point, nous introduisons une définition alternative des top-k motifs avec la propriété 13 :

Propriété 13 *Le seuil minimal d'appartenance aux top-k motifs selon la mesure m, dénoté* $\rho_{k,m}$*, est* $\min\{m(\varphi)|\varphi \in \mathcal{L} \wedge top_{k,m}(\varphi)\}$*, et on a* $top_{k,m}(\varphi) \equiv m(\varphi) \geq \rho_{k,m}$*.*

Preuve. Soient m une mesure et $k > 0$, on fixe $\rho_{k,m} = \min\{m(\varphi)|\varphi \in \mathcal{L} \wedge top_{k,m}(\varphi)\}$. Soit $\varphi \in \mathcal{L}$, si $m(\varphi)$ est supérieure à $\rho_{k,m}$, on a bien $top_{k,m}(\varphi)$ qui est vraie par définition. Sinon, si $m(\varphi)$ est strictement inférieure à $\rho_{k,m}$, φ ne peut satisfaire $top_{k,m}$ car par définition, $\rho_{k,m}$ est inférieur ou égal à tous les motifs satisfaisant $top_{k,m}$. □

Cette reformulation de la contrainte des top-k motifs pour m est à nouveau une contrainte globale. Le seuil $\rho_{k,m}$ concentre implicitement les comparaisons entre motifs nécessaires pour vérifier la contrainte $top_{k,m}$. Néanmoins, cette reformulation rend possible la définition d'une contrainte locale en fixant le seuil $\rho_{k,m}$ (même arbitrairement). Nous verrons que ce point est essentiel par la suite.

Dans la suite, nous proposons d'exploiter cette propriété avec l'approche Approximer-et-Pousser en considérant :

1. **Approximer :** cette étape d'approximation permettra de déterminer un seuil ρ tendant à égaler $\rho_{k,m}$ à partir d'une collection de motifs candidats.
2. **Pousser :** cette étape poussera la contrainte $m(\varphi) \geq \rho$ pour réduire l'espace de recherche.

Chacune de ces deux étapes est difficile. La première doit permettre de fixer le seuil temporaire ρ de façon à ne pas éliminer de motifs satisfaisant $top_{k,m}$. Contrairement à la contrainte de liberté, la contrainte à pousser $m(\varphi) \geq \rho$ n'est pas forcément anti-monotone. Nous utiliserons alors les résultats du chapitre précédent pour obtenir une relaxation anti-monotone. Ainsi, avec un algorithme d'extraction de contraintes anti-monotones (comme l'algorithme par niveaux), notre approche permettra de traiter n'importe quelle mesure basée sur les primitives pour un langage quelconque (même si, dans la suite, nous illustrons cette approche uniquement à partir du langage $\mathcal{L}_{\mathcal{I}}$).

7.3.2 Description des deux étapes

Approximer les top-k motifs

L'étape d'approximation conserve les k motifs maximisant la mesure m parmi les motifs déjà extraits. De cette façon, lorsque l'algorithme d'extraction aura parcouru l'intégralité de l'espace de recherche, les k motifs candidats retenus seront exactement les top-k motifs selon la mesure m.

À l'initialisation de l'extraction, la collection des motifs candidats $\mathcal{C}and$ ne contient aucun motif. La maintenance de cette collection commence alors par une phase de remplissage. Tous les motifs extraits sont ajoutés sans condition jusqu'à obtenir une collection de k motifs candidats. Durant cette phase, aucun motif de $\mathcal{C}and$ n'est supprimé. Ensuite, l'évolution de $\mathcal{C}and$ entre dans une phase sélective guidée par la propriété suivante :

Propriété 14 *Soit un ensemble de motifs C tel que $|C| \geq k$, si la mesure m d'un motif donné est strictement inférieure à celle de chacun des motifs de C, alors ce motif ne satisfait pas la contrainte $top_{k,m}$.*

Preuve. Soit un motif φ et $C \subseteq \mathcal{L}$ tel que $|C| \geq k$. Fixons ρ' à $\min_{\gamma \in C} m(\gamma)$. Comme les motifs satisfaisant la contrainte $top_{k,m}$ maximisent m, on a $\rho_{k,m} \geq \rho'$. Or $m(\varphi) < \rho'$, on obtient que $m(\varphi) < \rho_{k,m}$ et la propriété 13 permet de conclure que φ ne satisfait pas la contrainte $top_{k,m}$. □

Dans notre approche, la collection C de cette propriété correspond aux motifs candidats $\mathcal{C}and$ (ou à un de ses sous-ensembles). Dès que $\mathcal{C}and$ a atteint k éléments, la propriété peut être appliquée sur un motif postulant pour savoir s'il est bien nécessaire de l'ajouter à la collection des motifs candidats. Plus précisément, un motif postulant φ est ajouté à la collection si la mesure de φ est supérieure à celle d'au moins un des motifs candidats. Dans le cas contraire, la propriété 14 nous garantit que le motif postulant ne pourra pas faire parti des top-k motifs selon m. En outre, un motif est supprimé de la collection dès que k autres motifs de $\mathcal{C}and$ ont une mesure supérieure à la sienne. En effet, la propriété 14 assure à nouveau que ce motif ne sera jamais parmi les k motifs de plus forte mesure m et donc, ne satisfera pas la contrainte $top_{k,m}$.

L'introduction du seuil d'ajout permet d'unifier ces deux phases distinctes de l'étape approximer :

Définition 24 (Seuil d'ajout) *Le seuil d'ajout, noté ρ, est défini de la manière suivante :*

$$\rho = \begin{cases} -\infty, & \textit{si } |\mathcal{C}and| < k \\ \min_{\varphi \in \mathcal{C}and} m(\varphi), & \textit{sinon} \end{cases}$$

L'intérêt de cette approche est que ce seuil évolue au fur et à mesure des modifications de la collection des motifs candidats $\mathcal{C}and$. Basiquement, un motif postulant est ajouté à la collection si et seulement si sa mesure m est supérieure à celle du seuil d'ajout ρ. Ainsi, durant la phase de remplissage, la collection accepte tous les motifs car leur mesure est toujours supérieure au seuil d'ajout alors égal à $-\infty$. Ensuite, les valeurs de la mesure de chacun des motifs de $\mathcal{C}and$, synthétisées par le seuil d'ajout, conditionne l'introduction ou non du motif postulant au sein de la collection.

Le tableau 7.2 décrit l'évolution des motifs candidats $\mathcal{C}and$ au cours du processus d'extraction de la contrainte $top_{3,area}$ avec Apriori pour le contexte donné au tableau 7.1. L'algorithme par niveaux envoie trois vagues successives de motifs. Pour chaque niveau, les motifs dont l'aire est supérieure à ρ entrent dans la collection des motifs candidats. La valeur de l'aire est donnée par le chiffre entre parenthèses dans la colonne de gauche et les motifs candidats sont rassemblés dans la colonne centrale. Le seuil ρ (colonne de droite) est ajusté au fur et à mesure. Tant que la taille de la collection $\mathcal{C}and$ est inférieure à k, le seuil ρ est initialisé à $-\infty$. Ensuite, ρ correspond à l'aire minimale satisfaite par un des motifs de $\mathcal{C}and$. Le motif E n'est pas exclu par l'entrée de B car son aire excède ρ. En revanche, B et E sont supprimés négativement à l'arrivée du motif AB. À la fin du dernier niveau, $\mathcal{C}and$ correspond exactement aux 3 motifs de plus forte mesure d'aire.

Niveau 1

Motif	$\mathcal{C}and$	ρ
A (5)	A	$-\infty$
B (3)	A, B	$-\infty$
C (4)	A, C, B	3
E (3)	A, C, B, E	3

Niveau 2

Motif	$\mathcal{C}and$	ρ
AB (6)	AB, A, C	4
AC (6)	AB, AC, A	5

Niveau 3

Motif	$\mathcal{C}and$	ρ
ABC (6)	AB, AC, ABC	6

TAB. 7.2 – Les top-3 motifs selon l'aire avec APRIORI.

Pousser l'approximation

Cette étape bénéficie de la collection obtenue des motifs candidats afin de réduire l'espace de recherche. Nous montrons maintenant comment il est possible de déduire de cette collection une contrainte anti-monotone afin de réutiliser des algorithmes efficaces bénéficiant de l'anti-monotonie.

Les seuls motifs pouvant satisfaire la contrainte $top_{k,m}$ sont ceux qui peuvent être ajoutés à la collection des motifs candidats. Ces motifs doivent donc avoir une mesure supérieure au seuil d'ajout i.e., ils satisfont la contrainte locale $m(\varphi) \geq \rho$. Malheureusement, cette contrainte n'est pas toujours anti-monotone. Typiquement, la contrainte $area(X) \geq \rho$ n'est pas anti-monotone. Par exemple, dans le contexte $\mathcal{D}$, le motif ABC satisfait la contrainte $area(X) \geq 6$, mais pas sa généralisation BC dont l'aire est seulement de 4. En fait, la contrainte $m(\varphi) \geq \rho$ est anti-monotone si et seulement si la mesure m est une fonction décroissante (e.g., la fréquence). Afin d'obtenir dans le cas général une contrainte anti-monotone, nous proposons d'approximer la contrainte $m(\varphi) \geq \rho$ par sa relaxation anti-monotone (cf. le chapitre 6). Par exemple, avec le contexte $\mathcal{D}$, la contrainte $area(X) \geq \rho$ est approximée par $freq(X) \times 4 \geq \rho$ (cf. le tableau 6.3 de la page 73). De cette manière, le seuil d'ajout qui tend vers $\rho_{k,m}$, donne une contrainte anti-monotone (et sa condition d'élagage) qui s'affine au fur et à mesure.

L'efficacité de cette approche Approximer-et-Pousser réside dans l'ajustement dynamique de la contrainte au cours de l'extraction. Plus précisément, la relaxation anti-monotone $\lceil m(\varphi) \geq \rho \rceil^{\top}$ devient de plus en plus sélective car le seuil d'ajout ρ croît pour tendre vers $\rho_{k,m}$. Nous avons vu dans le chapitre précédent que lorsque la sélectivité d'une contrainte augmente, la relaxation est d'autant plus performante. Cette approche Approximer-et-Pousser diminue donc significativement l'espace de recherche pour donner un processus d'extraction rapide.

7.3.3 Expérimentations

L'objectif de ces expérimentations est de montrer l'efficacité de l'approche Approximer-et-Pousser pour différentes mesures et différents jeux de données. Au-delà de la rapidité, nous souhaitons montrer la faisabilité de notre approche générique. Aussi, nous ne nous comparons pas aux algorithmes de la littérature limités à la seule mesure de fréquence, mais nous confrontons trois stratégies différentes d'extraction des top-k motifs basées sur l'algorithme APRIORI :

- **Approximer-et-Pousser** : cette stratégie extrait les top-k motifs en s'appuyant sur l'approche Approximer-et-Pousser.
- **Optimale à 50%** : cette stratégie exploite la relaxation anti-monotone de $m(X) \geq \rho$ en fixant le seuil ρ à 50% du seuil idéal $\rho_{k,m}$. Ce seuil idéal est le seuil permettant d'obtenir

exactement et directement les top-k motifs. Bien sûr, dans la réalité, ce seuil n'est pas connu et l'utilisateur procède plutôt par tâtonnement à partir de son intuition.
- **Post-traitement** : les motifs sont extraits avec un seuil de fréquence minimale de 10%. Puis, les k motifs maximisant la mesure sont conservés. Le seuil de 10% est un compromis entre faisabilité et exhaustivité (i.e., ne manquer aucun des top-k motifs).

Pour toutes ces expériences, nous utilisons la même implémentation d'Apriori avec de légères variantes suivant la stratégie et la mesure considérées. Les temps d'extractions sont donc comparables. Toutes les expériences sont effectuées sur un ordinateur doté d'un processeur Xeon 2.2 GHz et de 3GB de mémoire RAM avec le système d'exploitation Linux.

La figure 7.3 reporte les temps des extractions en fonction du nombre de motifs désirés k pour 3 jeux de données : `mushroom`, `letter` et `chess` (cf. annexe B). Sur chaque base, deux mesures ont alors été utilisées, à savoir la fréquence et l'aire. En plus, des trois stratégies exposées ci-dessus, nous ajoutons le temps d'extraction optimal comme courbe de référence. Cette valeur de référence consiste à fixer directement la relaxation anti-monotone $m(X) \geq \rho_{k,m}$ pour obtenir exactement les k meilleurs motifs. Rappelons qu'en pratique, la valeur du seuil $\rho_{k,m}$ ne peut être connue par avance. Remarquons aussi que pour l'extraction des $top_{k,area}$ sur `chess`, toutes les approches (et même la courbe de référence) sont trop longues voire échouent. Cela s'explique par la faiblesse de l'algorithme Apriori (et/ou de notre implémentation).

La stratégie Post-traitement se distingue des deux autres car, quelque soit la valeur de k, le temps d'extraction est le même. Le plus souvent cette stratégie est la moins bonne (surtout lorsque k est peu élevé). Dans de rares situations où k est de valeur moyenne, cette stratégie dépasse les deux autres. En revanche, pour des valeurs de k trop grandes, il arrive que cette approche manque des top-k motifs (car ces derniers se situent sous le seuil de fréquence minimale choisi). Cela se traduit par un arrêt des courbes sur les graphiques de la figure 7.3 car l'approche n'effectue plus la tâche demandée. Par exemple, avec le jeu de données `letter`, quelque soit la valeur de k, cette stratégie manque des motifs. Sur `chess`, la courbe n'apparaît pas car le temps d'extraction à 10% dépasse plusieurs heures. Cette stratégie de post-traitement ne fournit pas toujours le résultat souhaité et elle échoue parfois en temps.

Il est intéressant de remarquer que, globalement, les deux stratégies Approximer-et-Pousser et Optimale-50% ont le même comportement. Plus le nombre de k motifs à extraire est grand, plus le temps d'extraction augmente. Par ailleurs, lorsqu'une mesure est plus difficile à traiter qu'une autre, elle l'est pour les deux stratégies. Comme attendu, dans tous les cas, la courbe de référence est en deçà des deux stratégies. Un résultat important est que pour toutes les expériences, la stratégie Optimale-50% a de plus mauvais résultats que l'approche Approximer-et-Pousser. Notre approche Approximer-et-Pousser s'intercale donc entre cette stratégie et la courbe de référence. Si le gain par rapport à l'approche Optimale-50% peut parfois être modeste, il peut devenir conséquent dans certaines situations. De plus, l'approche Approximer-et-Pousser évite la fixation du seuil souvent hasardeuse avec les deux autres stratégies.

Plus k est petit, plus notre approche est proche de la référence. Cela s'explique par une phase de remplissage rapide de la collection des candidats et une approximation immédiate de la contrainte globale. Cette approche novatrice, en laissant à l'utilisateur le choix de la mesure, est suffisamment efficace. Elle fonctionne d'autant mieux que l'utilisateur demande assez peu de motifs ce qui est généralement le cas en pratique. Par ailleurs, notre approche peut également rechercher les top-k motifs contraints selon une mesure en choisissant un algorithme d'extraction de motifs contraints. Par exemple, une longueur minimale peut être exigée sur les motifs comme c'est le cas pour certaines approches d'extraction des top-k motifs fréquents [Han *et al.*, 2002, Cong, 2001].

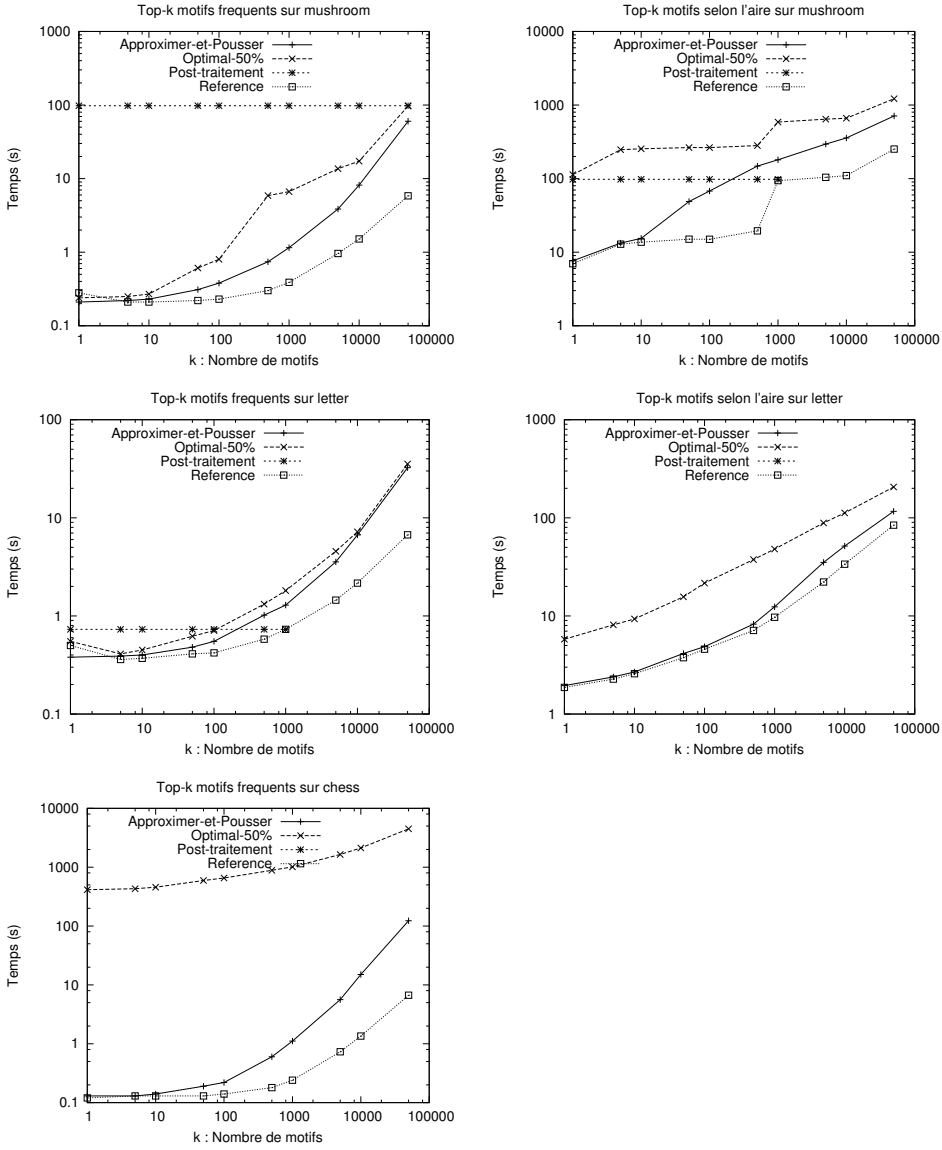


FIG. 7.3 – Temps d'extraction des top-k motifs.

7.4 Conclusion

Notre approche Approximer-et-Pousser d'extraction des top-k motifs selon une mesure permet d'extraire les k motifs maximisant une mesure. Son efficacité repose grandement sur la bonne qualité de la méthode de relaxation présentée au chapitre précédent. Cette méthode est généralisable à d'autres langages ou à des motifs contraints suivant l'algorithme d'extraction employé.

Plus généralement, l'efficacité de l'approche Approximer-et-Pousser réside dans la qualité de l'approximation et sur la manière de l'exploiter pour réduire au mieux l'espace de recherche. Par ailleurs, la collection des motifs candidats tend progressivement vers la solution finale (i.e., les motifs satisfaisant la contrainte globale). À tout moment, le processus peut fournir une solution approchée par le biais de ces motifs. Dans le cas des top-k motifs, l'utilisateur obtiendrait des motifs avec de fortes mesures, mais pas forcément les meilleurs. Enfin, des formes plus sophistiquées de réservoirs pourraient être imaginées pour traiter des contraintes globales de forme différentes. Par exemple, si plusieurs contextes de fouille étaient mis en jeu des réservoirs séparés pourraient leur être assignés.

Chapitre 8

MUSIC-DFS : un nouvel algorithme d'extraction de motifs contraints

Sommaire

Le chapitre 6 a étendu l'usage des solveurs usuels à l'extraction de motifs satisfaisant une PBC grâce à la relaxation de contraintes. Cependant, dans des contextes denses, les approches fondées sur les classes d'équivalence s'avèrent particulièrement performantes. En effet, celles-ci rassemblent de nombreux motifs partageant les mêmes propriétés. Dans ce chapitre, nous proposons un nouvel algorithme appelé MUSIC-DFS, tirant profit de ces propriétés communes pour le langage des motifs ensemblistes. Son atout majeur est de pleinement exploiter les élagages positif et négatif (cf. fin de la section 2.1.3) sur des intervalles et non seulement sur les spécialisations ou les généralisations comme cela était le cas au chapitre 6. D'autre part, MUSIC-DFS extrait une collection d'intervalles qui forme une représentation condensée des motifs satisfaisant la contrainte tout comme les classes d'équivalence constituent une représentation condensée des motifs fréquents. Au final, MUSIC-DFS s'avère pour l'utilisateur un outil efficace entièrement automatique offrant une grande flexibilité au niveau de la contrainte.

La section 8.1 définit l'opérateur d'élagage et donne les points clés de sa mise en œuvre pratique. Ensuite, cet opérateur est utilisé au sein de l'algorithme MUSIC-DFS (section 8.2). Pour cela, un nouvel opérateur de fermeture est introduit. La section 8.3 analyse le comportement de MUSIC-DFS en le comparant à d'autres algorithmes et en quantifiant ses principales caractéristiques.

8.1 Opérateur d'élagage

8.1.1 Définition et propriétés

Dans cette section, nous définissons un opérateur d'élagage qui évite de parcourir des intervalles lors de la recherche de motifs satisfaisant une PBC. Nous présentons le principe de cet opérateur à partir de l'exemple de la contrainte d'aire $freq(X) \times count(X) \geq 6$. Nous avons vu à la section 5.2.2 que pour tout motif Z contenu dans un intervalle $[X, Y]$, on a $(freq(Y) \times count(X) \geq 6) \leq (area(Z) \geq 6) \leq (freq(X) \times count(Y) \geq 6)$. Deux stratégies différentes sont alors envisageables pour élaguer l'intervalle $[X, Y]$ en utilisant les bornes. Si le minorant de la contrainte d'aire est égal à $true$, tous les motifs inclus dans $[X, Y]$ satisfont q car ils sont tous plus grands que $true$ (car $false < true$). Dans ce cas, on dit qu'on élague positivement l'intervalle $[X, Y]$. À l'inverse, l'élagage négatif de l'intervalle $[X, Y]$ est effectué lorsque le majorant est égal à $false$ car aucun motif de l'intervalle ne peut satisfaire la contrainte.

À partir de ces observations, nous définissons maintenant la condition d'élagage d'un intervalle pour n'importe quelle PBC :

Définition 25 (Opérateur d'élagage) *Soit q une PBC, la condition d'élagage sur un intervalle de q, dénotée par $[q]$, est égale à $\lfloor q \rfloor \vee \neg \lceil q \rceil$. $[.]$ est appelé l'opérateur d'élagage.*

L'opérateur d'élagage s'applique à une PBC pour obtenir une condition d'élagage définie sur des intervalles. L'expression de cette condition d'élagage fait bien apparaître l'élagage positif avec $\lfloor q \rfloor$ et l'élagage négatif avec $\neg \lceil q \rceil$. Par exemple, l'application de l'opérateur d'élagage à la contrainte d'aire minimale donne la condition d'élagage suivante $(freq(Y) \times count(X) \geq 6) \vee (freq(X) \times count(Y) < 6)$.

Le théorème suivant justifie l'élagage issu de l'opérateur $[.]$:

Théorème 4 *Soient q une PBC et $[\varphi, \gamma]$ un intervalle, si $[q]\langle \varphi, \gamma \rangle$ est égal à true, alors tous les motifs inclus dans l'intervalle $[\varphi, \gamma]$ ont la même valeur pour q.*

Preuve. Soient q une PBC et $[X, Y]$ un intervalle. Deux cas se distinguent pour que $[q]\langle X, Y \rangle$ soit vraie. Si $\lfloor q \rfloor \langle X, Y \rangle = true$, tous les motifs contenus dans $[X, Y]$ sont plus grand que $true$ (cf. le théorème 1, page 61). Ainsi, tous les motifs satisfont q. Sinon, nous avons $\neg \lceil q \rceil \langle X, Y \rangle = true$ (i.e. $\lceil q \rceil \langle X, Y \rangle = false$) et le théorème 1 nous permet d'affirmer que tous les motifs ont une valeur pour q inférieure à $false$. De cette manière, aucun motif de $[X, Y]$ ne satisfait q. □

Ce théorème est le fondement de notre méthode d'élagage. Lorsque la condition d'élagage obtenue avec l'opérateur est satisfaite sur un intervalle, tous les motifs contenus dans l'intervalle ont la même valeur (soit ils satisfont tous la contrainte, soit aucun ne satisfait la contrainte). Plutôt que de visiter tous les motifs de cet intervalle, il est alors possible de se contenter de la valeur d'un seul de ses motifs. Ainsi, l'espace de recherche est réduit. La section suivante décrit la mise en œuvre pratique de cette condition d'élagage.

Remarquons qu'en général la réciproque du théorème 4 est fausse. En d'autres termes, même si la condition d'élagage est fausse, tous les motifs de l'intervalle peuvent avoir la même valeur. Cela signifie que notre opérateur d'élagage traduit une condition suffisante mais pas toujours nécessaire. Ce résultat provient du fait que nous avons privilégié un cadre générique conduisant à approximer les bornes issues de $\lfloor . \rfloor$ et $\lceil . \rceil$. Cependant, l'optimalité de l'élagage est garantie avec certaines contraintes :

Propriété 15 (Optimalité) *Soient $[\varphi, \gamma]$ un intervalle et q une contrainte monotone ou anti-monotone détectable, $[q]\langle \varphi, \gamma \rangle = false$ est équivalent à $q(\varphi) = \neg q(\gamma)$.*

Preuve. Soit $q \in \lfloor \mathcal{Q} \rfloor^M$ et un intervalle $[\varphi, \gamma]$ tel que $[q]\langle\varphi, \gamma\rangle = false$. Comme q est anti-monotone détectable, on a $\lfloor q \rfloor\langle\varphi, \gamma\rangle \equiv q(\gamma)$ et $\lceil q \rceil\langle\varphi, \gamma\rangle \equiv q(\varphi)$. De cette manière, on obtient que $[q]\langle\varphi, \gamma\rangle = q(\gamma) \vee \neg q(\varphi)$. Or $[q]\langle\varphi, \gamma\rangle = false$, on déduit que $q(\gamma) = false$ et $\neg q(\varphi) = false$. On procède de la même manière pour une contrainte monotone détectable. Enfin, le théorème 4 montre le sens indirect de la propriété 15. □

L'équivalence $[q]\langle\varphi, \gamma\rangle = false \Leftrightarrow q(\varphi) = \neg q(\gamma)$ de la propriété 15 est bien synonyme d'optimalité puisque lorsque l'opérateur retourne $false$, tous les motifs de l'intervalle n'ont pas la même valeur pour la contrainte. Pour les contraintes monotones ou anti-monotones détectables, une réponse négative de l'opérateur d'élagage sur un intervalle signifie donc qu'il n'est pas possible de l'élaguer.

8.1.2 Mise en œuvre de la condition d'élagage

Le chapitre 2 a montré que, de manière générale, la mise en œuvre pratique des conditions d'élagage était liée au choix du parcours de l'espace de recherche (en largeur ou en profondeur). L'utilisation d'intervalles pour l'élagage, comme nous venons de l'introduire, pose la problématique supplémentaire et non triviale de leur construction. Pour un langage donné $\mathcal{L}$, le nombre d'intervalles est compris entre $|\mathcal{L}|$ et $|\mathcal{L}|^2$. Parmi tous ces intervalles, sur lesquels doit-on tester la condition d'élagage issue de l'opérateur $[.]$?

Le choix des intervalles est guidé par des impératifs et des souhaits. D'abord, il est nécessaire que l'ensemble des intervalles choisis R couvre l'intégralité de l'espace de recherche. En d'autres termes, pour tout motif θ de $\mathcal{L}$ présent dans $\mathbf{r}$, il doit y avoir un intervalle $[\varphi, \gamma] \in R$ tel que $\theta \in [\varphi, \gamma]$. Ensuite, pour qu'une telle approche ait un intérêt pratique, le nombre d'intervalles de R doit être inférieur au nombre de motifs du langage $\mathcal{L}$, i.e. $|R| < |\mathcal{L}|$ (sinon il n'y aurait pas de gain par rapport à l'énumération naïve de $\mathcal{L}$).

À ces conditions impératives, l'ensemble R des intervalles choisis peut aussi satisfaire des critères optimisants. Une représentation est d'autant meilleure que le chevauchement entre les intervalles est minimal. En effet, si un motif est contenu dans plusieurs intervalles, il peut être testé plusieurs fois. Cela signifie que la même partie de l'espace de recherche est explorée à plusieurs reprises, ce qui est évidemment inutile et coûteux. Par ailleurs, quand un intervalle ne peut être élagué grâce à la condition d'élagage issue de l'opérateur $[.]$, il est alors nécessaire de le subdiviser en d'autres intervalles. Idéalement, un bon ensemble d'intervalles pour une contrainte q ne rassemble que des intervalles où la condition d'élagage de q est satisfaite. Pour cela, les intervalles où la valeur de la contrainte q est partout la même, sont particulièrement efficaces (la probabilité que la réponse de l'opérateur d'élagage soit vraie est alors très grande). On parle de la *conservation* de q sur un intervalle. Nous reviendrons sur ce point dans le chapitre suivant.

D'autres problématiques se posent. Lorsque la condition d'élagage n'est pas satisfaite pour un intervalle donné, il est nécessaire de le subdiviser en d'autres intervalles pour y appliquer à nouveau la condition d'élagage. La construction de ces sous-intervalles implique les mêmes difficultés que la construction des intervalles originaux R : couverture, cardinalité faible, non-chevauchement, conservation.

8.1.3 Algorithme en largeur : Music

Dans un premier temps, nous avons développé Music (Mining with a User-SpecifIed Constraint) qui exploite une condition d'élagage sur un intervalle pour extraire les motifs ensemblistes satisfaisant une PBC [Soulet et Crémilleux, 2005a]. Le choix du langage des motifs

ensemblistes est essentiellement lié à des besoins applicatifs (cf. les chapitres 10 et 11). À partir d'une PBC q quelconque (et éventuellement d'une contrainte anti-monotone supplémentaire q_{AM}) et d'une base de données $\mathbf{r}$, MUSIC retourne la théorie $\mathcal{T}h(\mathcal{L}_{\mathcal{I}}, \mathbf{r}, q \wedge q_{AM})$ sous forme d'une représentation condensée d'intervalles. En fait, chacun des motifs appartenant à la théorie est contenu dans au moins un intervalle.

D'un point de vue technique, l'algorithme MUSIC construit des intervalles basés sur la fermeture de Galois introduite à la section 3.3.1. Un parcours par niveaux énumère les intervalles délimités par un motif libre et sa fermeture. Lorsque l'intervalle satisfait la condition d'élagage, on l'élague. Sinon, un algorithme type APRIORI détaille ses sous-intervalles en appliquant à nouveau la condition d'élagage jusqu'à sa satisfaction. Au final, l'ensemble des intervalles constitués des motifs libres et de leurs fermetures garantit un parcours exhaustif de l'espace de recherche. Par ailleurs, ces intervalles sont généralement moins nombreux que les motifs qu'ils décrivent, et ce gain devient très important dans les données corrélées.

Cet algorithme démontre un grand intérêt pratique en rendant faisable des extractions de motifs contraints jusqu'alors impossibles. De plus, la diversité des PBC en fait un outil particulièrement générique. Malheureusement, deux limites majeures entravent son fonctionnement. D'une part, les intervalles sur lequel MUSIC s'appuie se chevauchent énormément. Cela est dû à la forme des classes d'équivalence de fréquence dont sont issus les motifs libres (cf. la figure 3.4, page 37). Ainsi, un même motif est parfois considéré plusieurs fois ralentissant l'extraction et augmentant l'espace mémoire nécessaire au stockage des intervalles candidats. D'autre part, nos expériences pratiques sur des bases de données larges (e.g., données d'expression de gènes) ont mis en échec l'algorithme MUSIC tout comme les autres algorithmes par niveaux (cf. la section 12.3). En effet, dès les premiers niveaux (même avec un seuil de fréquence minimale élevé), le nombre de candidats explose et devient impossible à stocker en mémoire. Pour ces deux raisons, dans la section suivante, nous proposons une variante de l'algorithme MUSIC en exploitant un parcours en profondeur.

8.2 Algorithme en profondeur : MUSIC-DFS

8.2.1 Fondements théoriques

Opérateur de fermeture préfixé

L'usage de la fermeture de Galois pour construire les intervalles pose deux problèmes majeurs. D'une part, cette fermeture n'est pas triviale à calculer avec un parcours en profondeur (même s'il existe des approches telle que CHARM [Zaki et Hsiao, 1999]). D'autre part, comme nous l'avons expliqué auparavant, la fermeture de Galois conduit à des intervalles se chevauchant à de multiples reprises. Or nous souhaitons éviter ces chevauchements. Afin de pallier ces deux problèmes, nous allons introduire un nouvel opérateur de fermeture basé sur un pré-ordre $\preceq_R$. Cet opérateur est au cœur d'une représentation condensée d'intervalles qui couvre intégralement l'espace de recherche sans chevauchement (cf. le théorème 5 ci-après).

La relation de pré-ordre $\preceq_R$ prend en compte une relation d'ordre arbitraire sur les items $\mathcal{I}$ à savoir $A <_R B <_R C <_R \ldots$ comme proposée dans [Pei *et al.*, 2001a]. Désormais dans ce chapitre, les motifs manipulés sont ordonnés i.e., un motif $X = x_1 x_2 \ldots x_n$ satisfait $x_i <_R x_j$ pour tout $i < j$. De cette manière, le motif $X = x_1 x_2 \ldots x_n$ est un préfixe du motif $Y = y_1 y_2 \ldots y_m$, et cette relation se note $X \preceq_R Y$, ssi on a $n \leq m$ et $\forall i \in \{1, \ldots, n\},\ x_i = y_i$. Par exemple, les préfixes de ABC sont les motifs A, AB et ABC. Au contraire, $AD \not\preceq_R ADC$ car la forme ordonnée de ADC est ACD et AD n'est pas un préfixe de ACD.

$\mathcal{D}$

Trans.	Items					
t_1	A				E	F
t_2		B	C	D		
t_3	A	B	C	D	E	F
t_4	A	B	C	D		

TAB. 8.1 – Un exemple de contexte transactionnel.

En s'appuyant sur le pré-ordre $\preceq_R$, nous définissons maintenant la notion de fermeture préfixée pour un motif :

Définition 26 (Fermeture préfixée) *La fermeture préfixée d'un motif X, dénotée par $\mathbf{cl}_R(X)$, est le motif $\{a \in \mathcal{I} | \exists Y \subseteq X$ tel que $Y \preceq_R Y \cup \{a\}$ et $freq(Y \cup \{a\}) = freq(Y)\}$.*

Le motif $\mathbf{cl}_R(X)$ rassemble tous les items apparaissant dans les mêmes transactions et contenant $Y \subseteq X$ tel que Y est un préfixe de $Y \cup \{a\}$[19]. Les points fixes de l'opérateur $\mathbf{cl}_R$ sont nommés les *motifs fermés par préfixe*. Illustrons cette définition avec le contexte du tableau 8.1. Le motif ABC n'est pas fermé par préfixe car ABC est un préfixe de $ABCD$ et $freq(ABCD) = freq(ABC)$. La définition 26 permet de déduire que n'importe quel motif et sa fermeture préfixée ont la même fréquence. Par exemple, comme $\mathbf{cl}_R(ABC) = ABCD$, $freq(ABC) = freq(ABCD) = 2$. Nous montrons maintenant que $\mathbf{cl}_R$ est un opérateur de fermeture :

Propriété 16 (Opérateur de fermeture) *La fermeture préfixée $\mathbf{cl}_R$ est un opérateur de fermeture.*

Preuve. *Extensivité :* Soit X un motif et $a \in X$. On a $\{a\} \subseteq X$ et évidemment, $a \preceq_R a$ et $freq(a) = freq(a)$. On obtient alors que $a \in \mathbf{cl}_R(X)$ et donc, $\mathbf{cl}_R$ est extensif. *Isotonie :* Soit $X \subseteq Y$ et $a \in \mathbf{cl}_R(X)$. Il existe $Z \subseteq X$ tel que $Z \preceq_R Za$ et $freq(Za) = freq(Z)$. Avec ces mêmes propriétés, comme on a aussi $Z \subseteq Y$, on obtient que $a \in \mathbf{cl}_R(Y)$ et on conclut bien que $\mathbf{cl}_R(X) \subseteq \mathbf{cl}_R(Y)$. *Idempotence :* Soit X un motif. Soit $a \in \mathbf{cl}_R(\mathbf{cl}_R(X))$. Il existe $Z \subseteq \mathbf{cl}_R(X)$ tel que $freq(Za) = freq(Z)$ avec $Z \preceq_R Za$. Comme $Z \subseteq \mathbf{cl}_R(X)$, pour tout $a_i \in Z$, il y a $Z_i \subseteq X$ tel que $freq(Z_i a_i) = freq(Z_i)$ avec $Z_i \preceq_R Z_i a_i$. On a $\bigcup_i Z_i \preceq_R \bigcup_i Z_i a$ et $freq(\bigcup_i Z_i) = freq(\bigcup_i Z_i a)$ (car $freq(\bigcup_i Z_i) = freq(Z)$). Comme le motif $\bigcup_i Z_i \subseteq X$, a appartient à $\mathbf{cl}_R(X)$ et alors, $\mathbf{cl}_R$ est idempotent. □

La propriété 16 est importante pour la suite car elle nous permet d'exploiter tous les résultats concernant les opérateurs de fermeture[20]. D'abord, cet opérateur de fermeture organise le treillis des motifs en classes d'équivalence. Plus précisément, deux motifs X et Y sont équivalents ssi ils ont la même fermeture préfixée (i.e., $\mathbf{cl}_R(X) = \mathbf{cl}_R(Y)$). Bien sûr, comme $\mathbf{cl}_R$ est idempotent, le motif maximal (au sens de l'inclusion) d'une classe d'équivalence donnée à laquelle appartient X, correspond au motif fermé par préfixe $\mathbf{cl}_R(X)$. Réciproquement, nous appelons les *motifs libres par préfixe* les motifs minimaux des classes d'équivalence. De plus, la propriété 16 garantit également que ces minimaux sont les motifs les plus généraux de $\mathcal{L}$. En d'autres termes, "être

[19] Par la suite, l'union $X \cup \{a\}$ est parfois notée Xa pour alléger les notations.

