Découverte de motifs n-aires utilisant la programmation par contraintes

Medhi Khiari

université de Caen
Basse-Normandie

UFR Sciences
Département d'Informatique

École doctorale : SIMEM

THÈSE

présentée par

Mehdi KHIARI

et soutenue
le 19 juin 2012

en vue de l'obtention du

Doctorat de l'UNIVERSITÉ de CAEN

spécialité : Informatique et applications
(Arrêté du 7 août 2006)

Découverte de motifs n-aires utilisant la programmation par contraintes

MEMBRES du JURY

Rapporteurs :	Jean-Marc PETIT	Professeur des Universités, INSA de Lyon
	Christel VRAIN	Professeur des Universités, Université d'Orléans
Examinateurs :	Arnaud LALLOUET	Professeur des Universités, Université de Caen
	Olivier RIDOUX	Professeur des Universités, Université de Rennes 1
	Michel RUEHER	Professeur des Universités, Université Nice Sophia Antipolis
	Arnaud SOULET	Maître de conférences, Université François Rabelais Tours
Directeurs :	Patrice BOIZUMAULT	Professeur des Universités, Université de Caen
	Bruno CRÉMILLEUX	Professeur des Universités, Université de Caen

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

GREYC

Remerciements

Je tiens en tout premier lieu à remercier Patrice Boizumault et Bruno Crémilleux, mes deux directeurs de thèse, pour leurs nombreux conseils et leur soutien constant. Un grand merci à eux d'avoir rendu cette expérience si enrichissante aussi bien sur le plan scientifique que sur le plan humain.

J'adresse aussi tous mes remerciements à Christel Vrain et Jean-Marc Petit pour avoir accepté d'être rapporteurs de cette thèse et pour le temps précieux qu'ils ont consacré à cette tâche. Je remercie également Michel Rueher, Arnaud Soulet, Olivier Ridoux et Arnaud Lallouet pour avoir accepté de faire partie de mon jury de thèse

Je remercie également tous mes collègues du laboratoire GREYC. Je pense en particulier à Jean-Philippe, Samir et Xavier pour les discussions souvent fructueuses que nous avons eues. Merci à Mariya et Xavier pour les bons moments passés à coté de la machine à café, et avec qui s'est installée une amitié dépassant le cadre du laboratoire. Merci à tous ceux qui m'ont soutenu de près ou de loin, je pense notamment à Lamia, Guillaume, Peggy, Bertrand et François. Je n'oublierais pas de remercier enfin nos Sysadmins ainsi que tout le personnel administratif du GREYC et de la SIMEM pour leur disponibilité.

Merci à tous mes amis de Caen pour m'avoir permis de ne pas oublier le monde extérieur pendant cette thèse.

Merci à ma famille de m'avoir soutenu le long de cette aventure. Ce fut un moment fort émouvant de voir les larmes de joie dans les yeux de mes parents après ma soutenance.

Enfin, je remercie affectueusement Vanya qui a su me supporter chaque jour, m'encourager et me procurer la force et le réconfort nécessaires pour mener à terme ces travaux.

Table des matières

Table des figures

Liste des tableaux

Première partie

Introduction

Introduction

Contexte général

La fouille de données et la Programmation Par Contraintes (PPC) sont deux domaines de l'informatique qui ont eu, jusqu'à très récemment, des destins séparés. Néanmoins, les travaux menés en fouille de données, notamment dans le domaine de l'extraction de motifs sous contraintes, présentent une assez forte analogie avec la PPC. En effet, les motifs recherchés sont modélisés à partir de contraintes décrivant les propriétés qu'ils doivent vérifier. De plus, l'extraction des motifs recherchés se fait (très souvent) à l'aide d'une recherche arborescente où des méthodes d'élagage (filtrage) permettent d'en réduire la taille. Que ce soit en fouille de données ou en PPC, les contraintes sont actives : outre l'aspect modélisation, elles permettent aussi d'élaguer l'espace de recherche améliorant ainsi la résolution.

Cette thèse est l'une des toutes premières à s'intéresser aux liens entre la fouille de données et la PPC, et notamment aux apports de cette dernière à l'extraction de motifs sous contraintes.

Dans ce chapitre introductif, nous présentons le contexte général de la fouille de données en nous focalisant sur le cadre de l'extraction de motifs sous contraintes pour lequel nous faisons un bilan des principales difficultés rencontrées. Puis, nous présentons le contexte général de la PPC en décrivant plus précisément les problèmes de satisfaction de contraintes. Nous évaluons alors leur aptitude à surmonter les difficultés et résoudre les problèmes rencontrés. Ensuite, nous présentons par ordre chronologique les différentes contributions dans ce domaine de recherche avant d'énoncer le plan de la thèse.

Ce travail s'inscrit dans le cadre du projet ANR Bingo2 [1] associant le CGMC (CNRS UMR 5534, Lyon), le GREYC (CNRS UMR 6072, Caen), le LaHC (CNRS UMR 5516, Saint-Etienne) et le LIRIS (CNRS UMR 5205, Lyon).

Fouille de données

Au cours des dernières décennies, on assiste à une explosion permanente des volumes des données provenant de diverses sources (qu'elles soient commerciales ou scientifiques) et le défi aujourd'hui n'est plus de stocker ces données mais d'en extraire de l'information. L'Extraction de Connaissances dans les Bases de Données (ECBD ou KDD [2]) ou «fouille de données» (data-mining en anglais) est un domaine de l'informatique se situant à l'intersection de l'intelligence artificielle (notamment de l'apprentissage automatique), des statistiques et des bases de données.

1. Le projet Bingo2 `https://bingo2.greyc.fr` soutenu par l'ANR a pour but la conception de requêtes inductives pour la découverte de la connaissance dans les données génomiques et intégrant de la connaissance (partielle) du domaine.
2. Knowledge Discovery in Databases

Les outils d'analyse statistique permettent d'apporter une réponse à un problème précis, à l'image des réponses issues de sondages. *La fouille de données*, quant à elle, *vise à extraire des connaissances pertinentes, non triviales, trouver des corrélations et des tendances cachées à partir d'une grande quantité de données.* Par exemple, les transactions commerciales d'un super marché renferment des informations sur la corrélation entre achats et peuvent permettre, dans certains cas, d'identifier différents profils de clients et ainsi d'améliorer les services à destination de la clientèle.

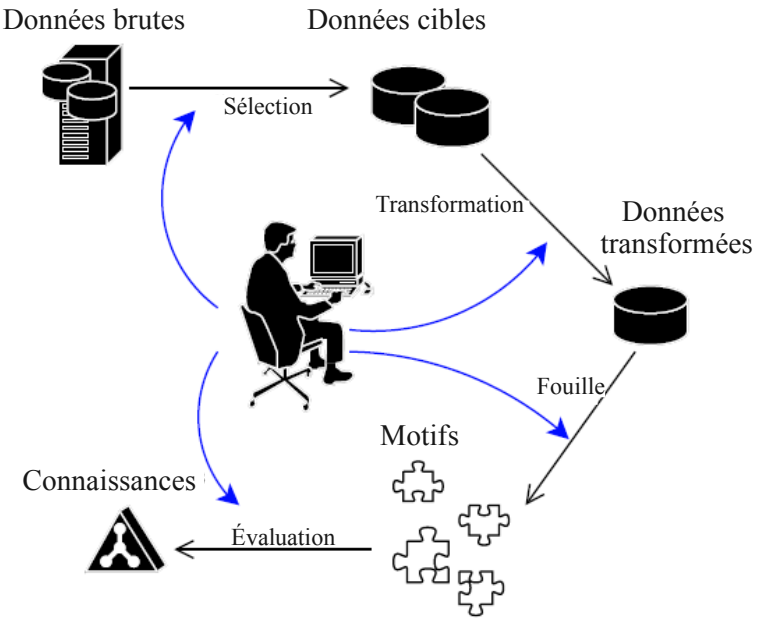

FIGURE 1 – Processus d'extraction de connaissances

Clairement, les données stockées contiennent plus d'informations que celles que l'on pourrait obtenir par des requêtes classiques du niveau base de données. Le processus de l'ECBD se situe dans ce contexte. Il permet à travers différentes étapes [Fayyad *et al.*, 1996] (cf. figure 1) de passer d'un grand volume de données à une ensemble de connaissances utilisables et utiles similairement à la transformation de la matière première vers un produit fini. *Ce travail de thèse contribue particulièrement à la phase d'extraction de motifs, un point clé dans ce processus.*

Les motifs extraits peuvent être de natures diverses. Par exemple la recherche de répétitions d'épisodes dans une séquence est utilisée pour la détection d'évènements [Mannila *et al.*, 1997], l'extraction de fragments moléculaires [Kramer et De Raedt, 2001], l'extraction de connaissances à partir de données textuelles [Azé, 2003] ou plus récemment l'extraction d'arbres ou de graphes fréquents [Termier, 2004, Kuramochi et Karypis, 2004].

Extraction de motifs sous contraintes

L'extraction de connaissances produit généralement un grand nombre d'informations qu'il n'est pas possible d'examiner manuellement et dans lequel l'information utile est très souvent noyée. *D'un point de vue modélisation*, l'extraction sous contraintes permet à l'utilisateur de focaliser la recherche de l'information à extraire suivant ses centres d'intérêt. *D'un point de vue résolution*, l'élagage (associé à ces contraintes) de l'espace de recherche permet d'améliorer les performances de l'extracteur de motifs et dans certains cas de rendre le processus faisable.

L'extraction de motifs sous contraintes s'est essentiellement développée dans le cas des *motifs locaux* (phénomènes valides sur certaines parties des données), ces derniers capturant les phénomènes valides sur une portion de la base de données [Hand, 2002, Morik *et al.*, 2005] (cf. section 1.1.3, page 15). Parmi les contraintes les plus utilisées, citons les contraintes de seuil sur des mesures telles que la fréquence (nombre d'occurrences d'un motif dans la base de données) ou encore la cardinalité (nombre d'attributs composant un motif).

De nombreux extracteurs de motifs locaux ont été développés : MUSIC-DFS [Soulet et Crémilleux, 2005], CONQUEST [Bonchi *et al.*, 2009], MAFIA [Burdick *et al.*, 2003a], MV-MINER [Rioult, 2005], etc. *La grande majorité de ces extracteurs mettent en oeuvre une méthode de recherche où l'élagage de l'espace de recherche s'effectue en utilisant les propriétés des contraintes, notamment la monotonie ou l'anti-monotonie* [Mannila et Toivonen, 1997].

Mais, la plupart de ces extracteurs ont du mal à traiter les contraintes ne présentant pas de bonnes propriétés d'élagage. C'est le cas des contraintes de seuil sur la mesure d'aire[3] [Geerts *et al.*, 2004], ou encore des requêtes combinant contraintes monotones et anti-monotones. [Soulet, 2006, De Raedt *et al.*, 2008] ont proposé des approches génériques permettant de combiner une large panoplie de contraintes locales (associées respectivement aux extracteurs de motifs locaux MUSIC-DFS et FIM_CP).

D'autre part, lors de l'extraction de motifs locaux, aucune attention n'est accordée aux interactions avec les autres motifs de la base. Or, *dans de nombreux cas, l'intérêt d'un motif dépend d'autres motifs extraits.* Citons comme exemples : les règles inattendues [Padmanabhan et Tuzhilin, 1998], les règles d'exception [Suzuki, 2002], la construction de modèles comme les classifieurs [Liu *et al.*, 1998, Li *et al.*, 2001, Lan *et al.*, 2005], ou bien encore le clustering [Berkhin, 2006]. *Contribuer à la découverte de motifs de plus haut niveau, fondés sur la combinaison de motifs locaux, permet ainsi la découverte de motifs plus riches, plus pertinents et plus proches des buts finaux de l'utilisateur.* Nous proposons dans ce manuscrit d'appeler *contraintes n-aires* les contraintes portant sur plusieurs motifs interdépendants, alors que les motifs locaux sont définis par des *contraintes unaires*.