[20] Remarquons que la fermeture préfixée est définie en exploitant $\preceq_R$, mais c'est un opérateur de fermeture par rapport à $\subseteq$.

libre par préfixe" est une contrainte anti-monotone suivant la spécialisation $\subseteq$ (cf. la propriété 18, page 115).

En plus des propriétés usuelles des opérateurs de fermetures, $\mathbf{cl}_R$ possède ses propres spécificités. En montrant que chaque classe d'équivalence ne possède qu'un unique motif libre par préfixe, la propriété 17 facilite la démonstration du théorème 5 :

Propriété 17 (Opérateur de liberté préfixée) *Soit un motif X, il existe un unique motif minimal (au sens de l'inclusion), dénoté par $\mathbf{fr}_R(X)$, dans la classe d'équivalence de X.*

Preuve. Supposons que X et Y sont deux motifs minimaux de la même classe d'équivalence : on a $\mathbf{cl}_R(X) = \mathbf{cl}_R(Y)$. Comme X et Y sont différents, il existe $a \in X$ tel que $a \notin Y$ et $a \leq \min_{\leq_R}\{b \in Y \backslash X\}$ (ou on inverse X et Y). Comme X est minimal, aucun motif $Z \subseteq X \cap Y$ ne satisfait à la fois $Z \preceq_R Za$ et $freq(Za) = freq(Z)$. De plus, pour tout Z tel que $Y \cap X \subset Z \subset Y$, on est sûr que $Z \not\preceq_R Za$ car a est strictement inférieur à tous les items de $Y \backslash X$. Donc, a n'appartient pas à $\mathbf{cl}_R(Y)$ (i.e., $\mathbf{cl}_R(X) \neq \mathbf{cl}_R(Y)$) et l'hypothèse initiale est contredite. Ainsi, on conclut que n'importe quelle classe d'équivalence contient un unique motif libre par préfixe. □

La propriété 17 signifie que l'opérateur $\mathbf{fr}_R$ relie chaque motif X au motif libre par préfixe de sa classe d'équivalence, i.e. $\mathbf{fr}_R(X)$. X est libre par préfixe ssi $\mathbf{fr}_R(X) = X$. N'importe quelle classe d'équivalence correspond en fait à un intervalle délimité par un motif libre par préfixe et son motif fermé par préfixe (i.e., $[\mathbf{fr}_R(X), \mathbf{cl}_R(X)]$). Dans notre exemple, AB (resp. $ABCD$) est le motif libre par préfixe (resp. motif fermé par préfixe) de la classe d'équivalence $[AB, ABCD]$.

Représentation condensée issue de $\mathbf{cl}_R$

Dans la suite, nous considérons les intervalles formés par tous les motifs libres par préfixe et leur motifs fermés par préfixe. La collection entière de ces intervalles est une *représentation condensée d'intervalles* i.e., $\mathcal{R}_{\preceq_R} = \{[\mathbf{fr}_R(X), \mathbf{cl}_R(X)] \subseteq \mathcal{L}_\mathcal{I} \times \mathcal{L}_\mathcal{I} \mid freq(X) \geq 1\}$. Chaque motif X (présent dans la base de données) est contenu dans un unique intervalle de cette représentation à savoir $[\mathbf{fr}_R(X), \mathbf{cl}_R(X)]$:

Théorème 5 (Représentation condensée formée d'intervalles) *Chaque motif présent dans la base de données est inclus dans un unique intervalle de $\mathcal{R}_{\preceq_R}$. De plus, le nombre de ces intervalles est inférieur au nombre de motifs qu'ils représentent.*

Preuve. Soit X un motif et $\mathcal{R}_{\preceq_R} = \{[\mathbf{fr}_R(X), \mathbf{cl}_R(X)] | freq(X) \geq 1\}$. La propriété 17 prouve que X est exactement contenu dans $[\mathbf{fr}_R(X), \mathbf{cl}_R(X)]$. Ce dernier est unique. L'intervalle $[\mathbf{fr}_R(X), \mathbf{cl}_R(X)]$ appartenant à $\mathcal{R}_{\preceq_R}$ par définition, on conclut que $\mathcal{R}_{\preceq_R}$ est une représentation de tous les motifs présents dans la base de données. Maintenant, l'extensivité et l'idempotence de l'opérateur de fermeture préfixé $\mathbf{cl}_R$ garantissent que $|\mathcal{R}_{\preceq_R}| \leq |\{X \in \mathcal{L}_\mathcal{I} \text{ tel que } freq(X) \geq 1\}|$. Ainsi, nous concluons que le théorème 5 est correct. □

Ce théorème montre que la représentation $\mathcal{R}_{\preceq_R}$ répond à nos deux attentes décrites à la section 8.1.2. Tout d'abord, chaque motif présent dans la base est présent dans un unique intervalle de $\mathcal{R}_{\preceq_R}$. Ce résultat la différencie de façon importante de la représentation issue des libres et des fermés de la fermeture de Galois (cf. la section 8.1.3). Cette unicité améliore l'efficacité de l'algorithme présenté à la section suivante en ne testant qu'une seule fois chaque motif. Deuxièmement, dans le pire des cas, la taille de la représentation condensée est le nombre de motifs (on a alors chaque motif qui est à la fois libre par préfixe et fermé par préfixe).

Mais en pratique, le nombre d'intervalles est très inférieur au nombre de motifs. Dans notre exemple du tableau 8.1, 23 intervalles résument les 63 motifs présents. Le tableau 8.2 illustre la condensation de cette représentation en comparant le nombre de motifs présents dans la base (seconde colonne) à la cardinalité de $\mathcal{R}_{\preceq_R}$ (troisième colonne). De manière expérimentale, on constate qu'en proportion, plus le nombre de motifs est important, plus la représentation condensée des intervalles est concise.

Contexte	Nombre de motifs	Nombre d'intervalles	Condensation
`abalone`	55439	44567	0.804
`cmc`	174127	98505	0.566
`wine`	2032121	346905	0.171
`mushroom`	1279963141	1045946	8.17e-4

TAB. 8.2 – Condensation de la représentation condensée.

Plus qu'une représentation condensée de tous les motifs du treillis, cette représentation structure le treillis des motifs. Il est alors immédiat d'adapter cette représentation aux seuls motifs fréquents. En effet, l'ensemble des intervalles $\mathcal{R}_{\preceq_R,minfr}\{[\mathbf{fr}_R(X), \mathbf{cl}_R(X)] \subseteq \mathcal{L}_\mathcal{I} \times \mathcal{L}_\mathcal{I} \mid freq(X) \geq minfr\}$ est une représentation condensée d'intervalles des motifs dont la fréquence excède $minfr$. Plus généralement, dans la section suivante, l'algorithme MUSIC-DFS transforme $\mathcal{R}_{\preceq_R}$ pour obtenir une représentation condensée d'intervalles adaptée à n'importe quelle PBC.

8.2.2 Description de l'algorithme

L'algorithme MUSIC-DFS prend en argument une PBC q, une contrainte anti-monotone q_{AM} et une base de données $\mathbf{r}$, et il retourne une représentation condensée d'intervalles de tous les motifs satisfaisant la contrainte $q \wedge q_{AM}$. Pour prendre en considération la fréquence, l'utilisateur peut choisir pour q_{AM} la contrainte de fréquence minimale. Afin que la contrainte q_{AM} n'élimine aucun motif satisfaisant q, on peut soit la fixer à $true$ (car $q \wedge true = q$), soit prendre pour q_{AM} la relaxation anti-monotone comme $\lceil q \rceil^\top$ (cf. le chapitre 6). La seconde alternative optimise l'extraction sans éliminer de candidats susceptibles de satisfaire q (cf. annexe C.3). D'un point de vue général, MUSIC-DFS (cf. l'algorithme 1) parcourt en profondeur les intervalles de $\mathcal{R}_{\preceq_R}$ présentés à la section précédente, grâce à la fonction GLOBALSCAN (cf. l'algorithme 2). Chacun de ces intervalles est alors raffiné jusqu'à satisfaire l'élagage sur l'intervalle grâce à la fonction LOCALSCAN (cf. l'algorithme 3). MUSIC-DFS bénéficie à la fois de la condition d'élagage sur les intervalles issue de q, de la condition d'élagage anti-monotone de q_{AM} et de la liberté par préfixe.

Tout d'abord, l'algorithme MUSIC-DFS (cf. l'algorithme 1) lance l'extraction des motifs sur chacun des items de $\mathcal{I}$ entre la ligne 2 et 7 en exécutant GLOBALSCAN (ligne 5). Au préalable, la ligne 3 vérifie que le motif a satisfait bien la contrainte anti-monotone q_{AM}. Les items candidats $\mathcal{IC}$ de GLOBALSCAN correspondent aux items qui peuvent venir grossir le préfixe a. Un item peut être éliminé de 2 manières distinctes : (1) pour éviter une redondance de parcours (s'il enfreint $b >_R a \wedge PrefixFree(ab)$ à la ligne 4), (2) pour éviter de violer q_{AM} (s'il enfreint $q_{AM}(ab)$ à la ligne 4). Le même principe de réduction de candidats est utilisé à la ligne 3 de GLOBALSCAN à partir d'un préfixe X.

Ensuite, GLOBALSCAN (cf. l'algorithme 2) construit récursivement les intervalles de $\mathcal{R}_{\preceq_R}$ de préfixe X grâce aux items candidats $\mathcal{IC}$. À chaque nouvel appel, le préfixe grossit (ligne 4). La liberté par préfixe est utilisée pour réduire l'espace de recherche et ne pas construire d'intervalles superflus en réduisant les items candidats (ligne 3). Sur chaque intervalle de la représentation

Algorithm 1 MUSIC-DFS

Input: Une PBC q, une contrainte anti-monotone q_{AM} et une base de données $\mathbf{r}$
Output: La représentation condensée d'intervalles de $\mathcal{T}h(\mathcal{L}_{\mathcal{I}}, \mathbf{r}, q \wedge q_{AM})$

1: $\mathcal{R}es := \emptyset$
2: **for** every $a \in \mathcal{I}$ **do**
3: **if** $q_{AM}(a)$ **then**
4: $cand := \{b \in \mathcal{IC} | b >_R a \wedge PrefixFree(ab) \wedge q_{AM}(ab)\}$ // *sélectionne les items candidats du préfixe a*
5: $\mathcal{R}es := \mathcal{R}es \cup$GLOBALSCAN$(a, cand, q, q_{AM}, \mathbf{r})$ // *recherche globale de préfixe a*
6: **end if**
7: **end for**
8: **return** $\mathcal{R}es$

condensée $\mathcal{R}_{\preceq_R}$, il recherche les intervalles dont les motifs satisfont $q \wedge q_{AM}$, en exécutant LOCALSCAN (ligne 1). Ces derniers sont alors ajoutés à $\mathcal{R}es$ (ligne 1) et sont finalement retournés (ligne 6).

Algorithm 2 GLOBALSCAN

Input: Un préfixe X, les items candidats $\mathcal{IC}$, une PBC q, une contrainte anti-monotone q_{AM} et la base de données $\mathbf{r}$
Output: La représentation condensée $\mathcal{R}es$ de la PBC q de préfixe X

// *Analyse de l'intervalle* $[X, \mathbf{cl}_R(X)]$ *:*
1: $\mathcal{R}es :=$LOCALSCAN$([X, \mathbf{cl}_R(X)], q, q_{AM}, \mathbf{r})$ // *analyse de l'intervalle*
// *Poursuite de la recherche globale :*
2: **for** $a \in \mathcal{IC}$ **do**
3: $cand := \{b \in \mathcal{IC} | b >_R a \wedge PrefixFree(Xb) \wedge q_{AM}(Xb)\}$
4: $\mathcal{R}es := \mathcal{R}es \cup$GLOBALSCAN$(Xa, cand, q, q_{AM}, \mathbf{r})$ // *énumération récursive*
5: **end for**
6: **return** $\mathcal{R}es$ // *retourne la représentation de préfixe X*

Enfin, LOCALSCAN (cf. l'algorithme 3) retourne les sous-intervalles de $[X, Y]$ dont les motifs satisfont la contrainte $q \wedge q_{AM}$. Pour cela, la condition d'élagage sur les intervalles intervient dans l'algorithme LOCALSCAN de la ligne 1 à 7. En fait, l'opérateur d'élagage est utilisé à la ligne 1, et le test de la ligne 2 détermine alors si l'élagage est positif (ligne 3) ou négatif (ligne 4). Lorsque cet élagage échoue, le motif X est tout de même ajouté à $\mathcal{R}es$ si X satisfait q (ligne 9). Ensuite, le reste de l'intervalle est découpé en sous-intervalles de sorte à ne pas avoir de chevauchement (ligne 10 à 15). En particulier, pour chaque préfixe Xa, la borne droite de l'intervalle est calculée grâce à l'ordre sur les items (ligne 12). On applique alors récursivement l'algorithme LOCALSCAN sur chacun de ces sous-intervalles (ligne 13) si la borne de gauche satisfait la contrainte q_{AM} (ligne 11). Au final, LOCALSCAN retourne un ensemble d'intervalles décrivant tous les motifs satisfaisant $q \wedge q_{AM}$ au sein de l'intervalle $[X, Y]$.

En pratique l'implémentation de MUSIC-DFS comporte plusieurs optimisations algorithmiques qui n'apparaissent pas ci-dessus. Typiquement, la construction du pré-ordre $\preceq_R$ n'est pas spécifiée ici. Remarquons que pour un motif X donné, de par la forme du parcours, l'anti-monotonie n'est pas testée sur chacun de ses sous-ensembles. Par exemple, pour le motif ABC, l'anti-monotonie est testée sur les seuls motifs A, AB et AC, et pas sur BC. Pour cette raison, l'ordre $\leq_R$ sur les items a une importance capitale. Nous choisissons d'ordonner les items du

Algorithm 3 LOCALSCAN

Input: Un intervalle $[X, Y]$, une PBC q, une contrainte anti-monotone q_{AM} et la base de données $\mathbf{r}$

Output: La représentation condensée de la PBC q sur l'intervalle $[X, Y]$

```
    // Elagage de l'intervalle [X,Y] ?
 1: if [q]⟨X,Y⟩ then
 2:   if q(X) then
 3:     return {[X,Y]}    // élagage positif de l'intervalle
 4:   else
 5:     return ∅    // élagage négatif de l'intervalle
 6:   end if
 7: end if
   // Enumération locale de l'intervalle [X,Y] :
 8: Res := ∅
 9: if q(X) then Res := {[X,X]}    // X satisfait q
10: for a ∈ Y\X do
11:   if q_AM(Xa) then
12:     right := Xa ∪ {b ∈ Y\X | b >_R a}
13:     Res := Res∪LOCALSCAN([Xa, right], q, q_AM, r)
14:   end if
15: end for
16: return Res
```

moins fréquent au plus fréquent. Cette heuristique améliore grandement l'extraction en diminuant le nombre d'intervalles car la contrainte de liberté préfixée devient plus efficace. Cette heuristique est déjà utilisée par [Zaki et Hsiao, 1999, Pei *et al.*, 2000].

Nous vérifions maintenant que l'algorithme MUSIC-DFS est correct :

Théorème 6 (Correction de MUSIC-DFS) MUSIC-DFS *est correct et complet.*

Preuve. La complétude de l'algorithme est assurée par le théorème 5 d'un point de vue global. D'un point de vue local, le théorème 4 garantissant que l'élagage est sûr, et le calcul des sous-intervalles (en particulier, la ligne 12 de LOCALSCAN) énumérant les motifs de $[X, Y]$ selon $\preceq_R$, aucun motif ne peut être manqué. Enfin, la consistance de l'algorithme est assurée par la vérification de la contrainte aux lignes 2 et 9 de LOCALSCAN (doublé du théorème 4 à la ligne 1). Ainsi, le théorème 6 est correct. □

D'autre part, MUSIC-DFS permet de quantifier la qualité des intervalles extraits avec une mesure basée sur des primitives m. Même si la valeur exacte de m sur chaque motif ne peut être connue, celle-ci est approximée en s'appuyant sur les bornes. En effet, pour un intervalle $[X, Y]$ et une mesure basée sur des primitives m, la valeur de m d'un motif $Z \in [X, Y]$ est approximée par $(\lceil m \rceil + \lfloor m \rfloor)/2$ (avec une erreur n'excédant jamais $(\lceil m \rceil - \lfloor m \rfloor)/2$). Nous voyons ainsi que MUSIC-DFS non seulement produit une représentation condensée des motifs satisfaisant une PBC, mais il donne aussi pour chaque motif une approximation de n'importe quelle mesure m basée sur des primitives avec une erreur de $(\lceil m \rceil - \lfloor m \rfloor)/2$. Le chapitre 9 offre des représentations condensées d'intervalles *exactes* et adéquates à certaines mesures.

8.3 Etude expérimentale de MUSIC-DFS

Cette étude expérimentale a pour objectif d'analyser les performances de MUSIC-DFS par rapport à d'autres solveurs et au type de contraintes utilisées. Nous identifions les contraintes où MUSIC-DFS est le plus compétitif et où la concision de la représentation condensée est la plus forte.

Une des originalité de MUSIC-DFS est de fournir des statistiques sur le nombre de motifs satisfaisant q_{AM}, le nombre de motifs satisfaisant $q \wedge q_{AM}$, le nombre d'intervalles, les nombres d'élagages sur un intervalle tentés et réussis. Nos expérimentations tirent profit de ces statistiques.

Toutes les expérimentations sont effectuées sur un ordinateur doté d'un processeur Xeon 2.2 GHz et de 3GB de mémoire RAM avec le système d'exploitation Linux.

8.3.1 Performances de MUSIC-DFS

Cette section analyse la rapidité d'extraction de MUSIC-DFS (sans utiliser les relaxations du chapitre 6). Dans un premier temps, nous le comparons à d'autres algorithmes. Ensuite, nous identifions les contraintes les plus efficacement traitées à l'aide de deux mesures (i.e., le taux de sélectivité et le taux de succès de l'élagage). Des expériences complémentaires sont aussi présentées à l'annexe C.

Comparaisons avec d'autres algorithmes

Aucun solveur ne proposant une généricité aussi forte que celle de MUSIC-DFS, il n'est pas possible de le comparer avec d'autres solveurs sur l'ensemble des contraintes du PBF. Une comparaison avec des algorithmes spécifiques traitant les contraintes monotones, succinctes, convertibles, etc serait longue et fastidieuse. Par ailleurs, certaines contraintes comme celle d'aire minimale ne disposent pas de solveurs dans la littérature.

Aussi, nous proposons de comparer MUSIC-DFS aux deux algorithmes d'extraction de motifs fréquents de références dans la littérature : APRIORI et ECLAT. En effet, à partir de l'extraction des motifs satisfaisant q_{AM}, il est possible d'obtenir les motifs satisfaisants $q \wedge q_{AM}$ par un post-traitement. Nous avons à nouveau utilisé les versions d'APRIORI et d'ECLAT réalisées par Borgelt et disponibles sur le site du FIMI. À titre indicatif, nous comparons également MUSIC-DFS à MUSIC (section 8.1.3).

La figure 8.1 compare l'efficacité des différents algorithmes pour l'extraction des seuls motifs fréquents. Plus précisément, l'axe des abscisses reporte le seuil de fréquence minimale et celui des ordonnées, les temps d'extraction. Cette expérience montre la qualité intrinsèque de chaque algorithme car ils effectuent tous la même tâche contrairement aux expériences suivantes où APRIORI et ECLAT ne peuvent pleinement pousser la contrainte q.

Même si MUSIC-DFS n'est pas optimisé pour la recherche des motifs fréquents comme c'est le cas d'APRIORI et d'ECLAT, il montre de très bonnes disponibilités. En particulier, MUSIC-DFS est nettement plus efficace qu'ECLAT sur `mushroom` car le bénéfice des classes d'équivalence est important. Ce gain est similaire à celui des algorithmes basés sur la fermeture de Galois. Sur `chess` où les classes d'équivalence sont moins avantageuses, ECLAT s'avère plus rapide que MUSIC-DFS, mais cette différence est peu significative.

La seconde expérience reportée sur la figure 8.2 observe les temps d'extraction des différents algorithmes pour les deux contraintes ($freq(X) \times count(X) \geq seuil$ et $sum(X.val)/count(X) \geq seuil$) en fonction du paramètre $seuil$. Pour `mushroom` (resp. `chess`) q_{AM} est fixé à $freq(X) \geq 100$

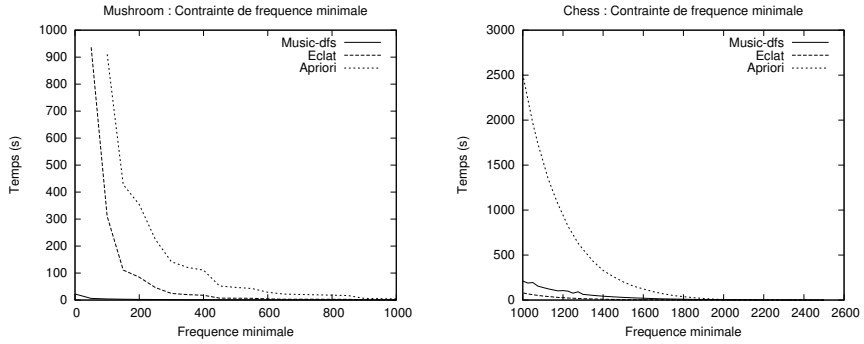


FIG. 8.1 – Impact du seuil de fréquence minimale sur les algorithmes.

(resp. $freq(X) \geq 1200$). En effet, cette contrainte anti-monotone poussée par les algorithmes APRIORI et ECLAT rend leurs extractions faisables (celles-ci ne sont pas faisables avec un seuil de 1). Pour ces derniers, le temps du post-traitement n'est pas inclus car il est souvent négligeable.

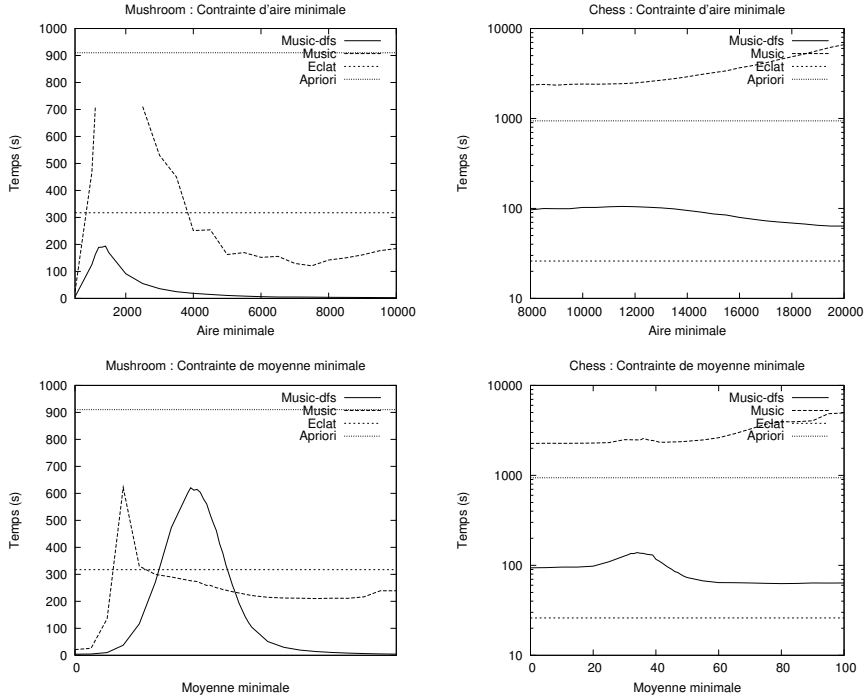


FIG. 8.2 – Comparaison de MUSIC-DFS avec d'autres algorithmes.

La première observation est que les temps d'extraction de MUSIC et MUSIC-DFS varient suivant la contrainte (à l'inverse de ceux d'APRIORI ou d'ECLAT). Cela démontre que l'élagage sur les intervalles issu de la contrainte q a un impact réel. Le paragraphe suivant explique en détail le comportement de MUSIC-DFS par rapport à la contrainte d'extraction.

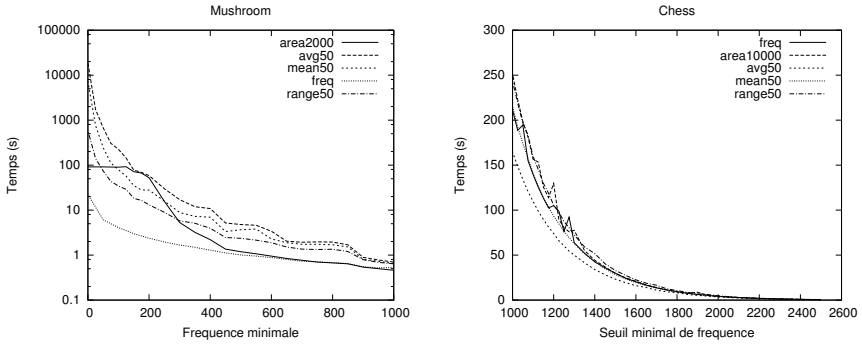
Sans surprise, MUSIC est généralement moins rapide que MUSIC-DFS. Cela s'explique par le chevauchement des intervalles pour MUSIC (section 8.1.3). De plus, lorsque le nombre de candidats devient trop important, les accès à la mémoire deviennent coûteux et MUSIC perd en

efficacité (c'est le cas pour la contrainte d'aire minimale sur `mushroom` pour un seuil compris entre 1200 et 2000).

APRIORI n'est jamais plus efficace que MUSIC-DFS et ne rivalise pas avec ce dernier. ECLAT est parfois plus efficace que MUSIC-DFS : cette situation se produit lorsque l'élagage issu de q_{AM} est prépondérant sur celui issu de q (e.g., pour la contrainte d'aire sur `chess`). Néanmoins, MUSIC-DFS surpasse régulièrement l'algorithme ECLAT (en particulier, sur `mushroom`) montrant le bénéfice de pousser pleinement la contrainte q.

Comportement de MUSIC-DFS en fonction de la fréquence minimale

Comme tout algorithme tirant bénéfice des conditions d'élagage issues des contraintes anti-monotones, l'efficacité de MUSIC-DFS est fortement liée à q_{AM}. La figure 8.3 quantifie l'efficacité de cet élagage pour un ensemble de 5 contraintes en prenant pour q_{AM} la contrainte de fréquence minimale. Remarquons que pour `mushroom`, l'axe des ordonnées utilise une échelle logarithmique.



freq :	$freq(X) \geq minfr$
areaα :	$freq(X) \times count(X) \geq \alpha$
avg50 :	$sum(X.val)/count(X) \geq 50$
mean50 :	$(max(X.val) + min(X.val))/2 \geq 50$
range50 :	$(max(X.val) - min(X.val)) \geq 50$

FIG. 8.3 – Impact de la fréquence sur l'algorithme MUSIC-DFS pour différentes contraintes.

Les courbes de la figure 8.3 montrent que moins la contrainte q_{AM} réduit l'espace de recherche, plus les temps d'extraction deviennent longs. Bien sûr, ce résultat était attendu. Dans le cas extrême, où tous les motifs sont extraits (i.e. $q_{AM} \equiv true$), l'extraction de MUSIC-DFS peut devenir très longue, mais ne faillit jamais. Sur `mushroom`, rappelons que contrairement à MUSIC-DFS, ECLAT (resp. APRIORI) ne peut pas descendre sous un seuil de fréquence minimale inférieur à 50 (resp. 100). Cette expérience démontre l'intérêt de pousser q_{AM} au cœur de MUSIC-DFS conjointement à la contrainte q. En particulier, elle renforce l'intérêt de l'usage d'une relaxation anti-monotone comme $\lceil q \rceil^{\top}$ proposée au chapitre 6.

D'autre part, on constate que la contrainte pour laquelle le temps d'exécution est le plus faible, est celle de la fréquence minimale. Contrairement aux autres contraintes, la contrainte de fréquence minimale est conservée pour tous les intervalles, i.e., pour un intervalle $[\mathbf{fr}_R(X), \mathbf{cl}_R(X)]$, on a toujours $[q]\langle \mathbf{fr}_R(X), \mathbf{cl}_R(X)\rangle$ qui est satisfait. De cette manière, l'élagage sur les intervalles est optimal et MUSIC-DFS atteint ses meilleures performances.

Comportement de MUSIC-DFS en fonction de la sélectivité

Nous avons observé sur la figure 8.2 que l'efficacité de l'extraction dépend du seuil choisi pour la contrainte. Afin de mieux comprendre ce comportement de MUSIC-DFS, nous introduisons

maintenant deux mesures. La première reflète la proportion de motifs qui satisfont la contrainte d'extraction q :

Définition 27 (Taux de sélectivité) *Le taux de sélectivité est le rapport entre le nombre de motifs satisfaisant $q \wedge q_{AM}$ et le nombre de motifs satisfaisant seulement q_{AM}.*

Cette définition 27 tient compte des seuls motifs satisfaisant q_{AM} plutôt que de tous les motifs présents dans la base de données afin de mieux analyser l'efficacité de MUSIC-DFS. La définition 27 donne une mesure de sélectivité comprise entre 0 et 1. Plus la sélectivité est proche de 1, moins la contrainte q est sélective car tous les motifs satisfaisant q_{AM} satisfont également q. À l'inverse, lorsque la sélectivité est proche de 0, la contrainte est très sélective car peu de motifs sont extraits parmi ceux satisfaisant q_{AM}.

Nous définissons maintenant une mesure pour rendre compte de l'efficacité de l'élagage sur les intervalles :

Définition 28 (Taux de succès de l'élagage) *Le taux de succès de l'élagage est le rapport entre le nombre de réussites de l'élagage sur les intervalles et le nombre de tentatives.*

Le taux de succès de l'élagage (ou taux d'élagage) est donc compris entre 0 et 1 et rend compte de l'efficacité de l'élagage sur les intervalles. Plus ce taux est grand, plus l'élagage sur les intervalles est efficace.

Nous avons reporté à la figure 8.4 pour les mêmes contraintes et les mêmes contextes transactionnels qu'à la figure 8.2, le temps d'extraction, le taux de sélectivité et le taux d'élagage, en fonction du seuil de la contrainte (abscisse des courbes). Le temps se réfère à l'axe des ordonnées de gauche et les deux taux, à l'axe des ordonnées de droite.

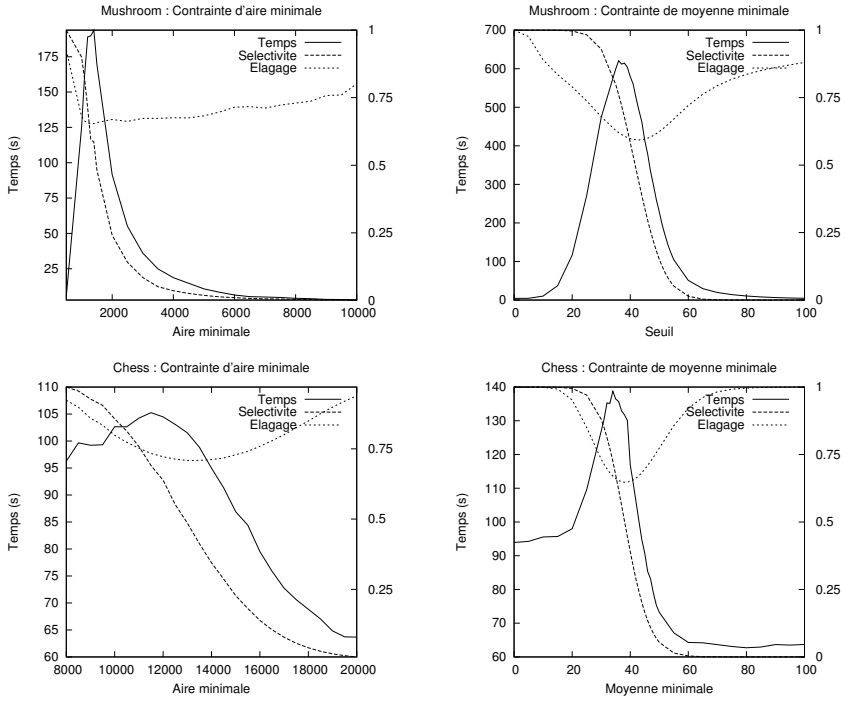


FIG. 8.4 – Comportement de MUSIC-DFS en fonction de la sélectivité.

Les différentes courbes de la figure 8.4 montrent que, pour chaque contrainte, le temps d'extraction forme une courbe en cloche. Le temps d'extraction maximal (i.e., le pic) intervient aux environs d'une sélectivité de 0.5, c'est-à-dire au point où environ la moitié des motifs satisfont la contrainte q. Une explication est, qu'à ce niveau, les classes d'équivalence sont très hétérogènes (certains motifs satisfont la contrainte, et d'autres pas) et l'élagage sur les intervalles est moins performant. On constate aussi que le minimum du taux d'élagage correspond au temps d'extraction maximal et une sélectivité de 0.5.

L'utilisateur est souvent intéressé par les motifs les plus significatifs au regard de sa contrainte d'extraction. En pratique, ces contraintes sont très sélectives (taux de sélectivité proche de 0) soit en augmentant les seuils, soit en combinant de multiples contraintes atomiques. Ainsi, l'extraction de motifs suffisamment significatifs est efficace.

Ces expériences (ainsi que celles proposées en annexe C) montrent à quel point, il est profitable de pousser la contrainte quand celle-ci est très sélective (ou que sa négation est très sélective). Par ailleurs, notre approche symétrique en exploitant à la fois l'élagage positif et l'élagage négatif, démontre sa complémentarité. Ainsi, pour une mesure m, les extractions des motifs suivant $m(X) \leq \mathit{seuil}$ ou $m(X) \geq \mathit{seuil}$ bénéficient des mêmes optimisations.

8.3.2 Condensation de la représentation

Cette section quantifie la concision de la représentation condensée d'intervalles des motifs contraints retournée par MUSIC-DFS. En plus, du contexte transactionnel considéré, elle montre que la condensation dépend fortement de la sélectivité de la contrainte. Au préalable, nous introduisons une nouvelle mesure afin de connaître le gain de la représentation condensée d'intervalles :

Définition 29 (Taux de condensation) *Le taux de condensation est le rapport entre le nombre d'intervalles et le nombres de motifs satisfaisant la contrainte.*

Lorsque le taux de condensation (abrégé condensation) approche de 0, la représentation condensée est extrêmement concise. Dans le cas inverse où le taux de condensation avoisine 1, la représentation obtenue est nettement moins avantageuse.

La figure 8.5 donne, pour les mêmes conditions expérimentales que précédemment (i.e., même contrainte q, même contrainte anti-monotone q_{AM} et même base de données $\mathbf{r}$), les courbes reportant le taux de sélectivité et le taux de condensation suivant les seuils des contraintes.

Un résultat important est que le taux de condensation est d'autant meilleur que le taux de sélectivité diminue. Autrement dit, moins une contrainte est sélective, plus la représentation condensée d'intervalles obtenue est concise. En effet, plus les motifs satisfaisant la contrainte sont nombreux, plus la probabilité qu'ils soient "adjacents" est grande. De cette manière, l'algorithme les regroupe plus efficacement.

Les intervalles produit par MUSIC-DFS ne sont pas forcément optimaux dans le sens où ils est parfois possible d'en fusionner plusieurs pour en constituer un plus large, sans perte de consistance par rapport à la contrainte. De telles fusions amélioreraient le taux de condensation.

8.4 Conclusion

Nous avons proposé une nouvelle forme d'élagage basée sur les intervalles pour réduire l'espace de recherche et automatisée grâce à l'opérateur d'élagage. L'introduction d'un nouvel opérateur de fermeture pour MUSIC-DFS, a permis de grandement améliorer l'efficacité de

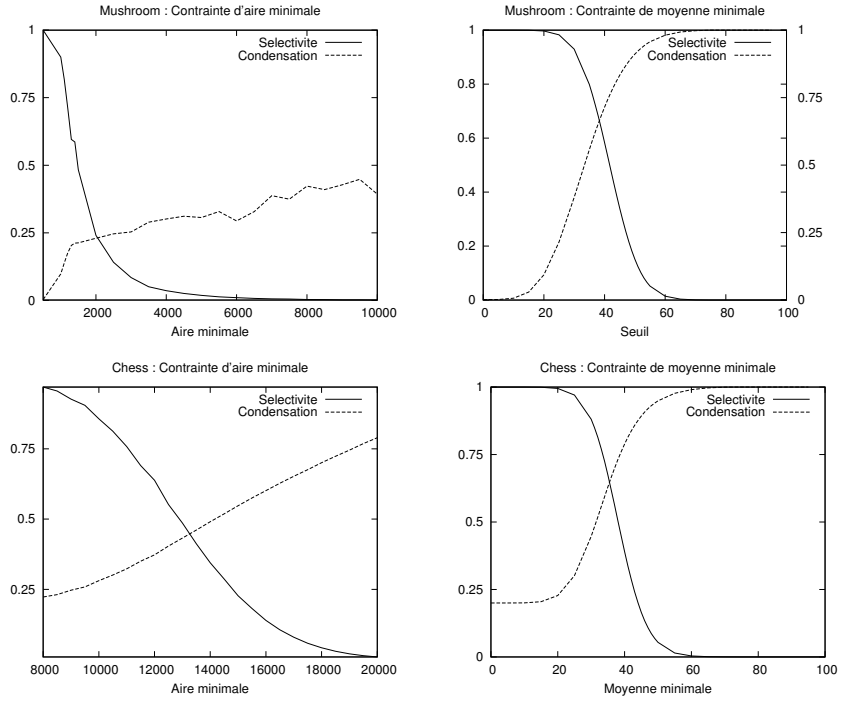


FIG. 8.5 – Condensation de MUSIC-DFS en fonction de la sélectivité.

notre approche initiale MUSIC. Par ailleurs, chaque motif satisfaisant la PBC est présent dans un seul intervalle de la représentation condensée produite par MUSIC-DFS. L'étude expérimentale de l'algorithme MUSIC-DFS a montré que son efficacité pratique est d'autant meilleure que la contrainte est soit très sélective, soit très faiblement sélective. La manière de construire les intervalles est cruciale. Dans le chapitre suivant, nous montrerons comment adapter la construction des intervalles afin que l'élagage sur les intervalles soit toujours réalisé avec diverses contraintes.

Deux perspectives importantes restent ouvertes. La première concerne l'adaptation de MUSIC-DFS pour effectuer des extractions itératives. L'idée est de tirer parti des extractions antérieures lorsqu'on en effectue une nouvelle avec, par exemple, un seuil différent. La version actuelle de MUSIC-DFS concerne le langage ensembliste et il serait intéressant de transposer les principes de MUSIC-DFS à d'autres langages. Clairement, cette tâche est réalisable pour les langages disposant d'un opérateur de fermeture (e.g., les séquences). En effet, tous les résultats sont alors généralisables. Pour les autres langages, la construction efficace des intervalles reste problématique (même si les motifs libres et fermés sont définis pour tous les langages).

Chapitre 9

Représentations condensées adéquates à une fonction

Sommaire

Ce chapitre montre que grâce à une généralisation de l'opérateur de fermeture, les représentations condensées de motifs (cf. la section 3.3) peuvent être étendues à un large ensemble de fonctions que nous appelons fonctions *conservées*. Rappelons qu'une représentation condensée exacte par rapport à une fonction permet d'inférer la valeur de cette fonction pour n'importe quel motif. Nous donnons l'algorithme MicMac qui permet d'extraire les représentations condensées *adéquates* à une fonction conservée. En généralisant la notion de représentation condensée à des fonctions autres que la fréquence, ce résultat étend les usages des représentations condensées. Par exemple, des tâches telles que la dérivation de règles d'association sont ainsi adaptables à d'autres mesures que la fréquence. D'autre part, nous expliquons comment le nouvel opérateur de fermeture permet d'optimiser les algorithmes Music et Music-dfs en améliorant la construction des intervalles.

La section 9.1 motive la généralisation des représentations condensées à des mesures autres que la fréquence tout en en montrant les difficultés. La section 9.2 introduit les fonctions conservées et le nouvel opérateur de fermeture. Ce dernier est nécessaire pour construire des représentations condensées adéquates aux fonctions conservées. L'extraction de ces représentations condensées est effectuée par l'algorithme MicMac décrit à la section 9.3.

Enfin, nous définissons la notion de motifs forts qui sont les motifs les plus significatifs d'une classe d'un jeu de données pour les mesures fondées sur la fréquence comme par exemple la confiance ou le lift.

9.1 Problématique des représentations condensées

9.1.1 Motivations

Le chapitre 3 consacré aux bases de données inductives a souligné l'importance des représentations condensées. Ces dernières facilitent les multiples usages des motifs. Par ailleurs, elles sont souvent plus efficaces à extraire que la collection des motifs représentés. Cependant comme nous l'avons indiqué à la section 3.4, ces représentations condensées sont soit dédiées aux contraintes monotones, soit à la fréquence. Dans ce chapitre, nous les généralisons à d'autres fonctions.

D'autre part, au chapitre précédent, nous avons vu que les algorithmes MUSIC et MUSIC-DFS utilisent de manière cruciale les représentations condensées pour construire des intervalles. Les expérimentations ont alors montré que la contrainte de fréquence minimale est plus efficacement extraite. L'élagage (positif ou négatif) sur les intervalles est toujours effectué car tous les motifs au sein d'un même intervalle possèdent la même fréquence. Dans ce chapitre, nous reprenons cette idée et nous l'adaptons à d'autres mesures pour obtenir des intervalles où les motifs possèdent les mêmes valeurs.

9.1.2 Illustration et intuitions clés

Avant de donner les intuitions de notre approche, observons la faiblesse des représentations condensées existantes pour représenter la collection de motifs par rapport à *min* (cf. la section A.1). Il n'est pas possible d'utiliser les bordures pour obtenir une représentation de la fonction *min*. En effet, les bordures sont limitées aux contraintes (anti-)monotones et ne permettent pas d'inférer les valeurs associées aux motifs pour des contraintes telles que la fréquence ou la valeur *min*. Par ailleurs, un exemple simple montre que l'usage des classes d'équivalence classiques basées sur la fréquence sont inefficaces. Dans le cas de la base de données du tableau 9.1, on a $h(AC) = ABCD$. Or, $min(AC.val) = 50$ est différent de $min(ABCD.val) = 10$. Les classes d'équivalence construites avec h sont inadaptées à la mesure *min* dans le sens où sa valeur varie entre deux motifs relevant d'une même classe d'équivalence. Afin de pallier ce problème, nous proposons de définir de nouvelles classes d'équivalence dont tous les motifs possèdent la même valeur pour une fonction donnée.

$\mathcal{D}$

Trans.	Items					
t_1	A	B			E	F
t_2	A				E	
t_3	A	B	C	D		
t_4	A	B	C	D	E	
t_5				D	E	
t_6			C			F

Item	A	B	C	D	E	F
val	50	30	75	10	30	15

TAB. 9.1 – Une base de données **r** constituée d'un contexte transactionnel $\mathcal{D}$ et d'une table de valeurs.

Comme pour la fréquence, un ou plusieurs représentants de chaque classe (e.g., le motif maximal) est alors choisi afin de constituer la représentation condensée désirée. Le point ardu et pourtant crucial est d'obtenir cette représentation condensée directement sans avoir à extraire tous les motifs. Ce dernier point distingue principalement notre approche de celle proposée pour le stockage dans [Diop, 2003].

Tout comme pour la fréquence, nous proposons d'exploiter un opérateur de fermeture pour former ces classes d'équivalence et en extraire facilement les motifs minimaux (i.e., libres) et/ou le motif maximal (i.e., fermé). Mais ce résultat ne doit pas être restreint à la fréquence : pour une fonction donnée f, ce nouvel opérateur de fermeture h_f garantit que pour tous motifs X et Y, on a $h_f(X) = h_f(Y) \Rightarrow f(X) = f(Y)$. De nombreux algorithmes d'extraction de représentations condensées adéquates à la fréquence et basés sur la fermeture h, peuvent directement être réutilisés avec les nouvelles fermetures (cf. la section 9.3). Au-delà de la problématique d'extraction, l'ensemble des usages fondés sur la fermeture adéquate à la fréquence est naturellement transposable à une fermeture adéquate à une fonction donnée. Typiquement, les règles d'association à prémisses minimales et à conclusion maximales formées chacune d'un motif libre et d'un motif fermé sont ainsi adaptables à d'autres mesures que la fréquence.

9.2 Représentations condensées adéquates à une fonction conservée

Dans cette section, nous montrons comment définir des représentations condensées adéquates à des fonctions quelconques : soit des mesures (e.g., la fréquence), soit directement les contraintes (e.g., la fréquence minimale). Dans un premier temps, nous introduisons les fonctions conservées. Nous définissons alors les fermetures adéquates à ces fonctions qui permettent d'aboutir aux représentations condensées.

9.2.1 Fonctions conservées

Nous proposons de nous focaliser sur certaines fonctions, que nous appelons fonctions *conservées* car nous verrons à la section suivante que celles-ci possèdent de bonnes propriétés pour définir les opérateurs de fermetures :

Définition 30 (Fonction conservée) *Une fonction f sur $\mathcal{L}_\mathcal{I}$ est conservée si et seulement si pour tout $a \in \mathcal{I}$ et pour tout $X \subseteq Y$ si $f(X \cup \{a\}) = f(X)$, alors $f(Y \cup \{a\})$ est égal à $f(Y)$.*

La propriété de conservation pour une fonction signifie qu'un item qui ne modifie pas la valeur de f pour un motif X, ne modifie pas la valeur de f pour une spécialisation de X. Cette propriété est centrale pour prouver les théorèmes 7 et 8 (cf. la section 9.2.2). Le terme *fonction conservée* est à prendre dans un sens générique. Selon les cas, nous parlons de primitive, mesure ou contrainte conservées. Typiquement, la fréquence vérifie la propriété énoncée par la définition 30. De nombreuses autres primitives satisfont également cette propriété comme la fréquence dans une sous-base (cf. la section 9.4.1), min, max, etc. À titre d'exemple, vérifions que la primitive min possède bien cette propriété. Soit un motif $X \in \mathcal{L}_\mathcal{I}$ et $a \in \mathcal{I}$ tel que $min(X.val) = min(X \cup \{a\}.val)$, la valeur val de a est donc supérieure à celle de X. Soit Y une spécialisation de X, on a alors $min(Y.val) \leq min(X.val)$ car la fonction min est décroissante. Comme $min(a.val) \geq min(X.val)$, $min(a.val)$ majore $min(Y.val)$. Ainsi, on conclut que $min(Y.val) = min(Y \cup \{a\}.val)$. Remarquons que *count* est aussi une primitive conservée mais nous verrons par la suite que sa fermeture adéquate est peu efficace (car aucun

motif X et item a ne satisfont $count(X \cup \{a\}) = count(X)$). Outre les primitives d'agrégats, certaines contraintes syntaxiques sont également conservées. Par exemple, la contrainte $X \cap A = \emptyset$ qui exclut les motifs X partageant des items avec A, est conservée.

Nous avons donné quelques fonctions conservées ci-dessus, mais il existe de nombreuses autre fonctions conservées. En particulier, les combinaisons de fonctions conservées sont souvent conservées. Par exemple, la mesure $max(X.val) - min(X.val)$ ou la contrainte $(min(X.val) + max(X.val))/2 \geq 50$ sont encore des fonctions conservées. En pratique, comme seuls les motifs présents dans la base de données sont utiles pour l'utilisateur, nous considérons donc le plus souvent la primitive *freq* en plus de la fonction désirée. Typiquement, plutôt que de considérer la seule fonction $min(X.val)$, on considère plutôt la fonction conservée $X \mapsto (min(X.val), freq(X))$. De cette manière, la fréquence aura aussi la même valeur au sein des classes d'équivalence formées par les nouveaux opérateurs de fermeture. Nous revenons sur ce point dans la section suivante en soulignant la différence entre la fermeture adéquate à *min* et celle adéquate à la fois à *min* et à *freq*.

9.2.2 Opérateurs de fermeture adéquats à une fonction conservée

Pour une fonction quelconque, il existe toujours un opérateur de fermeture qui la conserve. Naïvement, l'identité (qui est un opérateur de fermeture) conserve n'importe quelle fonction (i.e., pour une fonction f, on a toujours $f(Id(X)) = f(X)$ car $Id(X) = X$). Malheureusement cet opérateur de fermeture a peu d'intérêt si il est utilisé pour construire des représentations condensées, car il n'apporte aucun gain de condensation. En effet, comme chaque motif est son propre représentant avec Id, chercher cette représentation revient à extraire tous les motifs. A contrario, la notion de fonction conservée permet de définir un nouvel opérateur de fermeture alternatif à l'identité conduisant à une condensation réelle.

Considérons une fonction conservée et un item a qui n'affecte ni la valeur d'un motif X (i.e., $f(X \cup \{a\}) = f(X)$), ni aucune de ses spécialisations (i.e., $f(Y \cup \{a\}) = f(Y)$ où $Y \supseteq X$). Cet item a semble alors associé à tous les motifs $Y \supseteq X$. Une forme de dépendance caractérisée par f émerge et justifie un regroupement entre X et a. La définition 31 traduit cette intuition en associant à chaque fonction conservée un opérateur de fermeture comme le justifiera le théorème 8 :

Définition 31 (Fermeture adéquate) *La fermeture h_f adéquate à une fonction conservée f associe à chaque motif $X \in \mathcal{L}_\mathcal{I}$, tous les items a tels que $f(X \cup \{a\}) = f(X)$:*

$$h_f(X) = \{a \in \mathcal{I} | f(X \cup \{a\}) = f(X)\}$$

La fermeture h_f associe à un motif tous les items qui dépendent de lui. À travers l'égalité $f(X \cup \{a\}) = f(X)$, cette définition exploite comme pour les classes d'équivalence de fréquence, la constance locale de la fonction f. D'ailleurs, pour la mesure de fréquence, la fermeture h_{freq} coïncide avec la connexion de Galois h (cf. la section 3.3). En effet, l'ajout à un motif X de tous les items a tels que $freq(X \cup \{a\}) = freq(X)$ associe à X sa spécialisation la plus large et de même fréquence.

Prenons maintenant l'exemple moins usuel de la primitive conservée *min*. La fermeture adéquate à *min* est $h_{min}(X) = \{a \in \mathcal{I} | min(X \cup \{a\}) = min(X)\}$. Ainsi, la fermeture adéquate à *min* de BDE est $ABCDEF$ car tous les items ont une valeur *val* supérieure à celle de D. Le motif $ABCDEF$ n'étant pas présent dans le contexte transactionnel $\mathcal{D}$, on utilise plutôt $h_{min,freq}(X) = \{a \in \mathcal{I} | (min(X \cup \{a\}), freq(X \cup \{a\})) = (min(X), freq(X))\} = \{a \in \mathcal{I} | min(X \cup \{a\}) = min(X) \wedge freq(X \cup \{a\}) = freq(X)\}$ qui considère simultanément les primitives *min* et

freq. De cette manière, l'item F est alors exclu de $h_{min,freq}$ car $freq(BDE \cup \{F\}) \neq freq(BDE)$ et le motif $h_{min,freq}(BDE) = ABCDE$ est bien dans $\mathcal{D}$. La figure 9.1 illustre les deux fermetures des motifs BE et BDE. Les abscisses correspondent aux motifs suivant l'ordre de la spécialisation et les ordonnées classent les items $\mathcal{I}$ suivant la valeur croissante de *val*. Sur le graphique de gauche, tous les points au-dessus du point noir le plus bas (i.e., l'item du motif dont la valeur pour *val* est minimale) appartiennent à la fermeture du motif. En revanche, sur la partie droite, la fréquence élimine l'item F.

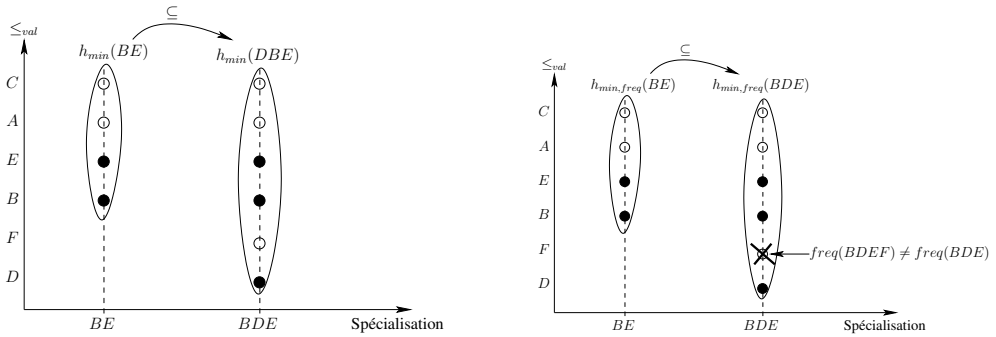


FIG. 9.1 – Exemple des fermetures h_{min} et $h_{min,freq}$.

Remarquons que h_{count} est l'identité car pour tout X et tout item a, on a toujours $count(X) \neq count(X \cup \{a\})$. Cette fermeture n'est donc pas efficace et ne permet pas d'aboutir à une représentation condensée satisfaisante (i.e., où une classe d'équivalence contient plusieurs motifs). Nous verrons dans la section expérimentale qu'à l'inverse de nombreuses fermetures apportent un gain significatif.

Sur la figure 9.1, le motif BE (resp. BDE) a la même valeur pour *min* (correspondant au point le plus bas) que celle de ses fermetures h_{min} ou $h_{min,freq}$. Plus généralement, la propriété suivante montre que le motif X et sa fermeture $h_f(X)$ possèdent la même valeur pour la fonction conservée f :

Théorème 7 (h_f conserve f) *Soit f une fonction conservée, la fermeture h_f conserve f i.e., pour tout $X \in \mathcal{L}_{\mathcal{I}}$, on a $f(h_f(X)) = f(X)$.*

Preuve. Soit $X \in \mathcal{L}_{\mathcal{I}}$, on fixe n items tels que $h_f(X) = X \cup \{a_1, \ldots, a_n\}$. Si $n = 0$, on a $h_f(X) = X$ et on a bien $f(h_f(X)) = f(X)$. Supposons que pour $n > 0$, on a bien $f(X) = f(X \cup \{a_1, \ldots, a_n\})$ et considérons le cas $n+1$. Comme $a_{n+1} \in h_f(X)$, on a $f(X \cup \{a_{n+1}\}) = f(X)$. Or $X \cup \{a_1, \ldots, a_n\}$ est une spécialisation de X. La définition 30 donne que $f(X \cup \{a_1, \ldots, a_n\}) = f(X \cup \{a_1, \ldots, a_{n+1}\})$. De plus, par hypothèse, $f(X)$ est égale à $f(X \cup \{a_1, \ldots, a_n\})$. Donc on obtient que $f(X)$ est égale à $f(X \cup \{a_1, \ldots, a_{n+1}\})$. Par récurrence, on conclut que le théorème 7 est juste. □

Le théorème 7 est essentiel pour la justesse des représentations proposées dans la section suivante. Illustrons le théorème 7 sur la base de données **r** du tableau 9.1. Comme $h_{min,freq}(BDE) = ABCDE$, les motifs BDE et $ABCDE$ ont le même minimum pour *val* (i.e., 10) et la même valeur pour la fréquence (i.e., 1). Tous les motifs X ayant la même valeur pour *min* et pour *freq* que $h_{min,freq}(X)$, les motifs X et $h_{min,freq}(X)$ ont aussi la même valeur pour les combinaisons des primitives *min* et *freq* telle que par exemple, $min(X.val) \times freq(X)$. De manière

plus générale, n'importe quelle combinaison de primitives conservées (e.g., les primitives de haut niveau) est conservée par la fermeture adéquate à chacune des primitives :

Corollaire 1 (Conservation des combinaisons de fonctions conservées) *Soit $f = F(f_1, \ldots, f_n)$ où $f_1, \ldots, f_n$ sont des fonctions conservées, la fermeture $h_{f_1,\ldots,f_n}$ conserve f.*

Preuve. Soit $f = F(f_1, \ldots, f_n)$. On note h_f la fermeture $h_{f_1,\ldots,f_n}$ adéquate aux fonctions conservées $f_1, \ldots, f_n$. Soit $X \in \mathcal{L}_\mathcal{I}$, $f(h_f(X)) = F(f_1(h_f(X)), \ldots, f_n(h_f(X)))$. Comme pour $i \in \{1, \ldots, n\}$, le théorème 7 indique que $f_i(h_f(X)) = f_i(X)$, $f(h_f(X)) = F(f_1(X), \ldots, f_n(X))$. Ainsi, on conclut que $f(h_f(X)) = f(X)$. □

Ce résultat signifie, par exemple, que la fermeture $h_{min,max}$ adéquate à *min* et à *max* est également adéquate à $h_{max(X.val)-min(X.val)}$ ou $h_{(max(X.val)+min(X.val))/2\geq 50}$. Par ailleurs, on observe alors que $h_{min,max}(X)$ correspond à l'intersection des motifs $h_{min}(X)$ et $h_{max}(X)$. Ce résultat se généralise naturellement à n'importe quelle fonction conservée $F(f_1, \ldots, f_n)$ où la fermeture $h_{F(f_1,\ldots,f_n)}$ est égale à l'intersection des fermetures h_{f_i}. Dans la suite, les combinaisons de fonctions conservées sont souvent confondues avec les fonctions conservées. La fermeture adéquate à la combinaison de fonctions conservées $F(f_1, \ldots, f_n)$ est équivalente à $h_{f_1,\ldots,f_n}$. Le corollaire 1 permet donc de définir des représentations condensées adéquates aux combinaisons de fonctions conservées. Il étend considérablement la diversité des fonctions dont on peut extraire des représentations condensées adéquates.

En nous appuyant sur la propriété 1 (cf. la page 38), nous montrons maintenant que la fonction h_f est bien un opérateur de fermeture :

Théorème 8 (Opérateur de fermeture adéquat à une fonction conservée) *Soit f une fonction conservée, la fonction h_f associant à X tous les items a tels que $f(X \cup \{a\}) = f(X)$, est un opérateur de fermeture.*

Preuve. *Extensivité :* Soit $X \in \mathcal{L}_\mathcal{I}$ et $a \in X$, on a $f(X \cup \{a\}) = f(X)$ car a est inclus dans X et $X \cup \{a\} = X$. *Idempotence :* Soit $X \in \mathcal{L}_\mathcal{I}$ et $a \in h_f(h_f(X))$, on a donc $f(h_f(X) \cup \{a\}) = f(h_f(X))$. Or $f(h_f(X)) = f(X)$ (théorème 7), on montre alors que $f(X \cup \{a\}) = f(h_f(X) \cup \{a\})$ avec une récurrence similaire à celle de la preuve du théorème 7. Ainsi, on conclut que $f(X \cup \{a\}) = f(X)$. *Isotonie :* Soit X et Y deux motifs ensemblistes tels que $X \subseteq Y$, soit $a \in \mathcal{I}$ tel que $a \in h_f(X)$. Par définition de la fonction h_f, on a $f(X \cup \{a\}) = f(X)$. La définition 30 garantit alors que $f(Y \cup \{a\}) = f(Y)$ (car Y est une spécialisation de X). Ainsi, l'item a appartient à $h_f(Y)$. □

La définition 31 intervient de manière cruciale pour garantir l'isotonie de l'opérateur h_f. La figure 9.1 illustre l'isotonie des fermetures h_{min} et $h_{min,freq}$. En effet, on observe bien que $BE \subseteq BDE$ et $h_{min}(BE) \subseteq h_{min}(BDE)$ (de même, $h_{min,freq}(BE) \subseteq h_{min,freq}(BDE)$).

Grâce au théorème 8, les résultats sur les opérateurs de fermeture sont donc applicables à h_f. Cela signifie que h_f structure les motifs du treillis en classes d'équivalence : deux motifs X et Y sont équivalents selon h_f ssi $h_f(X) = h_f(Y)$. Dans la section suivante, les représentations condensées seront basées sur les motifs minimaux et maximaux de ces classes. Nous verrons ensuite que la contrainte "être un motif minimal" est anti-monotone et nous exploitons cette propriété dans l'algorithme MicMac (cf. la section 9.3.1).

9.2.3 Représentations condensées exactes et adéquates à une fonction conservée

Définition

Nous utilisons maintenant les opérateurs de fermeture adéquats aux fonctions conservées (ou à leurs combinaisons) afin de définir deux sortes de représentations condensées adéquates. Comme pour les représentations condensées des motifs fréquents à la section 3.3, les motifs extrêmes des classes d'équivalence associées à h_f sont de bons représentants. Ils permettent d'inférer facilement la valeur de f pour un motif quelconque et ils sont faciles à extraire (cf. la section 9.3). Nous en donnons maintenant la définition :

Définition 32 (Motifs libres et fermés adéquats) *Soit f une fonction conservée, les motifs minimaux (resp. maximaux) des classes d'équivalence associées à h_f sont appelés les motifs libres (resp. fermés) adéquats à f.*

Un seul motif est fermé dans une classe d'équivalence car la fermeture est idempotente. Le motif fermé de la classe d'équivalence à X correspond donc à $h_f(X)$. En revanche, plusieurs libres peuvent coexister au sein de la même classe d'équivalence. Pour la fermeture $h_{min,freq}$, nous avons mentionné que $ABCDE$ est un motif fermé adéquat à min (et à $freq$). Dans la même classe d'équivalence, les motifs libres adéquats à min (et à $freq$) sont ADE, BDE et CDE. Toutes les généralisations de ces derniers ont soit une valeur minimale plus grande, soit une fréquence plus élevée.

De même que pour la fréquence, les motifs libres ou fermés adéquats donnent des représentations condensées, mais ces dernières sont alors adaptées à la fonction f choisie :

Définition 33 (Représentations condensées adéquates de motifs) *Soit f une fonction conservée, l'ensemble de tous les motifs libres (ou fermés) adéquats à f constitue une représentation condensée adéquate à f.*

Plus précisément, la représentation condensée adéquate à f de motifs fermés regroupe tous les motifs fermés en leur associant la valeur f. La valeur f d'un motif X est alors égale à celle de $h_f(X)$ où $h_f(X)$ correspond au plus petit motif contenant X. De manière similaire, la représentation condensée adéquate à f de motifs libres est constituée de chaque motif libre auquel on associe sa valeur f. Comme pour la représentation condensée des motifs libres usuels, une démarche en deux temps permet de retrouver la valeur d'un motif. Tout d'abord, la bordure négative des motifs présents dans la base de données assure la présence du motif X dans la base de données. Ensuite, la valeur f de ce motif se déduit alors en prenant la valeur du plus large motif libre contenu dans X. Comme plusieurs libres peuvent appartenir à la même classe d'équivalence (contrairement à l'unicité du motif fermé), la représentation condensée des motifs fermés est plus compacte que celle des motifs libres.

Dans la pratique, l'utilisateur est généralement intéressé par les motifs présents dans la base de données, aussi nous considérons la fréquence conjointement à la fonction dont on souhaite tenir compte. Malheureusement, cette considération de la fréquence détériore la qualité des représentations condensées adéquates à f par rapport à celles adéquates à la seule fréquence. En effet, nous avons remarqué à la section précédente que la fermeture $h_{f_1,\ldots,f_n}$ est égale à $h_{f_1} \cap \ldots \cap h_{f_n}$. Cette relation donne, par exemple, que les motifs fermés sont plus petits pour les combinaisons de primitives $h_{min,freq}$ que pour h_{freq} seule. Ainsi, les classes d'équivalence selon $h_{min,freq}$ regroupent moins de motifs que celles selon h_{freq}. Comme elles doivent couvrir un espace similaire, elles sont donc plus nombreuses. En général, les représentations condensées

adéquates à une fonction conservée sont donc moins concises que celles correspondant à la fermeture usuelle. Cette observation se vérifiera dans la partie expérimentale (cf. la section 9.3.2).

Intérêts et usages

Ces deux représentations condensées adéquates à une fonction dégagent un intérêt pour nos travaux à travers l'optimisation de MUSIC. Mais surtout, elles permettent d'étendre des travaux relatifs aux usages des représentations condensées.

La plupart des usages classiques des représentations condensées de la fréquence s'étendent naturellement à ces nouvelles représentations condensées. L'usage le plus classique des représentations condensées de motifs libres et fermés est probablement la dérivation des règles d'association [Agrawal et Srikant, 1994]. Rappelons que la règle d'association $X \rightarrow Y$ (où $X \cap Y = \emptyset$) est à prémisse minimale et conclusion maximale si à confiance égale (i.e., $freq(X \cup Y)/freq(X)$), aucune prémisse plus générale que X ne conclut sur Y et aucune conclusion plus spécifique que Y n'est la conclusion de X. Pour obtenir de telles règles (exactes ou approximées), les motifs libres coïncident avec les prémisses minimales et les motifs fermés (privés de leurs prémisses), avec les conclusions maximales [Bastide *et al.*, 2000, Zaki, 2000a]. Avec les motifs libres et les motifs fermés adéquats, ce principe se généralise à toute fonction conservée et il est désormais possible d'obtenir des règles adaptées à une fonction. Par exemple, en prenant les motifs libres adéquats à $max(X.prix)$ et les motifs fermés adéquats à $min(X \cup Y.prix)$, la règle $X \rightarrow Y$ désigne la prémisse minimale de prix $max(X.prix)$ impliquant l'achat des articles Y dont le prix minimal est $min(X \cup Y.prix)$. On sélectionne alors les règles dont la prémisse n'excède pas un certain prix et dont la conclusion dépasse le prix minimal des articles considérés comme chers. Les règles obtenues permettent de détecter les articles bon marché amenant à l'achat d'articles plus coûteux.

Dans la section 1.2.3, nous avons évoqué la construction de modèles descriptifs et prédictifs. Plusieurs dont l'algorithme ECCLAT [21] [Durand et Crémilleux, 2002] utilisent des motifs fermés. Le changement de fermeture permet alors d'aboutir à des modèles contraints. Par exemple, pour ECCLAT, l'usage de motifs adéquats à f construit une catégorisation où chaque groupe est homogène selon f. La section 11.2 s'inspire de ce principe pour résumer une base de données relative à la fibrose du foie en respectant l'impact des différents stades de la maladie.

Un autre intérêt des représentations condensées adéquates est de permettre une construction d'intervalles optimisant MUSIC. La première étape est de reconnaître si la PBC exprimée par l'utilisateur est une fonction conservée (ou une combinaison de fonctions conservées). Pour cela, un parcours de l'arbre syntaxique de la contrainte détermine les primitives terminales $p_1, \ldots, p_n$. Si ces dernières sont toutes des primitives conservées, la fermeture $h_{p_1,\ldots,p_n}$ est alors choisie pour construire les intervalles adéquats (en s'appuyant sur le corollaire 1). Avec l'algorithme en profondeur MUSIC-DFS, cette fermeture doit être adaptée à la fermeture préfixée. Il suffit alors de considérer l'intersection des fermetures $\mathbf{cl}_R$ et $h_{p_1,\ldots,p_n}$ qui donne une nouvelle fermeture facilement calculable avec un parcours en profondeur. Bien entendu, deux motifs au sein d'une classe d'équivalence construite avec $\mathbf{cl}_R \cap h_{p_1,\ldots,p_n}$ ont encore des valeurs égales pour chaque primitive p_i. Plus généralement, certaines méthodes d'extractions de motifs vues au chapitre 3 et fondées sur la fermeture de Galois bénéficient de ces nouvelles fermetures adéquates. Nous en donnons un exemple à la section suivante en reprenant un algorithme classique d'extraction de motifs libres et fermés.

[21] C'est algorithme est différent de ECLAT [Zaki, 2000b].

Nous ne développons pas davantage les usages possibles des représentations condensées adéquates aux fonctions conservées. Cependant, nous pensons que ces usages sont très prometteurs. L'adéquation à une fonction conservée introduit, au sein des approches basées sur la fermeture, une sémantique additionnelle sur les motifs (locaux ou globaux) voire les modèles.

9.3 Algorithme d'extraction : MicMac

Cette section propose un algorithme d'extraction de représentations condensées adéquates aux fonctions conservées et évalue ces représentations.

9.3.1 Description de l'algorithme MicMac

L'objectif de l'algorithme MicMac (MInimal Constrained and MAximal Constrained patterns) est d'extraire des représentations condensées adéquates à une fonction conservée f (e.g., une contrainte conservée). En fait, il fournit tous les motifs libres adéquats à f (i.e., les motifs minimaux au sens de l'inclusion des classes d'équivalence) et les complète pour obtenir les motifs fermés adéquats correspondant (i.e., les motifs maximaux). Cet algorithme est donc entièrement indépendant de Music et Music-dfs présentés dans le chapitre précédent.

Pour illustrer la possibilité de réutiliser les méthodes traditionnelles liées à la fermeture de Galois, nous avons choisi d'adapter l'algorithme AC-miner [Boulicaut et Bykowski, 2000] (proche de Close [Pasquier *et al.*, 1999]). En effet, MicMac est basé sur les mêmes principes : (1) algorithme par niveaux et (2) restriction de l'espace de recherche avec l'anti-monotonie de la liberté adéquate à une fonction conservée. L'originalité de MicMac réside dans l'utilisation de la fermeture h_f.

Avant de détailler l'algorithme, la propriété suivante montre que la liberté (i.e., "être-libre") selon n'importe quel opérateur de fermeture est une contrainte anti-monotone :

Propriété 18 (Anti-monotonie de la liberté) *Soit h un opérateur de fermeture, la contrainte de liberté selon h est anti-monotone.*

Preuve. Soit h un opérateur de fermeture. Soit $X \in \mathcal{L}_{\mathcal{I}}$ non-libre par rapport à h. Soit Y une spécialisation de X. Tout d'abord, il existe $Z \subset X$ tel que $h(X) = h(Z)$ car X n'est pas libre. On a $h(Z \cup (Y \backslash X)) = h(h(Z) \cup (Y \backslash X))$ (à cause de l'isotonie et de l'idempotence). Or $h(Z) = h(X)$, on obtient que $h(h(Z) \cup (Y \backslash X)) = h(h(X) \cup (Y \backslash X))$. À nouveau, l'isotonie et l'idempotence donnent $h(h(X) \cup (Y \backslash X)) = h(X \cup (Y \backslash X) = h(Y)$. Ainsi, Y n'est pas libre et donc, la propriété 18 est correcte. □

Le résultat de cette propriété offre une condition d'élagage naturelle en utilisant la condition d'élagage associée à une contrainte anti-monotone (cf. la condition d'élagage 1 de la page 24).

L'algorithme 4 formalise le parcours par niveaux dont l'élagage des motifs s'effectue grâce à la contrainte de liberté suivant h_f. Il retourne l'ensemble des couples des motifs libres et de leurs fermetures. Bien entendu, dans la pratique, chacun de ces couples est aussi accompagné de la valeur de f. Notons aussi que l'algorithme MicMac tolère une contrainte anti-monotone additionnelle. En utilisant la contrainte de fréquence minimale, cette dernière permet par exemple de se limiter à une représentation condensée des motifs fréquents, adéquate à la fonction f.

Détaillons maintenant chacune des lignes de l'algorithme 4. À l'initialisation, les candidats $\mathcal{C}_1$ correspondent aux items. L'indice i désigne pour chaque itération, la longueur des motifs libres $\mathcal{F}_i$ (satisfaisant q_{AM}). Au début, cet indice est donc fixé à 1 (ligne 2) et il est incrémenté

Algorithm 4 MicMac

Input: Une fonction conservée f, une contrainte anti-monotone q_{AM} et une base de données $\mathbf{r}$
Output: Retourne tous les motifs libres adéquats à f satisfaisant q_{AM} et leurs fermetures adéquates à f

1: $\mathcal{C}_1 := \mathcal{I}$
2: $i := 1$
3: **while** $\mathcal{C}_i \neq \emptyset$ **do**
4: $\quad \mathcal{F}_i := \{X \in \mathcal{L}_\mathcal{I} \mid X \in \mathcal{C}_i$ et X est libre selon h_f et satisfait $q_{AM}\}$
5: $\quad \mathcal{C}_{i+1} := \{X \in \mathcal{L}_\mathcal{I} \mid \forall Y \subset X,$ on a $Y \in \bigcup_{j \leq i} \mathcal{F}_j\} \setminus \bigcup_{j \leq i} \mathcal{C}_j$
6: $\quad i := i + 1$
7: **od**
8: **return**$\{(X, h_f(X)) \mid X \in \bigcup_{j<i} \mathcal{F}_j\}$

à chaque itération (ligne 6). Le processus est itéré tant qu'il reste des candidats (ligne 3). Pour chaque itération, les motifs libres adéquats à f et satisfaisant la contrainte q_{AM} sont sélectionnés parmi les candidats $\mathcal{C}_i$ (ligne 4). Ensuite, les candidats du niveau suivant (i.e., $\mathcal{C}_{i+1}$) sont générés en fusionnant des motifs libres plus petits $\mathcal{F}_j$ (ligne 5). En privant $\mathcal{C}_{i+1}$ de $\bigcup_{j \leq i} \mathcal{C}_j$, on veille à ne pas générer 2 fois le même motif. Enfin, l'ensemble des libres adéquats à f satisfaisant la contrainte q_{AM} est retourné à la ligne 8 en leur associant leur fermeture adéquate à f.