Peu de travaux concernant l'extraction de tels motifs ont été menés et les méthodes développées sont toutes ad hoc. La difficulté de la tâche explique l'absence de méthodes génériques : en effet, l'extraction de motifs locaux nécessite déjà l'exploration d'un espace de recherche de très grande taille, même lorsqu'il s'agit de motifs simples tels que les motifs ensemblistes. Or, cet espace est encore beaucoup plus grand pour l'extraction de motifs sous contraintes n-aires car le passage de un à plusieurs motifs augmente fortement la combinatoire. Certains problèmes particuliers portant sur des contraintes n-aires ont été traités [Suzuki, 2002, Lakshmanan *et al.*, 1999] à l'aide d'algorithmes dédiés, mais

3. L'aire d'un motif X est définie par : $aire(X) = \texttt{freq}(X) \times \texttt{size}(X)$.

aucune méthode générique n'est proposée. Ce manque de généricité est un frein à la découverte de motifs pertinents et intéressants car chaque contrainte n-aire entraîne la conception et le développement d'une méthode ad hoc.

Ainsi, l'extraction de motifs sous contraintes est limitée par les constats suivants :

- L'intégration d'une contrainte dans un extracteur de motifs locaux est liée au fait que l'on sache ou non lui associer un élagage performant.
- La difficulté de traiter des requêtes combinant des contraintes de natures différentes.
- L'absence de méthode générique pour l'extraction de motifs sous contraintes portant sur plusieurs motifs à la fois.

Par conséquent, l'utilisateur doit se contenter d'un jeu de contraintes assez réduit ne permettant pas une extraction efficace de l'information que ce soit en terme de qualité ou de quantité. En effet, ce manque d'expressivité est, entre autre, à l'origine d'un problème majeur dans les problèmes d'extraction de motifs locaux, à savoir le nombre conséquent de motifs produits, chose qui rend leur utilisation très difficile.

Programmation par contraintes

> *Constraint programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming : the user states the problem, the computer solves it.*
>
> *Eugene C. Freuder*

La PPC repose sur la déclarativité : l'utilisateur modélise son problème (le «Quoi») sans se soucier de sa résolution (le «Comment»). Cette résolution est généralement effectuée par un solveur de contraintes mettant en oeuvre des méthodes de recherche où une propriété de cohérence (faible) est maintenue afin d'élaguer l'espace de recherche.

Les Problèmes de Satisfaction de Contraintes (CSP) offrent un cadre générique pour modéliser et résoudre de nombreux problèmes combinatoires. Formellement, un CSP $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ est défini par :

- $\mathcal{X} = \{X_1, ..., X_n\}$ est un ensemble de variables,
- $\mathcal{D} = \{D_1, ..., D_n\}$ est un ensemble de domaines (chaque D_i est l'ensemble fini de valeurs que peut prendre la variable X_i),
- $\mathcal{C}$ est un ensemble de contraintes limitant les combinaisons de valeurs que peuvent prendre simultanément les variables.

Résoudre un CSP consiste à trouver une affectation pour chaque variable telle que toutes les contraintes soient satisfaites.

Le formalisme des CSP est utilisé dans de nombreux domaines tels que :

- Les problèmes de planification et ordonnancement : planifier une production, gérer un trafic ferroviaire, ...
- Les problèmes d'affectation de ressources : établir un emploi du temps, allouer de l'espace mémoire, du temps CPU par un système d'exploitation, affecter du personnel à des tâches, des entrepôts à des marchandises, ...
- Les problèmes d'optimisation : optimiser des placements financiers, des routages de réseaux de télécommunication, ...

Objectifs et motivations

Le principal objectif de cette thèse est de proposer une approche d'extraction de motifs s'affranchissant des propriétés des contraintes (relatives au filtrage) ainsi que du nombre de motifs impliqués.

D'emblée la PPC offre un cadre approprié pour répondre à cette problématique. En effet, comme nous l'avons déjà fait remarquer, les motifs recherchés sont modélisés à partir de contraintes décrivant les propriétés qu'ils doivent vérifier (le «Quoi»). De plus, l'extraction des motifs recherchés se fait à l'aide d'une recherche arborescente où des méthodes d'élagage (filtrage) permettent d'en réduire la taille (le «Comment»).

On peut alors modéliser, sous forme d'un CSP (certes très général), l'extraction de motifs dans le cas n-aire. Soit $\mathcal{I}$ l'ensemble des items. Alors, $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ avec :

- $\mathcal{X} = \{X_1, ..., X_n\}$ où chaque X_i désigne un motif inconnu,
- $\mathcal{D} = \{D_1, ..., D_n\}$ où chaque domaine fini $D_i = [\{\} \,..\, \mathcal{I}]$,
- $\mathcal{C}$ est un ensemble de contraintes portant sur les X_i.

Les contraintes peuvent être de natures très différentes. Elles peuvent être :

- ensemblistes ($X_1 \subseteq X_2$, $i_{12} \in X_4$, ...),
- numériques portant sur des mesures ($|\texttt{freq}(X_1) - \texttt{freq}(X_2)| \leq \alpha_1$, $\texttt{size}(X_4) < \texttt{size}(X_1) + 1$, ...),
- dédiées au type de problème à traiter,
- ...

Mais, la transposition d'un problème d'extraction de motifs sous forme d'un CSP est loin d'être immédiate : en effet, en fouille de données filtrage et heuristiques de parcours de l'arbre de recherche sont «fondus» ensemble. Il est donc difficile, voire impossible, d'ajouter d'autres types de contraintes sans avoir à tout réécrire (de même si l'on voulait modifier les stratégies de recherche). Cela explique pourquoi la plupart des travaux en fouille de données définissent des contraintes à partir de propriétés permettant un élagage de l'espace de recherche, l'exemple le plus typique est encore celui des contraintes (anti)-monotones.

Etat de l'art et contributions

Dans cette section, nous présentons les contributions de cette thèse et nous les situons par rapport aux autres travaux du domaine.

Le premier article traitant de la problématique "extraction de motifs et PPC" a été publié en 2008 par de L. de Raedt, T. Guns et S. Nijssen [De Raedt *et al.*, 2008]. Les auteurs proposent une approche permettant de traiter, dans un cadre unifié, un large ensemble de motifs locaux et de contraintes telles que la fermeture, la maximalité, les contraintes monotones ou anti-monotones et des variations de ces contraintes. Cette approche, qui marque un cadre fondateur, est malgré tout limitée aux motifs locaux. Dans un second temps, elle a été étendue et appliquée à l'extraction de motifs discriminants [Nijssen *et al.*, 2009].

Nous avons proposé dans [Khiari *et al.*, 2009, Khiari *et al.*, 2010a, Khiari *et al.*, 2010b, Khiari *et al.*, 2012] une approche générique pour l'extraction de motifs sous contraintes n-aires. Cette approche, appelée *Hybride*, repose sur l'utilisation conjointe d'un extracteur de motifs locaux et d'un solveur de CSP (cf. chapitre 5). A notre connaissance, il s'agit de la première approche générique pour l'extraction de motifs sous contraintes n-aires.

Dans [Khiari *et al.*, 2010c, Khiari *et al.*, 2010, Khiari *et al.*, 2011], nous avons proposé une seconde approche générique pour l'extraction de motifs sous contraintes n-aires (cf. chapitre 6). Cette approche (appelée *Pure-CP*) étend la modélisation proposée dans [De Raedt *et al.*, 2008] et, à la différence de l'approche *Hybride*, repose uniquement sur l'utilisation d'un solveur de CSP. Outre l'amélioration des performances, *Pure-CP* permet aussi de modéliser et résoudre un ensemble plus large de contraintes n-aires que celui traité par *Hybride*.

[Nijssen et Guns, 2010] propose une comparaison entre les performances des solveurs des CSP génériques et les extracteurs de motifs classiques.Il propose aussi un solveur dédié à l'extraction de motifs dont les méthodes de filtrage et de propagation sont inspirés d'algorithmes issus de la communauté PPC notamment AC-5 [Hentenryck *et al.*, 1992].

Notons que [Guns *et al.*, 2011a, Guns *et al.*, 2011b] présentent une méthode pour modéliser et résoudre des problèmes d'extraction portant sur k motifs ce qui est similaire aux contraintes n-aires. Les même auteurs présentent dans [Guns *et al.*, 2011c] un récapitulatif et une évaluation des performances des approches CSP pour la résolution de problèmes de fouille de données.

Dans [Métivier *et al.*, 2011, Métivier *et al.*, 2012], nous proposons un langage de contraintes permettant d'écrire des requêtes n-aires de manière déclarative. Ce langage permet d'écrire aussi bien des requêtes complètement définies, que des requêtes raffinées par ajouts successifs de contraintes au fur et à mesure qu'on veut raffiner les propriétés des motifs ciblés. Cette dernière démarche est particulièrement utile lorsque les spécifications de l'utilisateur sont plus ou moins incomplètes, ou lorsqu'elles sont à préciser au cours du processus d'extraction. L'ensemble des contraintes primitives de ce langage a été intégré dans l'approche *Pure-CP*.

Organisation du mémoire

La deuxième partie de ce mémoire est consacrée à un état de l'art des domaines centraux de notre étude que sont l'extraction de motifs sous contraintes et les CSP.

Dans le chapitre 1, nous présentons un état de l'art de l'extraction de motifs locaux sous contraintes en nous attardant sur le problème du «pattern flooding» i.e. la difficulté de retrouver l'information utile lorsqu'elle est noyée dans un très grand volume d'informations redondantes et souvent triviales. Nous présentons alors dans le chapitre 2 un récapitulatif d'approches cherchant à extraire des motifs de plus haut niveau. Nous y définissons aussi les contraintes n-aires dont l'extraction est un des objectifs majeurs de cette thèse.

Dans le chapitre 3, nous présentons de manière synthétique les méthodes de résolution et de filtrage pour les CSP. Puis, nous décrivons plus en détail les CSP ensemblistes et les CSP quantifiés. En effet, ces 2 familles de CSP sont utilisées pour décrire nos contributions aux chapitres 5 et 7 respectivement.

La troisième partie décrit nos principales contributions dans l'apport de la PPC pour l'extraction de motifs sous contraintes.

Le chapitre 4 décrit le langage de requêtes à base de contraintes que nous proposons pour modéliser les différents problèmes traités. Ce langage est un langage de haut niveau grâce auquel l'utilisateur pourra modéliser ses requêtes n-aires, les modifier et les compléter dans une démarche par raffinements successifs jusqu'à obtenir le résultat souhaité. L'intérêt et les apports d'un tel langage sont illustrés par un large ensemble d'exemples : les règles d'exception, les règles inattendues, la construction de modèles et enfin différentes formes de clustering.

Le chapitre 5 présente notre première approche générique pour l'extraction de motifs sous contraintes n-aires. Cette approche, appelée *Hybride*, repose sur l'utilisation conjointe d'un extracteur de motifs locaux et d'un solveur de CSP. Un premier prototype utilisant l'extracteur de motifs locaux MUSIC-DFS et les CSP ensemblistes d'ECL^iPS^e a pu être rapidement développé afin de tester et valider notre démarche ainsi que nos premières propositions.

Le chapitre 6 présente notre seconde approche générique pour l'extraction de motifs sous contraintes n-aires. Cette approche (appelée *Pure-CP*) étend la modélisation proposée dans [De Raedt *et al.*, 2008] et, à la différence de l'approche *Hybride*, repose uniquement sur l'utilisation d'un solveur de CSP, en l'occurrence *Gecode*. Outre l'amélioration des performances, *Pure-CP* permet aussi de modéliser et résoudre un ensemble plus large de contraintes n-aires que celui traité par *Hybride*. Nous présentons la mise en œuvre des différentes primitives de notre langage de contraintes. Différentes expérimentations menées sur plusieurs exemples présentés au chapitre 4 montrent l'intérêt et la faisabilité de cette approche.

Mais, le formalisme des CSP n'est pas suffisant pour pouvoir modéliser l'ensemble des contraintes n-aires. C'est pourquoi nous présentons, au chapitre 7, à l'aide des CSP quantifiés (QCSP) la modélisation de deux exemples de contraintes nécessitant l'utilisation de la quantification universelle : la contrainte de *peak* et les synexpressions. Nous décrivons alors la mise en œuvre effectuée en *QeCode* ainsi que les premiers résultats expérimentaux obtenus.

Enfin, et en conclusion de ce mémoire, nous dressons un bilan des travaux menés au cours de cette thèse. Puis, nous ouvrons et discutons différentes perspectives.