Le théorème suivant montre que l'algorithme MicMac est correct et complet :

Théorème 9 (Correction de MicMac**)** *L'algorithme* MicMac *est correct et complet.*

Preuve. La conjonction de deux contraintes anti-monotones (i.e., la liberté et q_{AM}) étant encore anti-monotone, l'algorithme par niveaux [Mannila et Toivonen, 1997] garantit que tous les motifs libres adéquats à f et satisfaisant q_{AM} sont bien extraits. Comme l'algorithme retourne chaque motif libre accompagné de sa fermeture adéquate, l'algorithme MicMac est correct. □

9.3.2 Expériences

Le principal enjeu de ces expériences est d'estimer la concision des représentations condensées adéquates aux fonctions conservées. Nous testerons peu les performances de l'algorithme MicMac car les atouts et les limites de ce type d'algorithmes sont déjà connus.

Comme dans les chapitres précédents, nous utilisons les jeux de données `mushroom` et `chess` (cf. annexe B). Nous les complétons alors avec une table de valeurs *val* générées aléatoirement entre 0 et 100, nécessaire pour certaines mesures (e.g., *min*). Toutes les expériences sont effectuées sur un ordinateur doté d'un processeur Xeon 2.2 GHz et de 3GB de mémoire RAM avec le système d'exploitation Linux. Nous utilisons à nouveau l'algorithme Eclat comme algorithme de référence, même si ce dernier extrait seulement des motifs fréquents. En particulier, en donnant le nombre de motifs fréquents, il permet de mesurer la concision des représentations condensées adéquates.

Performances de MicMac

Dans cette première expérience, la rapidité d'extraction des représentations condensées adéquates est comparée à celle de l'extraction des motifs fréquents. Pour cela, la contrainte de fréquence minimale est choisie comme contrainte q_{AM} de MicMac. Cette même contrainte

est également poussée par l'algorithme Eclat. Enfin, nous testons 4 représentations condensées adéquates par rapport à h_{freq} (notée *freq* dans les légendes des courbes), $h_{min,freq}$ (notée min), $h_{max,freq}$ (notée max) et $h_{min,max,freq}$ (notée min, max). Les différents temps d'extractions par rapport au seuil de fréquence minimale sont reportés sur la figure 9.2. Notons que les échelles des ordonnées sont logarithmiques.

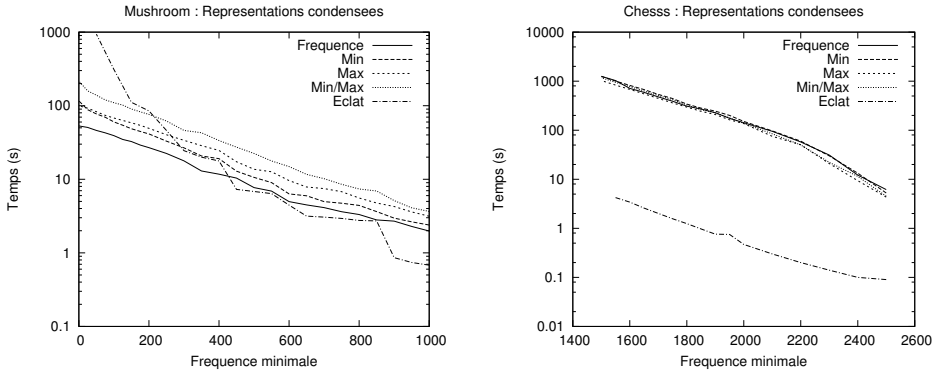


Fig. 9.2 – Comparaison des performances entre MicMac et Eclat.

Sur `mushroom`, Eclat est plus rapide pour les seuils de fréquence minimale élevés. Dans cette situation, le coût du calcul de la fermeture de l'algorithme MicMac est prépondérant sur le gain qu'il apporte (via la contrainte de liberté). Malgré ce dernier atout, lorsque le seuil de fréquence minimale devient très bas, l'extraction des représentations condensées adéquates est plus rapide que celle de tous les motifs fréquents du jeu de données. Pour `chess`, l'algorithme MicMac n'est visiblement pas adapté car il est dépassé par Eclat.

Cependant, la figure 9.2 montre surtout que les temps d'extraction des représentations condensées adéquates sont comparables. En particulier, leur temps d'extraction est proche de celui de la classique représentation condensée des motifs fréquents.

L'intérêt des représentations condensées adéquates réside aussi par leur gain qualitatif. En effet, le paragraphe suivant montre que la taille des représentations condensées adéquates est très restreinte.

Concision des représentations condensées adéquates

Dans les mêmes conditions expérimentales, nous comparons maintenant le nombre de motifs des représentations condensées de libres et de fermés adéquates par rapport au nombre de motifs fréquents. La figure 9.3 donne la taille des quatre représentations condensées adéquates (i.e., adéquates à h_{freq}, $h_{min,freq}$, $h_{max,freq}$ et $h_{min,max,freq}$) et de la collection de tous les motifs fréquents en fonction du seuil de fréquence minimale. Les courbes à gauche (resp. droite) sont relatives aux motifs libres (resp. fermés). À nouveau, une échelle logarithmique est choisie pour les axes des ordonnées.

Bien sûr, la taille des représentations comme le nombre de motifs augmente lorsque la fréquence diminue. Globalement, le nombre de motifs est identique pour les représentations condensées de libres et de fermés. Seule la courbe reportant le nombre de motifs contenus dans le jeu de données se détache nettement sur les deux graphiques. Toutes les représentations condensées adéquates sont environ 1000 fois moins grandes sur `mushroom`. Les quatre représentations condensées adéquates ont des tailles similaires quelque soit le seuil de

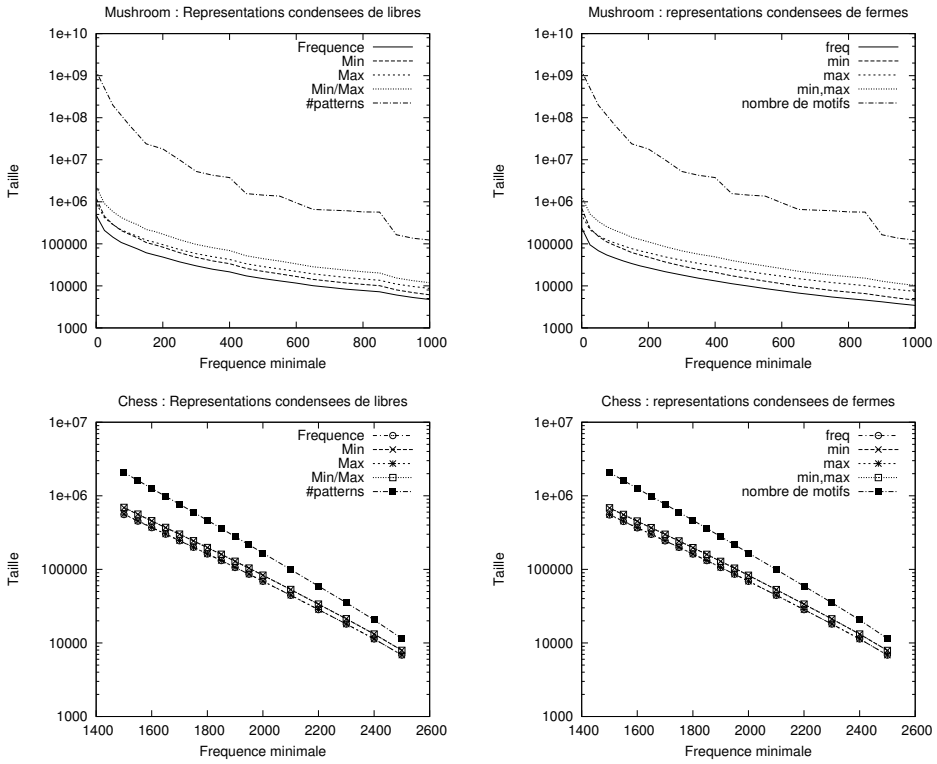


FIG. 9.3 – Concision des représentations condensées adéquates en fonction du seuil de fréquence minimale (à gauche, les représentations basées sur les motifs libres et à droite, celles basées sur les motifs fermés).

fréquence minimale et le type de motif (i.e., libre ou fermé). Néanmoins, la représentation condensée adéquate à la fréquence est la plus compacte des représentations. Comme nous l'avons expliqué à la section 9.2.3, les autres représentations condensées, qui tiennent aussi compte de la fréquence, ont des classes d'équivalence plus resserrées et donc plus nombreuses. De la même manière, la représentation condensée adéquate simultanément à min et à max a une cardinalité plus importante que celle adéquate à min ou à max. Les représentations condensées de motifs fermés sont à peine plus concises que celles de motifs libres.

9.4 Cas particulier des mesures de fréquences

Cette section s'intéresse aux mesures de fréquences qui sont un cas particulier de combinaisons de fonctions conservées. Ces mesures sont très utilisées pour évaluer par exemple la qualité des règles de classification ou de caractérisation. Parmi ces mesures de fréquences, nous mettons alors en évidence les mesures fortes et montrons leurs bonnes propriétés.

9.4.1 Mesures de fréquences

Dans de nombreuses applications, l'objectif est de rechercher des motifs caractérisant une partie de la base de données par rapport à une autre (e.g., caractérisation de classes). Au sein d'un même contexte transactionnel, on distingue alors plusieurs sous-bases correspondant aux différentes classes. Par exemple, le tableau 9.2 présente le contexte transactionnel $\mathcal{D}$ subdivisé en deux sous-bases notées respectivement $\mathcal{D}_1$ et $\mathcal{D}_2$.

	$\mathcal{D}$					
Trans.	Items					
t_1	A	B	C	D		$\mathcal{D}_1$
t_2	A	B	C	D		
t_3	A	B	C			
t_4	A			D	E	
t_5	A	B	C			$\mathcal{D}_2$
t_6		B	C	D	E	
t_7		B	C		E	
t_8		B			E	

TAB. 9.2 – Exemple d'un contexte transactionnel $\mathcal{D}$ avec deux sous-bases $\mathcal{D}_1$ et $\mathcal{D}_2$.

Les motifs caractéristiques de la sous-base $\mathcal{D}_i$ s'obtiennent en sélectionnant les motifs dont la mesure d'intérêt M_i est supérieure à un seuil donné. Typiquement, la contrainte d'émergence sélectionne les motifs n fois plus présents dans une sous-base que dans les autres réunies (cf. la section 1.1.2). Cette contrainte d'émergence se ré-écrit sous la forme normalisée $GR_i(X) \geq \rho$ où GR_i est le taux de croissance de la sous-base $\mathcal{D}_i$ par rapport aux autres (voir le tableau 9.3 qui sera aussi commenté dans la section suivante).

De manière générale, de nombreuses mesures de fréquences permettent d'évaluer la qualité d'un motif pour une classe donnée :

Définition 34 (Mesure de fréquences) *Une mesure de fréquences se définit comme une combinaison des primitives* $freq(X, \mathcal{D}_1), \ldots, freq(X, \mathcal{D}_n)$.

Le tableau 9.3 donne de nombreuses mesures de fréquences. Ces dernières sont inspirées de mesures statistiques définies à l'origine en terme de probabilités. Elles évaluent la qualité d'un motif de la classe i par rapport aux autres classes d'un contexte $\mathcal{D}$.

Dans les sections précédentes, nous avons indiqué que *freq* est une primitive conservée. En fait, ce résultat se généralise à la fréquence dans une sous-base :

Propriété 19 *La primitive $freq(X, \mathcal{D}_i)$ est une primitive conservée.*

Preuve. Soit X un motif ensembliste et a un item tel que $freq(X \cup \{a\}, \mathcal{D}_i) = freq(X, \mathcal{D}_i)$. En d'autres termes, l'item a est présent dans toutes les transactions de $\mathcal{D}_i$ contenant X. Pour une spécialisation Y du motif X, les transactions de $\mathcal{D}_i$ contenant Y sont un sous-ensemble de celles contenant X. Ainsi, l'item a est également présent dans chacune des transactions de $\mathcal{D}_i$ contenant Y et on conclut que $freq(Y \cup \{a\}, \mathcal{D}_i) = freq(Y, \mathcal{D}_i)$. □

La propriété 19 nous permet d'utiliser les résultats de la section 9.2. En particulier, comme toutes les mesures de fréquences du tableau 9.3 sont des combinaisons de $freq(X, \mathcal{D}_i)$ et de $freq(X)$, la fermeture $h_{freq(X,\mathcal{D}_i),freq(X)}$ est adéquate à toutes ces mesures (cf. corollaire 1). Ainsi, nous pouvons facilement définir des représentations condensées adéquates aux mesures de fréquences. En particulier, même si le taux de croissance (tout comme la contrainte d'émergence) n'a pas de bonne propriété de monotonie, nous disposons d'une méthode efficace d'extraction des motifs émergents car il existe des algorithmes qui extraient de façon efficace les représentations condensées adéquates $h_{freq(X,\mathcal{D}_i),freq(X)}$. Par exemple, l'algorithme MicMac vu ci-avant est adapté. Néanmoins, remarquons que $h_{freq(X,\mathcal{D}_i),freq(X)}$ est égal à $h_{freq(X)}$ car le motif $h_{freq(X,\mathcal{D}_i)}$ est toujours une spécialisation de $h_{freq(X)}$. Ainsi, les algorithmes d'extraction de motifs libres ou de fermés classiques sont parfaitement adaptés à l'extraction des représentations condensées adéquates à une mesure de fréquences M_i. En particulier, cela signifie que les intervalles de Music (construits avec $h_{freq(X)} = h$) conservent les mesures de fréquences et garantissent ainsi un élagage optimal. De même, comme pour tous les motifs X, on a $\mathbf{cl}_R(X) \subseteq h_{freq(X)}(X)$. L'élagage sur les intervalles de Music-dfs pour les mesures de fréquences est toujours effectué.

9.4.2 Motifs forts

Les motifs les plus significatifs au regard d'une mesure de fréquences M_i (cf. la définition 34) sont souvent ceux qui maximisent cette mesure. Plutôt que d'extraire tous les motifs à travers les représentations condensées adéquates et de filtrer ceux de plus forte mesure, il est pertinent de n'extraire que ces derniers. Contrairement à l'approche du chapitre 7 qui sélectionne ceux qui maximisent M_i, nous souhaitons dans cette section extraire une "couverture maximisante". En effet, dans le cas des mesures de fréquences, se contenter des motifs de plus forte mesure n'est pas souhaitable car ils sont souvent trop spécifiques. Par exemple, les motifs de plus fort taux de croissance (i.e., appelés *Jumping Emerging Patterns* et qui ont un taux de croissance infini) sont souvent des motifs de faible fréquence (e.g., $GR_1(ADE) = \infty$ et $freq(ADE) = 1$, $GR_2(BCDE) = \infty$ et $freq(BCDE) = 1$). Le motif ABC caractérise bien la classe 1 avec un taux de croissance de 3, mais il possède l'avantage d'avoir une bien meilleure fréquence. Le motif ABC semble donc plus pertinent que le motif ADE dont la fréquence est uniquement de 1.

Nous proposons d'extraire une collection de motifs à la fois représentative de la base de données et de bonne qualité au regard de la mesure de fréquences M_i considérée. Nous nous focalisons sur des mesures qui augmentent avec le nombre d'exemples de la classe i contenant le motif :

Mesure de fréquences	Définition	Forte	P_3
J-Measure (J) [Smyth et Goodman, 1991]	$\frac{freq(X,\mathcal{D}_i)}{\lvert\mathcal{D}\rvert} \times log(\frac{freq(X,\mathcal{D}_i)\times\mathcal{D}}{\lvert\mathcal{D}_i\rvert\times freq(X,\mathcal{D})})$ $+\frac{freq(X,\mathcal{D}\setminus\mathcal{D}_i)}{\lvert\mathcal{D}\rvert} \times log(\frac{freq(X,\mathcal{D}\setminus\mathcal{D}_i)\times\mathcal{D}}{freq(X,\mathcal{D})\times\lvert\mathcal{D}\setminus\mathcal{D}_i\rvert})$	non	non
Support [Agrawal *et al.*, 1993]	$freq(X,\mathcal{D}_i)/\lvert\mathcal{D}\rvert$	oui	non
Confidence [Agrawal *et al.*, 1993]	$freq(X,\mathcal{D}_i)/freq(X,\mathcal{D})$	oui	non
Sensitivity	$freq(X,\mathcal{D}_i)/\lvert\mathcal{D}_i\rvert$	oui	non
Success rate	$\frac{freq(X,\mathcal{D}_i)}{\lvert\mathcal{D}\rvert} + \frac{\lvert\mathcal{D}\setminus\mathcal{D}_i\rvert - freq(X,\mathcal{D}\setminus\mathcal{D}_i)}{\lvert\mathcal{D}\rvert}$	oui	oui
Specificity	$\frac{\lvert\mathcal{D}\setminus\mathcal{D}_i\rvert - freq(X,\mathcal{D}\setminus\mathcal{D}_i)}{\lvert\mathcal{D}\rvert}$	oui	oui
Piatetsky-Shapiro's (PS) [Piatetsky-Shapiro, 1991]	$\frac{freq(X,\mathcal{D}_i)}{\lvert\mathcal{D}\rvert} - \frac{freq(X,\mathcal{D})}{\lvert\mathcal{D}\rvert} \times \frac{\lvert\mathcal{D}_i\rvert}{\lvert\mathcal{D}\rvert}$	oui	oui
Lift [International Business Machines, 1996]	$\frac{\lvert\mathcal{D}\rvert\times freq(X,\mathcal{D}_i)}{\lvert\mathcal{D}_i\rvert\times freq(X,\mathcal{D})}$	oui	oui
Odds ratio (α)	$\frac{freq(X,\mathcal{D}_i)\times(\lvert\mathcal{D}\setminus\mathcal{D}_i\rvert - freq(X,\mathcal{D}\setminus\mathcal{D}_i))}{(freq(X,\mathcal{D}) - freq(X,\mathcal{D}_i))\times(\lvert\mathcal{D}_i\rvert - freq(X,\mathcal{D}_i))}$	oui	oui
Laplace (L) [Clark et Boswell, 1991]	$\frac{freq(X,\mathcal{D}_i)/\lvert\mathcal{D}\rvert+1}{freq(X,\mathcal{D})/\lvert\mathcal{D}\rvert+k}$ avec $k>1$	oui	oui
Growth rate (GR) [Dong et Li, 1999]	$\frac{\lvert\mathcal{D}\rvert-\lvert\mathcal{D}_i\rvert}{\lvert\mathcal{D}_i\rvert} \times \frac{freq(X,\mathcal{D}_i)}{freq(X,\mathcal{D})-freq(X,\mathcal{D}_i)}$	oui	oui

TAB. 9.3 – Exemples de mesures de fréquences pour évaluer X dans la sous-base $\mathcal{D}_i$.

Définition 35 (Mesure forte de fréquences) *Une mesure de fréquences M_i qui décroît avec $freq(X, \mathcal{D})$ quand $freq(X, \mathcal{D}_i)$ reste constant, est une mesure forte de fréquences.*

Une autre façon de formuler cette définition est de dire que M_i croît selon $freq(X, \mathcal{D}_i)$ quand $freq(X, \mathcal{D})$ reste constant. Cette propriété est souvent souhaitable en pratique. Par exemple, le lift est défini par $\frac{|\mathcal{D}| \times freq(X,\mathcal{D}_i)}{|\mathcal{D}_i| \times freq(X,\mathcal{D})}$. Quand $freq(X, \mathcal{D}_i)$ reste inchangé et que la fréquence $freq(X, \mathcal{D})$ augmente, le lift décroît car le dénominateur croît. Ainsi, le lift est une mesure forte de fréquences.

Relions la définition 35 au cadre de Piatetsky-Shapiro [Piatetsky-Shapiro, 1991] qui évalue la qualité d'une mesure objective. Ce dernier a, en effet, proposé trois propriétés clés pour caractériser une bonne mesure d'intérêt. D'un point de vue formel, la définition 35 est presque similaire à la troisième propriété P_3 donné par Piatetsky-Shapiro : *M_i croît strictement avec $P(X)$ quand les autres paramètres (i.e. $P(X, C_i)$ et $P(C_i)$) restent inchangés.* En effet, on observe que $P(X) = freq(X, \mathcal{D})/|\mathcal{D}|$, $P(X, C_i) = freq(X, \mathcal{D}_i)/|\mathcal{D}|$ et $P(C_i) = |\mathcal{D}_i|/|\mathcal{D}|$. En comparaison avec la définition 35, la seule différence, très mineure, est que M_i doit strictement décroître quand $freq(X, \mathcal{D})$ augmente alors que, dans notre définition, M_i peut rester inchangée. Dans la pratique, la plupart des mesures de fréquences sont fortes [Tan *et al.*, 2002] car elles vérifient la propriété P_3. Le tableau 9.3 donne, pour plusieurs mesures, celles qui satisfont ou non la définition 35 et la propriété P_3.

À partir de la définition 35, nous définissons la notion de motif fort et surtout, la caractérisons par le théorème suivant :

Théorème 10 *Soit M_i une mesure de fréquences forte et X un motif, on a $M_i(X) \leq M_i(h_{freq(X,\mathcal{D}_i)}(X))$. $h_{freq(X,\mathcal{D}_i)}(X)$ est appelé un motif fort de la classe i.*

Preuve. Soit M_i une mesure forte de fréquences et X un motif. la propriété 7 donne que $freq(X, \mathcal{D}_i) = freq(h_{freq(X,\mathcal{D}_i)}(X), \mathcal{D}_i)$. Comme $X \subseteq h_{freq(X,\mathcal{D}_i)}(X)$ et que la fréquence est décroissante, on a $freq(X) \geq freq(h_{freq(X,\mathcal{D}_i)}(X))$. La mesure M_i étant forte, la définition 35 permet de conclure que le théorème 10 est juste. □

Les motifs forts correspondent aux motifs fermés adéquats à $freq(X, \mathcal{D}_i)$. Il sont appelés *forts* car ils maximisent toutes les mesures fortes M_i. Illustrons le théorème 10 sur le contexte du tableau 9.2. Le motif CD n'est pas un motif fort de la classe 1 (car $h_{freq(X,\mathcal{D}_i)}(CD) = ABCD$), sa mesure de Piatetsky-Shapiro est 0.0625 et on a $PS_1(CD) \leq PS_1(ABCD) = 0.125$ comme attendu.

En plus d'optimiser les mesures fortes, le motif fort $h_{freq(X,\mathcal{D}_i)}(X)$ a la même fréquence dans le jeu de données $\mathcal{D}_i$ que celle du motif X sur lequel il est basé. Ainsi, les motifs forts ne dégradent pas le critère de fréquence.

Les motifs forts d'une classe peuvent être extraits directement avec l'algorithme MicMac en utilisant la fonction conservée $freq(X, \mathcal{D}_i)$. L'opération doit alors être répétée pour chacune des classes. Ainsi, nous verrons que les motifs émergents et les motifs émergents forts (notés SEPs pour *strong emerging patterns*) sont largement utilisés dans les chapitres 10 et 11. Ils ont permis d'obtenir des informations à forte valeur ajoutée. Nous montrerons également que la collection des motifs émergents forts est très restreinte par rapport à la représentation condensée des motifs émergents.

Conclusion

Le premier chapitre de cette partie a introduit une nouvelle classe de contraintes, les PBC, dont l'originalité est d'être fondée sur des primitives à la sémantique très simple, mais combinables à volonté. Au final, cette vaste classe englobe les classes plus classiques comme celles des contraintes monotones, anti-monotones et convertibles. Par ailleurs, les PBC permettent de formuler des contraintes originales et nouvelles comme nous le verrons dans le chapitre 12. Le chapitre 5 a alors défini des opérateurs pour identifier des propriétés de monotonies et déduire des bornes sur un intervalle. Ce sont ces opérateurs formels qui autorisent des extractions automatisables et génériques de motifs satisfaisant une PBC dans les chapitres 6, 7 et 8.

Plus précisément, le chapitre 6 a proposé une méthode d'extraction des motifs satisfaisant une PBC en relaxant la contrainte. Les relaxations monotones et anti-monotones obtenues grâce aux motifs virtuels peuvent alors être exploitées par des algorithmes portant sur des langages variés même complexes. Ces relaxations sont reprises dans le chapitre 7 pour traiter des contraintes globales. De telles contraintes, dont la vérification nécessite la comparaison de plusieurs motifs entre eux, sont relaxées dynamiquement par une approche Approximer-et-Pousser. Un exemple d'application de cette dernière est la recherche des k motifs maximisant une mesure basée sur des primitives.

Le chapitre 8 présente un nouvel algorithme d'extraction de motifs ensemblistes satisfaisant une PBC. En exploitant un élagage sur les intervalles combiné à un parcours en profondeur, Music-dfs s'avère particulièrement efficace pour traiter les contraintes du PBC (surtout les plus sélectives), y compris dans les larges jeux de données. Music-dfs peut aussi bénéficier de notre méthode de relaxation anti-monotone. En outre, l'introduction des opérateurs de fermeture adéquats aux fonctions conservées dans le chapitre 9, permet d'optimiser la construction des intervalles et d'améliorer ainsi l'élagage. Dans ce chapitre, nous avons aussi défini de nouvelles représentations condensées adéquates à d'autres fonctions que la fréquence. Un algorithme élémentaire MicMac a permis de les extraire et d'en montrer la concision. Pour les mesures de fréquence, une représentation condensée des motifs les plus significatifs, appelés motifs forts, est aussi proposée.

La figure 1 (page 124) schématise les différentes contributions du cadre fondé sur des primitives, elle complète la figure 3.5 de la page 41. La méthode de relaxation et la recherche des top-k motifs selon une mesure reposent sur les PBC et sont dédiés à tout langage. En revanche, l'algorithme Music-dfs et les représentations condensées adéquates bien que traitant n'importe quelle PBC, sont restreintes au langage des motifs ensemblistes. Ces méthodes, tout en dépassant les limites classiques de l'extraction de motifs, rappellent le compromis entre diversité du langage et efficacité. En effet, l'efficacité de Music-dfs réside sur des principes qui ne sont pas aisément généralisables à tout langage.

Par ailleurs, plusieurs de nos méthodes dépassent le cadre fondé sur les primitives. Par exemple, la relaxation à base de motifs virtuels peut bénéficier à d'autres solveurs et le nouvel opérateur de fermeture étendre des usages des bases de données inductives comme l'intégration

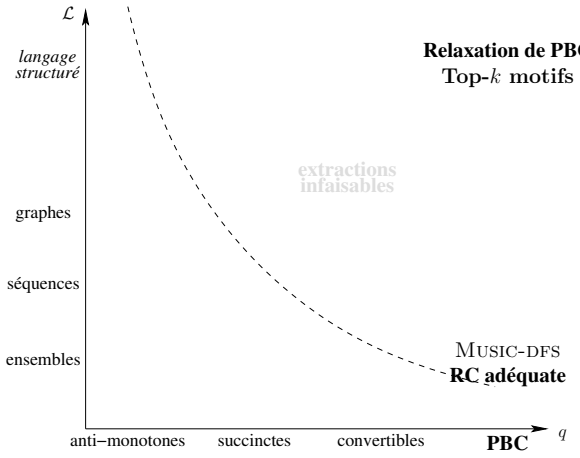


FIG. 1 – Contributions à l'extraction de motifs contraints.

de nouvelles mesures dans les requêtes.

Troisième partie

Usages et applications

Introduction

Cette partie montre l'apport de notre travail sur des problèmes réels et présente des applications menées en collaboration avec les experts des données. Les résultats précédemment obtenus sur la découverte de motifs contraints sont ici développés en terme de méthodes de découverte de connaissances sur des problèmes réels.

Le chapitre 10 caractérise des lots défectueux d'une chaîne de production de plaques de silicium à l'aide de motifs émergents forts. Il s'agit d'une collaboration avec la société Philips. Le chapitre 11 regroupe différentes expérimentations effectuées sur des données médicales relatives à l'athérosclérose et aux hépatites. À chaque fois, nous nous intéressons à la caractérisation des groupes sain et pathologique. Pour les données hépatites, un résumé discrimine chaque stade de la fibrose. Enfin, le chapitre 12 décrit un processus de découverte de gènes jouant un rôle dans le développement du cancer. À cette fin, les motifs contraints ont permis de prendre efficacement en compte les connaissances du domaine issues de plusieurs jeux de données.

Chapitre 10

Détection d'équipements défectueux dans une chaîne de production de plaques de silicium

Sommaire

Nous rapportons dans ce chapitre nos expériences visant à identifier les étapes posant des problèmes lors de la production de plaques de silicium. En terme de découverte de connaissance dans les bases de données, nous nous appuyons sur les méthodes de motifs émergents forts définis à la section 9.4.2 pour caractériser les lots défectueux par rapport à ceux valides. Nous faisons ainsi émerger les différences de réglage d'équipements. Ce travail a été effectué en collaboration avec Gilles Ferru de la société Philips et François Rioult du GREYC.

La section 10.1 présente la problématique des plaques de silicium défaillantes et les deux problèmes posés par la société Philips. La section 10.2 explique comment sont préparées les données pour rechercher leurs caractérisations. La section 10.3 donne les résultats des expérimentations et les équipements défectueux pour ces deux problèmes.

Pour des raisons de confidentialité, les noms des étapes ainsi que les données techniques présentées dans les flow-charts (cf. annexe D), ont été volontairement modifiés dans ce mémoire.

10.1 Contexte et problématique

10.1.1 Présentation du problème

La fabrication de plaques de silicium est une tâche délicate et cruciale dans la production de composants (e.g., micro-circuits). Un défaut dans le procédé de fabrication peut entraîner une chute importante du taux de composants valides. Étant donnés le temps et le coût de fabrication d'un composant, il est nécessaire de limiter au maximum le nombre de puces défectueuses le plus

tôt possible au sein de la chaîne. Pour cela des tests de qualité sont effectués dès la fabrication des plaques (tests sous pointes), puis une seconde série de vérifications a lieu après le montage.

Pour obtenir un bon rendement final, il est donc nécessaire de diagnostiquer rapidement les défaillances au sein de la chaîne de production. La première difficulté provient du nombre important d'étapes du procédé de fabrication qui sont autant de facteurs potentiels de défaillance après la phase de montage. La seconde difficulté réside dans le fait qu'un rendement acceptable lors de la première série de tests, avant le montage, n'implique pas automatiquement un rendement acceptable lors de la seconde série. Précisons qu'il peut être onéreux et long de vérifier les hypothèses de diagnostic (en effet, la fabrication, le montage et les tests sont effectués dans des pays différents).

10.1.2 Présentation des données

Un premier problème, déjà résolu par la société PHILIPS, nous a d'abord été présenté afin de tester l'approche "fouille de données" dans ce type d'applications. Il a permis de calibrer notre chaîne d'extraction. À la vue des bons résultats obtenus sur ce problème, PHILIPS nous a alors présenté un autre problème non résolu par cette société.

Premier problème. Ce premier jeu de données concerne le problème d'un four de diffusion. La caractérisation de ce problème est effectuée à partir des données de 127 lots. Il s'agit de retrouver le four incriminé et éventuellement de trouver d'autres dysfonctionnements.

Second problème. Ce second problème concerne un problème d'isolation entre deux transistors. Il s'agit de caractériser cette défectuosité à partir de 51 lots.

10.2 Pré-traitement des données

Dans un premier temps, il est nécessaire de diviser les lots en plusieurs classes afin de caractériser la "bonne" classe par rapport à la "mauvaise". Étant donné que le rendement des lots est uniformément réparti entre le plus mauvais et le meilleur rendement, il semble difficile d'effectuer avec pertinence une séparation « bon lot » et « mauvais lot ». C'est pourquoi nous effectuons un découpage en 3 classes correspondant à 3 niveaux de qualité nommés `Bonne`, `Moyenne` et `Mauvaise` (ces noms désignent aussi les jeux de données respectifs de chaque classe). Les meilleurs lots et les moins bons sont ainsi caractérisés avec plus de précision, ce découpage restant quelque peu arbitraire. Pour les deux problèmes, le tableau 10.1 donne la répartition des lots par classe en fonction du rendement.

Classe	Intervalle	Nbr. de lots
`Mauvaise`	<71%	45
`Moyenne`	[71%,85%]	37
`Bonne`	>85%	45

Classe	Intervalle	Nbr. de lots
`Mauvaise`	<65%	10
`Moyenne`	[65%,70%]	28
`Bonne`	>70%	13

TAB. 10.1 – Répartition des lots (problème 1, à gauche et problème 2, à droite).

La succession des étapes, avec l'équipement correspondant pour produire les lots, est indiquée par un *flow-chart* (l'annexe D en donne un extrait). Seules les données discrètes issues de celui-ci sont exploitées pour décrire les lots. L'objectif de ce prétraitement est d'associer pour chaque

lot l'outil utilisé à chaque étape. Nous donnons maintenant quelques précisions techniques sur le pré-traitement. Dans un premier temps, pour chaque lot, nous associons l'équipement utilisé à chaque étape (dans le flow-chart, l'étape est spécifiée dans la colonne `Step Id.` et l'équipement utilisé dans la colonne `Equipement`, cf. l'annexe D.1).

La principale difficulté est que l'ensemble d'un lot ne suit pas toujours exactement le même cheminement. Parfois, un lot est divisé en sous-lots (cela est alors noté dans la colonne `Sub Lot`, cf. l'annexe D.2). Comme le rendement final qualifie le lot dans son ensemble, il est difficile d'effectuer la caractérisation à partir des sous-lots. De même, l'opération sur le sous-lot ne concerne pas l'ensemble du lot. Ainsi, dans un premier temps, nous avons décidé d'éliminer les opérations effectuées sur les sous-lots. Nous verrons que pour ce problème, cette simplification ne nuit pas à la qualité des résultats.

Certains lots pour une même étape transitent par plusieurs équipements (par exemple, lorsqu'une opération est annulée et réitérée). Quel(s) équipement(s) doit-on choisir pour rendre compte au mieux de l'étape ? Dans un souci de lisibilité des résultats, nous avons choisi de ne tenir compte que du premier équipement.

Notons aussi que certaines étapes n'ont pas été retenues car tous les lots transitent par le même équipement pour ces étapes. En effet, lorsqu'à une étape donnée, le même outil est utilisé pour les lots des trois classes, cet outil ne peut pas discriminer l'une de ces classes. En l'éliminant, on ne perd donc pas d'information.

Dans la réalité, le problème comporte d'autres données : réglages des équipements, les données environnementales, les données externes (e.g., la température ou la pression), etc. Dans cette étude, nous faisons l'hypothèse que les variations de ces données ne sont pas significatives.

10.3 Identification des équipements défectueux

Cette section donne les résultats de la caractérisation pour les deux problèmes. Les motifs émergents forts ont été extraits grâce au cadre dépeint au chapitre 9. En particulier, la méthode d'extraction est basée sur celle de l'algorithme MicMac. Elle est clairement détaillée dans [Soulet *et al.*, 2004c]. Plusieurs expériences supplémentaires sont menées dans [Soulet *et al.*, 2004a] montrant que le nombre de motifs émergents forts (notés SEPs) est avantageusement faible par rapport au nombre total de motifs émergents.

10.3.1 Résultats du premier problème

Cette section montre que les motifs émergents forts ont permis de retrouver la cause principale de la défaillance dans la chaîne de production. Elle identifie aussi une deuxième cause potentielle.

La caractérisation des classes a été effectuée avec une recherche des motifs fréquents avec un support absolu de 10. Dans cette expérience, on s'est intéressé aux motifs émergents forts (définis à la section 9.4.2) de longueur inférieure ou égale à 5. La fréquence relative (par classe) et le nombre de motifs émergents sont donnés pour chaque classe dans le tableau 10.2.