Deuxième partie

Etat de l'art

Chapitre 1

Extraction de motifs locaux sous contraintes

Sommaire

Ce chapitre introduit la problématique de l'extraction de motifs locaux sous contraintes. La section 1.1 présente le cadre formel de l'extraction de motifs sous contraintes et nous introduisons les notions essentielles de fouille de données que nous utiliserons tout au long de ces travaux de thèse. La section 1.2 présente la structuration de l'espace de recherche pour le cas des motifs ensemblistes et les principales techniques d'élagage dans cet espace. Nous présentons ensuite les représentation condensées des motifs relativement à des contraintes. Enfin, nous présentons dans la section 1.4 deux exemples d'extracteurs génériques de motifs locaux.

1.1 Cadre formel - Définitions

Nous présentons dans cette section les définitions relatives au cadre formel introduit par Mannila et Toivonen dans [Mannila et Toivonen, 1997] pour la découverte de motifs sous contraintes dans une base de données.

1.1.1 Cadre formel

Soit une base de données $\boldsymbol{r}$ composée d'un ensemble de transactions $\mathcal{T}$ décrites par un ensemble $\mathcal{I}$ de littéraux distincts appelés items, un *motif ensembliste d'items* (aussi appelé *itemset*) correspond à un sous-ensemble non vide de $\mathcal{I}$. Ces motifs forment le langage $\mathcal{L}_{\mathcal{I}} = 2^{\mathcal{I}} \backslash \emptyset$.

Définition 1.1.1 (contexte transactionnel)**.**
Une base de données $\boldsymbol{r}$ est décrite par un triplet $(\mathcal{I}, \mathcal{R}, \mathcal{T})$ où $\mathcal{I} = \{i_1, ..., i_n\}$est l'ensemble des attributs (items), $\mathcal{T} = \{t_1, ..., t_m\}$ celui des objets (transactions) et $\mathcal{R}$ une relation binaire entre $\mathcal{I}$ et $\mathcal{T}$. $\mathcal{R}$ indique quels attributs i sont recensés dans les objets t et on notera de façon équivalente $\mathcal{R}(i,t) = present$ (noté par 1) pour $i\mathcal{R}t$ et $\mathcal{R}(i,t) = absent$ (noté par 0) pour $\neg i\mathcal{R}t$.

Un contexte transactionnel peut être vu comme une matrice binaire (cf. table 1.1). Le tableau 1.1 représente un contexte transactionnel $\boldsymbol{r}$ où 5 transactions étiquetées $t_1, \ldots, t_5$ sont décrites par 5 items $A, B, \ldots, E$.

$\mathcal{T} \setminus \mathcal{I}$	A	B	C	D	E
t_1	1	1	1	0	0
t_2	0	1	1	0	0
t_3	0	1	1	0	1
t_4	0	1	1	1	1
t_5	1	1	1	1	1

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

TABLE 1.1 – Deux représentations d'une base de données transactionnelle $\boldsymbol{r}$.

Définition 1.1.2 (Motif)**.**
On distingue deux types de motifs :

Un motif ensembliste d'items ou itemset *est un sous ensemble de $\mathcal{I}$*

Un motif ensembliste de transactions *est un sous ensemble de $\mathcal{T}$*

Un motif constitué de k items est appelé un k-motif.

Remarque 1.1.1.

- *Dans ce manuscrit, nous nous intéressons uniquement aux motifs ensemblistes. De plus, nous nous intéressons principalement aux motifs ensemblistes d'items. Ainsi, lorsqu'aucune précision n'est apportée, un motif est un motif ensembliste d'items.*

- *Dans la suite, les items seront décrits par des lettres majuscules du début de l'alphabet (A,B,C,...).*
- *Pour alléger les écritures, les motifs seront notés sous forme de chaînes plutôt que sous forme d'ensembles (i.e AB au lieu de $\{A, B\}$).*

Définition 1.1.3 (Langage).
L'espace de recherche des motifs est défini comme le langage $\mathcal{L}_\mathcal{I}$ construit sur l'alphabet $\mathcal{I} = \{i_1, i_2, ..., i_n\}$ constitué de tous les items de $\mathcal{I}$. C'est aussi l'ensemble des parties de $\mathcal{I}$.

1.1.2 Extraction sous contraintes

Une contrainte évalue si un motif est intéressant ou non, selon des critères fixés par l'utilisateur. Elle permet à la fois de cibler les motifs recherchés par l'utilisateur et de réduire l'ensemble des motifs extraits dans la perspective d'une interprétation plus facile. De plus, nous verrons que les contraintes permettent d'élaguer considérablement l'espace de recherche (cf. section 1.2.2).

Définition 1.1.4 (Contrainte).
Une contrainte q est un prédicat défini sur un langage.

Dans le but de décrire avec plus de précision les motifs recherchés, plusieurs contraintes peuvent être combinées formant ainsi une *requête*.

L'extraction de motifs sous contraintes consiste à extraire d'une base $\boldsymbol{r}$ décrite par des éléments du langage $\mathcal{L}_\mathcal{I}$ les motifs satisfaisant une contrainte q. Plus formellement, il s'agit de déterminer la théorie correspondante (l'ensemble des motifs satisfaisant q dans $\boldsymbol{r}$) :

Définition 1.1.5 (Théorie). *[Mannila et Toivonen, 1997]*
Pour un langage donné $\mathcal{L}_\mathcal{I}$, une base de données $\boldsymbol{r}$ et une contrainte q, la théorie $\mathcal{T}h(\mathcal{L}_\mathcal{I}, \boldsymbol{r}, q)$ est l'ensemble des motifs de $\mathcal{L}_\mathcal{I}$ satisfaisant la contrainte q dans $\boldsymbol{r}$.

1.1.3 Motifs locaux - Contraintes locales

Les *motifs locaux* sont des régularités observées dans certaines parties des données [Hand, 2002, Morik *et al.*, 2005]. Ils sont extraits via les *contraintes locales* (unaires).

Il existe de nombreuses contraintes permettant d'évaluer la pertinence et la qualité des motifs locaux, comme par exemple les contraintes exprimées à l'aide de mesures définissant l'intérêt des motifs extraits [Geng et Hamilton, 2006]. Le nombre d'occurrences d'un motif dans la base de données, défini par la mesure de sa fréquence, est l'une des mesures les plus classiques. La contrainte de fréquence découlant de cette mesure (définition 1.1.6) servira d'exemple pour présenter les différentes notions dans la suite de ce chapitre. D'autres critères sont utiles pour construire des motifs intéressant l'utilisateur. Par exemple, supposons que l'utilisateur recherche des motifs suffisamment fréquents et longs. Un tel compromis s'exprime par la mesure d'aire [Geerts *et al.*, 2004] définie par le produit de la fréquence d'un motif par sa longueur : $\texttt{area}(X) = \texttt{freq}(X) \times \texttt{size}(X)$.

Exemple 1.1.1. *Un premier exemple de motifs locaux, est défini par la conjonction des contraintes locales suivantes :* $\mathit{freq}(X) \geq 3 \wedge \mathit{size}(X) \geq (2)$ *vérifiant des propriétés de fréquence et de taille sur un seul motif* X*. L'ensemble des solutions obtenues à partir du jeu de données exemple du tableau 1.1 est* BC*,* BE*,* CE *et* BCE*.*

Les notions de support et de fréquence d'un motif sont introduites comme suit :

Définition 1.1.6 (Support, fréquence)**.**

Une transaction $t \in \mathcal{T}$ *supporte le motif* X *(ou* X *est présent dans* t*) ssi* $X \subseteq t$ *(ou* $\forall i \in X, R(i,t) = 1$*). Le* ***support*** $\mathit{support}(X)$ *d'un motif* X *d'attributs est l'ensemble des transactions qui le supportent.*

Sa mesure de fréquence $\mathit{freq}(X)$ *est le cardinal du support*[1]*.*

Définition 1.1.7 (Motif fréquent)**.**

Un motif est fréquent (ou min_{freq}*-fréquent) ssi sa fréquence dépasse un seuil de fréquence minimale* min_{freq} *fixé par l'utilisateur.*

On note $\mathcal{F}(\mathbf{r}, min_{\mathit{freq}})$ *l'ensemble des motifs fréquents dans la base de données* $\mathbf{r}$ *pour une fréquence minimale* min_{freq}*.*

Dans notre exemple du tableau 1.1 (page 14) les motifs B, C, E, BC, BE, CE et BCE sont 3-fréquents.

Les motifs fréquents sont à la base, entre autres, de la construction de *règles d'association*, une méthode non supervisée, permettant l'identification des relations significatives entre les données.

Définition 1.1.8 (Règle d'association - Confiance)**.**

Soit $\mathbf{r}$ *une base de données et* $Z \subseteq \mathcal{I}$ *un motif. Une règle d'association basée sur* Z *est une expression* $X \rightarrow Y$ *avec* $X \subsetneq Z$ *et* $Y = Z \backslash X$*.* X *est la prémisse,* Y *est la conclusion. La fréquence de la règle est celle de* XY *(i.e.,* $X \cup Y$*).*

La confiance notée $conf(X \rightarrow Y)$*, est la proportion d'objets contenant* X *qui contiennent aussi* Y *[Agrawal et Srikant, 1994] :* $conf(X \rightarrow Y) = \frac{\mathit{freq}(X \cup Y)}{\mathit{freq}(X)}$*.*

Une règle exacte dans $\mathbf{r}$ *est une règle de confiance 1, i.e nous notons* $\models_r X \rightarrow Y$ *quand* $\mathit{freq}(X \cup Y) = \mathit{freq}(X)$*.*

La définition 1.1.8 repose sur un calcul relatif de la confiance. Une valeur absolue du nombre d'éléments ne vérifiant pas la conclusion est aussi utilisée pour le calcul de la confiance d'une règle d'association et donne lieu aux *règles d'association* δ*-fortes* [Boulicaut *et al.*, 2000].

Définition 1.1.9 (Règle d'association δ-forte)**.**

Une règle δ*-forte est une règle d'association de la forme* $X \rightarrow Y$ *qui admet au plus* δ *exceptions, soit* $\mathit{freq}(X) - \mathit{freq}(X \cup Y) \leq \delta$

Le calcul de la confiance d'une règle $X \rightarrow Y$ revient à calculer la différence exacte entre la fréquence du motif X et celle de $X \cup Y$.

1. Dans la littérature, le terme support est parfois utilisé pour désigner le nombre d'occurrences d'un motif (ce que nous avons appelé la fréquence), tandis que la fréquence est utilisée pour exprimer en pourcentage la fraction d'occurrences relativement au nombre total d'objets.

1.2 Espace de recherche et élagage

1.2.1 Espace de recherche

Bien que la formulation du problème de l'extraction de motifs contraints soit simple, sa résolution est une tâche algorithmiquement difficile car l'espace de recherche des motifs est très grand : si $|\mathcal{I}| = n$, alors la taille de l'espace de recherche est $\mathcal{O}(2^n)$.

Pour les motifs ensemblistes, l'espace $\mathcal{L}_\mathcal{I}$ dans lequel les motifs sont recherchés correspond à un treillis (cf. figure 1.1). En plaçant l'ensemble vide en haut, la deuxième ligne contient les singletons, la troisième les paires, etc...

Définition 1.2.1 (Spécialisation - Généralisation)**.**
Une relation de spécialisation $\preceq$ est un ordre partiel défini sur les motifs de $\mathcal{L}_\mathcal{I}$. Un motif X est plus spécifique qu'un motif Y (respectivement plus général) si $Y \preceq X$ (respectivement $X \preceq Y$).

L'inclusion entre deux motifs définit une relation de **spécialisation** [Mitchell, 1981]. Ainsi, si un motif X est inclus dans un motif Y, on dit que Y est plus spécifique que X ou que X est plus général que Y. Par exemple BC est inclus dans ABC.