L'essentiel des motifs émergents a un taux de croissance faible i.e., inférieur à 2 (cf. le tableau 10.3). Étonnamment, seule `Moyenne` possède de nombreux JEPs (i.e., EPs de taux de croissance infini) et `Mauvaise` comporte relativement peu de motifs émergents forts. En fait, il est plus que probable qu'en s'intéressant aux EPs (et non aux SEPs), ces répartitions seraient plus homogènes. Par exemple, `A=284` est un EP de `Mauvaise` (avec GR=1.89) mais il n'est pas

Classe	Seuil de fréquence minimale	Nombre de SEPs
`Mauvaise`	0.22	6532
`Moyenne`	0.27	8116
`Bonne`	0.22	14782

TAB. 10.2 – Résultats globaux (premier problème).

un SEP. De plus, l'item `E=727` présent dans chaque *flow-chart* de `Mauvaise` et `Bonne` allonge les fermetures d'un item. Or, nous nous intéressons aux SEPs de longueur au plus égale à 5.

Classe	$GR \in [1,2[$	$GR \in [2,5[$	$GR \in [5,\infty[$	JEPs
`Mauvaise`	5442	896	194	0
`Moyenne`	6379	1438	112	287
`Bonne`	10750	3680	351	1

TAB. 10.3 – Répartition des SEPs (premier problème).

Le tableau 10.4 propose plusieurs SEPs intéressants. Tout d'abord, on remarquera qu'il n'y a pas de SEP caractéristique de longueur 1. Par exemple, le motif `E=727` n'est pas discriminant. Non seulement son taux de croissance est proche de 1, mais en plus, `E=727` est présent dans `Mauvaise` et `Bonne`.

Au contraire, les SEPs de longueur 2 semblent pertinents. L'opposition entre le motif `E=727 A=284` pour `Mauvaise` et le motif `E=727 A=222` (pour `Bonne`) semble souligner un problème majeur au niveau de l'étape `A` étant donné que `E=727` n'est pas un item discriminant. De plus, l'étape `A` ne comporte que deux équipements le 222 et le 284. Ce résultat tendrait à montrer la nécessité de modifier les réglages de l'équipement 284 (pour les faire tendre vers ceux de l'équipement 222).

Motifs émergents forts de longueur 1			
Classe	Motif	GR	Fréquence
`Mauvaise`	`E=727`	1.01	100% (45)
`Moyenne`	`F=232`	1.03	100% (37)
`Bonne`	`E=727`	1.01	100% (45)
Motifs émergents forts de longueur 2 avec un GR>1.5			
Classe	Motif	GR	Fréquence
`Mauvaise`	`E=727 A=284`	3.64	75.6 % (34)
`Moyenne`	`I=504 F=232`	1.84	91.9 % (34)
`Moyenne`	`L=490 F=232`	1.62	54.0 % (20)
`Bonne`	`E=727 B=288`	2.92	71.1 % (32)
`Bonne`	`E=727 A=222`	2.33	91.1 % (41)

TAB. 10.4 – Exemples de motifs émergents forts (premier problème).

Dans le tableau 10.5, nous avons procédé à une analyse plus minutieuse de l'étape `A` et de `B=288`. En particulier, nous avons observé que `A=284`, `A=222` et `B=288` sont des motifs émergents

avec des supports importants. Comment expliquer la qualité des lots issus de l'équipement 288 à l'étape `B` ? Le réglage de cet appareil est-il plus efficace que les autres employés à cette étape ?

Motif	Classes considérées	Taux de croissance	Fréquence
`A=284`	de {`Moyenne`, `Bonne` } dans `Mauvaise`	3.44	75.6 % (34)
`A=284`	de {`Bonne` } dans `Mauvaise`	11.33	75.6 % (34)
`A=222`	de {`Mauvaise`, `Moyenne` } dans `Bonne`	2.33	91.1 % (41)
`A=222`	de {`Mauvaise` } dans `Bonne`	4.1	91.1 % (41)
`B=288`	de {`Mauvaise`, `Moyenne` } dans `Bonne`	2.92	71.1 % (32)
`B=288`	de {`Mauvaise` } dans `Bonne`	3.56	71.1 % (32)

TAB. 10.5 – Détail de `A` et `B=288`.

Conclusion. Par la suite, les experts nous ont confirmé que l'étape `A` que nous suspections problématique, était dans la réalité la cause de la chute du rendement. Plusieurs éléments montrent que le réglage de l'équipement 222 est moins efficace que celui de l'équipement 284. De même, on peut s'interroger dans une moindre mesure sur la pertinence de l'équipement 288 à l'étape `B` qui semble performant.

10.3.2 Résultats du second problème

Cette section, en reprenant la méthodologie exposée ci-dessus, montre comment l'utilisation des SEPs aboutit à l'identification d'un équipement défectueux.

La caractérisation des classes a été effectuée avec une recherche des motifs fréquents avec un support absolu de 1. Pour cette caractérisation, on s'est intéressé aux motifs émergents forts de longueur inférieure ou égale à 5. La fréquence relative et le nombre de motifs émergents forts sont donnés pour chaque classe dans le tableau 10.6.

Classe	Seuil de fréquence minimale	Nombre de SEPs
`Mauvaise`	0.10	61
`Moyenne`	0.04	249
`Bonne`	0.08	59

TAB. 10.6 – Résultats globaux (second problème).

Le nombre important de SEPs pour `Moyenne` se justifie par un nombre plus important de lots. Le tableau 10.7 donne la répartition des motifs émergents forts selon la classe et le taux de croissance. Pour `Mauvaise`, il est surprenant qu'il y ait si peu de SEPs avec un taux de croissance inférieur à 2.

Parmi l'ensemble des SEPs, le tableau 10.8 en propose quelques exemples. Comme pour le premier problème, bien que `T=118` soit un motif émergent de `Mauvaise`, `T=118` ne caractérise absolument pas `Mauvaise` puisqu'il est aussi un EP de `Bonne`. En revanche, comme précédemment, les SEPs de longueur 2 semblent plus caractéristiques des classes. Outre des taux de croissance respectables (i.e., > 1.5), les fréquences sont importantes (e.g., environ 80% pour le SEP de `Bonne`).

Classe	$GR \in [1,2[$	$GR \in [2,5[$	$GR \in [5,\infty[$	JEPs
`Mauvaise`	2	30	29	3
`Moyenne`	136	35	1	77
`Bonne`	25	17	8	9

TAB. 10.7 – Répartition des EPs (second problème).

Motifs émergents forts de longueur 1			
Classe	Motif	GR	Fréquence
`Mauvaise`	`T=118`	2.56	100% (10)
`Moyenne`	`L=439T`	1.09	28.6% (8)
`Bonne`	`T=118`	1.15	100% (13)
Motifs émergents forts de longueur 2 avec un GR>1.5			
Classe	Motif	GR	Fréquence
`Mauvaise`	`K=462 T=118`	3.2	80.0% (8)
`Mauvaise`	`T=118 C=158`	9.2	40.0% (4)
`Moyenne`	`V=248 D=492C`	1.77	46.4% (13)
`Moyenne`	`V=248 C=248`	1.77	46.4% (13)
`Bonne`	`T=118 C=248`	1.69	84.6% (11)

TAB. 10.8 – Exemples de motifs émergents forts (second problème).

Afin de préciser la caractérisation, nous avons détaillé le cas de l'item `K=462` et de l'étape `C` (cf. tableau 10.9). Cela montre que `K=462` est bien un EP de la classe `Mauvaise` (avec un taux de croissance évidemment inférieur à celui du SEP `K=462 K=462`). De même, l'item `C=248` caractérise `Bonne` avec un taux de croissance de 1.40 par rapport aux classes `Mauvaise` et `Moyenne`. Cependant, `C=248` est un très bon EP de `Mauvaise` dans `Bonne`. Pour l'étape `C`, on peut s'interroger sur la nécessité de modifier les réglages du four 158 pour les faire tendre vers ceux du four 248. En effet, tous les lots qui sont passés par le four 158 à l'étape `C`, appartiennent à `Mauvaise`.

Motif	Classes considérées	Taux de croissance	Fréquence
`K=462`	de {`Moyenne`, `Bonne` } dans `Mauvaise`	1.21	80.0% (8)
`K=462`	de {`Bonne` } dans `Mauvaise`	1.49	80.0% (8)
`C=158`	de {`Moyenne`,`Bonne` } dans `Mauvaise`	3.28	40.0% (4)
`C=158`	de {`Bonne` } dans `Mauvaise`	∞	40.0% (4)
`C=248`	de {`Mauvaise`, `Moyenne` } dans `Bonne`	1.40	84.6% (11)
`C=248`	de {`Mauvaise` } dans `Bonne`	2.11	84.6% (11)

TAB. 10.9 – Détail de `K=462` et `C`.

Conclusion. Dans ce second problème, nous avons suspecté un dysfonctionnement à l'étape `C`. L'équipement 158 semble moins bien réglé que l'équipement 248. Néanmoins, ce résultat est moins probant que ceux du premier problème. En effet, on dispose de trop peu de lots et la répartition des lots dans les trois classes est déséquilibrée (la classe `Moyenne` est trop dense).

Les experts de PHILIPS ont estimé que vraisemblablement un problème survient à l'étape `C`. Malheureusement, la destruction de la chaîne de production suite à l'incendie du 12 décembre 2003 n'a pas permis de vérifier ces hypothèses.

Chapitre 11

Découverte de facteurs de risque pour la maladie de l'athérosclérose et de la fibrose du foie

Sommaire

Ce chapitre présente plusieurs expérimentations sur des données médicales qui ont fait l'objet de nombreuses collaborations notamment au sein de l'Action Spécifique "Discovery Challenge" (septembre 2003-octobre 2004). Le premier cas d'étude concerne la maladie de l'athérosclérose et a pour objectif de caractériser les patients sains par rapport à ceux atteints ou décédés de l'athérosclérose. Nous verrons que nous avons à nouveau utilisé les motifs émergents forts pour effectuer cette caractérisation. Le second cas d'étude est dédié aux données relatives aux hépatites afin d'étudier les différents stades de la fibrose du foie. Nous proposons une méthode de catégorisation contrainte pour produire une caractérisation des différents stades de la fibrose sous forme d'un résumé. Cette approche illustre un usage original de la fermeture adéquate (cf. chapitre 9).

La section 11.1 relate trois expériences distinctes sur les données de l'athérosclérose. La section 11.2 présente les résultats relatifs à la caractérisation des différents stades de la fibrose du foie.

11.1 Facteurs de risque des maladies issues de l'athérosclérose

Le principal objectif de cette étude est d'identifier les facteurs de risque de l'athérosclérose (et leurs combinaisons) et de suivre leur développement et leurs impacts. Pour cela, nous cherchons ce qui distingue les patients décédés ou atteints de la maladie de l'athérosclérose des autres personnes de l'étude. Pour cette tâche de caractérisation, nous allons employé la méthode des

motifs émergents forts (i.e., SEPs) présentée à la section 9.4.2. La caractérisation est effectuée à partir des données recueillies durant 20 ans auprès d'une étude longitudinale de facteurs de risque de l'athérosclérose sur une population de 1417 Tchéquoslovaques (le projet a débuté dans les années 70)[22]. La prévention et la détection de l'athérosclérose et des maladies cardiovasculaires est une tâche essentielle pour la santé publique.

Ce travail résulte d'échanges variés au sein de l'Action Spécifique "Discovery Challenge". En particulier, nous avons collaboré avec Guillaume Cleuziou du LIFO, Nicolas Durand et Céline Hébert du GREYC.

11.1.1 Présentation et préparation des données

Cette section présente les données et leur préparation utilisées dans les expériences appelées `Experiment 1` et `Experiment 2` (section 11.1.2). Dans l'expérience `Experiment 3` (section 11.1.3), une étape de catégorisation est ajoutée. Enfin, ces données ont également été exploitées pour comparer deux algorithmes de classification non-supervisée [Durand *et al.*, 2004].

Description des données

Les données se présentent sous la forme d'une base de données relationnelles composée de 4 tables que nous décrivons brièvement.

La table `Entry` contient 1417 hommes qui ont été examinés au cours de leur examen d'entrée. Chaque patient est décrit par 64 attributs. La plupart d'entre eux sont qualitatifs (l'examen physique et l'examen biochimique rassemblent principalement des attributs continus). Nous employons cette table pour obtenir les caractéristiques décrivant les patients à leur entrée dans l'étude (i.e., au cours de l'examen initial).

La table `Control` collecte les facteurs de risque et l'observation clinique de l'athérosclérose au cours des examens des patients suivis pendant 20 ans (à savoir les patients de `normal studied group`, de `intervened risk group` et de `control risk group`). Il y a 10572 examens. Nous employons cette table pour suivre des patients affectés par une maladie due à l'athérosclérose pendant l'étude. Cette table a 66 attributs.

La table `Death` indique les 389 patients qui sont morts pendant l'étude. Les causes de la mort sont diverses et peuvent être différentes de l'athérosclérose. Nous employons cette table pour sélectionner les patients qui sont décédés de l'athérosclérose pendant l'étude. Les attributs de cette table sont le numéro d'identification du patient, la date et la cause du décès.

Pour finir, la table `Letter` fournit des informations additionnelles (recueillies grâce à un questionnaire postal) au sujet de l'état de santé des 403 patients. Nous n'employons pas cette table dans ce travail.

Pré-traitement des données

Nous rappelons que le but est de caractériser des patients (en employant des SEPs) selon qu'ils sont affectés ou pas par une maladie due à l'athérosclérose. Nous avons ainsi besoin

[22]Cette étude a été réalisée par le 2ème Département de Médecine, la 1ère Faculté de Médecine de Charles University et Charles University Hospital, U nemocnice 2, Prague 2 (directeur Prof. M. Aschermann, MD, SDr, FESC), sous la supervision du Prof. F. Boudik, MD, ScD, en collaboration avec M. Tomeckova, MD, PhD et Ass. Prof. J. Bultas, MD, PhD. Les données ont été transformées sous format électronique par European Centre of Medical Informatics, Statistics and Epidemiology of Charles University et Academy of Sciences (directeur Prof. RNDr. J. Zvarova, DrSc). Les ressources sont disponibles sur le site `http ://euromise.vse.cz/STULONG`.

de savoir si un patient est mort ou est malade de l'athérosclérose durant l'étude. Ainsi, nous nous concentrons sur les patients de `normal studied group`, de `intervened risk group` et de `control risk group` parce que seuls ces groupes de patients sont suivis pendant cette période. Nous obtenons alors 899 patients.

Dans `Experiment 1` et `Experiment 3`, à partir des caractéristiques disponibles dans la table `Entry`, nous voulons distinguer les patients qui meurent de l'athérosclérose des autres. Grâce à l'observation à long terme, en employant la table `Death`, nous connaissons les patients qui sont morts et l'attribut (`PRICUMAR`) de cette table fournit la cause de la mort. D'un point de vue médical, `myocardial infarction`, `coronary heart disease`, `stroke` et `general atherosclerosis` indiquent des causes de mort dues à l'athérosclérose. Ces quatre valeurs correspondent à 165 patients. Quand nous les recoupons avec les groupes de patients qui sont suivis pendant toute l'étude, 124 patients restent (en supposant que tous les patients morts de l'athérosclérose sont enregistrés dans la table `Death`).

Nous avons effectué un travail semblable dans la seconde expérience (appelée `Experiment 2`), sauf que le but est de distinguer les patients qui sont atteints d'une maladie cardiovasculaire[23] de ceux qui sont restés en bonne santé (mais ces patients peuvent souffrir d'une autre maladie). Un patient a été affecté par une maladie cardiovasculaire quand il a une maladie basée sur l'un des attributs suivants : `HODN1`, `HODN2`, `HODN3`, `HODN11`, `HODN12`, `HODN13`, `HODN14`, `HODN21`, `HODN23` (ce groupe d'attributs provient « du questionnaire A_2 » de la table `Control`). Comme dans `Experiment 1`, nous supposons que tous les patients qui souffrent de l'athérosclérose sont enregistrés dans la base de données. On obtient 281 patients atteints par les maladies cardiovasculaires observées et qui appartiennent à un des groupes de patients suivis.

Nous avons décidé a priori de garder tous les attributs de la table `Entry`. Néanmoins, nous avons supprimé certains attributs : l'attribut `KONKSUP` (groupe étudié de patients) parce que la valeur `normal studied group` peut présenter un biais ; les attributs concernant des facteurs de risque (l'information représentée par ces attributs est déjà prise en considération par d'autres attributs) ; les attributs concernant l'anamnesis personnel à cause des fréquences très basses des valeurs. Nous avons remplacé les attributs `ROKNAR` (année de naissance) et le `ROKVSTUP` (année d'entrée dans l'étude) par l'âge du patient quand il s'est présenté dans l'étude. Pour les attributs qui ont seulement deux valeurs, seul l'item correspondant à `true` pour cette valeur (i.e., présence de la caractéristique) a été gardé. Les attributs `CHLST` (cholestérol) et `TRIGL` (triglycérides) ont été segmentés binairement selon les seuils indiqués par les médecins. Nous avons employé les équivalences suivantes : pour `CHLST` : 5,2 mmol/l = 200 mg/dL et pour le `TRIGL` : 2,0 mmol/l = 150 mg/dL. Les autres attributs continus (par exemple, `VYSKA` (taille)) ont été coupés en attributs qualitatifs, chacun des items ayant le même nombre de patients. En conclusion, nous obtenons un total de 119 items, chaque patient étant décrit par au plus 37 items.

La première partie du tableau 11.1 indique les caractéristiques des données obtenues pour `Experiment 1` et `Experiment 2`. Les caractéristiques de `Experiment 3` sont fournies à la section 11.1.3. Nous appelons `atherosclerosis` le nom des données des patients qui sont morts de l'athérosclérose (`Experiment 1` et `Experiment 3`) ou qui sont atteints par les maladies cardiovasculaires observées (`Experiment 2`). `healthy` désigne les données contenant les autres patients.

11.1.2 Caractérisation des patients

Pour les deux expériences, nous avons fixé le seuil de fréquence minimale à 15 pour chercher les SEPs. Pour chaque base de données, le tableau 11.1 donne le seuil minimum en fréquence

[23] Ces maladies découlent de l'athérosclérose.

relative (*minfr*) et le nombre de SEPs (contenant au plus 8 items) par rapport à leur taux de croissance (noté GR).

	Experiment 1		Experiment 2	
	atherosclerosis	healthy	atherosclerosis	healthy
Nbr. de patients	124	624	281	618
minfr (%)	12.1%	2.4%	5.3%	2.4%
$GR \in [1..2[$	32606	2,278,346	510,901	2,845,756
$GR \in [2..5[$	6254	1,229,359	69609	605,312
$GR \in [5..\infty[$	47	94,921	1038	61168
JEP	132	387,203	2690	16916

TAB. 11.1 – Nombres de patients, seuils minimum de fréquence et nombres de SEPs par rapport à leur taux de croissance.

Le tableau 11.1 récapitule les résultats. Il prouve que, pour les deux expériences, le nombre de SEPs de `healthy` est plus grand que celui de `atherosclerosis`. Ceci peut être expliqué par les fréquences relatives qui sont inférieures dans `healthy` par rapport à `atherosclerosis` ou par le fait que certains facteurs médicaux accentuent davantage `healthy` que `atherosclerosis`. Même si les SEPs sont une couverture des EPs (cf. la section 9.4.2), on constate que le nombre de SEPs demeure très important.

Le tableau 11.2 (resp. 11.3) détaille des SEPs de `Experiment 1` (resp. `Experiment 2`) avec les meilleurs taux de croissance et de bons résultats en fréquence. Les items d'un SEP sont séparés par des « ; ». Les premières parties de ces tables fournissent les SEPs de `atherosclerosis` et les deuxièmes parties ceux de `healthy`. La fréquence (notée *freq*) d'un SEP pour une base $\mathcal{D}_i$ est sa fréquence relative à cette base $\mathcal{D}_i$ (par exemple, une fréquence de 11.3% d'un SEP de `atherosclerosis` signifie que 11.3% des patients de `atherosclerosis` sont caractérisés par ce SEP).

atherosclerosis		
items of SEPs	GR	*freq* (%)
the way to work takes around 1 hour ; smoker of 21 and more cigarettes per day ; smoking during 21 and more years ; do not drink liquors	6.71	11.3
weight $\leq$ 74 kg ; blood pressure II diastolic $>$ 92 mm Hg ; normal urine	6.29	11.3
height $\leq$ 1.72 m ; blood pressure II diastolic $>$ 92 mm Hg	3.91	16.9
blood pressure II diastolic $>$ 92 mm Hg	1.72	32.3
age of entry in the study $\in$ [43,47] ; moderate activity after his job ; level of total cholesterol $\geq$ 200 mg/dL	∞	18.5
healthy		
items of SEPs	GR	*freq* (%)
partly independent worker ; blood pressure II systolic $\leq$ 118 mm Hg	11.7	9.46
reached education : university ; level of total cholesterol $<$ 200 mg/dL	8.35	6.7
age of entry in the study $\in$ [44,47] ; level of total cholesterol $<$ 200 mg/dL	8.15	6.6
age of entry in the study $\leq$ 43 years	2.21	30.3
reached education : university ; blood pressure II diastolic $\leq$ 78 mm Hg	∞	8.7
age of entry in the study $\leq$ 43 years ; mainly standing at work	∞	5.0
non-smoker ; blood pressure I systolic $\leq$ 120 mm Hg	∞	4.5

TAB. 11.2 – SEPs de `Experiment 1`.

`atherosclerosis`		
items of SEPs	GR	*freq* (%)
1 or 2 cups of coffee per day ; height < 1.72 m ; blood pressure I diastolic $\in [75,92]$; skinfold above musculus triceps > 11	7.48	6.0
more than 6 sugar lumps per day ; skinfold above musculus triceps > 11	2.30	8.2
height ≤ 1.72 m ; blood pressure I systolic > 135	2.00	14.6
drinking of alcohol : occasionally ; drinking of vine ; up to half a litre of vine per day ; level of triglycerides > 150 mg/dL	∞	14.2
reached education : secondary school ; drinking of vine ; up to half a litre of vine per day ; blood pressure II diastolic $\in [78,92]$	∞	12.5
`healthy`		
items of SEPs	GR	*freq* (%)
single ; do not drink coffee	8.64	3.1
lower limbs pain is non-ischaemic ; blood pressure I diastolic ≤ 75 mm Hg	8.64	3.1
mainly walks at work ; drink daily more than 1 litre of beer	5.12	7.3
partly independent worker ; blood pressure I systolic ≤ 120 mm Hg ; blood pressure I diastolic ≤ 75 mm Hg ;	5	7.1
drinking of 10° beer ; daily consumption of 2 at 6 sugar lumps ; blood pressure I diastolic ≤ 75 mm Hg ; normal urine	∞	7.3
blood pressure II systolic $\in [118,138]$; blood pressure II diastolic > 92 mm Hg	∞	3.8

TAB. 11.3 – SEPs de `Experiment 2`.

Dans `Experiment 1`, beaucoup de SEPs de `atherosclerosis` ont l'item « fumer pendant 21 ans et plus » (par exemple, 67 JEPs[24] parmi les 132 incluent cet item). La tension artérielle semble aussi avoir un rôle important. Tous les JEPs de `healthy` ont au moins deux items relatifs à la tension. Dans `Experiment 2`, il y a 11 JEPs de `atherosclerosis` ayant 4 items (il n'y a aucun JEP avec moins d'items) et 7 SEPs composés de 2 items ont un taux de croissance supérieur à 2. Contrairement à `Experiment 1`, il n'y a aucun SEP significatif ayant un seul item. Dans `atherosclerosis`, la taille (`height`) semble jouer un rôle important. Dans `healthy`, il y a des JEPs simples ayant 2 items (voir le tableau 11.3).

Il est clair que beaucoup d'associations soulignées par les SEPs sont attendues (et déjà connues) par les médecins. Néanmoins, un résultat intéressant apporté par cette méthode de fouille est de quantifier l'intérêt de telles associations (par exemple, de combien augmente le risque d'athérosclérose en fonction de certaines caractéristiques).

11.1.3 Caractérisation des patients suivant leur catégorie sociale

Dans cette expérience `Experiment 3`, nous avons voulu approfondir l'impact des facteurs sociaux comme le niveau d'éducation ou le type de travail, sur la maladie de l'athérosclérose (cf. tableaux 11.2 et 11.3). Pour cela, nous avons caractérisé plus finement chaque catégorie sociale en comparant leurs SEPs avec ceux obtenus dans la base complète. Ce travail a été mené en collaboration avec Céline HÉBERT du GREYC et est plus précisément détaillé dans [Soulet et Hébert, 2004].

[24] JEP : Jumping Emerging Pattern (cf. section 9.4.2)

Description du processus

Nous donnons d'abord un aperçu de l'expérience et la préparation des données nécessaire pour la réaliser.

Rappelons que nous souhaitons pour chaque catégorie sociale caractériser les patients décédés suite à une maladie de l'athérosclérose. Pour cela, les SEPs sont extraits à partir de clusters construits suivant les facteurs sociaux. Puis ces SEPs sont comparés à ceux obtenus sur le jeu de données entier (i.e., les SEPs de `Experiment 1`) afin de faire ressortir les spécificités des groupes sociaux.

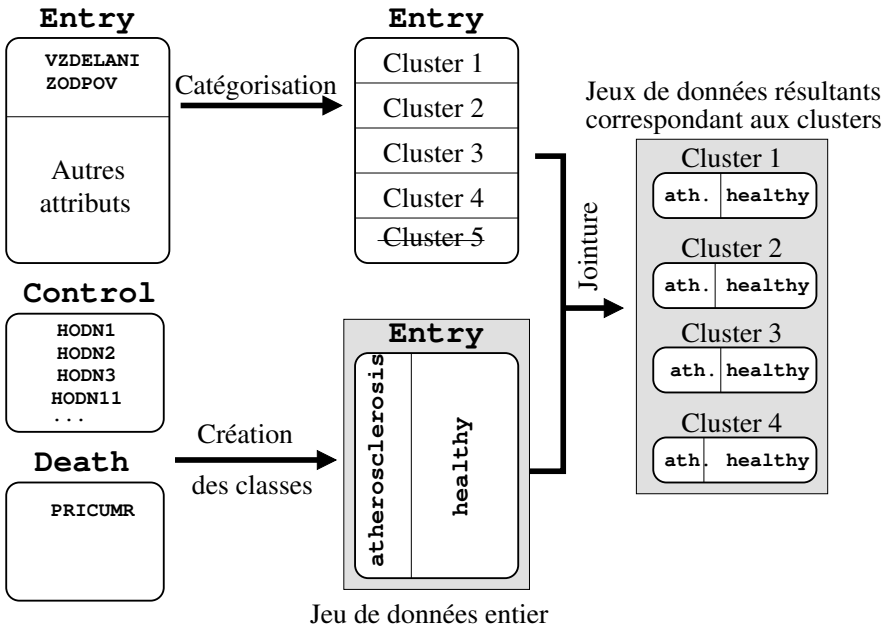


FIG. 11.1 – Adaptation du jeu de données (`Experiment 3`).

Pour cette expérience, la préparation des données de la section 11.1.1 a été complétée par une catégorisation des patients suivant leur groupe social et une projection des classes `healthy` et `atherosclerosis` sur ces groupes (cf. la figure 11.1). Cette caractérisation a été réalisée à partir des motifs fermés parce que ceux-ci possèdent de bonnes propriétés pour la constitution de clusters. En effet, les motifs fermés sont des clusters candidats intéressants car ils rassemblent l'ensemble maximal d'items partagés par un ensemble de transactions données. En d'autres termes, il capture la plus grande quantité de similarité entre les transactions. On trouvera dans [Durand et Crémilleux, 2002] une discussion plus approfondie sur ce point. Les 5 clusters ont été obtenus à partir de l'analyse manuelle des 11 motifs fermés restreints aux items issus des attributs `VZDERLANI` (i.e., niveau d'éducation) et `ZODPOV` (i.e., responsabilité au travail). Le tableau 11.4 donne les principales caractéristiques des différents clusters. Le cluster 5 est un groupe de patients hétérogène car il résulte des patients non classés parmi les 4 autres clusters. Ce cluster n'est donc plus considéré par la suite.

Résultats de l'expérience `Experiment 3`

Nous présentons les résultats de la caractérisation des classes `atherosclerosis` et `healthy` pour chaque groupe social. Les tableaux 11.5 et 11.6 donnent des SEPs caractéristiques de

	Description sociale		Nombre de patients		
Cluster	Étude supérieure	Responsabilité au trvail	`healthy`	`atherosclerosis`	Total
1	oui	cadre	150	60	210
2	oui	travailleur indépendent	227	82	309
3	oui	autres	127	59	186
4	non	autres	221	122	343
5	-	-	94	57	151
		Total des patients	819	380	1199

TAB. 11.4 – Description des clusters.

chaque cluster ainsi que leur quantification. Le taux de croissance et la fréquence relative dans chaque cluster sont spécifiés respectivement par la colonne 3 et 4. Le taux d'amélioration (*improvement rate* noté IR) d'un SEP X pour un cluster donné k correspond au rapport du taux de croissance de X dans le cluster k et de son taux de croissance dans le jeu de données complet. Plus ce taux est grand, plus le SEP est caractéristique du groupe social considéré. Par exemple, dans le cluster 1, le SEP `smoking during 21 and more years ; weight` $\leq$ `84kg ; level of triglycerides`$>$ `150 ; level of total cholesterol` $>$ `200 mg/dL` a un taux d'amélioration de 3.78. Cela signifie que le taux de croissance de ce SEP pour le cluster 1 est 3.78 fois plus important que le taux de croissance du même motif dans le jeu de données complet.

Cluster	Description of SEPs	GR	*freq* (%)	IR
1	smoking during 21 and more years ; weight $\leq$ 84kg ; level of triglycerides$>$ 150 ; level of total cholesterol $>$ 200 mg/dL	8.33	16.7	3.78
2	smoking daily between 15 and 20 cigarettes ; smoking during 21 and more years ; blood pressure systolic 1 in [116,135] mm Hg ; blood pressure diastolic 2 in [78,93] mm Hg	6.33	19.5	2.73
	drinking daily more than 3 cups of coffee	1.82	30.5	1.33
3	he mainly walk at work ; do not drink tea ; no lower limbs pain ; level of total cholesterol $>$ 200 mg/dL	5.17	20.3	3.66
	2 smoking during 21 and more years ; blood pressure systolic 1 ¿135 ; normal urine	2.92	32.2	1.77
4	around 1/2 hour to get to work ; height $>$ 178 ; level of triglycerides$\leq$ 150	∞	10.7	∞
	height$\leq$172 ; weight $>$ 84	6.52	14.8	2.54

TAB. 11.5 – SEPs pour `atherosclerosis` (`Experiment 3`).

Les différents SEPs obtenus permettent de retrouver les facteurs de risques majeurs de l'athérosclérose à savoir la consommation d'alcool et de tabac. Sans surprise, un taux de cholestérol élevé caractérise à nouveau les patients décédés (SEPs des clusters 1 et 3 de `atherosclerosis`). Les SEPs du quatrième cluster, correspondant aux patients qui n'ont pas eu d'étude supérieure et ont des reponsabilité au travail faible, sont plus déroutants car la consommation d'alcool ou de tabac y est absente. Globalement les patients de ce groupe sont caractérisés par des facteurs relevant peu du comportement et au contraire, on trouve des SEPs comme `height`$\leq$`172 ; weight` $>$ `84` ou `age of entry in the study` $\leq$ `43 years`. La différence des attributs mis en jeux dans les SEPs entre les groupes sociaux tend à montrer la nécessité de distinguer leur caractérisation.

Cluster	Description of SEPs	*GR*	*freq* (%)	*IR*
1	drinking daily up to 1 liter of beer ; do not drink coffee ; daily drinking 1 or 2 cups of tea	5.80	19.3	4.00
	non-smoker	2.28	38.0	1.20
	level of total cholesterol < 200 mg/dL	1.88	31.3	1.19
2	blood pressure systolic $2 \leq 118$; level of total cholesterol < 200 mg/dL	7.22	8.8	2.95
	blood pressure systolic $1 \leq 116$ mm Hg	5.96	29.1	2.80
	level of total cholesterol < 200 mg/dL	1.58	25.1	1.00
3	around 1/2 hour to get to work ; drinking of wine ; do not drink liquors	∞	18.9	∞
	mainly walking at work ; non-smoker ; no asthma ; normal urine	∞	9.4	∞
	level of total cholesterol < 200 mg/dL	3.16	26.8	2.00
4	non-smoker ; no asthma ; level of triglycerides ≤ 7 mg/dL	9.94	8.1	3.18
	age of entry in the study ≤ 43 years	2.27	33.5	1.34

TAB. 11.6 – SEPs pour `healthy` (`Experiment 3`).

11.2 Caractérisation des différents stades de la fibrose du foie

Cette section présente une méthode de construction d'un modèle global construit à partir de motifs locaux émergents. Des résultats sont obtenus à partir des données relatives aux différents stades de la fibrose du foie, collectées à l'hôpital universitaire de Chiba au Japon. Ce travail est issue d'une collaboration avec Nicolas DURAND.

La section 11.2.1 donne un aperçu de l'approche générique de construction d'un modèle global caractérisant. La section 11.2.2 décrit les données hépatites et les pré-traitements effectués. Enfin, les résultats de notre approche appliquée aux données hépatites sont présentés à la section 11.2.3.

11.2.1 Approche de découverte de clusters émergents avec chevauchement

Nous souhaitons obtenir un modèle qui décrive l'évolution de la fibrose du foie, c'est-à-dire isoler les caractéristiques majeures de chaque stade tout en couvrant le maximum de patients. Les motifs émergents permettraient de trouver les contrastes, mais seraient locaux et nombreux. Une idée est de produire une classification non supervisée, qui est un modèle global dont la définition de chaque cluster est une façon de caractériser l'ensemble des patients. Mais, il est peu probable que, de façon naturelle, chaque cluster soit "pur" par rapport aux stades de la fibrose que nous cherchons ici à caractériser. Aussi, nous proposons de combiner des motifs locaux avec la construction d'un modèle global (cf. figure 11.2) afin de produire un ensemble de clusters émergents (avec des chevauchements éventuels), i.e., un ensemble de motifs émergents caractérisant chaque stade et décrivant tous les patients. En d'autres termes, cette approche générique s'apparente à une catégorisation contrainte des données. Plus précisément, notre processus de découverte de clusters émergents se fonde sur deux étapes : extraction de clusters potentiels avec MUSIC et sélection des clusters les plus pertinents pour construire le modèle global avec ECCLAT. Le principe d'ECCLAT, ainsi que les deux étapes de ce processus sont donnés ci-après.

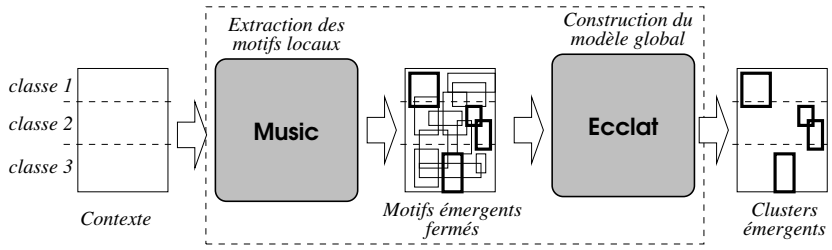


FIG. 11.2 – Processus de découverte des clusters émergents.

Extraction des motifs émergents fermés : MUSIC

Les motifs émergents fermés (CEP pour *Closed Emerging Pattern*) sont des motifs à la fois fermés et émergents (i.e., dont le taux de croissance, noté GR, excède un seuil donné *mingr*). La collection de tous les motifs émergents fermés correspond exactement à la représentation condensée adéquate au taux de croissance, des motifs fermés. Rappelons que le chapitre 9 garantit que cette représentation est une bonne couverture de tous les motifs émergents présents dans la base de données. Bien que les CEPs soient plus nombreux que les SEPs, ils sont un choix cohérent dans notre processus. En effet, la réduction utile qu'apportent les motifs émergents forts, est ici remplacée par l'étape de sélection d'ECCLAT.

L'extraction des CEPs peut indifféremment être effectuée avec l'algorithme MUSIC (cf. la section 8.1.3) et l'algorithme MICMAC (cf. la section 9.3.1).

Sélection des clusters émergents : ECCLAT

ECCLAT (Extraction of Clusters from Concepts LATice) [Durand et Crémilleux, 2002] produit des concepts (i.e., un motif d'items associés aux transactions où il apparaît) à partir de données catégorielles. Ces concepts constituent un ensemble de clusters autorisant des chevauchements. L'une des originalités d'ECCLAT est de ne pas fixer par avance le nombre de clusters pour effectuer sa classification non-supervisée. À l'origine, ECCLAT sélectionne ses clusters à partir d'un réservoir de motifs fermés dont la fréquence excède *minfr*. Rappelons que nous avons indiqué à la section 11.1.3 que les motifs fermés sont de bons candidats pour produire des clusters. Par ailleurs, la fréquence minimale imposée assure une certaine représentativité à chaque cluster en lui imposant un nombre minimal de transactions. En outre, ECCLAT sélectionne parmi les motifs fermés les clusters qui maximisent une mesure d'intérêt qui est la moyenne de deux mesures : l'*homogénéité* et la *concentration*. L'homogénéité est d'autant plus forte que les transactions partagent beaucoup d'items. La concentration évite des chevauchements excessifs entre les différents clusters. Ce chevauchement est aussi controlé par un paramètre M qui est le nombre minimal de transactions différentes entre deux clusters sélectionnés.