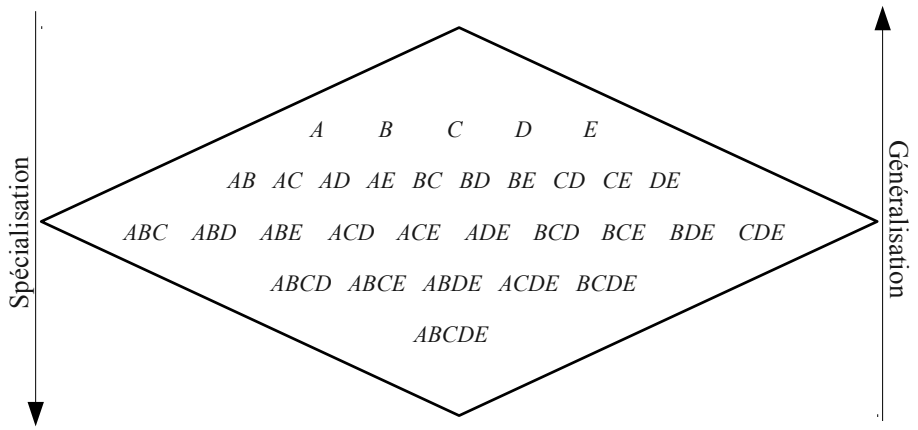

FIGURE 1.1 – Treillis représentant le langage $\mathcal{L}_\mathcal{I}$ avec $\mathcal{I} = \{A, B, C, D, E\}$

Plusieurs travaux traitent de la découverte de motifs locaux sous contraintes [Agrawal et Srikant, 1994, Ng *et al.*, 1998, Bonchi et Lucchese, 2007]. Leur idée maîtresse est d'exploiter les propriétés de monotonie/anti-monotonie de certaines contraintes afin d'élaguer l'espace de recherche (filtrage) [Mannila et Toivonen, 1997]. Ce point fait l'objet de la section suivante.

1.2.2 Les contraintes monotones et anti-monotones

La taille de l'espace de recherche est liée à la cardinalité du langage. Pour la recherche de motifs ensemblistes, cette taille augmente exponentiellement en fonction du nombre d'items. Un langage composé de 200 items, ce qui est très courant, comporte 2^{200} motifs

à tester dans le pire cas 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). Ainsi, il est illusoire de parcourir exhaustivement l'espace de recherche et il est nécessaire d'adopter des stratégies d'élagage. C'est pourquoi la communauté fouille de données s'est intéressée à définir des classes de contraintes pour lesquelles il existe des propriétés d'élagage de l'espace de recherche telles que la monotonie et l'anti-monotonie.

Donnons l'exemple de la fréquence d'un motif : celle-ci a une propriété remarquable : elle ne peut que diminuer quand on spécialise le motif. Sur notre exemple (tableau 1.1, page 14), le motif BC a une fréquence de 5 alors que ABC n'est présent que 2 fois. La spécialisation diminue le support du motif concerné. Inversement, quand on généralise un motif, sa fréquence ne peut qu'augmenter. Ainsi, les généralisations d'un motif fréquent sont fréquentes également. Cette propriété est qualifiée d'anti-monotone par rapport à la relation de spécialisation des items, par opposition à une propriété monotone.

Définition 1.2.2 (Anti-monotonie)**.**
Une contrainte $q(X, \boldsymbol{r})$ est anti-monotone suivant la spécialisation ssi

$$\forall X, Y \in \mathcal{L}_{\mathcal{I}},\ X \subseteq Y \Rightarrow (q(Y, \boldsymbol{r}) \Rightarrow q(X, \boldsymbol{r})).$$

$\texttt{freq}(X) \geq min_{\texttt{freq}}$ est un exemple de contrainte anti-monotone.

Définition 1.2.3 (Monotonie)**.**
Une contrainte $q(X, \boldsymbol{r})$ portant sur un motif X de $\mathcal{L}_{\mathcal{I}}$ relativement à une base de données $\boldsymbol{r}$, est monotone suivant la spécialisation si et seulement si

$$\forall X, Y \in \mathcal{L}_{\mathcal{I}},\ X \subseteq Y \Rightarrow (q(X, \boldsymbol{r}) \Rightarrow q(Y, \boldsymbol{r})).$$

$\texttt{size}(X) \geq \delta$ est un exemple de contrainte monotone.

Remarque 1.2.1.
Dans la suite de ce manuscrit, pour simplifier, nous ne précisons pas la relation entre les motifs et la relation de spécialisation est sous-entendue.

L'(anti-)monotonie conduit à une importante propriété d'élagage.

Propriété 1.2.1 (Élagage fondé sur les contraintes (anti-)monotones)**.**
Si un motif X ne satisfait pas une contrainte anti-monotone (respectivement monotone) q, alors toutes ses spécialisations (respectivement généralisations) ne la satisfont pas.

1.2.3 L'algorithme Apriori

Nous présentons dans cette section l'algorithme historique de découverte des motifs fréquents, Apriori [Agrawal et Srikant, 1994]. Étant donné un seuil de fréquence $min_{\texttt{freq}}$, cet algorithme recherche tous les motifs $min_{\texttt{freq}}$-fréquents d'une base de données transactionnelle. C'est un algorithme de type "générer et tester" par niveau : à la k^{ieme} itération l'algorithme génère des k-motifs candidats en se fondant sur les $(k-1)$-motifs fréquents trouvés à l'itération précédente. La propriété 1.2.1 permet un élagage de l'espace de recherche : l'algorithme ne génère que les k-motifs qui sont des spécialisations des $(k-1)$-motifs fréquents. Ensuite, la fréquence des k-motifs candidats est évaluée par

un balayage de la base de données. Pour éviter les redondances dans les motifs candidats produits, les motifs sont munis de l'ordre lexicographique. La réduction significative du nombre des motifs engendrés par rapport à la méthode naïve (l'énumération de tous les motifs) est due à la propriété d'anti-monotonie de la contrainte de fréquence minimale. Le pseudo-code d'Apriori est donné dans l'algorithme 1. Les notations utilisées sont définies au tableau 1.2.

Symbole	Signification
C_k	Ensemble des k-motifs candidats
F_k	Ensemble des k-motifs fréquents
Apriori_Gen	fonction qui engendre les motifs candidats.

Table 1.2 – Notations utilisées dans l'algorithme Apriori

Fonction Apriori($\mathcal{D}$: **bd transactionnelle**, $min_{\texttt{freq}}$: **entier**) : **ensemble de motifs**
  $F_k \leftarrow \{1\text{-motifs fréquents}\}$;
  Pour $(k = 2 ; F_{k-1} \neq \emptyset ; k++)$ **faire**
    $F_k \leftarrow 0$;
    $C_k \leftarrow Apriori\_Gen(F_{k-1})$;
    Pour tout $c \in C_k$ **faire**
      $\texttt{freq}(c) \leftarrow CalculFreq(c, \mathcal{D})$;
      Si $(\texttt{freq}(c) \geq min_{\texttt{freq}})$ **Alors**
        $F_k \leftarrow F_k \cup \{c\}$;
      Fin Si
    Fin Pour
  Fin Pour
  Retourner $\bigcup_k F_k$;
Fin

Algorithme 1: L'algorithme Apriori

1.2.4 Les bordures d'une collection de motifs

Étant donné une contrainte anti-monotone, les bordures [Mitchell, 1981, Mannila et Toivonen, 1997] partitionnent l'espace de recherche en deux parties : d'un côté les motifs satisfaisant la contrainte, de l'autre ceux ne la satisfaisant pas.

Définition 1.2.4 (Bordure positive - Motifs maximaux)**.**
Soit S un ensemble de motifs de $\mathcal{L}_\mathcal{I}$ satisfaisant une contrainte anti-monotone q. La bordure positive de S, notée $\mathcal{B}d^+(S)$ est l'ensemble des motifs X de $\mathcal{L}_\mathcal{I}$ tels que :

$$X \in \mathcal{B}d^+(S) \quad ssi \quad q(X, \boldsymbol{r}) \wedge (\forall Y, X \subsetneq Y \Rightarrow \neg q(Y, \boldsymbol{r}))$$

$\mathcal{B}d^+(S)$ rassemble les motifs maximaux de S.

De manière duale, on définit la bordure négative :

Définition 1.2.5 (Bordure négative - Motifs minimaux)**.**

Soit S un ensemble de motifs de $\mathcal{L}_\mathcal{I}$ satisfaisant une contrainte anti-monotone q. La bordure négative de S, notée $\mathcal{B}d^-(S)$ est l'ensemble des motifs X de $\mathcal{L}_\mathcal{I}$ tels que :

$$X \in \mathcal{B}d^-(S) \quad ssi \quad X \in \mathcal{L}_\mathcal{I} \setminus S \text{ et } \forall Y \subsetneq X,\ q(Y, \boldsymbol{r})$$

$\mathcal{B}d^-(S)$ rassemble les motifs minimaux de $\mathcal{L}_\mathcal{I} \setminus S$.

Exemple 1.2.1.

Soit le jeux de données de la figure 1.1 (page 14). La figure 1.2 illustre le calcul de $Bd^+(\mathcal{F}(\boldsymbol{r}, 2))$ et $Bd^-(\mathcal{F}(\boldsymbol{r}, 3))$.

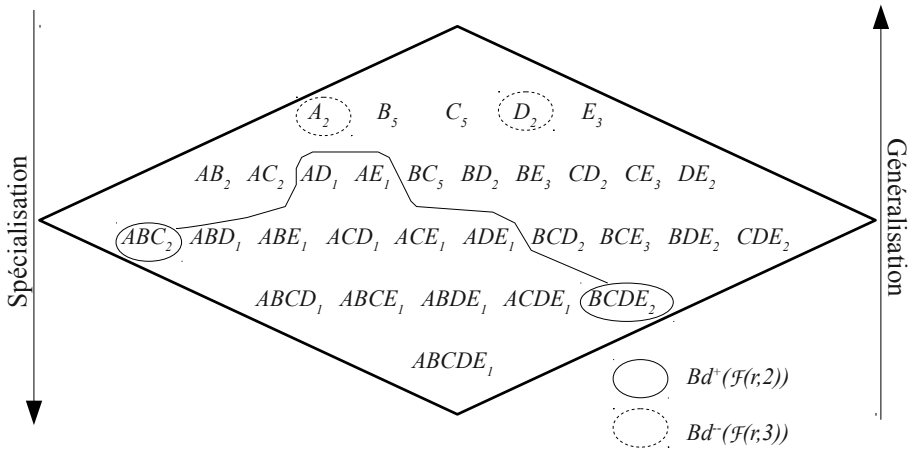

FIGURE 1.2 – $Bd^+(\mathcal{F}(\boldsymbol{r}, 2))$ et $Bd^-(\mathcal{F}(\boldsymbol{r}, 3))$

1.2.5 Motifs fermés

Dans cette section, nous commençons par donner la définition intuitive des motifs fermés, liée à l'extraction des motifs fréquents. En examinant le treillis des motifs (cf. figure 1.1, page 17), on remarque que les motifs peuvent être regroupés suivant leurs supports formant des classes d'équivalence, les motifs d'une classe d'équivalence partagent le même support et ont donc la même fréquence (cf. définition 1.1.6). La figure 1.3 illustre ce phénomène sur le jeu de données de la table 1.1 (page 14). Elle représente les motifs munis de leur fréquence, générés par un algorithme qui recherche les motifs 2-fréquents. Les motifs maximaux de chaque classe d'équivalence sont appelés *motifs fermés* et les minimaux *motifs libres*. Les libres et les fermés sont utilisés pour différentes tâches comme les représentations condensées [Calders *et al.*, 2005], la recherche de contraste [Novak *et al.*, 2009a] ou encore le clustering [Durand et Cremilleux, 2002]. À la figure 1.3, on constate par exemple qu'une classe regroupe les motifs A, AB, AC et ABC. Cela signifie que les items B et C apparaissent dans toutes les transactions contenant l'item A. Un algorithme d'extraction de motifs peut exploiter cette propriété pour élaguer l'espace de recherche : il évitera de produire des candidats d'une même classe d'équivalence et se limitera aux seuls motifs libres ou fermés.