Pour notre processus, nous adaptons ECCLAT de sorte que les clusters obtenus soient émergents. Pour cela, il suffit que le réservoir de clusters potentiels à savoir les motifs fermés, soit remplacé par la collection des motifs émergents fermés. En effet, les bonnes propriétés des motifs fermés perdurent avec les CEPs. Le processus de sélection d'ECCLAT (en particulier, les mesures) reste donc inchangé. Au final, cette approche garantit, en plus de la catégorisation, que chaque cluster décrit majoritairement une classe.

L'intérêt de la coopération des deux approches repose sur des atouts complémentaires dont les deux premiers sont apportés par ECCLAT et le troisième, par les motifs locaux :

- **Sélection forte** : ECCLAT sélectionne peu de clusters comparé au nombre de CEP obtenus par MUSIC. L'analyse de la catégorisation est donc aisée.
- **Chevauchement** : l'intersection entre les clusters est autorisée (i.e., une transaction peut appartenir à plusieurs clusters). Cela permet de capturer différents aspects des données.
- **Bonne pureté des clusters émergents** : chaque cluster émergent sélectionné appartient principalement à une seule classe. Cela permet alors à la catégorisation de fournir une caractérisation des données.

À notre avis, le processus de découverte de clusters émergents a l'avantage de traduire des phénomènes locaux en leur donnant une dimension sur l'ensemble du jeu de données. Cette approche construit un modèle global qui prend en compte les caractéristiques de chaque classe. Il peut aussi être vu comme une méthode de sélection globale et cohérente d'informations locales issues des CEPs.

11.2.2 Préparation des données

Nous indiquons maintenant la phase de préparation des données (des détails supplémentaires sont fournis dans [Durand et Soulet, 2005]).

Description des données

Parmi les 7 tables des données hépatites disponibles sur le site du Discovery Challenge `lisp.vse.cz/challenge/`, nous en utilisons 4 :

- La table `patient` contient 771 patients dont la majorité sont des hommes (i.e., 70.69%).
- La table `biopsy` contient 694 examens. L'attribut majeur `Fibrosis` renseigne le stade de la fibrose du moins avancé `F0` au plus avancé `F4`.
- La table `out-hospital_examinations` reporte les résultats des examens en dehors de l'hôpital pour 31040 patients. Certains attributs comportent de nombreuses valeurs manquantes et d'autres sont inexploitables sans l'aide de connaissances médicales approfondies.
- La table `in-hospital_examinations` enregistre 1565876 examens de patients à l'hôpital.

En ce qui concerne les tables `out-hospital_examinations` et `in-hospital_examinations`, nous nous intéresserons particulièrement à l'interprétation médicale `Qualitative_Interpretation` de chaque examen afin de déterminer si le résultat de l'examen est normal ou non. Les associations de ces résultats peuvent être caractéristiques de certains stades de la fibrose.

Contextes transactionnels résultants

Les contextes transactionnels ont été construits de la manière suivante : chaque transaction regroupe la biopsie et les examens pour un même patient. L'idée est alors de découvrir des clusters décrits par les examens et qui sont assez purs au regard du stade de la fibrose du foie. L'intérêt, du point de vu médical, est de pouvoir prédire le stade de la fibrose sans biopsie qui est un examen invasif. Nous construisons deux contextes transactionnels : `bioexain` se réfère aux examens de `in-hospital_examinations` et `bioexaout`, aux examens de `out-hospital_examinations`. Au sein de la même table, plusieurs examens similaires sont parfois effectués en relation avec la même biopsie. Nous associons alors à la biopsie l'examen le plus proche dans le temps. Par ailleurs, le stade de la fibrose est connu grâce à l'attribut `Fibrosis` de la table `biopsy`.

Pour chaque type d'examen de `in-hospital_examinations` ou `out-hospital_examinations` nous codons l'état normal et l'état anormal. Par exemple, l'examen `GLU` est codé `GLU+` pour l'état normal et `GLU-` pour l'état anormal.

Les examens concernent 499 patients distincts. Pour l'attribut `Fibrosis`, 13 patients ont la valeur `F0`, 216 ont la valeur `F1`, 109 ont la valeur `F2`, 78 ont la valeur `F3` et 83 ont la valeur `F4`.

	Nbr. de patients	Nbr. d'examens (final)	Nbr. d'items
`bioexaout`	342	1,400	42
`bioexain`	499	14,226	168

TAB. 11.7 – Caractéristiques de `bioexaout` et `bioexain`.

11.2.3 Résultats et discussion

À partir des deux contextes transactionnels que nous venons de présenter, nous recherchons les clusters émergents avec chevauchement pour construire un modèle estimant le stade de la fibrose. Les CEPs sont extraits pour chaque stade et ensuite, sélectionnés.

Résultats quantitatifs

Les tableaux 11.8 et 11.9 donnent un aperçu général des CEPs extraits, puis les clusters sélectionnés pour différents paramètres. Ces tableaux indiquent le seuil de fréquence minimale *minfr* (d'abord avec sa valeur relative, puis sa valeur absolue entre parenthèses), le taux de croissance minimal *mingr*, le nombre total de CEPs (i.e., les clusters candidats), le nombre minimal M de transactions nouvelles par cluster, le chevauchement moyen et le nombre de transactions non classées (i.e., appartenant au cluster appelé poubelle).

minfr	*mingr*	Nbr. de CEPs	M	Nbr. de clusters	Chevauchement moyen	# poubelle
3 (10)	3	24	1	10	2	265
2 (7)	3	41	1	14	1.11	244
0 (1)	3	180	1	52	0.22	218

TAB. 11.8 – Résultats quantitatifs sur `bioexaout`.

Dans le tableau 11.8, les résultats de `bioexaout` sont peu concluants. Notre approche de caractérisation requiert de nombreux CEPs pour bien couvrir toutes les données. Pour obtenir suffisamment de CEPs avec un taux de croissance raisonnable (i.e., supérieur à 3), le seuil de fréquence a été diminué jusqu'à 0%. Ainsi, les clusters sont peu représentatifs car ils contiennent peu de patients. En contrepartie, le cluster poubelle est très grand. Nous ne poussons donc pas l'analyse plus profondément sur `bioexaout`.

Examinons plus en détail le tableau 11.9 présentant les résultats pour `bioexain`. Plusieurs ensembles de clusters émergents semblent pertinents. Par exemple, avec $minfr = 8\%$ et $mingr = 3.5$, on obtient 57707 motifs émergents fermés. Lorsque M est fixé à 20, on obtient 13 clusters avec un chevauchement moyen de 11.15 et seulement 196 transactions non classées. Le compromis entre ces différentes valeurs nous paraît satisfaisant, d'autant que le taux de croissance de 3.5 est de bonne qualité. La section suivante analyse plus précisément cet ensemble de clusters.

minfr	*mingr*	Nbr. de CEPs	M	Nbr. de clusters	Chevauchement moyen	# poubelle
8 (40)	3	106,237	1	269	12.24	17
			20	15	9.43	148
	3.5	57,707	1	273	13.91	37
			20	13	11.15	196
	4	30,481	1	234	15.27	64
			20	11	8.14	214
	5	8,119	1	174	17.86	154
			20	6	12	340
6 (30)	3	287,122	1	310	10.38	13
			20	16	8.29	141
	3.5	176,136	1	314	7.24	20
			20	13	4.64	208
	4	106,024	1	298	8.5	32
			20	13	6.96	211
	5	41,303	1	273	10.15	83
			20	9	5.3	294

TAB. 11.9 – Résultats quantitatifs sur `bioexain`.

Résultats qualitatifs

Nous détaillons la description des clusters émergents pour une seule expérience où $minfr = 8\%$, $mingr = 3.5$ et $M = 20$ (cf. les tableaux 11.10 et 11.11). Ces deux tableaux s'appuient sur les mesures d'intérêt que sont la fréquence et le taux de croissance. En particulier, le taux de croissance permet d'établir le stade majoritairement caractérisé par le cluster. Par exemple, le plus fort taux de croissance du cluster 1 étant 6.12 pour le stade `F4`, il caractérise celui-ci (cf. le tableau 11.10). Le tableau 11.11 détaille la caractérisation des clusters en précisant les taux de croissance pour chacun des stades.

Le tableau 11.10 ordonne les clusters suivant la mesure d'intérêt utilisée par ECCLAT pour leur sélection. Cet ordre de sélection des clusters est important. De manière surprenante le stade `F4` (seulement 83 patients) est le mieux caractérisé avec 9 clusters sur 13 et, la plupart de ces clusters sont parmi les mieux classés. Plus précisément, les résultats d'examens ZTT+ et F-ALB- présents dans de nombreux clusters caractérisant **F3** et **F4** semblent spécifiques de ces deux stades. Pour les mêmes raisons, le résultat d'examen F-A1.GL- caractérise les stades **F0** et **F1**. En revanche, certains items (e.g., GOT+ ou GPT+) apparaissent dans de nombreux clusters voire dans tous, et sont peu significatifs.

cluster	CEP	*freq*	*GR*	stade
1	ALB- GOT+ GPT+ **ZTT+** ALP+ **F-ALB-** G-GTP+ TTT+ CHE- D-BIL+ G.GL+ I-BIL+ LDH+ T-BIL+	40	6.12	F4
2	ALB- GOT+ GPT+ **ZTT+** ALP+ CRE- **F-ALB-** G-GTP+ TTT+ G.GL+ LDH+ T-BA+	43	3.61	F4
3	GOT+ GPT+ **ZTT+ F-ALB-** G-GTP+ LAP+ TTT+ AMY+ CHE- D-BIL+ G.GL+ I-BIL+ T-BIL+ IG-G+ PT+	40	4.53	F4
4	GOT+ GPT+ **ZTT+** ALP+ F-A2.GL+ **F-ALB-** TTT+ D-BIL+ G.GL+ I-BIL+ T-BIL+	46	3.85	F4
5	ALB- GOT+ GPT+ **ZTT+ F-ALB-** F-B.GL+ TTT+ CHE- D-BIL+ G.GL+ T-BIL+ PT+	53	3.84	F4
6	ALB- GOT+ GPT+ **ZTT+** ALP+ **F-ALB-** G-GTP+ TTT+ G.GL+ PT+	77	3.56 2.02	F4 F3
7	GOT+ GPT+ LDH- TP- **F-ALB-** LAP+ TTT+ CHE- D-BIL+ G.GL+ T-BIL+	52	3.67	F4
8	GOT+ GPT+ **ZTT+** G-GTP+ TTT+ D-BIL+ I-BIL+ LDH+ T-BIL+	70	3.54	F4
9	ALP- F-A/G+ GOT+ GPT+ LDH- TTT+ G.GL+ HBD-	42	3.62	F2
10	ALB- GOT+ GPT+ **ZTT+ F-ALB-** CHE- G.GL+ F-A/G-	89	3.56 2.47	F4 F3
11	ALB- F-A1.GL- F-A/G+ GOT+ GPT+ TG+	48	3.93	F1
12	F-A/G+ GOT+ GPT+ I-BIL+ T-BIL+ F-CHO-	48	5.34	F0
13	F-A1.GL- GOT+ GPT+ ALB+	42	3.93	F0

TAB. 11.10 – Résultats sur `bioexain` (*minfr*=8%, *mingr*=3.5, M=20).

cluster	F0	F1	F2	F3	F4
1	0	0.11	0.52	1.8	**6.12**
2	0	0.4	0.83	1.05	**3.61**
3	0	0.23	0.52	1.8	**4.53**
4	1.69	0.41	0.65	0.81	**3.85**
5	0	0.13	1.06	1.75	**3.84**
6	0	0.29	0.54	***2.02***	**3.56**
7	0	0.2	0.97	1.62	**3.67**
8	0.54	0.42	0.53	1.35	**3.54**
9	0	0.41	**3.62**	0.73	0.83
10	0	0.2	0.56	***2.47***	**3.56**
11	0.79	**3.93**	0.72	0.36	0
12	**5.34**	1.31	0.52	0.49	1
13	**3.93**	1.19	0.85	0.57	0.83

TAB. 11.11 – Taux de croissance (pour chaque stade) sur les résultats de `bioexain`.

Chapitre 12

Utilisation de la connaissance du domaine pour la découverte de gènes co-régulés

Sommaire

Ce chapitre a pour but de montrer l'apport de la connaissance du domaine (e.g., la littérature disponible, l'ontologie de gènes) pour la découverte de gènes co-régulés afin d'identifier des fonctions cancéreuses au sein de librairies SAGE (*Serial Analysis of Gene Expression*). Les biologistes ont besoin d'obtenir des motifs interprétables et généralisables, et nous montrons comment la richesse sémantique des PBC permet de traduire les exigences des experts pour fouiller avec précision la matrice d'expression de gènes. Pour réaliser les extractions, nous utilisons Music-dfs présenté au chapitre 8. Il se révèle particulièrement efficace pour extraire les motifs malgré la complexité des contraintes et les bases de données très larges.

La section 12.1 décrit les différents jeux de données utilisés et leur pré-traitement. La section 12.2 montre la nécessité d'utiliser les connaissances du domaine et explique comment la contrainte effectue cette intégration. Enfin, dans la section 12.3, une analyse quantitative des extractions montre la difficulté de l'extraction, et est suivie par les résultats qualitatifs.

Ce travail est issu d'une collaboration avec les biologistes du CGMC (Centre de Génétique Moléculaire et Cellulaire, UMR 5534) au sein de l'ACI Masse de Données Bingo*, MD 46. Plus précisément, Jiří* Kléma *de Cezch Technical University, au GREYC durant l'année 2005/2006, a effectué l'élaboration et la mise en oeuvre de l'approche. Sylvain* Blachon *et Olivier* Gandrillon *du CGMC, ont analysé et interprété les résultats.*

12.1 Description des données et du pré-traitement

Cette section présente dans un premier temps les données à partir desquelles les motifs sont recherchés, à savoir les données SAGE (ces données sont appelées les données internes) et les données externes constituant les connaissances du domaine.

12.1.1 Données SAGE

Les données SAGE (*Serial Analysis of Gene Expression*) permettent de construire le contexte transactionnel pour trouver les associations de gènes intéressantes. Nous décrivons ici brièvement la construction de cette matrice.

La technique SAGE a pour objectif de mesurer le niveau d'expression des gènes au sein d'une population de cellules [Velculescu *et al.*, 1995]. Cela est effectué en séquençant les *tags* (des séquences courtes de 14 à 21 paires de base (bp)) qui sont spécifiques à chaque mRNA (i.e., ARN messager). 207 librairies SAGE (i.e., 207 situations biologiques ou expériences) ont été téléchargées sur le site web NCBI (`www.ncbi.nlm.nih.gov/`). Pour éliminer les séquences erronées, un prétraitement des données décrit dans [Becquet *et al.*, 2002] a été appliqué, donnant un ensemble de 125985 tags 14 bp. Les tags ont été identifiés grâce à Identitag [Keime *et al.*, 2004], en utilisant les séquences RefSeq mRNA. Seuls les 11082 tags non-ambigus ont été sélectionnés. La matrice d'expression de gènes ainsi obtenue comporte 207 lignes et 11082 colonnes. Une sous-matrice regroupe aussi les tags appartenant au transcriptome minimal [Velculescu *et al.*, 1999]. Cette matrice minimale d'expression de gènes contient les informations relatives à seulement 447 tags. Les deux matrices ont été binarisées pour encoder la sur-expression de chaque tag en utilisant la méthode MidRange décrite dans [Becquet *et al.*, 2002].

12.1.2 Données externes

Cette section présente les données externes constituant les connaissances du domaines génomiques (i.e., littérature, ontologie des gènes, etc). Elle décrit aussi leur pré-traitement qui est nécessaire lors de prise en compte pour la découverte de motifs.

Littérature

Pour accéder aux données d'annotation de gènes, pour chaque tag considéré, les identifiants RefSeq ont été convertis en identifiant EntrezGene avec MatchMiner (`discover.nci.nih.gov/matchminer/`). Seules 11 RefSeq n'ont pu être converties, 24 RefSeq ont été reliées à plus de un identifiant et 203 identifiant apparaissent plus d'une fois. Connaissant les identifiants des gènes, l'accès aux annotations à la base de données EntrezGene a été réalisé automatiquement grâce à des requêtes d'hypergraphe. Ces annotations ont ensuite été analysées selon la méthode de [Zelezny *et al.*, 2005]. Les enregistrements de textes non-triviaux sont obtenus pour 6302 identifiants qui forment 58% des 10858 identifiants uniques (3926 gènes ont un résumé et 5109 ont au moins un *abstract*).

Les annotations textuelles des gènes ont été converties en vecteurs (i.e., "sacs de mots"). Après l'élimination des termes trop courants, seuls les termes présents plus de 5 fois ont été conservés. Au final, le vocabulaire est donc constitué de 10373 termes. Une matrice de similarité a alors été constituée en utilisant le cosinus des angles entre les TFIDF [Salton et Buckley, 1988]. Plus la valeur entre deux gènes est proche de 1, plus leur connexion est vraisemblable.

Ontologie des gènes

Les gènes peuvent aussi être reliés fonctionnellement sur la base des termes de l'ontologie des gènes. Dans la suite, l'expression "termes GO" référence les termes de l'ontologie des gènes. Brièvement, plus les gènes partagent des termes généraux, plus ils sont fonctionnellement dépendants. [Martin *et al.*, 2004] définit une distance basée sur la formule de Czekanowski-Dice, la méthodologie est implémentée à travers l'outil GOProxy de GOToolBox (`crfb.univ-mrs.fr/GOToolBox/`).

Les identifiants de tag RefSeq originaux sont convertis en des identifiants UnitProt (`www.gene.ucl.ac.uk/nomenclature/data/gdlw\_index.html`). Sur les 11082 tags, 7670 ont des identifiants connus. Comme cet ensemble est trop large pour être traité par GOToolBox, nous confinons l'approche à la matrice minimale d'expression de gènes où 366 RefSeq peuvent être converties. Les identifiants résultant sont utilisés avec GOToolBox pour générer deux matrices de similarité. Pour l'ontologie du processus biologique, 254 entrées sont validées tandis que 271 tags peuvent être diagnostiqués à travers l'ontologie des fonctions moléculaires.

Les termes GO peuvent être analysés de la même manière que la littérature pour produire une matrice de similarité.

Description des librairies

De courtes annotations textuelles d'une longueur de 10 termes sont rattachées à chaque librairie SAGE. Ces annotations représentent des documents très courts et leur vocabulaire associé est restreint. En conséquence, ils peuvent être traités de la même manière que les documents relatifs aux tags. En considérant tous les termes apparaissant dans au moins 3 librairies, on obtient ainsi un vocabulaire de 83 termes. On génère alors la matrice de similarité correspondante.

12.2 Intégration des données externes à travers la contrainte

Dans un premier temps, cette section montre que se focaliser sur le contexte transactionnel (i.e., les données SAGE) est insuffisant pour dégager des motifs pertinents. Nous montrons alors comment la contrainte interconnecte les connaissances du domaine avec la matrice SAGE pour recueillir les motifs les plus pertinents.

12.2.1 Nécessité d'exploiter les connaissances du domaine

Considérons tous les motifs ayant une aire supérieure à 20[25]. On obtient environ un demi million de motifs distincts qui sont regroupés en 37852 intervalles. Bien que ces intervalles forment une bonne représentation condensée, le nombre de motifs obtenus interdit une recherche manuelle car l'interprétation d'un motif est une tâche non triviale qui demande des consultations répétées des bases médicales. Les biologistes préfèrent seulement une dizaine de motifs ou intervalles.

Augmenter le seuil de la contrainte d'aire minimale pour obtenir un nombre raisonnable de motifs est contre productif. La contrainte $area(X) \geq 75$ donne un petit nombre de motifs, seulement 56, mais ils contiennent essentiellement des protéines ribosomiques qui sont extrêmement fréquentes dans le jeu de données. Les biologistes jugent ces motifs comme justes mais inintéressants.

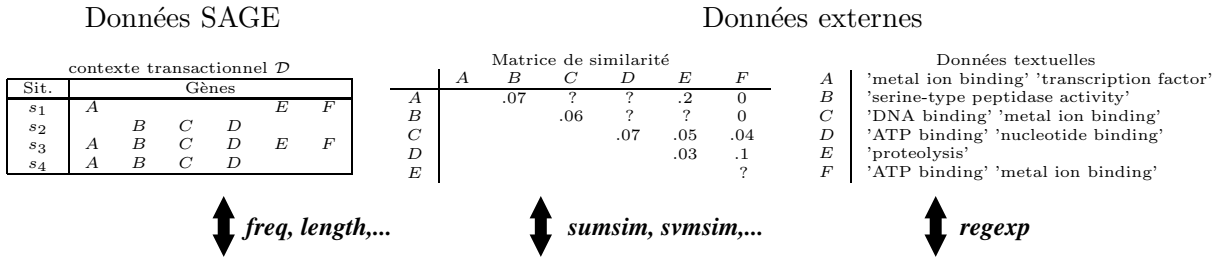
[25] Ce seuil a été fixé par une analyse statistique d'un jeu de données généré aléatoirement avec les mêmes propriétés que la matrice SAGE originale [Klema *et al.*, 2006].

Les motifs les plus intéressants attendus par les biologistes ont une aire certes importante, mais surtout, contiennent des gènes et des situations qui peuvent être généralisés, connectés, interprétés et ainsi, transformés en connaissance. Pour obtenir de tels motifs, des contraintes basées sur les données externes doivent être ajoutées à celle d'aire minimale comme nous allons le voir.

12.2.2 Définir une contrainte à travers plusieurs jeux de données

Considérons la figure 12.1 qui récapitule la forme de la base de données pour les données génomiques. Celle-ci est composée d'un contexte transactionnel (i.e., la matrice issue des données SAGE), d'une matrice de similarité (correspondant ici à la littérature) et des données textuelles (i.e., les termes pertinents relatifs à chaque gène). Ces différents jeux de données ont été obtenus grâce aux prétraitements décrits à la section 12.1.

Afin de tirer bénéfice de chacun des jeux de données, la contrainte exploite des informations sur chacun d'entre eux. En fait, la contrainte d'extraction pousse les connaissances du domaine dans l'extraction de motifs. Pour cela, elle combine des primitives (cf. la section A.1) portant sur chaque jeu de données. Par exemple, la contrainte q (figure 12.1) exploite simultanément les 3 jeux de données.



$$
\begin{aligned}
q(X) \equiv \quad & freq(X) \times count(X) \geq 24 && \text{(a)}\\
& \wedge length(regexp(X, '*ribosom*', \texttt{TEXT\_terms})) \leq 1 && \text{(b)}\\
& \wedge svsim(X, \texttt{TEXT})/(svsim(X, \texttt{TEXT}) + mvsim(X, \texttt{TEXT})) \geq 0.7 && \text{(c)}\\
& \wedge sumsim(X, \texttt{TEXT})/svsim(X, \texttt{TEXT}) \geq 0.025 && \text{(d)}
\end{aligned}
$$

FIG. 12.1 – Exemple simplifié d'une base de données génomique et d'une contrainte.

Décrivons maintenant le sens de chacune des contraintes atomiques de q. La contrainte atomique (a) repose sur le contexte $\mathcal{D}$ et signifie que les biologistes sont intéressés par des motifs d'aire minimale supérieure à 24. Les autres parties de q exploitent les données externes : (b) élimine les motifs contenant plus d'un gène ribosome, (c) privilégie les motifs dont on connaît des informations et (d) assure une cohésion entre les différents gènes.

La contrainte q montre à quel point les PBC sont flexibles. Elles offrent la possibilité d'adresser et de combiner plusieurs jeux de données. D'autre part, elles permettent une grande finesse d'expression pour l'utilisateur. Toutes ses attentes ont pu être traduites et permettent une extraction de motifs de bonne qualité comme nous le verrons dans la section 12.3.2.

12.3 Résultats des expérimentations

Cette section montre l'efficacité de MUSIC-DFS malgré la taille du contexte transactionnel et la complexité des contraintes d'extraction. Ensuite, elle donne les résultats concrets des expériences et leur interprétation.

12.3.1 Complexité de l'extraction

Rappelons que nos expériences sont menées avec un processeur Xeon 2.2 GHz avec 3 Go de mémoire RAM sous le système d'exploitation Linux.

Nécessité d'un parcours en profondeur

Cette première expérience souligne l'importance du parcours en profondeur. Nous considérons la contrainte extrayant des motifs dont l'aire est supérieure à 70 et apparaissant dans au moins 4 transactions du jeu de données. MUSIC-DFS nécessite 7 secondes pour extraire les 212 motifs contraints. En comparaison, pour la même matrice, l'approche par niveaux présentée dans [Soulet et Crémilleux, 2005a] échoue après 963 sec dès que la matrice contient plus de 3500 gènes. En effet, les motifs candidats d'un niveau donné ne tiennent plus en mémoire.

La comparaison avec des prototypes du FIMI [Goethals, 2003a] montre que les implémentations les plus efficaces telles que KDCI, LCM (ver. 2), COFI ou APRIORI (de Borgelt) échouent dans l'extraction des motifs de fréquence supérieure à 4. L'implémentation d'ECLAT et de AFOPT basées sur un parcours en profondeur, sont capables d'extraire ces motifs fréquents. Mais elles nécessitent un post-traitement pour prendre en compte d'autres contraintes que celle de fréquence (e.g., contraintes d'aire minimale ou basées sur la matrice de similarité).

Gain de l'élagage sur les intervalles

Cette expérience a pour but de montrer le rôle important de la stratégie d'élagage sur les intervalles. Nous comparons l'algorithme MUSIC-DFS avec une variante de celui-ci qui n'exploite pas l'élagage sur les intervalles. Celle-ci, nommée MUSIC-DFS-FILTER, extrait tous les motifs qui satisfont la contrainte de fréquence minimale, puis les autres contraintes sont appliquées dans une étape de post-traitement. Nous utilisons deux contraintes typiques du domaine génomique qui nécessitent des données externes. Les temps d'exécution de MUSIC-DFS et MUSIC-DFS-FILTER pour chacune de ces contraintes sont reportés sur la figure 12.2. Les résultats montrent que l'approche par post-traitement est faisable jusqu'à ce que le seuil de fréquence devienne trop bas. Pour un seuil de fréquence minimale trop peu élevé, le nombre de motifs explose et les élagages sur les intervalles se multiplient. L'élagage sur les intervalles diminue les temps d'exécution et ainsi, MUSIC-DFS devient vraiment plus avantageux.

12.3.2 Résultats et interprétation

Nous nous focalisons dans cette section sur l'extraction d'un motif jugé pertinent par les experts biologistes.

Comme nous l'avons expliqué à la section 12.2.2, l'extraction de motifs significatifs repose sur les différentes sources de données à l'aide de la contrainte. Alors qu'il y a 46671 motifs satisfaisant la contrainte d'aire minimale (partie (a) de la contrainte q), seulement 9 motifs satisfont q dans son intégralité. Cela montre l'intérêt d'utiliser les connaissances du domaine pour réduire le nombre de motifs et dégager les plus intéressants. Rappelons que cette réduction du nombre de motifs n'avait pas pu être menée par le biais du seuil de l'aire minimale (cf. la section 12.2.1). Étant peu nombreux, ces motifs ont été analysés individuellement par les experts biologistes. Ces derniers ont alors isolé un motif important correspondant aux 4 gènes KHDRBS1, NONO, TOP2B et FMR1. Ce motif est interprété ci après.

D'autre part, afin de prendre en compte la connaissance issue de Gene Ontology, nous avons aussi utilisé la contrainte q' (ci-dessous). Celle-ci est similaire à q mais se base sur les fonctions

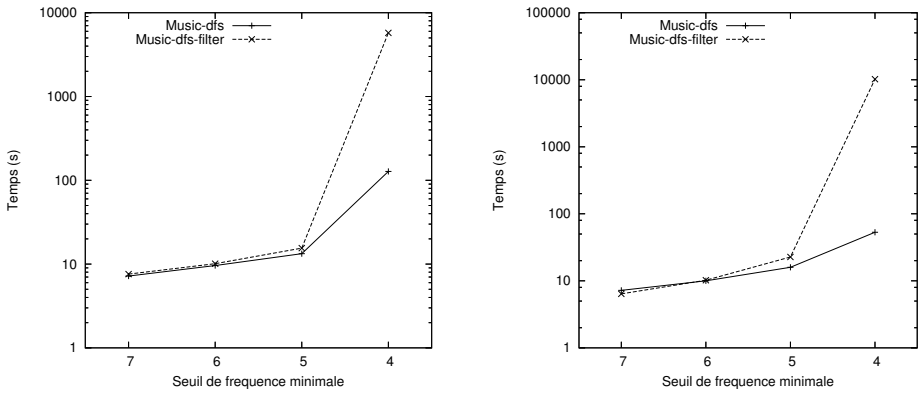


FIG. 12.2 – Efficacité de l'élagage sur les intervalles suivant le seuil de fréquence minimale. Le graphique de gauche est relatif à la contrainte $freq(X) \geq minfr \wedge lenght(X) \geq 4 \wedge sumsim(X)/svsim(X) \geq 0.9 \wedge svsim(X)/(svsim(X)+mvsim(X)) \geq 0.9$. Le graphique de droite correspond à la contrainte $freq(X) \geq minfr \wedge length(regexp(X, '*ribosom*', \texttt{GO\_terms})) = 0$.

de l'ontologie de gènes au lieu des ressources textuelles NBCI. Une autre contrainte atomique (e) est aussi ajoutée pour se concenter sur des gènes dont la similarité est assez forte.

$$
\begin{array}{lll}
q'(X) \equiv & area(X) \geq 24 & \text{(a)} \\
& \wedge\, length(regexp(X, '*ribosom*', \texttt{GO\_terms})) \leq 1 & \text{(b)} \\
& \wedge\, svsim(X, \texttt{GO})/(svsim(X, \texttt{GO}) + mvsim(X, \texttt{GO})) \geq 0.7 & \text{(c)} \\
& \wedge\, sumsim(X, \texttt{GO})/svsim(X, \texttt{GO}) \geq 0.025 & \text{(d)} \\
& \wedge\, insim(X, 0.5, 1, \texttt{GO})/svsim(X, \texttt{GO}) \geq 0.6 & \text{(e)}
\end{array}
$$

Seuls deux motifs sont extraits par q'. Par ailleurs, le motif extrait grâce à la contrainte q et jugé intéressant par les experts est à nouveau extrait avec q'. Ce motif correspond donc au 4 gènes KHDRBS1, NONO, TOP2B et FMR1, et il apparaît dans les 6 situations biologiques 48, 52, 54, 56, 62 et 65. En se basant sur la description des gènes et des librairies, ce motif présente un intérêt auprès des biologistes pour les raisons suivantes :

- 3 des 4 gènes (KHDRBS1, NONO et FMR1) sont connus pour coder les protéines qui montrent une activité de fixation de l'ARN [Lukong et Richard, 2003, Shav-Tal et Zipori, 2002, Zalfa et Bagni, 2004]. Le terme "RNA-bind" apparaît dans la liste des termes associée au motif. Parmi ces gènes, deux (KHDRBS1 et NONO) sont plus précisément connus pour augmenter l'épissage de l'ARN.
- le quatrième gène (TOP2B) encode une topoisomérase [Shav-Tal et Zipori, 2002]. Il est intéressant de noter que le gène NONO est connu pour interagir fonctionnellement avec la Topoisomerase 1 (un membre de la famille à laquelle TOP2B appartient). En outre une isoforme de TOP2B, TOP2A, est aussi trouvée différentiellement exprimée dans des cellules de médulloblastomes par rapport à des librairies SAGE normales [Boon *et al.*, 2003]. Les auteurs notent aussi l'existence de médicaments anti-cancéreux et variés dirigés contre le TOP2A. Ces médicaments doivent agir sur l'isoforme du TOP2B, augmentant l'effet anti-cancer. Une topoisomerase II inhibitrice est aussi connue pour montrer une activité antitumeur significative dans une xenogreffe de médulloblastome.
- un article récent utilisant des bio-puces, a démontré l'importance d'un processus d'épissage des ARN pour la neurogénèse chez l'adulte [Lim *et al.*, 2006]. Dans cette étude, le gène

KHDRBS1 a été trouvé parmi les gènes importants pour les cellules souches neuronales de l'adulte.

– toutes les situations dans lesquelles ces gènes sont sur-exprimés (i.e., 48, 52, etc) sont des médulloblastomes. Ce sont des tumeurs très agressives chez les enfants. Un faisceau de preuves montre que les cellules les plus agressives des médulloblastomes se comportent comme les cellules souches du cerveau [Al-Hajj et Clarke, 2004, Derrington *et al.*, 1998].

Ces évidences biologiques prises dans leur globalité permettent de formuler l'hypothèse suivante sur le sens biologique du motif : l'activité de fixation de l'ARN en général et l'épissage de l'ARN en particulier, connectée avec la conformation de l'ADN génomique via TOP2B, est tout aussi essentielle pour le médulloblastome qu'elle l'est pour les cellules souches nerveuses d'un adulte normal. En inhibant cette activité de fixation de l'ARN, on pourrait espérer un bénéfice thérapeutique, de la même manière que pour l'inhibition de la topoisomerase II.

Conclusion

Ces différentes confrontations aux données réelles ont permis d'éprouver nos méthodes de découverte de motifs contraints. Au delà de la faisabilité ou de la rapidité des extractions, ces applications ont permis de juger de la validité des motifs forts et plus généralement, du paradigme des motifs locaux contraints.

Nos méthodes combinatoires se sont révélées d'un grand secours pour déterminer les étapes défectueuses dans une chaîne de production de plaques de silicium. En particulier, les experts ont été surpris par la finesse de notre analyse qui a permis de déduire exactement l'équipement mal réglé. Nous regrettons que la seconde expérience n'ait pas pu être confirmée ou invalidée.

Les résultats applicatifs sur des expériences relatives à des données médicales (athérosclérose et hépatite) ont été plus modestes. Malgré plusieurs tentatives de caractérisation (motifs émergents forts, motifs émergents forts par catégorie sociale ou catégorisation contrainte), les résultats n'ont pas réellement surpris les médecins. Il est vrai que ces deux maladies sont déjà bien étudiées.

Sur les larges données SAGE, outre la démonstration de la faisabilité concrète des extractions, l'extraction d'un motif de gènes intéressant a été réalisée avec une contrainte synthétisant les connaissances multiples de la littérature. Grâce à l'analyse de ces gènes sur-exprimés simultanément, les biologistes ont pu formuler l'hypothèse que l'inhibition d'un des gènes aurait un impact anti-cancéreux. Par ailleurs, la valeur ajoutée calculée à partir de la littérature (similarité entre les gènes, termes caractéristiques des gènes, etc) et directement fournie par MUSIC-DFS, a facilité l'interprétation du motif en orientant la recherche bibliographique.

Bilan et perspectives

Bilan

Notre travail de recherche a porté sur la découverte de motifs locaux contraints dans les bases de données. Au cours de ce travail, nous nous sommes attachés à ne pas nous enfermer autour de l'extraction de motifs satisfaisant seulement les contraintes monotones. Nous avons privilégié la généricité des contraintes traitées en adoptant une démarche reposant sur des conditions suffisantes pour pousser les contraintes au cœur de l'extraction.

Un cadre générique fondé sur des primitives

Nous avons proposé un cadre générique qui permet à l'utilisateur de définir de façon flexible, un large éventail de contraintes. Celles-ci reposent sur un ensemble de primitives dont la seule condition est d'être monotone suivant chacune de leurs variables. En pratique, l'ajout aisé de nouvelles primitives permet d'étendre à volonté la richesse des contraintes. Ce cadre laisse une grande liberté à l'utilisateur dans la formulation de sa contrainte. Les primitives peuvent être combinées pour obtenir des contraintes complexes offrant une forte expressivité pour décrire le type de connaissance à cerner. Par ailleurs, ces contraintes fondées sur les primitives englobent les classes usuelles et leurs combinaisons.

Cette généricité dans la définition des contraintes ne s'oppose pas à la possibilité de les pousser au cœur de l'extraction. Nous avons montré que ce cadre permettait naturellement la définition d'opérateurs d'élagage conduisant à des méthodes génériques et automatiques d'extraction pour tout langage. En particulier, le comportement d'une contrainte sur un intervalle est évalué par les opérateurs de minoration et de majoration. Ces derniers étendent les conditions d'élagage suivant les spécialisations et les généralisations à tout intervalle.

Extraction de motifs contraints pour tout langage

Pour n'importe quel langage, nous avons étendu l'usage des algorithmes usuels dédiés aux contraintes monotones et anti-monotones à toute PBC en nous appuyant sur une méthode de relaxation. L'originalité de notre approche est de pousser les caractéristiques de la base de données au cœur de l'extraction par le biais de la contrainte. Les motifs virtuels le plus général et le plus spécifique en résumant ces spécificités, encadrent au mieux la valeur des primitives pour chacun des motifs. L'association de ces motifs virtuels avec les opérateurs de bornes produit alors des relaxations monotones et anti-monotones qui réduisent de façon fiable l'espace de recherche. Notre démarche a été entièrement automatisée pour toute PBC à travers les opérateurs de relaxation monotone et anti-monotone.