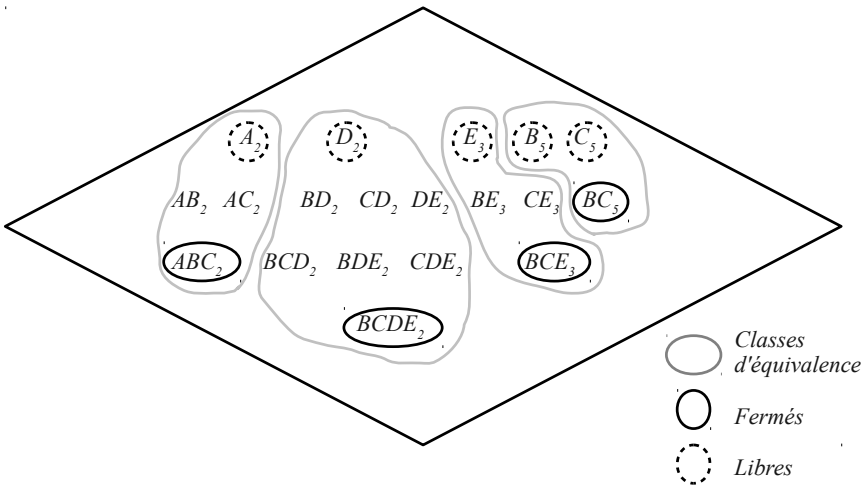

FIGURE 1.3 – Classes d'équivalence - motifs fermés - motifs libres

1.2.5.a Propriétés

Dans le contexte de la fouille de données, les motifs fermés ont des propriétés multiples et intéressantes : ils permettent, entre autre, de calculer efficacement les supports [Bastide *et al.*, 2002], déterminer les ensembles bases minimales de règles d'association [Pasquier *et al.*, 1999], d'oprechercher des contrastes [Soulet *et al.*, 2004a, Li *et al.*, 2007], de favoriser le clustering [Durand et Cremilleux, 2002], etc.

Définition 1.2.6.
Un motif X est fermé ssi toutes ses spécialisations strictes ont une fréquence strictement inférieure à celle de X.
X est un motif fermé ***ssi*** $\forall X' \supsetneq X,\ \mathit{freq}(X') < \mathit{freq}(X)$

En d'autres termes, X est fermé s'il n'a pas de spécialisation de même fréquence.

Propriété 1.2.2. *[Bastide* et al.*, 2000]*
Un opérateur de fermeture partitionne l'ensemble des motifs fréquents en classes d'équivalences disjointes par rapport à la propriété d'appartenance à la même fermeture.

Nous montrons dans la section 1.3.2 (page 24) comment cette dernière propriété est exploitée pour la génération de représentations condensées de motifs fréquents.

1.2.5.b Connexion de GALOIS

Formellement, les motifs maximaux des classes d'équivalence sont appelés des motifs *fermés* en référence à l'opérateur de fermeture de GALOIS. Il existe deux opérateurs de connexion de GALOIS f et g sur un contexte transactionnel $\boldsymbol{r}$.

Définition 1.2.7 (connexion de GALOIS)**.**
Soit $\boldsymbol{r} = (\mathcal{I}, \mathcal{R}, \mathcal{T})$ un contexte transactionnel, alors :
$f(T) = \{i \in \mathcal{I} | \forall t \in \mathcal{T},\ \mathcal{R}(i,t) = 1\}$
$g(X) = \{t \in \mathcal{T} | \forall i \in \mathcal{I},\ \mathcal{R}(i,t) = 1\}$

$f(T)$ représente l'ensemble de tous les items communs à un groupe de transactions T (on parle d'intension) et $g(X)$ représente l'ensemble de toutes les transactions partageant les même items de X (extension). Le couple (f, g) définit une connexion de GALOIS [Birkhoff, 1967] entre $\mathcal{I}$ et $\mathcal{T}$. $h = f \circ g$ et $h' = g \circ f$ sont les opérateurs de fermeture de GALOIS.

Rappel 1.2.1.

L'opérateur h est bien un opérateur de fermeture car il vérifie les 3 *propriétés suivantes [Stadler et Stadler, 2002] :*

- *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* $X \subseteq Y \Rightarrow h(X) \subseteq h(Y)$

Les motifs *fermés* sont les motifs invariants par l'opérateur de fermeture h (*i.e.*, $X = h(X)$). L'association d'un motif fermé et de son extension (motif fermé de transactions) forme un concept :

Définition 1.2.8 (motif fermé - concept)**.**

Un motif X est fermé ssi h(X)=X.

Un motif T de transactions est fermé ssi $h'(T) = T$.

Un concept [Wille, 1982] (X, T) *associe deux fermés X d'items et T de transactions, tels que* $X = f(T)$ *ou* $T = g(X)$.

1.2.6 Motifs libres - motifs δ-libres

En revenant sur la structure des classes d'équivalence (cf. figure 1.3), de façon duale aux motifs maximaux appelés motifs fermés, les motifs minimaux de chaque classe sont appelés motifs *libres*[2] [Boulicaut *et al.*, 2000].

La propriété de minimalité des motifs libres dans les classes d'équivalence souligne le fait que ces motifs ne contiennent pas de corrélation intrinsèque. Leur caractérisation est la suivante :

Définition 1.2.9 (motif libre)**.**

Un motif Z est libre s'il n'existe sur Z aucune règle d'association exacte $X \rightarrow Y$, *avec* $X \subsetneq Z$ *et* $Y = Z \setminus X$.

Les motifs δ-libres permettent de relaxer la contrainte de liberté, en considérant non plus des règles exactes pour calculer les fréquences, mais des règles dont le nombre d'exceptions est borné par δ. Un motif δ-libre est défini comme suit :

Définition 1.2.10 (Motif δ-libre)**.**

Un motif Z est δ*-libre s'il n'existe sur Z aucune règle* δ*-forte* $X \rightarrow Y$ *(avec* $X \subsetneq Z$ *et* $Y = Z \setminus X$*).*

Nous présentons dans la section 1.3.2 comment les motifs libres et δ-libres peuvent être utilisés pour la génération de représentations condensées de motifs fréquents.

2. appelés aussi dans la littérature motifs générateurs ou clés.

1.3 Représentation condensée de motifs

Les *représentations condensées* de motifs satisfaisant une contrainte q [Mannila et Toivonen, 1996], ont été introduites pour limiter le nombre de motifs extraits en s'appuyant sur les propriétés des classes d'équivalence. L'idée centrale des représentations condensées est de s'appuyer sur un nombre limité de motifs pivots à partir desquels il est possible de régénérer, si on le souhaite, tous les motifs vérifiant la contrainte q. Cette démarche facilite à la fois l'extraction des motifs et leur interprétation.

La recherche de représentations condensées a donné lieu à de nombreux travaux [Pasquier *et al.*, 1999, Boulicaut et Bykowski, 2000, Calders et Goethals, 2003, Boulicaut *et al.*, 2003, Soulet *et al.*, 2004b, El-Hajj et Zaïane, 2005, Calders et Goethals, 2007, Soulet et Crémilleux, 2008].

Il existe deux types de représentations condensées : les représentations dites exactes et celles dites approximatives. Avec une représentation condensée exacte, il est possible de régénérer la valeur exacte de la fréquence de chaque motif, alors que celle-ci est connue seulement avec une approximation contrôlée dans le cas des représentations approximatives.

Il existe de nombreuses représentations condensées de motifs ensemblistes telles que les représentations par motifs fermés [Pasquier *et al.*, 1999], les représentations par itemsets maximaux [Burdick *et al.*, 2005, Gouda et Zaki, 2005], les représentations par itemsets non-dérivables [Calders et Goethals, 2007], les représentations par ensembles libres [Boulicaut *et al.*, 2003], les représentations condensées par intervalles selon l'opérateur de fermeture préfixée [Soulet, 2006]. etc...

Définition 1.3.1 (Représentation condensée)**.**
La représentation condensée d'un ensemble de motifs $\mathcal{F}$ selon une contrainte q est un ensemble de motifs $\mathcal{C}$ de cardinalité inférieure à celle de $\mathcal{F}$ tel que $\mathcal{C} \subseteq \mathcal{F}$ et tous les éléments de $\mathcal{F} \setminus \mathcal{C}$ peuvent être générés de manière efficace à partir de $\mathcal{C}$ sans aucun accès à la base de données.

La qualité d'une représentation condensée des motifs satisfaisant une contrainte q est généralement évaluée selon trois critères :

1. **La taille de la représentation :** La taille de la représentation obtenue est un critère essentiel pour évaluer sa qualité : plus une représentation est petite, plus elle est intéressante.
2. **L'efficacité de l'extraction :** Extraire une représentation condensée doit être plus efficace en terme de temps de calcul et d'espace mémoire utilisé qu'extraire directement tous les motifs satisfaisant q. Plus l'extraction est rapide, plus la représentation condensée est intéressante.
3. **La complétude et l'efficacité de la régénération :** La régénération doit permettre de passer de la représentation condensée à l'ensemble correct et complet de tous les motifs satisfaisant q.

1.3.1 Représentation par motifs maximaux

La représentation par motifs maximaux est composée des motifs formant la bordure positive (cf. définition 1.2.4). C'est une représentation approximative puisqu'elle ne permet pas de calculer précisément la fréquence des motifs mais de dériver une borne inférieure de celle-ci.

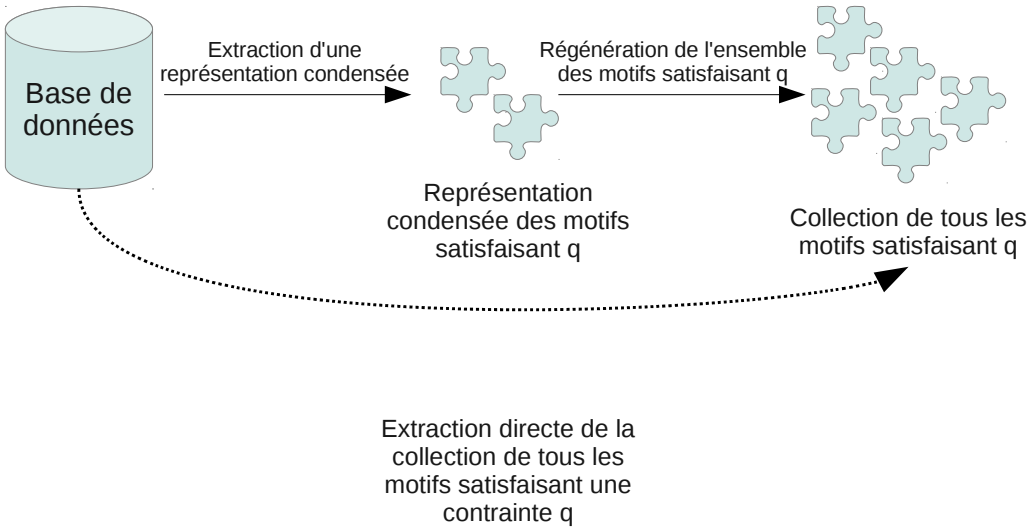

FIGURE 1.4 – Processus d'extraction de représentations condensées des motifs satisfaisant une contrainte q

Exemple 1.3.1.

Considérons la base de données transactionnelles $\boldsymbol{r}$ *représentée dans la table 1.1 et soit* $min_{\mathit{freq}} = 3$ *la valeur de la fréquence minimale pour l'extraction des motifs.*

L'ensemble des motifs fréquents et leurs fréquences associées est : $\{B : 5,\ C : 5,\ E : 3,\ BC : 5,\ BE : 3,\ CE : 3,\ BCE : 3\}$.

La représentation par motifs maximaux fréquents est : $\mathcal{C}(\boldsymbol{r}, 3) = \{BCE : 3\}$

A partir de $\mathcal{C}(\boldsymbol{r}, 3)$*, l'utilisateur ne peut pas régénérer de manière précise toutes les fréquences des motifs fréquents inclus dans* BCE*, mais grâce à la propriété d'anti-monotonie, l'utilisateur peut déduire, par exemple, que comme* $E \subseteq BCE$ *alors* $\mathit{freq}(E) \geq \mathit{freq}(BCE)$ *et approximer de cette manière la fréquence des motifs* 3*-fréquents.*

Un algorithme pionnier d'extraction de motifs maximaux est celui présenté par Bayardo [Roberto J. Bayardo, 1998]. Il fut suivi d'autres algorithmes et implémentations efficaces tels que [Burdick *et al.*, 2003b, Gouda et Zaki, 2005].

1.3.2 Représentation par motifs fermés

La représentation fondée sur les motifs fermés (cf. section 1.2.5) est une représentation exacte [Pasquier *et al.*, 1999, Boulicaut et Bykowski, 2000]. Elle tire son origine de la théorie des treillis et plus précisément des travaux autour de l'analyse de concepts formels [Ganter et Wille, 1997].

D'après les propriétés des motifs fermés (cf. section 1.2.5), les classes d'équivalence sont disjointes et les motifs d'une classe ont tous la même fréquence que l'unique motif fermé de la classe. Ce motif suffit pour indiquer la fréquence des autres motifs de la classe.

Soit $\mathcal{CF}(\boldsymbol{r}, min_{\mathtt{freq}})$ la représentation condensée des motifs $min_{\mathtt{freq}}$-fréquents obtenus à partir de la base de données $\boldsymbol{r}$. Il est possible de régénérer de manière exacte l'ensemble des motifs fréquents à partir uniquement de l'ensemble des motifs fermés comme suit :

Soit X un motif dont on veut déterminer la fréquence,
- si X ne possède pas de spécialisation contenue dans la collection $\mathcal{CF}(\mathbf{r},min_{\texttt{freq}})$, alors X n'est pas fréquent,
- sinon, il faut rechercher le plus petit fermé Y (au sens de l'inclusion) tel que $X \subset Y$. $\texttt{freq}(X)$ est alors égale à $\texttt{freq}(Y)$.

Cette notion de régénération exacte est illustrée dans l'exemple suivant :

Exemple 1.3.2.
Considérons le jeu de données représenté par le table 1.1 et un seuil minimal de fréquence fixé à 2.

Il y a 19 *motifs, mais seulement* 4 *classes d'équivalence (cf. figure 1.3). Les* 19 *motifs fréquents sont résumés par les* 4 *motifs maximaux des classes d'équivalence.*

Ainsi, en possédant uniquement la collection des motifs fermés ainsi que leurs fréquences : $\{ABC : 2,\ BCDE : 2,\ BCE : 3,\ BC : 5\}$ *on peut régénérer complètement la collection de tous les motifs 2-fréquents.*

Par exemple, la fréquence du motif B *est celle du plus petit fermé contenant* B*, ainsi* $\mathit{freq}(B) = \mathit{freq}(BC) = 5$.

1.3.3 Représentation par motifs libres

La collection des motifs libres fréquents munis de leur bordure négative est une représentation condensée exacte des motifs fréquents [Boulicaut *et al.*, 2003].

Étant donné un motif X, un seuil de fréquence minimale $min_{\texttt{freq}}$, l'ensemble des motifs libres fréquents $\mathcal{FL}(\mathbf{r},min_{\texttt{freq}})$ et leur bordure négative $\mathcal{B}d^{-}(\mathcal{FL}(\mathbf{r},min_{\texttt{freq}}))$, la fréquence exacte de X est déduite comme suit :
- si $\exists Y \in \mathcal{B}d^{-}(\mathcal{FL}(\mathbf{r},min_{\texttt{freq}}))$ tel que $Y \subseteq X$, alors X n'est pas fréquent,
- sinon la fréquence de X est la plus petite fréquence des libres fréquents inclus dans X.

Exemple 1.3.3. *Considérons le jeu de données représenté par le table 1.1 et un seuil minimal de fréquence fixé à* 3*. D'après les figures 1.2 représentant les bordures et la figure 1.3 représentant, entre autre, les motifs libres relatifs à ce jeu de données, on a l'ensemble des libres fréquents est* $\mathcal{FL}(\boldsymbol{r},3) = \{E : 3,\ B : 5,\ ,C : 5\}$ *et sa bordure négative est* $\mathcal{B}d^{-}(\mathcal{FL}(\boldsymbol{r},3)) = \{A : 2,\ D : 2\}$.

Par exemple, BDE *n'est pas fréquent car c'est une spécialisation d'un motif d'un motif de* $\mathcal{B}d^{-}(\mathcal{FL}(\boldsymbol{r},3))$ *(i.e,* D*). La fréquence de* BE *est la plus petite fréquence des libres fréquents inclus dans* BE*, d'où* $\mathit{freq}(BE) = \mathit{freq}(E) = 3$.

1.3.4 Représentation condensée issue de la fermeture préfixée

L'opérateur de fermeture préfixée introduit dans [Soulet, 2006] produit une représentation condensée sous forme d'intervalles disjoints. Cet opérateur de fermeture est basé sur un pré-ordre $\preceq_R$ qui prend en compte une relation d'ordre arbitraire sur les items.

Définition 1.3.2 (fermeture préfixée)**.**
La fermeture préfixée d'un motif X*, dénotée par* $\boldsymbol{cl}_R(X)$*, est le motif* $\{a \in \mathcal{I} | \exists Y \subseteq X$ *tel que* $Y \preceq_R Y \cup \{a\}$ *et* $\mathit{freq}(Y \cup \{a\}) = \mathit{freq}(Y)\}$

Le motifs $\mathbf{cl}_R(X)$ rassemblent tous les items apparaissant dans les mêmes transactions et contenant $Y \subseteq X$ tel que Y est un préfixe de $Y \cup \{a\}$. Les points fixes de l'opérateur $\mathbf{cl}_R(X)$ sont nommés les motifs fermés par préfixe.

Illustrons cette définition avec le contexte du tableau 1.1. On considère que la relation de pré-ordre $\preceq_R$ est tel que $A <_R B <_R C <_R D <_R E$. Le motif ACD n'est pas fermé par préfixe car ACD est un préfixe de ACDE et $\texttt{freq}(ACD) = \texttt{freq}(ACDE)$.

Propriété 1.3.1 (opérateur de liberté préfixée)**.**
Soit un motif X, il existe un unique motif minimal (au sens de l'inclusion), dénoté $fr_R(X)$, dans la classe d'équivalence de X.

Dans notre exemple, **AD** est libre par préfixe.

Définition 1.3.3 (Représentation condensée issue de $\mathbf{cl}_R$)**.** *On considère 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 $\mathcal{C}_{\preceq_R} = \{[\boldsymbol{fr}_R(X), \boldsymbol{cl}_R(X)] \subseteq \mathcal{L}_\mathcal{I} \times \mathcal{L}_\mathcal{I} | \boldsymbol{freq}(X) \geq 1\}$.*

Exemple 1.3.4. *La représentation condensée par fermeture préfixée de la base du tableau 1.1 est la suivante :* $\mathcal{C}_{\preceq_R} = \{[A, ACB] : 2,\ [AD, ADCEB] : 1,\ [AE, AECB] : 1,\ [D, DCBE] : 2,\ [E, ECB] : 3,\ [B, BC] : 5,\ C : 5\}$
$= \{[A, ABC] : 2,\ [AD, ABCDE] : 1,\ [AE, ABCE] : 1,\ [D, BCDE] : 2,\ [E, BCE] : 3,\ [B, BC] : 5,\ C : 5\}$

Théorème 1.3.1. *[Soulet, 2006] Chaque motif présent dans la base de données est inclus dans un unique intervalle de $\mathcal{C}_{\preceq_R}$.*

Cette propriété nous intéresse particulièrement et elle sera un des arguments justifiant le choix de la représentation condensée par fermeture préfixée pour la construction des domaines des variables dans une de nos approches CSP de résolution de problèmes d'extraction de motifs (cf. chapitre 5).

1.4 Extracteurs génériques de motifs locaux

Nous présentons dans cette section deux extracteurs génériques de motifs locaux Music-dfs et FIM_CP, sur lesquels nous reviendrons dans la suite de ce travail. Ces deux extracteurs reposent sur des paradigmes différents. Music-dfs se base sur une décomposition des contraintes en primitives tandis que FIM_CP utilise un solveur générique issu de la communauté PPC en proposant une modélisation CSP du problème d'extraction de motifs locaux sous contraintes. Notons qu'il existe d'autres méthodes génériques [Negrevergne, 2011] pour l'extraction de motifs locaux tel que la modélisation de ce problème sous la forme d'un système d'ensembles afin de tirer profit des résultats obtenus par M. Boley [Boley *et al.*, 2010] et H. Arimura [Arimura et Uno, 2009] sur l'extraction de motifs fermés. Ce choix permet une parallélisation de l'algorithme. Cette méthode traite de la recherche de motifs suivant plusieurs langages (itemsets fréquents fermés, graphes relationnels fermés et itemsets graduels fermés).

1.4.1 Music-dfs

Music-dfs[3] [Soulet et Crémilleux, 2005, Soulet, 2006] est un extracteur correct et complet de motifs locaux qui permet d'extraire efficacement l'ensemble des motifs satisfaisant un large éventail de contraintes locales [Soulet *et al.*, 2007].

3. `http://www.info.univ-tours.fr/~soulet/music-dfs/music-dfs.html`

Dans [Soulet, 2006], Arnaud Soulet propose un cadre fondé sur les primitives (Primitive-Based Framework, PBF), un cadre générique pour définir de façon flexible les contraintes locales. Ce cadre met à la disposition de l'utilisateur un dictionnaire de primitives plutôt qu'un dictionnaire de contraintes. Ces primitives sont des fonctions d'un fin niveau de granularité vérifiant une propriété de monotonie et qui peuvent être combinées les unes avec les autres pour décrire une large panoplie de contraintes.

À titre d'exemple, les fonctions `freq` et `size` sont des primitives du PBF (la première conduisant à la définition d'une contrainte anti-monotone et la seconde à une contrainte monotone). Ces primitives sont dites *terminales* vu qu'elles permettent d'évaluer un motif, elles peuvent être combinées les unes avec les autres à l'aide d'autres primitives dites *non-terminales* (e.g., $\times$). À partir de ces primitives, l'utilisateur peut librement construire sa requête (Primitive-Based Constraint, PBC).

L'efficacité de MUSIC-DFS est due à sa stratégie de recherche en profondeur d'abord et à l'élagage de l'espace de recherche se basant sur les propriétés de monotonie et d'anti-monotonie dans l'espace des motifs candidats. L'originalité de MUSIC-DFS est d'effectuer des élagages efficaces sur des intervalles à la place des seules généralisations ou spécialisations. Les motifs locaux sont produits sous forme de représentation condensée composée d'intervalles disjoints où chaque intervalle synthétise un ensemble de motifs satisfaisant la contrainte [Soulet *et al.*, 2007]. La représentation condensée produite est une représentation condensée par fermeture préfixée présentée dans la section 1.3.4. Chaque intervalle obtenu représente un ensemble de motifs satisfaisant les contraintes posées, et chaque motif solution appartient à un et un seul intervalle (cf. section 1.3.4).

Exemple 1.4.1.

Soit le jeu de données décrit par le tableau 1.1 et la requête $Q = \mathit{freq}(X) \geq 3 \wedge \mathit{size}(X) \geq (2)$. *Avec* MUSIC-DFS, *cette requête est écrite comme suit :*

```
-------------------------------------------------------
music-dfs -i DataFile.bin -q "count(X)>=3 and length(x)>=2;"

Solutions :
X in [CE..BCE] U [BE..BE] -- freq(X) = 3 ;
X in [BC..BC] -- freq(X) = 5 ;
-------------------------------------------------------
```

Le principe de décomposition de contraintes sur des primitives est aussi au cœur des contributions de Loïc Cerf [Cerf, 2010] sur l'extraction de motifs fermés sous contraintes dans des relations n-aires, c'est-à-dire des tenseurs.

1.4.2 FIM_CP

FIM_CP est le premier extracteur de motifs locaux utilisant la PPC [De Raedt *et al.*, 2008]. FIM_CP permet de traiter, dans un cadre unifié, un large ensemble de motifs locaux et de contraintes telles que la fréquence, la fermeture, la maximalité, etc.FIM_CP est implémenté sous *Gecode* (cf section C.2).

1.4.2.a Modélisation et lien entre les transactions et le motif recherché

L'originalité de FIM_CP est d'effectuer des tâches de fouille de données (i.e, extraction de motifs locaux sous contraintes) en utilisant uniquement un solveur de CSP (cf. chapitre 3). Ses points clés sont la modélisation CSP proposée et la technique utilisée pour établir le lien entre le motif local recherché et les transactions de la base de données. Nous détaillons ce lien dans cette section.

Soit $\boldsymbol{r}$ un contexte transactionnel où $\mathcal{I}$ est l'ensemble de ses n items et $\mathcal{T}$ l'ensemble de ses m transactions. Soit $\boldsymbol{d}$ la matrice $0/1$ de dimension $(m,\ n)$ telle que : $\forall t \in \mathcal{T}, \forall i \in \mathcal{I},\ (d_{t,i} = 1)\ ssi\ (i \in t)$.