Nous avons ensuite mis en exergue des contraintes globales qui associent plusieurs motifs et au delà, considèrent la structure du treillis. Ces contraintes globales permettent donc de fournir

une couverture ou de sélectionner les meilleurs motifs. Ne pouvant pas les évaluer directement, nous avons proposé l'approche Approximer-et-Pousser qui relaxe dynamiquement une contrainte globale en une contrainte locale évolutive. Au fur et à mesure de l'extraction, l'analyse des motifs déjà extraits approxime avec précision la collection finale recherchée et permet de réduire efficacement l'espace de recherche. Cette approche Approximer-et-Pousser a été illustrée avec la recherche des k motifs maximisant une mesure d'intérêt, qui fournit à l'utilisateur les motifs les plus significatifs au regard de son critère. Pour cela, nous avons exploité à nouveau l'opérateur de relaxation anti-monotone.

Extraction de motifs ensemblistes contraints

Notre deuxième contribution est relative aux bases de données denses pour le langage des motifs ensemblistes. Dans de telles données, le nombre de motifs explose et, en complément des relaxations, l'exploitation des classes d'équivalence devient particulièrement profitable.

Nous avons d'abord proposé un algorithme d'extraction de représentations condensées constituées d'intervalles contenant tous les motifs satisfaisant une contrainte fondée sur des primitives. L'approche exploite un opérateur d'élagage, combinant les opérateurs de bornes, qui garantit que tous les motifs d'un intervalle satisfont ou pas la contrainte. D'autre part, l'introduction d'un nouvel opérateur de fermeture basé sur un préordre structure le treillis de sorte que chaque classe d'équivalence est exactement décrite par un intervalle. Grâce aux bonnes propriétés de cette fermeture, l'algorithme en profondeur Music-dfs énumère alors tous les intervalles en leur appliquant récursivement l'opérateur d'élagage. Au final, chaque motif satisfaisant la contrainte est contenu dans un unique intervalle de la représentation condensée d'intervalles produite par Music-dfs. Cet algorithme performant se révèle aussi très efficace pour les bases de données comportant un très grand nombre d'items par transaction.

Dans un second temps, nous avons introduit les représentations condensées exactes et adéquates aux fonctions conservées et à leurs combinaisons. Ces représentations permettent d'inférer la valeur de nombreuses contraintes et mesures relatives à la fréquence et à d'autres primitives d'agrégats plus originales. La force de ces représentations condensées est d'être issues d'un nouvel opérateur de fermeture adéquat à la fonction considérée. Il est alors aisé de les extraire en adaptant les solveurs dédiés à l'opérateur de fermeture classique. Notre algorithme d'extraction MicMac a démontré la grande concision de ces nouvelles représentations condensées adéquates. Puis, nous nous sommes intéressés aux mesures de fréquences, comme le taux de croissance, utiles pour dériver des règles de classification et de caractérisation. Parmi elles, nous avons identifié les mesures fortes. Ces mesures sont toutes simultanément optimisées par les motifs forts.

Usages des motifs contraints

Nous avons ensuite montré comment ces résultats sur les motifs contraints permettaient de traiter différents problèmes applicatifs réels. Dans le domaine industriel, les motifs forts ont permis d'identifier des équipements défectueux dans une chaîne de production de plaques de silicium en collaboration avec la société Philips. Sur les données médicales relatives aux maladies de l'athérosclérose, la découverte de facteurs de risques généraux ou restreints à des catégories sociales a validé des connaissances du domaine tout en les quantifiant. Une synthèse des différents stades de la fibrose a aussi été effectuée sur les données liées à l'hépatite. Pour cela, nous avons proposé une méthode d'obtention de clusters sous contraintes en utilisant la fermeture adéquate au taux de croissance. Ainsi, le modèle obtenu considère au mieux les différents stades

de la fibrose.

Enfin, des motifs contraints ont été extraits dans de très larges jeux de données d'expression de gènes SAGE. La richesse sémantique des PBC a été particulièrement efficace pour pousser les connaissances du domaine médical, issues de plusieurs jeux de données, au cœur de l'extraction. Un motif a alors pu être isolé et interprété par les experts biologistes, qui ont ainsi pu identifier un gène dont l'inhibition apporterait un bénéfice thérapeutique dans le traitement du cancer.

Perspectives

Nos perspectives de recherche découlent des prolongements de nos travaux et de façon plus générale, s'intéressent à la formulation des contraintes en s'appuyant sur des modèles de primitives ou l'apprentissage.

Extension et usages des extractions de motifs contraints : vers la production de motifs globaux

Pour les contraintes locales, les résultats obtenus avec le solveur Music-dfs sont très probants car l'opérateur d'élagage réduit fortement le parcours de l'espace de recherche. Comme cet opérateur est défini pour tout langage, l'algorithme Music-dfs pourrait être adapté à d'autres langages. Pour cela, il est nécessaire d'adapter la structuration de l'espace en intervalles utilisée pour les motifs ensemblistes à tout langage. Des efforts plus intenses doivent être consacrés à l'extraction de contraintes globales. En effet, les motifs globaux essentiellement produits en post-traitant des motifs locaux peuvent sûrement être définis en terme de contraintes globales. L'approche Approximer-et-Pousser serait alors une méthode raisonnable pour leur extraction directe, mais surtout ouvrirait la voie à d'autres motifs globaux dont la génération est encore trop complexe.

Les usages fondés sur les motifs locaux sont encore trop confinés aux seuls motifs fréquents ou à leurs représentations. Nous estimons que l'expressivité des PBC en découvrant des motifs locaux de meilleure qualité, peut aussi participer à la construction de motifs globaux et de modèles plus pertinents. Les représentations condensées adéquates poursuivent un but similaire. Leurs propriétés utiles de minimalité et de maximalité adaptées à de nombreuses mesures d'intérêt ouvrent la voie à la généralisation des usages actuels limités à la seule mesure de fréquence. Par exemple, la construction de bases génériques de règles [Bastide *et al.*, 2000] peut être étendue aux règles contraintes.

Primitives et bases de données

Un atout important du PBF est la liberté d'ajouter des primitives à volonté au gré des besoins de l'utilisateur. Néanmoins, il serait intéressant de lui proposer, d'emblée, des primitives a priori pertinentes. Afin de découvrir des informations significatives au sein de la base de données, les primitives doivent en être représentatives. Le cadre de Mannila et Toivonen, exploité dans notre travail, reste vague sur la forme de cette base de données. Pourtant, des dimensions essentielles se dégagent comme la séquentialité (e.g., évolution des données) ou la granularité (e.g., structuration d'un document). Des archétypes de primitives telle que la fréquence qui traduit les régularités, peuvent être définis en fonction de ces axes indépendamment de la base de données. Par exemple, étant donné la structure d'un document, les primitives de fréquences correspondant à chaque niveau (e.g., section, paragraphe, phrase) peuvent être proposées directement à

l'utilisateur. Ces méta-primitives permettraient, par exemple, de formuler la notion de contraste entre les granularités pour n'importe quelle base de données comportant ces différents grains.

Apprentissage de contraintes

La contrainte d'extraction synthétise l'intérêt que l'utilisateur porte sur les données intéressantes. Elle constitue en elle-même une connaissance importante parfois difficile à formuler. Nous pensons que la démarche suivante pourrait aider l'expert à formuler cette contrainte. Dans un premier temps, l'expert dégage les motifs qu'il estime pertinents pour traduire des connaissances ciblées. Puis il s'agit d'apprendre par induction la contrainte à partir de ces motifs. Cet apprentissage pourrait être effectué grâce aux primitives en observant leurs valeurs pour les motifs virtuels le plus spécifique et le plus général. Par exemple, de nombreux motifs pertinents de gènes sur-exprimés sont déjà connus à travers la littérature. En observant leurs valeurs pour des primitives classiques comme la fréquence, nous pensons qu'il est possible d'apprendre une contrainte utile ou tout au moins, une de ses formes préliminaires.

Annexe A

Liste des primitives du cadre fondé sur les primitives

Cette annexe présente les primitives du PBF utilisées dans ce mémoire. Cette liste peut évidemment être complétée.

A.1 Les primitives des motifs ensemblistes

Nous donnons ci-dessous un sous-ensemble des primitives de $\mathcal{P}$ dédiées aux motifs ensemblistes. Les domaines de définition de certaines primitives sont parfois restreints. Par exemple, les opérateurs arithmétiques ne sont pas définis sur les réels négatifs. Toutes les primitives présentées sont implémentées dans MUSIC-DFS. Notons que l'illustration de chaque primitive s'appuie sur la base de données de la figure 12.1 (page 154).

Primitives sur le contexte $\mathcal{D}$:

$freq(X)$	fréquence du motif X	$freq(ABC) = 2$
$freq(X, \mathcal{D})$	fréquence du motif X dans le contexte $\mathcal{D}$	
$g(X)$	extension du motif X	$g(ABC) = \{t_3, t_4\}$
$f(T)$	intension de l'ensemble T	$f(\{t_3, t_4\}) = ABCD$

Primitives sur la table de valeur val :

$sum(S.val)$	somme des valeurs val des items de S
$min(S.val)$	valeur val minimale des items de S
$max(S.val)$	valeur val maximale des items de S

Primitives sur la matrice de similarité `sim` :

$sumsim(S, \texttt{sim})$	somme des similarités de chaque paire d'items de S de la matrice de similarité `sim`	$sumsim(ABC, \texttt{sim}) = 0.13$
$minsim(S, \texttt{sim})$	similarité minimale entre les items de S	$minsim(ABC, \texttt{sim}) = 0.06$
$maxsim(S, \texttt{sim})$	similarité maximale entre les items de S	$maxsim(ABC, \texttt{sim}) = 0.07$

$svsim(S, \texttt{sim})$	nombre de paires d'items de S renseignées dans `sim`	$svsim(ABC, \texttt{sim}) = 2$
$mvsim(S, \texttt{sim})$	nombre de paires d'items de S non-renseignées dans `sim`	$mvsim(ABC, \texttt{sim}) = 1$
$insim(S, x, y, \texttt{sim})$	nombre de paires d'items de S dont la similarité est comprise entre x et y	$insim(ABC, 0.07, 1, \texttt{sim}) = 1$

Primitive sur le texte `text` :

$regexp(S, RE, \texttt{text})$	items de S dont une expression de `text` satisfait RE	$regexp(ABC, '*ion*', \texttt{text}) = AC$

Autres primitives :

$count(S)$	longueur (ou cardinalité) de S	$count(ABC) = 3$
$\cup, \cap, \setminus$	opérateurs ensemblistes	
$+, -, \times, /$	opérateurs arithmétiques	
$<, \leq, \subset, \subseteq$	opérateurs de comparaison	
$\neg, \wedge, \vee$	opérateurs booléens	

A.2 Les primitives des motifs séquentiels

Nous présentons maintenant les primitives dédiées aux motifs séquentiels. Pour les primitives *sum*, *min*, *max* et *count*, les items de chaque motif ensembliste s_i sont considérés où la séquence $S = s_1 s_2 \dots s_n$. Par exemple, pour la séquence $\langle (C)(AD)(A) \rangle$, on considère le multi-ensemble d'items $\{C, A, D, A\}$ et $count(\langle (C)(AD)(A) \rangle) = 4$. Les exemples ci-dessous se réfèrent au tableau A.1.

$\mathcal{D}_\mathcal{S}$

Trans.	Séquence
t_1	$\langle (C)(A) \rangle$
t_2	$\langle (AB)(C)(ADF) \rangle$
t_3	$\langle (ACE) \rangle$
t_4	$\langle (C)(AD)(A) \rangle$
t_5	$\langle (B)(A) \rangle$

Item	A	B	C	D	E	F
val	50	30	75	10	30	15

TAB. A.1 – Exemple d'une base de données séquentielles.

Primitives sur le contexte $\mathcal{D}_\mathcal{S}$:

$freq(X)$	fréquence du motif X	$freq(\langle (C)(AD) \rangle) = 2$
$freq(X, \mathcal{D},)$	fréquence du motif X dans le contexte $\mathcal{D}$	

Primitives sur la table de valeur val :

$sum(S.val)$	somme des valeurs val des items de S	$sum(\langle (C)(AD) \rangle.val) = 75$
$min(S.val)$	valeur val minimale des items de S	$min(\langle (C)(AD) \rangle.val) = 10$
$max(S.val)$	valeur val maximale des items de S	$max(\langle (C)(AD) \rangle.val) = 50$

Autres primitives :

$count(S)$	longueur (ou cardinalité) de S	$count(\langle(C)(AD)\rangle) = 3$
$+, -, \times, /$	opérateurs arithmétiques	
$<, \leq, \subset, \subseteq$	opérateurs de comparaison	
$\neg, \wedge, \vee$	opérateurs booléens	

Annexe B

Liste des jeux de données

Cette annexe présente succinctement les jeux de données utilisés pour les motifs ensemblistes et séquentiels dans les différentes expérimentations. Ils sont utilisés dans les expériences du chapitre 6 au chapitre 9 et dans l'annexe C.

Motifs ensemblistes

Tous les jeux de données proviennent de l'UCI Machine Learning Repository (`www.ics.uci.edu/~mlearn/MLRepository.html`) [D.J. Newman et Merz, 1998], excepté le jeu de données RETAIL[26]. Les attributs à valeur continue pour ABALONE, CMC, WINE et LETTER ont été segmentés en 3 items de sorte que chaque item appartiennent au même nombre de transactions. Enfin, les versions utilisées des jeux de données `pumsb*`, `mushroom` et `chess` ont été préparés par Roberto Bayardo. Ces données sont disponibles sur le site du FIMI (`fimi.cs.helsinki.fi/data/`) tout comme le jeu de données `retail`.

Le tableau suivant récapitule les caractéristiques principales de ces jeux de données :

Nom	Nbr. de transaction	Nbr. d'items	Nbr. d'items au maximum par transaction
`abalone`	4177	28	9
`cmc`	1473	28	10
`wine`	178	45	14
`letter`	20000	74	17
`chess`	3196	75	37
`mushroom`	8124	119	23
`pumsb*`	49046	2088	63
`retail`	88162	16470	76

Motifs séquentiels

Pour les motifs séquentiels, nous avons utilisé un seul jeu de données de séquentiel (cf. la section 6.3.4). Ce jeu de données, dénoté par $C100T2.5S10I2.5$, a été généré en utilisant la procédure standard décrite dans [Agrawal et Srikant, 1995]. Dans celui-ci, le nombre d'items est fixé à 1000 et il y a 10000 séquences dans le jeu de données. Le nombre moyen d'items par ensemble est 2.5 (dénoté $T2.5$). Le nombre moyen d'ensembles par séquence est de 10

[26] Ces données correspondent aux ventes d'une grande surface [Brijs *et al.*, 1999].

(dénoté par $S10$). $C100T2.5S10I2.5$ a été produit avec l'outil de génération disponible sur le site `http://illimine.cs.uiuc.edu/`.

Annexe C

Expériences complémentaires pour MUSIC-DFS

Cette annexe regroupe des expérimentations additionnelles effectuées avec l'algorithme MUSIC-DFS. Plus précisément, ces diverses expérimentations complètent celles présentées au chapitre 8 en se focalisant sur des jeux de données comportant beaucoup de transactions. À nouveau, elles ont été effectuées sur un ordinateur doté d'un processeur Xeon 2.2 GHz et de 3GB de mémoire RAM avec le système d'exploitation Linux.

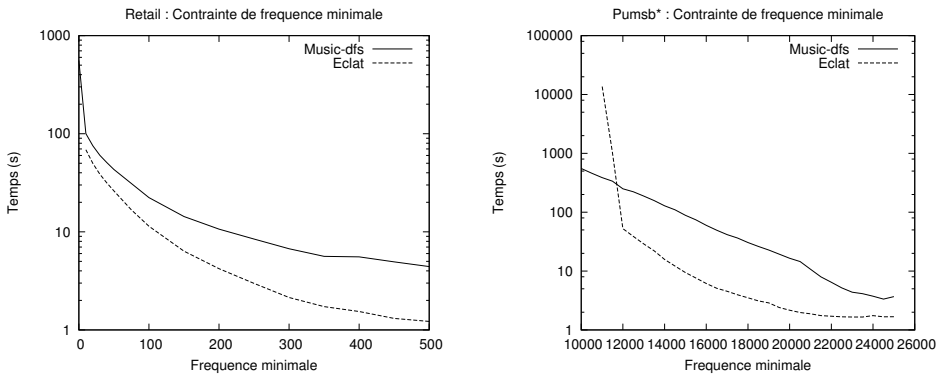
La section C.1 observe les performances de MUSIC-DFS en fonction de la fréquence minimale d'extraction. La section C.2 décrit le comportement de MUSIC-DFS (i.e., rapidité, efficacité de l'élagage et condensation) en fonction de la sélectivité. Enfin, l'apport de la relaxation anti-monotone avec MUSIC-DFS est illustré à la section C.3.

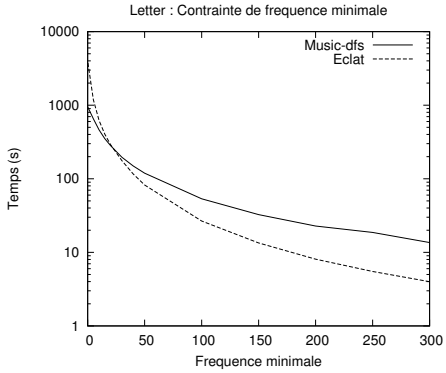
C.1 Performances de MUSIC-DFS en fonction de la fréquence

Les différentes expérimentations de cette section portent sur des jeux de données de grandes dimensions (i.e., `letter`, `retail` et `pumsb*`) dont les principales caractéristiques sont présentées à l'annexe B.

C.1.1 MUSIC-DFS vs. ECLAT

Nous comparons l'efficacité de l'algorithme MUSIC-DFS par rapport à celle de l'algorithme ECLAT [Zaki, 2000b] pour l'extraction des motifs fréquents. Les courbes ci-dessous reportent les temps d'exécution en fonction du seuil de fréquence minimale :



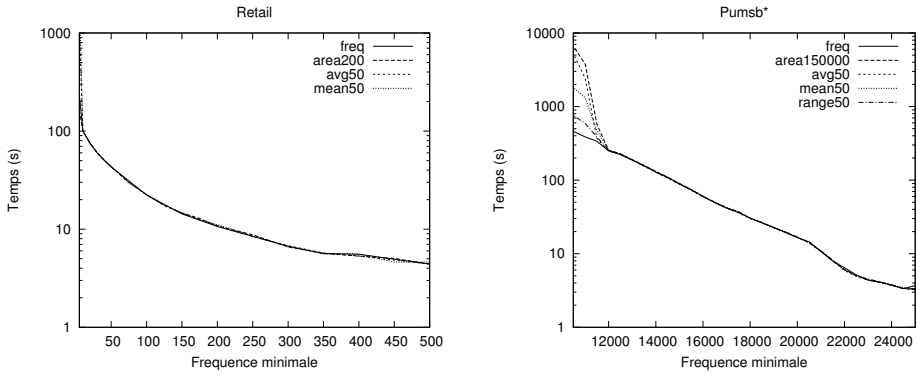


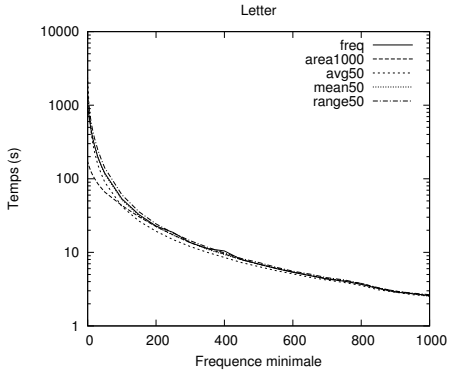
Lorsque la fréquence minimale devient très basse MUSIC-DFS devient plus performant que ECLAT. Ainsi, sur ces trois jeux de données l'algorithme ECLAT finit par échouer.

C.1.2 MUSIC-DFS avec des contrainte variées

Nous détaillons maintenant l'efficacité de MUSIC-DFS pour extraire les motifs satisfaisant des contraintes variées lorsque le seuil de fréquence minimale diminue. La légende des courbes correspond à :

freq : $freq(X) \geq minfr$
areaα : $freq(X) \times count(X) \geq \alpha$
avg50 : $sum(X.val)/count(X) \geq 50$
mean50 : $(max(X.val) + min(X.val))/2 \geq 50$
range50 : $(max(X.val) - min(X.val)) \geq 50$





Sur `retail` et `letter`, les temps d'exécution des extractions pour les différentes contraintes sont comparables. En revanche pour `pumsb`*, lorsque la fréquence minimale est en dessous de 12000, l'explosion du nombre de motifs influence les temps d'extraction. En particulier, l'élagage sur les intervalles devient significatif pour certaines contraintes comme la fréquence. Remarquons que pour un seuil de fréquence inférieur à 12000, toutes les extractions sont plus efficaces que celle précédemment effectuée avec ECLAT.

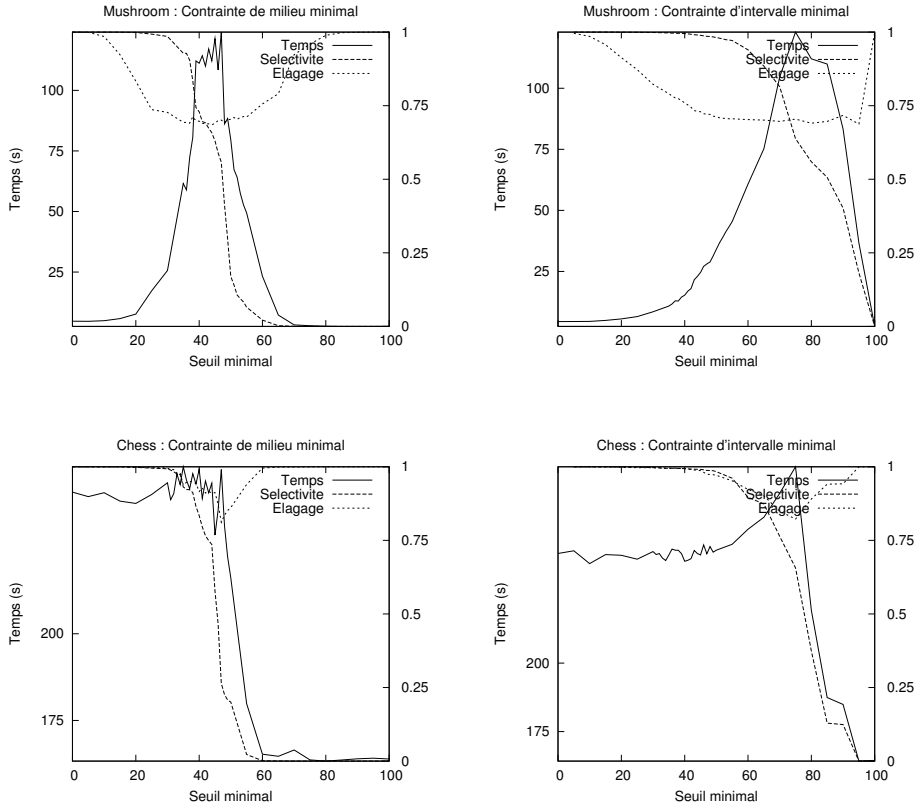
C.2 Comportement de MUSIC-DFS en fonction de la sélectivité

La section 8.3.1 a montré l'importance de la sélectivité pour l'efficacité de MUSIC-DFS au niveau des temps d'exécution et de la condensation. Les différentes expérimentations menées dans cette section confirment ces observations pour deux autres contraintes :

- contrainte de milieu minimal (*mean*) : $(max(X.val) + min(X.val))/2 \geq seuil$
- contrainte d'intervalle minimal (*range*) : $max(X.val) - min(X.val) \geq seuil$

C.2.1 Peformances d'exécution

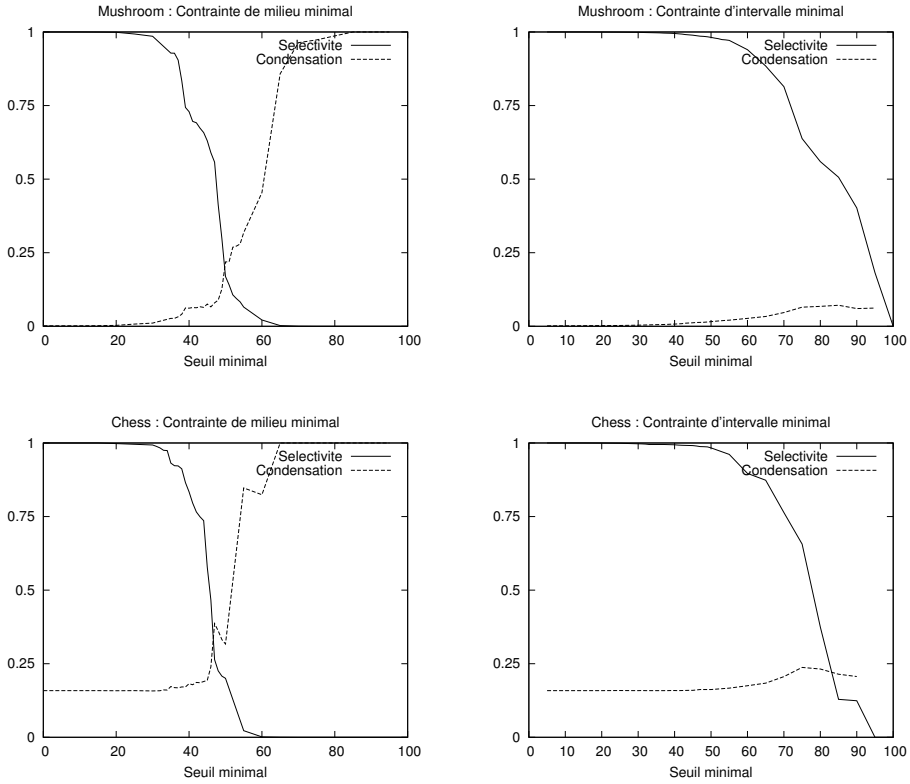
Les courbes ci-dessous reportent les temps d'exécution (axes des ordonnées de gauche), le taux d'élagage et le taux de sélectivité (axes des ordonnées de droite) en fonction du seuil minimal de la contrainte considérée :



Le temps d'exécution le plus long pour chaque expérimentation coïncide avec un taux de sélectivité de 0.5 et le moins bon taux d'élagage.

C.2.2 Qualité de la condensation

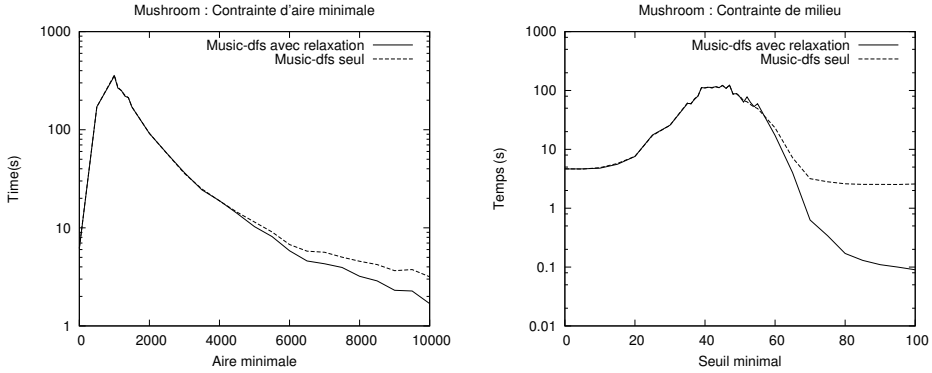
Les courbes ci-dessous reportent le taux de sélectivité et le taux de condensation :



Les courbes du taux de condensation sont interrompues dès que plus aucun motif ne satisfait la contrainte. À nouveau, la concision est d'autant plus forte que le nombre de motifs extraits est important (taux de sélectivité proche de 1).

C.3 MUSIC-DFS avec relaxation

Cette section montre le bénéfice de l'usage de la relaxation anti-monotone calculée au chapitre 6 en conjonction de l'algorithme MUSIC-DFS. Plus précisément, les courbes reportent les temps d'exécution de MUSIC-DFS avec et sans l'usage de la relaxation anti-monotone. Rappelons que les relaxations anti-monotones correspondant aux contraintes d'aire minimale ou de milieu minimal sont données dans le tableau 6.3 de la page 73.



L'impact de la relaxation anti-monotone se ressent particulièrement lorsque la sélectivité de la contrainte est forte comme pour les expérimentations effectuées avec ECLAT et APRIORI (cf. la section 6.3.4).

Annexe D

Exemples de flow-charts

Cette annexe donne des extraits de flow-charts utilisés lors de notre collaboration avec la société PHILIPS (cf. le chapitre 10).

D.1 Extrait de flow-chart

```
ERIC : PHILIPS Composants     ARC_LOT    Archive Lot Progress.                    L104-R0   Date : 10-SEP-2002 08:13   Page : 0001

Lot-book of lot:  J85516C       Product type: RS3332A2/1C           Process: R00606     Account status: P     Lot owner     : PROD
Substrate 12NC : 351160086203   Ingot number: 1T24153111+1VMM+D020                      defined by    : BAC   delivered by  : FLD
Date      Time  Sub B Event                 Tech.   Step        Equip- Empl S #Wf Message /
dd-mmm-yy hh:mm lot                         Stage   Id.           ment            Due    #Obs   Step checked        EqmChk
-------------------------------------------------------------------------------------------------------------------------------
06-Jul-02 06:12       ERIC-SYS   LOTDEFINED                                BAC        Lot Defined.
06-Jul-02 06:12       Remarque   Lot                                       BAC        Remarques sur les lots
                                                                                     Groupe de Procédé ECD
                                                                                     025 plaque(s) du lot   X3956
                                                                                      Type XBOX         Procede XBOX
06-Jul-02 06:29       LOT MOVIN              02-BOT  A                    BAC     25   2
06-Jul-02 06:29       ERIC-SYS   Keuze Aanm  02-BOT  A                    BAC        Choix effectué au move in :
                                                                                     NR SUBSTRAT BOITE ????     :  P331100779154
06-Jul-02 06:29       LOT MOVOUT             02-BOT  A                    BAC     25
06-Jul-02 06:29       InlineTest             02-BOT  A                    BAC      1        1      02-LOT  A
                              Parameter: 033761 1=MEMC2=WACKER     unit:                              low       high    target
                                                                                     Req. limits            1         4         1
                                                                                     Outlier limits         1         4
                                        Statistics:  average         1.0, stand.dev  n.a.      , #values   1
                                        Details   : wafer 01          1
08-Jul-02 01:25       LOT MOVIN              02-BOT  B                992 BAC     25   0
08-Jul-02 01:25       LOT MOVOUT             02-BOT  B                992 BAC     25
08-Jul-02 06:24       LOT MOVIN              02-BOT  C                159 DTR     25   0
08-Jul-02 06:52       LOT MOVOUT             02-BOT  C                159 DTR     25
08-Jul-02 07:18       LOT MOVIN              03-MAT  D               709P JF      25   0 Used mask :99999CC
08-Jul-02 07:46       LOT MOVOUT             03-MAT  D               709P JF      25     Used mask : 99999CC
08-Jul-02 07:46       InlineTest             03-MAT  D               709P JF       1        1      03-MA   D             709P
```

1

D.2 Exemple de traitement sur un sous-lot

```
13-Jul-02 12:12      ERIC-SYST. Opr.inter.  11-PC   F             482 XML       Operation interrompue.
                                                                               ESSAI SUR LAM 933 AVEC ST
13-Jul-02 12:13    B LOT MOVIN              11-PC   F             433 BEB    22   1
13-Jul-02 12:22      Remarque   Lot         11-PC   F             433 XML       Remarques sur les lots
                                                                               1 ere plaque sur 433 ok
                                                                                vu avec st
                                                                                reste du lot a graver sur 433
                                                                                aureole sur les 3 premieres pls passees sur 482
13-Jul-02 14:52    B LOT MOVOUT             11-PC   F             433 BEB    22
13-Jul-02 14:58    B InlineTest             11-PC   F             433 BEB     1       1      11-PC   F             433
                           Parameter: 034980 TEMPS PS    BIMOS  unit: sec                          low       high    target
                                                                               Req. limits           35        85        59
                                                                               Outlier limits         0       150
                                                                               Control limits        35        85        59
                                    Statistics:  average        78.0, stand.dev  n.a.     , #values   1
                                    Details    : wafer 01         78F
13-Jul-02 14:52      ERIC-SYST. Perc.Rej.   11-PC   F               BEB       Perc rejected values exceeded.
                                                                               Parameter : 034980 / TEMPS PS    BIMOS
                                                                               Lot bloque par le Systeme
13-Jul-02 14:56      Remarque   Lot         11-PC   F             433 TEG       Remarques sur les lots
                                                                               temps de gravure legerement superieur a LSC suite
                                                                               epaisseur de poly forte
13-Jul-02 15:01      ERIC-SYST. Lot rls.    11-PC   F               TEG       Lot debloque.
                                                                               FAIBLE AUREOLES SUR 3 PREMIERES PLS => ACC
```

Bibliographie

[Agrawal *et al.*, 1993] Agrawal, R., Imielinski, T. et Swami, A. N. (1993). Mining association rules between sets of items in large databases. *In* Buneman, P. et Jajodia, S., éditeurs : *SIGMOD Conference*, pages 207–216. ACM Press.

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

[Agrawal et Srikant, 1994] Agrawal, R. et Srikant, R. (1994). Fast algorithms for mining association rules in large databases. *In* Bocca, J. B., Jarke, M. et Zaniolo, C., éditeurs : *VLDB*, pages 487–499. Morgan Kaufmann.

[Agrawal et Srikant, 1995] Agrawal, R. et Srikant, R. (1995). Mining sequential patterns. *In* Yu, P. S. et Chen, A. L. P., éditeurs : *ICDE*, pages 3–14. IEEE Computer Society.

[Al-Hajj et Clarke, 2004] Al-Hajj, M. et Clarke, M. F. (2004). Self-renewal and solid tumor stem cells. *Oncogene*, 23:7274–7282.

[Antunes et Oliveira, 2004] Antunes, C. et Oliveira, A. L. (2004). Constraint relaxations for discovering unknown sequential patterns. *In* Goethals, B. et Siebes, A., éditeurs : *KDID 2004, Knowledge Discovery in Inductive Databases, Proceedings of the Third International Workshop on Knowledge Discovery inInductive Databases, Pisa, Italy, September 20, 2004, Revised Selected and Invited Papers*, volume 3377 de *Lecture Notes in Computer Science*, pages 11–32. Springer.

[Bastide *et al.*, 2000] Bastide, Y., Pasquier, N., Taouil, R., Stumme, G. et Lakhal, L. (2000). Mining minimal non-redundant association rules using frequent closed itemsets. *In* Lloyd, J. W., Dahl, V., Furbach, U., Kerber, M., Lau, K.-K., Palamidessi, C., Pereira, L. M., Sagiv, Y. et Stuckey, P. J., éditeurs : *Computational Logic*, volume 1861 de *Lecture Notes in Computer Science*, pages 972–986. Springer.

[Bayardo, 1998] Bayardo, R. J. (1998). Efficiently mining long patterns from databases. *In* Haas, L. M. et Tiwary, A., éditeurs : *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA.*, pages 85–93. ACM Press.

[Bayardo, 2005] Bayardo, R. J. (2005). The hows, whys, and whens of constraints in itemset and rule discovery. *In Proc. of the Workshop on Inductive Databases and Constraint Based Mining (IDW'05).*

[Becquet *et al.*, 2002] Becquet, C., Blachon, S., Jeudy, B., Boulicaut, J.-F. et Gandrillon, O. (2002). Strong association rule mining for large gene expression data analysis : A case study on human sage data. *Genome Biology*, 3(12):16 pages.

[Beyer et Ramakrishnan, 1999] Beyer, K. S. et Ramakrishnan, R. (1999). Bottom-up computation of sparse and iceberg cubes. *In* Delis, A., Faloutsos, C. et Ghandeharizadeh, S., éditeurs : *SIGMOD Conference*, pages 359–370. ACM Press.

[Birkhoff, 1967] Birkhoff, G. (1967). Lattices theory. *American Mathematical Society, vol. 25.*

[Bistarelli et Bonchi, 2005] Bistarelli, S. et Bonchi, F. (2005). Interestingness is not a dichotomy : Introducing softness in constrained pattern mining. *In* Jorge, A., Torgo, L., Brazdil, P., Camacho, R. et Gama, J., éditeurs : *PKDD*, volume 3721 de *Lecture Notes in Computer Science*, pages 22–33. Springer.

[Bonchi et Giannotti, 2004] Bonchi, F. et Giannotti, F. (2004). Pushing constraints to detect local patterns. *In* Morik, K., Boulicaut, J.-F. et Siebes, A., éditeurs : *Local Pattern Detection*, volume 3539 de *Lecture Notes in Computer Science*, pages 1–19. Springer.