Soit M le motif (unique) recherché par FIM_CP. M est décrit par n variables de décision $\{M_1, M_2, ..., M_n\}$ (de domaine $\{0, 1\}$) telles que :

$$\forall i \in \mathcal{I},\ (M_i = 1) \quad ssi \quad (i \in M) \tag{1.1}$$

Le support T du motif recherché M est représenté par m variables de décision $\{T_1, T_2, ..., T_m\}$ (de domaine $\{0, 1\}$) telles que :

$$\forall t \in \mathcal{T},\ (T_t = 1) \quad ssi \quad (M \subseteq t) \tag{1.2}$$

Cette représentation permet d'exprimer simplement les mesures suivantes :
- la fréquence d'un motif : $\texttt{freq}(M) = \sum_{t \in \mathcal{T}} T_t$
- la cardinalité d'un motif : $\texttt{size}(M) = \sum_{i \in \mathcal{I}} M_i$.

La relation entre le motif recherché M, son support T et la base de données $\boldsymbol{r}$ est établie via des contraintes réifiées (cf. section 3.1.2, page 43) imposant que, pour chaque transaction t, $(T_t = 1)$ ssi $(M \subseteq t)$, ce qui se reformule en :

$$\forall t \in \mathcal{T}, (T_t = 1) \Longleftrightarrow \sum_{i \in \mathcal{I}} M_i \times (1 - d_{t,i}) = 0 \tag{1.3}$$

Démonstration 1.4.1.
M a comme support $T = \{t \in \mathcal{T} | \forall i \in M : d_{t,i} = 1\}$
$ssi \quad \forall t \in \mathcal{T},\ t \in T \Longleftrightarrow \forall i \in\ M : d_{t,i} = 1$
$ssi \quad \forall t \in \mathcal{T},\ t \in T \Longleftrightarrow \forall i \in\ M : 1 - d_{t,i} = 0$
$ssi \quad \forall t \in \mathcal{T},\ T_t = 1 \Longleftrightarrow \sum_{i \in \mathcal{I}} M_i(1 - d_{t,i}) = 0$

Le filtrage associé à chaque contrainte réifiée (cf. équation 1.3) procède comme suit : si le solveur déduit que $(T_t = 1)$ (resp. $T_t = 0$), alors la somme doit être nulle (resp. non-nulle). La propagation s'effectue de manière analogue de la partie droite vers la partie gauche de l'équivalence.

1.4.2.b Reformulation des contraintes

La dernière étape de FIM_CP consiste à reformuler les contraintes sur les motifs à fournir directement afin de pouvoir les résoudre utilisant un solveur de CSP. [De Raedt *et al.*, 2008] propose des reformulations couvrant un large ensemble de motifs locaux et de contraintes telles que la fréquence, la fermeture, la maximalité, l'émergence, les contraintes monotones ou anti-monotones sur les coûts et des variations de ces contraintes.

Exemple 1.4.2.

Soit le jeu de données décrit par le tableau 1.1 et la requête $Q = \mathit{freq}(X) \geq 3 \wedge \mathit{size}(X) \geq (2)$. *Avec* FIM_CP, *cette requête est écrite comme suit :*

```
-------------------------------------------------------
fimcp_size -datafile DataFile.txt -freq 0.60 -bound1 GQ 2

Solutions :
X={0,1,1,0,0} (BC)  -- T={1,1,1,1,1}
X={0,1,0,0,1} (BE)  -- T={0,0,1,1,1}
X={0,0,1,0,1} (CE)  -- T={0,0,1,1,1}
X={0,1,1,0,1} (BCE) -- T={0,0,1,1,1}
-------------------------------------------------------
```

1.5 Conclusion : des motifs locaux aux ensembles de motifs

Les motifs locaux peuvent, dans certains cas, servir à une analyse directe des données par les analystes. Mais ils sont le plus souvent utilisés dans des approches découvrant des motifs de plus haut niveau (couverture d'un ensemble de motifs, ensembles de règles) [Raedt et Zimmermann, 2007, Khiari *et al.*, 2010b] ou à construire des modèles globaux (classification, clustering) [Knobbe *et al.*, 2008, Giacometti *et al.*, 2009].

Néanmoins, l'exploitation des motifs locaux est confrontée au problème du nombre conséquent de motifs produits rendant leur interprétation fastidieuse : les motifs les plus significatifs restent noyés au milieu d'informations triviales et souvent redondantes. Plusieurs approches se sont alors intéressées à la réduction du nombre de motifs extraits. Nous les présentons dans le chapitre suivant.

Chapitre 2

Vers l'extraction de motifs de plus haut niveau

Sommaire

Le chapitre précédent a présenté la problématique de l'extraction de motifs locaus sous contraintes. Cependant, un problème majeur bien connu en fouille de données est la taille du volume d'informations extraites rendant fastidieuse son interprétation par les analystes. Typiquement, un processus d'extraction de motifs locaux retourne de nombreux motifs intéressants, mais aussi un grand nombre de motifs inintéressants ou redondants rendant très délicate la découverte des motifs pertinents. De plus, en pratique, l'utilisateur est souvent intéressé par la découverte de motifs de plus haut niveau reposant sur des caractéristiques impliquant plusieurs motifs locaux. Nous présentons dans ce chapitre deux approches permettant de réduire le nombre de motifs extraits tout en n'en gardant que les plus pertinents. La première (cf. section 2.1) est fondée sur l'élimination de la redondance dans l'ensemble des motifs extraits. La seconde (cf. section 2.2) repose sur la notion de *contraintes n-aires* qui permet la prise en compte de caractéristiques portant sur plusieurs motifs. Finalement, nous présentons dans la section 2.3 quelques exemples d'utilisations des motifs pour la construction de modèles globaux.

2.1 Élimination de la redondance

Nous présentons dans cette section les approches fondées sur l'élimination de la redondance entre les motifs extraits, le résultat formant un résumé de ces motifs. Pour cela,

nous commençons par définir la notion de couverture.

Définition 2.1.1 (Couverture d'un ensemble de motifs)**.**
La couverture d'un ensemble de motifs $\mathbb{S}$ *est l'ensemble des transactions qui appartiennent au support d'au moins un des motifs de* $\mathbb{S}$*.*

La redondance d'un motif est définie à l'aide de mesures de redondance souvent liées à la notion de couverture [Knobbe et Ho, 2006, Xin *et al.*, 2006, Bringmann et Zimmermann, 2007]

2.1.1 Exemples d'approches

2.1.1.a The chosen few

L'approche «the chosen few» [Bringmann et Zimmermann, 2007, Bringmann et Zimmermann, 2009] élimine la redondance à partir de techniques heuristiques. Ces heuristiques construisent un sous ensemble de la collection de motifs extraits de telle sorte que les classes d'équivalence (par rapport à la couverture) produites soient les mêmes que celles de la collection initiale de motifs.

Deux types de redondance sont traitées : la redondance de couverture et la redondance basée sur une mesure. Pour la redondance de couverture, un motif est considéré redondant ssi son ajout au résumé courant ne change pas l'ensemble des classes d'équivalence [Bringmann et Zimmermann, 2007]. Soit un ensemble de motifs $\mathbb{S} = \{X_1, ..., X_n\}$, deux transactions sont dans une même classe d'équivalence ssi elle sont couvertes par le même ensemble de motifs dans $\mathbb{S}$. Les classes d'équivalence sont définies par par la relation $\sim_{\mathbb{S}}$ suivante :

$$\sim_{\mathbb{S}} = \{(t_1, t_2) \in \mathcal{T} \times \mathcal{T} | \forall X \in \mathbb{S}, (t_1 \in \texttt{support}(X) \Leftrightarrow t_2 \in \texttt{support}(X))\}$$

Le second type de redondance élimine les motifs à l'aide d'une mesure de redondance Φ en fonction des motifs déjà sélectionnés :
$\Phi(\mathcal{T}, \mathbb{S}^*, X) \rightarrow [0, 1] \subset \mathbb{R}$
où $\mathbb{S}^*$ est l'ensemble de motifs construit à partir d'un ensemble initial $\mathbb{S}$ en éliminant la redondance dans ce dernier. La mesure Φ évalue l'intérêt de l'ajout d'un motif X de $\mathbb{S}$ à $\mathbb{S}^*$. Un exemple d'intérêt proposé par les auteurs est la mesure Φ_Q évaluant la variation du nombre de classes d'équivalence dans l'ensemble $\mathbb{S}^*$ par l'ajout d'un motif candidat X :
$\Phi_Q(\mathcal{T}, \mathbb{S}^*, X) = 1 - \frac{|\mathcal{T}/\sim_{\mathbb{S}^*}|}{|\mathcal{T}/\sim_{(\mathbb{S}^* \cup X)}|}$
où $\mathcal{T}/\sim_{\mathbb{S}^*}$ est l'ensemble des classes d'équivalence dans $\mathcal{T}$ par rapport à l'ensemble $\mathbb{S}^*$ et $\mathcal{T}/\sim_{\mathbb{S}^* \cup X}$ est celui des classes d'équivalence dans $\mathcal{T}$ par rapport à l'ensemble $\mathbb{S}^*$ auquel on ajoute le motif candidat X. Par exemple, l'ajout d'un motif X qui ne change pas les classes d'équivalence de $\mathbb{S}$ conclut à $\Phi_Q(\mathcal{T}, \mathbb{S}^*, X) = 0$ c'est à dire un intérêt nul.

Les études expérimentales montrent que l'ensemble des motifs sélectionnés par cette méthode améliore les performances en classification supervisée par rapport à celles obtenues à partir de l'ensemble initial de motifs.

2.1.1.b Les top-k tenant compte de la redondance

Une tâche classique en fouille de données est l'extraction des top-k motifs c'est à dire des k-motifs maximisant une mesure d'intérêt [Fu *et al.*, 2000, Han *et al.*, 2002]. Ces

méthodes ne prennent pas en compte la redondance entre les motifs extraits qui est en pratique particulièrement forte comme, par exemple, avec la mesure de fréquence. A contrario, les top-k motifs tenant compte de la redondance (redundancy-aware top-k patterns) [Xin *et al.*, 2006] construisent un résumé des données de taille prédéfinie k en tenant compte de l'intérêt d'un motif par rapport à ceux précédemment extraits. Le but est de trouver k motifs satisfaisant un bon compromis entre forte mesure d'intérêt (par exemple le taux de croissance) et limitation de redondance entre les motifs extraits. La redondance entre deux motifs X et Y est définie dans [Xin *et al.*, 2006] comme étant fonction de l'intérêt de X, celui de Y et celui du couple de motifs (X, Y).

2.1.1.c Les équipes de motifs (Pattern Teams)

Toujours dans cette lignée, [Knobbe et Ho, 2006] propose de filtrer l'ensemble des motifs extraits en se basant sur des mesures de qualité telles que l'interdépendance des motifs, le recouvrement entre les motifs extraits et le pouvoir de prédiction des motifs combinés. Les sous ensembles de motifs satisfaisant de telles mesures sont appelés *pattern teams* (équipes de motifs). De même que dans le cas des top-k motifs tenant compte de la redondance, les pattern teams recherchés ont une taille prédéfinie k. Les mesures utilisées permettent de réduire considérablement la redondance dans l'ensemble des motifs extraits tout en essayant de garantir l'obtention du meilleur classifieur possible par la suite, ce qui est exprimé par le choix de mesures exprimant des corrélations avec la classe.

2.1.2 Discussion

Les approches par élimination de redondance ont prouvé leur efficacité dans la réduction de la collection des motifs extraits. De plus, elles permettent d'améliorer la qualité des modèles qui sont ensuite construits, comme les classifieurs. En réduisant considérablement la redondance des motifs, ces approches contribuent à faire face au grand nombre de motifs extraits. Néanmoins, ces approches ne permettent pas de modéliser des contraintes portant simultanément sur plusieurs motifs, ce qui limite l'expressivité offerte à l'utilisateur.

2.2 Contraintes sur des ensembles de motifs

2.2.1 Généralités - Définitions

En pratique, l'utilisateur est souvent intéressé par la découverte de motifs combinant plusieurs motifs, comme par exemple les règles de classification les plus simples afin d'éviter le sur-apprentissage [Yin et Han, 2003], ou encore les paires de règles d'exceptions de Suzuki [Suzuki, 2002, Duval *et al.*, 2007]. La communauté a défini de nouveaux types de contraintes pour la découverte de ce type de motifs. Dans [Raedt et Zimmermann, 2007], ces contraintes sont définies comme étant des *contraintes sur des ensembles de motifs*. Dans [Soulet, 2006, Crémilleux et Soulet, 2008], elles sont appelées *contraintes globales*. Dans [Giacometti *et al.*, 2011], pour un langage donné $\mathcal{L}_\mathcal{I}$, une base de données $\boldsymbol{r}$ et une contrainte q, les auteurs classifient les contraintes selon leur interactions avec les différentes composantes de la théorie $\mathcal{T}h(\mathcal{L}_\mathcal{I}, \boldsymbol{r}, q)$. Les motifs impliquant plusieurs motifs locaux sont ainsi définis comme étant *motifs globaux par rapport au langage* $\mathcal{L}_\mathcal{I}$.