[Bonchi *et al.*, 2003] Bonchi, F., Giannotti, F., Mazzanti, A. et Pedreschi, D. (2003). Exante : Anticipated data reduction in constrained pattern mining. *In* Lavrac, N., Gamberger, D., Blockeel, H. et Todorovski, L., éditeurs : *PKDD*, volume 2838 de *Lecture Notes in Computer Science*, pages 59–70. Springer.

[Bonchi et Goethals, 2004] Bonchi, F. et Goethals, B. (2004). FP-bonsai : The art of growing and pruning small FP-trees. *In* Dai, H., Srikant, R. et Zhang, C., éditeurs : *Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference, PAKDD 2004, Sydney, Australia, May 26-28, 2004, Proceedings*, volume 3056 de *Lecture Notes in Computer Science*, pages 155–160. Springer.

[Bonchi et Lucchese, 2004] Bonchi, F. et Lucchese, C. (2004). On closed constrained frequent pattern mining. *In Proceedings of the 4th IEEE International Conference on Data Mining (ICDM 2004), 1-4 November 2004, Brighton, UK*, pages 35–42. IEEE Computer Society.

[Bonchi et Lucchese, 2005] Bonchi, F. et Lucchese, C. (2005). Pushing tougher constraints in frequent pattern mining. *In* Ho, T. B., Cheung, D. et Liu, H., éditeurs : *Advances in Knowledge Discovery and Data Mining, 9th Pacific-Asia Conference, PAKDD 2005, Hanoi, Vietnam, May 18-20, 2005, Proceedings*, volume 3518 de *Lecture Notes in Computer Science*, pages 114–124. Springer.

[Boon *et al.*, 2003] Boon, K., Edwards, J. B., Siu, I. M. et et al. (2003). Comparison of medulloblastoma and normal neural transcriptomes identifies a restricted set of activated genes. *Oncogene*, 23:7687–7694.

[Borgelt et Kruse, 2002] Borgelt, C. et Kruse, R. (2002). Induction of association rules : Apriori implementation. *In 15th Conference on Computational Statistics (Compstat 2002, Berlin, Germany)*, pages 395–400. Physica Verlag.

[Boulicaut et Bykowski, 2000] Boulicaut, J.-F. et Bykowski, A. (2000). Frequent closures as a concise representation for binary data mining. *In Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 62–73.

[Boulicaut *et al.*, 2000] Boulicaut, J.-F., Bykowski, A. et Rigotti, C. (2000). Approximation of frequency queris by means of free-sets. *In* Zighed, D. A., Komorowski, H. J. et Zytkow, J. M., éditeurs : *Principles of Data Mining and Knowledge Discovery, 4th European Conference, PKDD 2000, Lyon, France, September 13-16, 2000, Proceedings*, volume 1910 de *Lecture Notes in Computer Science*, pages 75–85. Springer.

[Boulicaut et Jeudy, 2000] Boulicaut, J.-F. et Jeudy, B. (2000). Using constraint for itemset mining : Should we prune or not ? *In* Doucet, A., éditeur : *BDA00*, pages 221–237, Blois, France. Université de Tours.

[Boulicaut et Jeudy, 2001] Boulicaut, J.-F. et Jeudy, B. (2001). Mining free itemsets under constraints. *In International Database Engineering and Application Symposium*, pages 322–329.

[Brachman et Anand, 1996] Brachman, R. J. et Anand, T. (1996). The process of knowledge discovery in databases. *In* Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P. et Uthurusamy, R., éditeurs : *Advances in Knowledge Discovery and Data Mining*, pages 37–57. AAAI/MIT Press.

[Brijs *et al.*, 1999] Brijs, T., Swinnen, G., Vanhoof, K. et Wets, G. (1999). Using association rules for product assortment decisions : A case study. *In Knowledge Discovery and Data Mining*, pages 254–260.

[Bucila *et al.*, 2002] Bucila, C., Gehrke, J., Kifer, D. et White, W. (2002). DualMiner : A dual-pruning algorithm for itemsets with constraints. *In Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*. ACM.

[Burdick *et al.*, 2001] Burdick, D., Calimlim, M. et 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. IEEE Computer Society.

[Calders et Goethals, 2002] Calders, T. et Goethals, B. (2002). Mining all non-derivable frequent itemsets. *In proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'02)*, pages 74–85.

[Calders et Goethals, 2003] Calders, T. et Goethals, B. (2003). Minimal k-free representations of frequent sets. *In proceedings of the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'03)*, pages 71–82. Springer.

[Calders *et al.*, 2004] Calders, T., Rigotti, C. et Boulicaut, J.-F. (2004). A survey on condensed representations for frequent sets. *In* Boulicaut, J.-F., Raedt, L. D. et Mannila, H., éditeurs : *Constraint-Based Mining and Inductive Databases, European Workshop on Inductive Databases and Constraint Based Mining, Hinterzarten, Germany, March 11-13, 2004, Revised Selected Papers*, volume 3848 de *Lecture Notes in Computer Science*, pages 64–80. Springer.

[Casas-Garriga, 2003] Casas-Garriga, G. (2003). Towards a formal framework for mining general patterns from ordered data. *In MRDM 2003 2nd Workshop on Multi-Relational Data Mining Preliminary schedule*.

[Chi *et al.*, 2005] Chi, Y., Xia, Y., Yang, Y. et Muntz, R. R. (2005). Mining closed and maximal frequent subtrees from databases of labeled rooted trees. *IEEE Trans. Knowl. Data Eng.*, 17(2):190–202.

[Clark et Boswell, 1991] Clark, P. et Boswell, R. (1991). Rule induction with CN2 : Some recent improvements. *In Proc. Fifth European Working Session on Learning*, pages 151–163, Berlin. Springer.

[Codd, 1970] Codd, E. F. (1970). A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387.

[Cong, 2001] Cong, S. (2001). Mining the top-k frequent itemset with minimum length m.

[de Knijf et Feelders, 2005] de Knijf, J. et Feelders, A. (2005). Monotone constraints in frequent tree mining. *In* van Otterlo, M., Poel, M. et Nijholt, A., éditeurs : *the 14 th Annual Machine Learning Conference of Belgium and the Netherlands*, pages 13–20.

[Dehaspe, 1998] Dehaspe, L. (1998). *Frequent pattern discovery in first-order logic.* Thèse de doctorat, Katholieke Universiteit Leuven.

[Derrington *et al.*, 1998] Derrington, E. A., Dufay, N., Rudkin, B. B. et Belin, M. F. (1998). Human primitive neuroectodermal tumour cells behave as multipotent neural precursors in response to FGF2. *Annu Rev Biochem*, 17:1663–1672.

[Diop, 2003] Diop, C. T. (2003). *Etude et mise en oeuvre des aspects itératifs de l'extraction de règles d'association dans une base de données.* Thèse de doctorat, Université de Tours.

[D.J. Newman et Merz, 1998] D.J. Newman, S. Hettich, C. B. et Merz, C. (1998). UCI repository of machine learning databases.

[Dong et Li, 1999] Dong, G. et Li, J. (1999). Efficient mining of emerging patterns : discovering trends and differences. *In Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'99)*, pages 43–52, New York, NY, USA. ACM Press.

[Dong *et al.*, 1999] Dong, G., Zhang, X., Wong, L. et Li, J. (1999). CAEP : Classification by aggregating emerging patterns. *In* Arikawa, S. et Furukawa, K., éditeurs : *Discovery Science*, volume 1721 de *Lecture Notes in Computer Science*, pages 30–42. Springer.

[Durand *et al.*, 2004] Durand, N., Cleuziou, G. et Soulet, A. (2004). Discovery of overlapping clusters to detect atherosclerosis risk factors. *In proceedings of the workshop Discovery Challenge, PKDD'04.*

[Durand et Crémilleux, 2002] Durand, N. et Crémilleux, B. (2002). ECCLAT : a New Approach of Clusters Discovery in Categorical Data. *In the 22nd Int. Conf. on Knowledge Based Systems and Applied Artificial Intelligence (ES'02)*, pages 177–190, Cambridge, UK.

[Durand et Soulet, 2005] Durand, N. et Soulet, A. (2005). Emerging overlapping clusters for characterizing the stage of fibrosis. *In proceedings of the workshop Discovery Challenge, PKDD'05*, pages 139–150.

[El-Hajj et Zaïane, 2005] El-Hajj, M. et Zaïane, O. R. (2005). Finding all frequent patterns starting from the closure. *In* Li, X., Wang, S. et Dong, Z. Y., éditeurs : *Advanced Data Mining and Applications, First International Conference, ADMA 2005, Wuhan, China, July 22-24, 2005, Proceedings*, volume 3584 de *Lecture Notes in Computer Science*, pages 67–74. Springer.

[El-Hajj *et al.*, 2005] El-Hajj, M., Zaïane, O. R. et Nalos, P. (2005). Bifold constraint-based mining by simultaneous monotone and anti-monotone checking. *In Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), 27-30 November 2005, Houston, Texas, USA*, pages 146–153. IEEE Computer Society.

[Fayyad *et al.*, 1996] Fayyad, U. M., Piatetsky-Shapiro, G. et Smyth, P. (1996). Knowledge discovery and data mining : Towards a unifying framework. *In KDD*, pages 82–88.

[Fischer, 2003] Fischer, J. (2003). Version spaces in constraint-based data mining.

[Flouvat *et al.*, 2004] Flouvat, F., Marchi, F. D. et Petit, J.-M. (2004). ABS : Adaptive borders search of frequent itemsets. *In* Bayardo, R. J., Goethals, B. et Zaki, M. J., éditeurs : *FIMI*, volume 126 de *CEUR Workshop Proceedings*. CEUR-WS.org.

[Fu *et al.*, 2000] Fu, A. W.-C., w. Kwong, R. W. et Tang, J. (2000). Mining n-most interesting itemsets. *In* Ras, Z. W. et Ohsuga, S., éditeurs : *ISMIS*, volume 1932 de *Lecture Notes in Computer Science*, pages 59–67. Springer.

[Ganter, 1984] Ganter, B. (1984). Two basic algorithms in concept analysis. *In Preprint 831, Technische Hochschule Darmstadt.*

[Garofalakis *et al.*, 1999] Garofalakis, M. N., Rastogi, R. et Shim, K. (1999). SPIRIT : Sequential pattern mining with regular expression constraints. *In The VLDB Journal*, pages 223–234.

[Gehrke et Hellerstein, 2004] Gehrke, J. et Hellerstein, J. M. (2004). Guest editorial to the special issue on data stream processing. *VLDB J.*, 13(4):317.

[Giacometti *et al.*, 2002] Giacometti, A., Laurent, D. et Diop, C. T. (2002). Condensed representations for sets of mining queries. *In proceedings of KDID'02*.

[Giannotti *et al.*, 2002] Giannotti, F., Gozzi, C. et Manco, G. (2002). Clustering transactional data. *In* Elomaa, T., Mannila, H. et Toivonen, H., éditeurs : *PKDD*, volume 2431 de *Lecture Notes in Computer Science*, pages 175–187. Springer.

[Goethals, 2003a] Goethals, B. (2003a). FIMI site web. fimi.cs.helsinki.fi.

[Goethals, 2003b] Goethals, B. (2003b). Survey on frequent pattern mining. Manuscript.

[Gouda et Zaki, 2005] Gouda, K. et Zaki, M. J. (2005). Genmax : An efficient algorithm for mining maximal frequent itemsets. *Data Min. Knowl. Discov.*, 11(3):223–242.

[Grahne *et al.*, 2000] Grahne, G., Lakshmanan, L. V. S. et Wang, X. (2000). Efficient mining of constrained correlated sets. *In ICDE*, pages 512–521.

[Gray *et al.*, 1996] Gray, J., Bosworth, A., Layman, A. et Pirahesh, H. (1996). Data cube : A relational aggregation operator generalizing group-by, cross-tab, and sub-total. *In* Su, S. Y. W., éditeur : *ICDE*, pages 152–159. IEEE Computer Society.

[Gray *et al.*, 1997] Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F. et Pirahesh, H. (1997). Data cube : A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *J. Data Mining and Knowledge Discovery*, 1(1):29–53.

[Gunopulos *et al.*, 1997] Gunopulos, D., Khardon, R., Mannila, H. et Toivonen, H. (1997). Data mining, hypergraph transversals, and machine learning. *In PODS*, pages 209–216. ACM Press.

[Hamrouni *et al.*, 2005] Hamrouni, T., Yahia, S. B. et Slimani, Y. (2005). Prince : An algorithm for generating rule bases without closure computations. *In* Tjoa, A. M. et Trujillo, J., éditeurs : *DaWaK*, volume 3589 de *Lecture Notes in Computer Science*, pages 346–355. Springer.

[Han *et al.*, 1997] Han, E.-H., Karypis, G., Kumar, V. et Mobasher, B. (1997). Clustering based on association rule hypergraphs. *In proceedings of the workshop on Research Issues on Data Mining And Knowledge Discovery, SIGMOD 97*.

[Han *et al.*, 2001] Han, J., Pei, J., Dong, G. et Wang, K. (2001). Efficient computation of iceberg cubes with complex measures. *In ACM SIGMOD International Conf. on Management of Data*.

[Han *et al.*, 2000] Han, J., Pei, J. et Yin, Y. (2000). Mining frequent patterns without candidate generation. *In* Chen, W., Naughton, J. F. et Bernstein, P. A., éditeurs : *SIGMOD Conference*, pages 1–12. ACM.

[Han *et al.*, 2002] Han, J., Wang, J., Lu, Y. et Tzvetkov, P. (2002). Mining top-k frequent closed patterns without minimum support. *In ICDM*, pages 211–218. IEEE Computer Society.

[Hipp *et al.*, 2000] Hipp, J., Güntzer, U. et Nakhaeizadeh, G. (2000). Mining association rules : Deriving a superior algorithm by analyzing today's approaches. *In* Zighed, D. A., Komorowski, H. J. et Zytkow, J. M., éditeurs : *Principles of Data Mining and Knowledge*

Discovery, 4th European Conference, PKDD 2000, Lyon, France, September 13-16, 2000, Proceedings, volume 1910 de *Lecture Notes in Computer Science*, pages 159–168. Springer.

[Hirate *et al.*, 2004] Hirate, Y., Iwahashi, E. et Yamana, H. (2004). TF2P-growth : Frequent itemset mining algorithm without any thresholds. *In Proc. of Workshop on Alternative Techniques for Data Mining and Knowledge Discovery (ICDM'04)*.

[Hébert *et al.*, 2007] Hébert, C., Bretto, A. et Crémilleux, B. (2007). Optimizing hypergraph transversal computation with an anti-monotone constraint. *In The 22nd Annual ACM Symposium on Applied Computing (SAC'06)*, Seoul, Korea. À paraître.

[Imielinski et Mannila, 1996] Imielinski, T. et Mannila, H. (1996). A database perspective on knowledge discovery. *In Communication of the ACM*, pages 58–64.

[International Business Machines, 1996] International Business Machines (1996). IBM intelligent miner, user's guide, version 1, release 1.

[Jeudy, 2002] Jeudy, B. (2002). *Optimisation de requètes inductives : application à l'extraction sous contraintes de règles d'association*. Thèse de doctorat, INSA de Lyon.

[Jeudy et Rioult, 2004] Jeudy, B. et Rioult, F. (2004). Database transposition for constrained closed pattern mining. *In proceedings of Third International Workshop on Knowledge Discovery in Inductive Databases (KDID) co-located with ECML/PKDD*.

[Keime *et al.*, 2004] Keime, C., Damiola, F., Mouchiroud, D., Duret, L. et Gandrillon, O. (2004). Identitag, a relational database for sage tag identification and interspecies comparison of sage libraries. *BMC Bioinformatics*, 5 :143.

[Kiefer *et al.*, 2003] Kiefer, D., Gehrke, J., Bucila, C. et White, W. (2003). How to quickly find a witness. *In Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 272–283.

[Klema *et al.*, 2006] Klema, J., Soulet, A., Crémilleux, B., Blachon, S. et Gandrillon, O. (2006). Mining Plausible Patterns from Genomic Data. *In 19th IEEE International Symposium on Computer-Based Medical Systems (CBMS'06)*, pages 90–101, Salt Lake City, Utah.

[Kramer *et al.*, 2001] Kramer, S., Raedt, L. D. et Helma, C. (2001). Molecular feature mining in hiv data. *In KDD*, pages 136–143.

[Kryszkiewicz, 2002] Kryszkiewicz, M. (2002). Inferring knowledge from frequent patterns. *In* Bustard, D. W., Liu, W. et Sterritt, R., éditeurs : *Soft-Ware*, volume 2311 de *Lecture Notes in Computer Science*, pages 247–262. Springer.

[Kuramochi et Karypis, 2001] Kuramochi, M. et Karypis, G. (2001). Frequent subgraph discovery. *In* Cercone, N., Lin, T. Y. et Wu, X., éditeurs : *Proceedings of the 2001 IEEE International Conference on Data Mining, 29 November - 2 December 2001, San Jose, California, USA*, pages 313–320. IEEE Computer Society.

[Lakshmanan *et al.*, 2003] Lakshmanan, L. V. S., Leung, C. K.-S. et Ng, R. T. (2003). Efficient dynamic mining of constrained frequent sets. *ACM Trans. Database Syst.*, 28(4):337–389.

[Lee et Raedt, 2003] Lee, S. D. et Raedt, L. D. (2003). An algebra for inductive query evaluation. *In Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), 19-22 December 2003, Melbourne, Florida, USA*, pages 147–154. IEEE Computer Society.

[Lee et Raedt, 2004] Lee, S. D. et Raedt, L. D. (2004). An efficient algorithm for mining string databases under constraints. *In* Goethals, B. et Siebes, A., éditeurs : *KDID 2004, Knowledge Discovery in Inductive Databases, Proceedings of the Third International Workshop*

on Knowledge Discovery in Inductive Databases, Pisa, Italy, September 20, 2004, Revised Selected and Invited Papers, volume 3377 de *Lecture Notes in Computer Science*, pages 108–129. Springer.

[Leung *et al.*, 2002] Leung, C. K.-S., Lakshmanan, L. V. S. et Ng, R. T. (2002). Exploiting succinct constraints using FP-trees. *SIGKDD Explorations*, 4(1):40–49.

[Li *et al.*, 2001] Li, W., Han, J. et Pei, J. (2001). CMAR : Accurate and efficient classification based on multiple class-association rules. *In* Cercone, N., Lin, T. Y. et Wu, X., éditeurs : *Proceedings of the 2001 IEEE International Conference on Data Mining, 29 November - 2 December 2001, San Jose, California, USA*, pages 369–376. IEEE Computer Society.

[Lim *et al.*, 2006] Lim, D. A., Suarez-Farinas, M., Naef, F. et et al. (2006). In vivo transcriptional profile analysis reveals RNA splicing and chromatin remodeling as prominent processes for adult neurogenesis. *Mol Cell Neurosci*, 31:131–148.

[Liu *et al.*, 1998] Liu, B., Hsu, W. et Ma, Y. (1998). Integrating classification and association rule mining. *In KDD*, pages 80–86.

[Liu *et al.*, 2000] Liu, B., Ma, Y. et Wong, C. K. (2000). Improving an association rule based classifier. *In* Zighed, D. A., Komorowski, H. J. et Zytkow, J. M., éditeurs : *Principles of Data Mining and Knowledge Discovery, 4th European Conference, PKDD 2000, Lyon, France, September 13-16, 2000, Proceedings*, volume 1910 de *Lecture Notes in Computer Science*, pages 504–509. Springer.

[Lukong et Richard, 2003] Lukong, K. E. et Richard, S. (2003). Sam68, the KH domain-containing superSTAR. *Biochim Biophys Acta*, 1653:73–86.

[Mannila, 1997] Mannila, H. (1997). Inductive databases and condensed representations for data mining. *In International Logic Programming Symposium*, pages 21–30.

[Mannila et Toivonen, 1997] Mannila, H. et Toivonen, H. (1997). Levelwise search and borders of theories in knowledge discovery. *Data Min. Knowl. Discov.*, 1(3):241–258.

[Mannila *et al.*, 1995] Mannila, H., Toivonen, H. et Verkamo, A. I. (1995). Discovering frequent episodes in sequences. *In KDD*, pages 210–215.

[Martin *et al.*, 2004] Martin, D., Brun, C., Remy, E., Mouren, P., Thieffry, D. et Jacq, B. (2004). GOToolBox : functional investigation of gene datasets based on gene ontology. *Genome Biology*, 5(12):R101.

[Meo *et al.*, 1996] Meo, R., Psaila, G. et Ceri, S. (1996). A new SQL-like operator for mining association rules. *In* Vijayaraman, T. M., Buchmann, A. P., Mohan, C. et Sarda, N. L., éditeurs : *VLDB*, pages 122–133. Morgan Kaufmann.

[Mielikäinen, 2004] Mielikäinen, T. (2004). Separating structure from interestingness. *In* Dai, H., Srikant, R. et Zhang, C., éditeurs : *Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference, PAKDD 2004, Sydney, Australia, May 26-28, 2004, Proceedings*, volume 3056 de *Lecture Notes in Computer Science*, pages 476–485. Springer.

[Mitchell, 1982] Mitchell, T. M. (1982). Generalization as search. *Artif. Intell.*, 18(2):203–226.

[Morik *et al.*, 2005] Morik, K., Boulicaut, J.-F. et Siebes, A., éditeurs (2005). *Local Pattern Detection, International Seminar, Dagstuhl Castle, Germany, April 12-16, 2004, Revised Selected Papers*, volume 3539 de *Lecture Notes in Computer Science*. Springer.

[Morishita et Sese, 2000] Morishita, S. et Sese, J. (2000). Traversing itemset lattice with statistical metric pruning. *In PODS*, pages 226–236. ACM.

[Ng *et al.*, 1998] Ng, R. T., Lakshmanan, L. V. S., Han, J. et Pang, A. (1998). Exploratory mining and pruning optimizations of constrained association rules. *In* Haas, L. M. et Tiwary, A., éditeurs : *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA.*, pages 13–24. ACM Press.

[Ngan *et al.*, 2005] Ngan, S.-C., Lam, T., Wong, R. C.-W. et Fu, A. W.-C. (2005). Mining n-most interesting itemsets without support threshold by the COFI-tree. *Int. J. Business Intelligence and Data Mining*, 1(1):88–106.

[Oyanagi *et al.*, 2001] Oyanagi, S., Kubota, S. et Nakase, A. (2001). Application of matrix clustering to web log analysis and access prediction. *In proceedings of the WebKDD workshop (WebKDD'01) co-located with the 7th ACM SIGKDD International Conference on Knowledge Discovery in Databases (KDD'01)*, San Francisco, CA.

[Pasquier *et al.*, 1999] Pasquier, N., Bastide, Y., Taouil, R. et Lakhal, L. (1999). Discovering frequent closed itemsets for association rules. *Lecture Notes in Computer Science*.

[Pei et Han, 2000] Pei, J. et Han, J. (2000). Can we push more constraints into frequent pattern mining ? *In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 350–354. ACM Press.

[Pei et Han, 2002] Pei, J. et Han, J. (2002). Constrained frequent pattern mining : a pattern-growth view. *SIGKDD Explorations*, 4(1):31–39.

[Pei *et al.*, 2001a] Pei, J., Han, J. et Lakshmanan, L. V. S. (2001a). Mining frequent item sets with convertible constraints. *In ICDE*, pages 433–442.

[Pei *et al.*, 2004] Pei, J., Han, J. et Lakshmanan, L. V. S. (2004). Pushing convertible constraints in frequent itemset mining. *Data Min. Knowl. Discov.*, 8(3):227–252.

[Pei *et al.*, 2000] Pei, J., Han, J. et Mao, R. (2000). CLOSET : An efficient algorithm for mining frequent closed itemsets. *In ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 21–30.

[Pei *et al.*, 2001b] Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U. et Hsu, M. (2001b). Prefixspan : Mining sequential patterns by prefix-projected growth. *In ICDE*, pages 215–224. IEEE Computer Society.

[Pei *et al.*, 2002] Pei, J., Han, J. et Wang, W. (2002). Mining sequential patterns with constraints in large databases. *In CIKM*, pages 18–25. ACM.

[Pensa et Boulicaut, 2005] Pensa, R. G. et Boulicaut, J.-F. (2005). From local pattern mining to relevant bi-cluster characterization. *In* Famili, A. F., Kok, J. N., Peña, J. M., Siebes, A. et Feelders, A. J., éditeurs : *IDA*, volume 3646 de *Lecture Notes in Computer Science*, pages 293–304. Springer.

[Perng *et al.*, 2002] Perng, C.-S., Wang, H., Ma, S. et Hellerstein, J. L. (2002). Discovery in multi-attribute data with user-defined constraints. *SIGKDD Explorations*, 4(1):56–64.

[Piatetsky-Shapiro, 1991] Piatetsky-Shapiro, G. (1991). Discovery, analysis and presentation of strong rules. *In* Piatetsky-Shapiro, G. et Frawley, W., éditeurs : *Knowledge Discovery in Databases*, pages 229–248, Cambridge, MA. MIT Press.

[Quinlan, 1986] Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

[Raedt et Kramer, 2001] Raedt, L. D. et Kramer, S. (2001). The levelwise version space algorithm and its application to molecular fragment finding. *In* Nebel, B., éditeur : *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pages 853–862. Morgan Kaufmann.

[Ronkainen, 1998] Ronkainen, R. (1998). Attribute similarity and event sequence similarity in data mining.

[Salton et Buckley, 1988] Salton, G. et Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing Management*, 24(5):513–523.

[Sebag et Schoenauer, 1988] Sebag, M. et Schoenauer, M. (1988). Generation of rules with certainty and confidence factors from incomplete and incoherent learning bases. *In* Boose, J., Gaines, B. et Linster, M., éditeurs : *Proc. of the European Knowledge Acquisition Workshop (EKAW'88)*, pages 28–1 – 28–20. Gesellschaft für Mathematik und Datenverarbeitung mbH.

[Shav-Tal et Zipori, 2002] Shav-Tal, Y. et Zipori, D. (2002). PSF and p54(nrb)/NonO–multifunctional nuclear proteins. *FEBS Lett*, 531:109–114.

[Siebes *et al.*, 2006] Siebes, A., Vreeken, J. et van Leeuwen, M. (2006). Item sets that compress. *In SIAM conference on data mining*.

[Smyth et Goodman, 1991] Smyth, P. et Goodman, R. M. (1991). Rule induction using information theory. *In* Piatetsky-Shapiro, G. et Frawley, W., éditeurs : *Knowledge Discovery in Databases*, pages 159–176, Cambridge, MA. AAAI/MIT Press.

[Soulet et Crémilleux, 2005a] Soulet, A. et Crémilleux, B. (2005a). An efficient framework for mining flexible constraints. *In* Ho, T. B., Cheung, D. et Liu, H., éditeurs : *Advances in Knowledge Discovery and Data Mining, 9th Pacific-Asia Conference, PAKDD 2005, Hanoi, Vietnam, May 18-20, 2005, Proceedings*, volume 3518 de *Lecture Notes in Computer Science*, pages 661–671. Springer.

[Soulet et Crémilleux, 2005b] Soulet, A. et Crémilleux, B. (2005b). Exploiting virtual patterns for automatically pruning the search space. *In* Bonchi, F. et Boulicaut, J.-F., éditeurs : *Knowledge Discovery in Inductive Databases, 4th International Workshop, KDID 2005, Porto, Portugal, October 3, 2005, Revised Selected and Invited Papers*, volume 3933 de *Lecture Notes in Computer Science*, pages 202–221. Springer.

[Soulet et Crémilleux, 2005c] Soulet, A. et Crémilleux, B. (2005c). Optimizing constraint-based mining by automatically relaxing constraints. *In Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), 27-30 November 2005, Houston, Texas, USA*, pages 777–780. IEEE Computer Society.

[Soulet *et al.*, 2004a] Soulet, A., Crémilleux, B. et Rioult, F. (2004a). Condensed representation of emerging patterns. *In 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Lecture Notes in Computer Science, pages 127–132, Sydney.

[Soulet *et al.*, 2004b] Soulet, A., Crémilleux, B. et Rioult, F. (2004b). Condensed representation of EPs and patterns quantified by frequency-based measures. *In* Goethals, B. et Siebes, A., éditeurs : *KDID 2004, Knowledge Discovery in Inductive Databases, Proceedings of the Third International Workshop on Knowledge Discovery in Inductive Databases, Pisa, Italy, September 20, 2004, Revised Selected and Invited Papers*, volume 3377 de *Lecture Notes in Computer Science*, pages 173–190. Springer.

[Soulet *et al.*, 2004c] Soulet, A., Crémilleux, B. et Rioult, F. (2004c). Représentation condensée de motifs émergents. *In 4èmes journées d'Extraction et de Gestion des Connaissances*, Revue des Nouvelles Technologies de l'Information, pages 265–276, Clermont-Ferrand, France. Cepaduès Editions.

[Soulet et Hébert, 2004] Soulet, A. et Hébert, C. (2004). Using emerging patterns from clusters to characterize social subgroups of patients affected by atherosclerosis. *In proceedings of the workshop Discovery Challenge, ECML-PKDD'04*.

[Soulet *et al.*, 2006] Soulet, A., Klema, J. et Crémilleux, B. (2006). Efficient Mining under Flexible Constraints through Several Datasets. *In 5th International Workshop on Knowledge Discovery in Inductive Databases (KDID'06) co-located with the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases PKDD'06* , pages 131–142, Berlin, Germany.

[Srikant et Agrawal, 1995] Srikant, R. et Agrawal, R. (1995). Mining generalized association rules. *In* Dayal, U., Gray, P. M. D. et Nishio, S., éditeurs : *VLDB*, pages 407–419. Morgan Kaufmann.

[Srikant *et al.*, 1997] Srikant, R., Vu, Q. et Agrawal, R. (1997). Mining association rules with item constraints. *In KDD*, pages 67–73.

[Stadler et Stadler, 2002] Stadler, B. M. R. et Stadler, P. F. (2002). Basic properties of filter convergence spaces.

[Tan *et al.*, 2002] Tan, P., Kumar, V. et Srivastava, J. (2002). Selecting the right interestingness measure for association patterns. *In In proceedings The Eighth ACM Special Interest Group on Knowledge Discovery in Data and Data Mining (SIGKDD'02)*, Edmonton, Alberta, Canada.

[Termier *et al.*, 2004] Termier, A., Rousset, M.-C. et Sebag, M. (2004). Dryade : A new approach for discovering closed frequent trees in heterogeneous tree databases. *In ICDM*, pages 543–546. IEEE Computer Society.

[Tzvetkov *et al.*, 2003] Tzvetkov, P., Yan, X. et Han, J. (2003). TSP : Mining top-k closed sequential patterns. *In Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), 19-22 December 2003, Melbourne, Florida, USA*, pages 347–354. IEEE Computer Society.

[Velculescu *et al.*, 1999] Velculescu, V., Madden, S., Zhang, L. et et al. (1999). Analysis of human transcriptomes. *Nat. Genet.*, 23:387–8.

[Velculescu *et al.*, 1995] Velculescu, V., Zhang, L., Vogelstein, B. et Kinzler, K. (1995). Serial analysis of gene expression. *Science*, 270:484–7.

[Wang et Han, 2004] Wang, J. et Han, J. (2004). BIDE : Efficient mining of frequent closed sequences. *In ICDE*, pages 79–90. IEEE Computer Society.

[Wang et Karypis, 2005] Wang, J. et Karypis, G. (2005). Harmony : Efficiently mining the best rules for classification. *In SDM*.

[Wang *et al.*, 2003] Wang, K., Jiang, Y. et Lakshmanan, L. V. S. (2003). Mining unexpected rules by pushing user dynamics. *In* Getoor, L., Senator, T. E., Domingos, P. et Faloutsos, C., éditeurs : *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*, pages 246–255. ACM.

[Wang *et al.*, 2005] Wang, K., Jiang, Y., Yu, J. X., Dong, G. et Han, J. (2005). Divide-and-approximate : A novel constraint push strategy for iceberg cube mining. *IEEE Trans. Knowl. Data Eng.*, 17(3):354–368.

[Yan *et al.*, 2003] Yan, X., Han, J. et Afshar, R. (2003). CloSpan : Mining closed sequential patterns in large databases. *In* Barbará, D. et Kamath, C., éditeurs : *Proceedings of the Third SIAM International Conference on Data Mining, San Francisco, CA, USA, May 1-3, 2003*. SIAM.

[Zaki, 2000a] Zaki, M. (2000a). Generating non-redundant association rules. *In ACM SIGKDD'00*, pages 34–43.

[Zaki et Hsiao, 1999] Zaki, M. et Hsiao, C. (1999). CHARM : an efficient algorithm for closed association rule mining.

[Zaki, 1999] Zaki, M. J. (1999). Parallel and distributed association mining : A survey. *IEEE Concurrency*, 7(4):14–25.

[Zaki, 2000b] Zaki, M. J. (2000b). Scalable algorithms for association mining. *IEEE Trans. Knowl. Data Eng.*, 12(2):372–390.

[Zaki, 2002] Zaki, M. J. (2002). Efficiently mining frequent trees in a forest. *In KDD*, pages 71–80. ACM.

[Zaki et Aggarwal, 2003] Zaki, M. J. et Aggarwal, C. C. (2003). Xrules : an effective structural classifier for xml data. *In* Getoor, L., Senator, T. E., Domingos, P. et Faloutsos, C., éditeurs : *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*, pages 316–325. ACM.

[Zaki *et al.*, 2005] Zaki, M. J., Parimi, N., De, N., Gao, F., Phoophakdee, B., Urban, J., Chaoji, V., Hasan, M. A. et Salem, S. (2005). Towards generic pattern mining. *In* Ganter, B. et Godin, R., éditeurs : *Formal Concept Analysis, Third International Conference, ICFCA 2005, Lens, France, February 14-18, 2005, Proceedings*, volume 3403 de *Lecture Notes in Computer Science*, pages 1–20. Springer.

[Zalfa et Bagni, 2004] Zalfa, F. et Bagni, C. (2004). Molecular insights into mental retardation : multiple functions for the fragile X mental retardation protein ? *Curr Issues Mol Biol*, 6:73–88.

[Zelezny *et al.*, 2005] Zelezny, F., Tolar, J., Lavrac, N. et Stepankova, O. (2005). Relational subgroup discovery for gene expression data mining. *In EMBEC : 3rd IFMBE European Medical & Biological Engineering Conf.*

[Zhang *et al.*, 2000] Zhang, X., Dong, G. et Ramamohanarao, K. (2000). Information-based classification by aggregating emerging patterns. *In* Leung, K.-S., Chan, L.-W. et Meng, H., éditeurs : *Intelligent Data Engineering and Automated Learning - IDEAL 2000, Data Mining, Financial Engineering, and Intelligent Agents, Second International Conference, Shatin, N.T. Hong Kong, China, December 13-15, 2000, Proceedings*, volume 1983 de *Lecture Notes in Computer Science*, pages 48–53. Springer.

Résumé La découverte de motifs est une tâche centrale pour l'extraction de connaissances dans les bases de données. Cette thèse traite de l'extraction de motifs locaux sous contraintes. Nous apportons un éclairage nouveau avec un cadre combinant des primitives monotones pour définir des contraintes quelconques. La variété de ces contraintes exprime avec précision l'archétype des motifs recherchés par l'utilisateur au sein d'une base de données. Nous proposons alors deux types d'approche d'extraction automatique et générique malgré les difficultés algorithmiques inhérentes à cette tâche. Leurs efficacités reposent principalement sur l'usage de conditions nécessaires pour approximer les variations de la contrainte. D'une part, des méthodes de relaxations permettent de ré-utiliser les nombreux algorithmes usuels du domaines. D'autre part, nous réalisons des méthodes d'extraction directes dédiées aux motifs ensemblistes pour les données larges ou corrélées en exploitant des classes d'équivalences. Enfin, l'utilisation de nos méthodes ont permi la découverte de phénomènes locaux lors d'applications industrielles et médicales.

Mots clés : Fouille de données, bases de données, motifs locaux, contraintes.

Title

A generic framework for discovering patterns under primitive-based constraints.

Abstract

Pattern mining is a significant field of Knowledge Discovery in Databases. This thesis deals with the mining problem of local patterns under constraints. We propose a new framework relying on monotone primitives in order to define and mine varied constraints. This broad spectrum of constraints enables users to accurately focus on the most interesting patterns. We provide two main approaches for automatically mining patterns which solve the intrinsic algorithmic difficulty of this task. Their efficiency mainly relies on necessary conditions approximating the variation of contraints. Firstly, relaxing methods enable us to re-use numerous usual algorithms. Secondly, we design pattern mining algorithms dedicated to wide or correlated datasets by exploiting the concept of equivalence classes. Finally, the use of these methods highlights several relevant local phenomena in real industrial and medical applications.

Keywords : Data mining, databases, local patterns, constraints.

Discipline : Informatique

Laboratoire : Groupe de Recherche en Informatique, Image, Automatique et Instrumentation de Caen (UMR 6072), Université de Caen Basse-Normandie, France.