Dans ce manuscrit, nous proposons d'appeler *contraintes n-aires* les contraintes portant sur n motifs ($n \geq 1$) [Khiari *et al.*, 2009, Khiari *et al.*, 2010c].

Définition 2.2.1 (Contrainte n-aire).
Une contrainte n-aire est une contrainte portant sur n motifs.

Définition 2.2.2 (Requête n-aire).
Une requête n-aire est une formule construite à partir de contraintes n-aires et des connecteurs logiques $\wedge$ (conjonction) et $\vee$ (disjonction).

Définition 2.2.3 (Motif n-aire).
Un motif n-aire est un motif défini par une requête n-aire.

L'extraction de motifs sous contraintes locales demande l'exploration d'un espace de recherche de très grande taille, même lorsqu'il s'agit de motifs simples tels que les motifs ensemblistes. Bien évidemment, l'extraction sous contraintes portant sur plusieurs motifs est encore plus complexe puisqu'il faut prendre en considération et comparer plusieurs motifs à la fois. En effet, si $|\mathcal{I}| = n$, alors la taille de l'espace de recherche est $\mathcal{O}(2^n)$ dans le cas des motifs locaux (unaires) unaires et $\mathcal{O}(2^{(k \times n)})$ dans le cas de contraintes portant sur k motifs.

Plusieurs travaux se sont intéressés aux contraintes portant sur plusieurs motifs, mais ils se limitent à des contraintes particulières ou au mieux à une classe particulière de contraintes. Nous présentons dans ce qui suit les principales approches.

2.2.2 Approches dédiées à des classes particulières de contraintes n-aires

A la fin des années 90, Lakshmanan constate que plusieurs requêtes portant sur les motifs fréquents portent naturellement sur deux variables interdépendantes. C'est un problème difficile puisque lors de la résolution, les deux variables changent simultanément de valeur, et les propriétés d'élagage de l'espace de recherche sont beaucoup plus dures à établir que lorsqu'il s'agit de contraintes portant sur une seule variable. [Lakshmanan *et al.*, 1999] propose une approche dédiée à la résolution d'une classe particulière de contraintes, à savoir les contraintes quasi-succintes portant sur deux variables interdépendantes [Lakshmanan *et al.*, 1999] . La technique consiste à transformer une contrainte quasi-succinte portant sur deux variables en deux contraintes succinctes portant chacune sur une seule variable.

On trouve dans la littérature d'autres travaux traitant de classes particulières de contraintes. Dans [Raedt et Zimmermann, 2007], les auteurs montrent comment des propriétés bien connues de l'extraction de motifs locaux telles que la monotonie et l'anti-monotonie, peuvent être adaptées au cas d'extraction d'ensembles de motifs. Deux algorithmes sont présentés, un algorithme par niveaux qui extrait les ensembles de motifs satisfaisant une conjonction de contraintes monotones et anti-monotones, et un algorithme permettant en plus d'extraire les $top - k$ motifs.

L'idée majeure utilisée est d'ajouter à la première phase d'extraction de motifs locaux, une seconde phase dans laquelle on cible des ensembles de motifs. En d'autres termes :

1. dans la première phase le langage est construit à partir de l'ensemble d'items $\mathcal{I}$ et l'extraction porte sur des ensembles d'items,
2. dans la seconde phase, le langage est construit ainsi de l'ensemble de motifs locaux précédemment extraits, et l'extraction porte sur des ensembles de motifs.

Plus formellement, à partir de l'ensemble $\mathbf{L} = Th(\mathcal{L}_{\mathcal{I}}, \boldsymbol{r}, q)$ représentant les motifs locaux extraits, on cherche à générer l'ensemble $\mathbf{M} = \mathbf{Th}(\mathbf{L}, \boldsymbol{r}, \mathbf{p})$ des motifs vérifiant les contraintes n-aires. L'ensemble d'ensembles de motifs $\mathbf{M}$ est obtenu en formulant des contraintes $\mathbf{p}$ portant sur des ensembles de motifs (sous ensembles de $\mathbf{L}$). Dans ce cas $\mathbf{L}$ représente l'ensemble des motifs satisfaisant les contraintes locales, $\mathbf{M}$ représente l'ensemble des ensembles de motifs résultant des contraintes portant sur des ensembles de motifs qui sont posées par l'utilisateur après l'extraction des motifs locaux.

2.2.3 Approches dédiées à des contraintes n-aires particulières

Nous présentons dans cette section quelques exemples de requêtes n-aires qui sont résolues à l'aide de méthodes qui leur sont spécialement dédiées.

2.2.3.a Règles inattendues

Dans [Padmanabhan et Tuzhilin, 1998, Padmanabhan et Tuzhilin, 1999, The et Padmanabhan, 2000] les auteurs introduisent la notion de règle *inattendue* c'est à dire d'une règle qui contredit une croyance préalablement donnée. La méthode extrait l'ensemble des règles inattendues $X \rightarrow Y$ par rapport à une croyance $U \rightarrow V$ où U et V sont des motifs. Plus formellement, une règle inattendue est définie dans [Padmanabhan et Tuzhilin, 1998] par :

1. $Y \wedge V$ n'est pas valide,
2. $X \wedge U$ est valide (XU est un motif fréquent),
3. $XU \rightarrow Y$ est valide ($XU \rightarrow Y$ est une règle fréquente et de confiance suffisante),
4. $XU \rightarrow V$ n'est pas valide (soit $XU \rightarrow V$ n'est pas une règle fréquente, soit $XU \rightarrow V$ est une règle de faible confiance).

L'algorithme `ZoomUR` (Zoom to Unexpected association Rules) extrait les règles inattendues par rapport aux croyances fixées par l'utilisateur. Dans [Padmanabhan et Tuzhilin, 1998, Padmanabhan et Tuzhilin, 1999] deux applications sur des bases de données réelles sont décrites, l'une concernant les journaux d'accès au web et l'autre un historique d'achats commerciaux. Par exemple, pour la deuxième base de données, la croyance donnée est *«Les clients ayant un emploi font plus d'achats les weekends qu'en semaine»*, l'algorithme extrait la règle inattendue suivante : «Au mois de décembre, les clients ayant un emploi font plus d'achats en semaine que pendant les les weekends», l'algorithme extrait aussi une généralisation de la règle précédente : «Au mois de décembre, les clients font plus d'achats en semaine que pendant les les weekends», les auteurs montrent que ces généralisations ne contredisent pas les règles inattendues extraites.

2.2.3.b Règles d'exception

E. Suzuki s'est intéressé à la découverte de paires de règles incluant une règle d'exception [Suzuki, 2002]. Une règle d'exception est une règle qui exprime une situation déviant d'un comportement général modélisé par une règle de sens commun : l'intérêt de cette définition est d'expliciter la nature d'une exception par rapport à un comportement général reconnu. Formellement, les règles d'exception sont définies comme suit (I est un item, par exemple une valeur de classe, X et Y sont des motifs locaux) :

Trans.	Items				
t_1	A	B		c_1	
t_2	A	B		c_1	
t_3			C	c_1	
t_4			C	c_1	
t_5			C	c_1	
t_6	A	B	C	D	c_2
t_7			C	D	c_2
t_8			C		c_2
t_9				D	c_2

TABLE 2.1 – Jeu de données exemple

$$exception(X \rightarrow \neg I) \equiv \begin{cases} vrai & \text{si } \exists Y \in \mathcal{L}_{\mathcal{I}} \text{ tq } Y \subset X, \\ & X \backslash Y \rightarrow I \wedge \\ & X \rightarrow \neg I \wedge \\ & Y \nrightarrow \neg I \\ faux & \text{sinon} \end{cases}$$

Dans cette définition :
- $X \backslash Y \rightarrow I$ est une règle générale,
- $X \rightarrow \neg I$ est une règle d'exception qui révèle ainsi une information inattendue par rapport à la règle générale,
- $Y \nrightarrow \neg I$ permet de vérifier que l'exception n'est pas déclenchée par un motif caractérisant la classe $\neg I$ [1].

Cette définition demande à ce que la règle générale soit de forte fréquence et de forte confiance tandis que la règle d'exception est peu fréquente mais de très forte confiance. La comparaison entre la règle générale et la règle d'exception nécessite l'utilisation de contraintes n-aires.

Exemple 2.2.1. *Donnons un exemple de règle d'exception à partir du tableau 2.1. En prenant 2/3 comme seuil de confiance d'une règle, la règle* $AC \rightarrow \neg c_1$ *est une règle d'exception puisque nous avons conjointement* $A \rightarrow c_1$ *et* $AC \rightarrow \neg c_1$.

E. Suzuki a proposé une méthode fondée sur une estimation probabiliste [Suzuki, 2002] et dédiée à l'extraction de telles règles.

2.2.3.c Les motifs pics

Les motifs *pic* ou sommet (ou encore *peak*) [Crémilleux et Soulet, 2008] sont un autre exemple de requête n-aire. Un motif pic est un motif qui possède une valeur d'une mesure m donnée assez grande par rapport à celles de tous ses voisins. En d'autres termes, un motif pic a un comportement exceptionnel par rapport à ses voisins (le voisinage d'un motif est calculé par rapport à une distance d donnée).

Soit $m : \mathcal{L}_{\mathcal{I}} \rightarrow \mathbb{R}$ une mesure, δ un entier et ρ un réel, et soit d une distance, on a :

$$peak(X) \equiv \begin{cases} vrai & \text{si } \forall Y \in \mathcal{L}_{\mathcal{I}} \textit{ tel que } d(X,Y) < \delta, m(X) \geq \rho \times m(Y) \\ faux & \text{sinon} \end{cases}$$

1. Dans la suite, pour simplifier la compréhension des exemples, nous considérons une version abrégée de la requête des règles d'exception n'intégrant pas cette dernière contrainte.

Exemple 2.2.2.
Soit $m(X) = \texttt{area}(X)$, $\delta = 1$, $\rho = 2$ *et* $d(X,Y) = |X \backslash Y| + |Y \backslash X|$, *et considérons le jeu de données du tableau 2.1 sans prendre en compte les items* c_1 *et* c_2.
Le motif **AB** *est un peak puisque* $\texttt{area}(AB) = 6$ *est au moins 2 fois plus grande que l'aire des 4 voisins de* AB *(*$\texttt{area}(A) = 3$, $\texttt{area}(B) = 3$, $\texttt{area}(ABC) = 3$, $\texttt{area}(ABD) = 3$ *et* $3 \times \rho \leq 6$*).*

Cette contrainte a été présentée dans [Crémilleux et Soulet, 2008] sans méthode de résolution associée.

2.2.3.d Les groupes de synexpression

Le domaine de l'analyse d'expression de gènes fournit un autre exemple de contrainte n-aire. Dans les jeux de données étudiés, les situations biologiques forment les lignes, et les tags les colonnes. Les motifs locaux, tels que les concepts formels, et satisfaisant la contrainte d'aire (cf. section 1.1.3) sont au cœur de la recherche de groupes de synexpression [Kléma *et al.*, 2008] . Néanmoins, dans des données réelles telles que celles du transcriptome, la recherche de motifs tolérants aux erreurs[2] y est capitale afin de tenir compte de l'incertain qui est présent dans les données [Besson *et al.*, 2006]. Les contraintes n-aires sont une façon naturelle de concevoir des motifs tolérants aux erreurs candidats à être des groupes de synexpression : ceux-ci sont définis par la combinaison de plusieurs motifs locaux satisfaisant la contrainte d'aire et ayant un fort recouvrement entre eux. Plus précisément, à partir de deux motifs locaux X et Y, on définit la contrainte n-aire suivante :

$$\texttt{area}(X) > min_{\texttt{area}} \wedge \texttt{area}(Y) > min_{\texttt{area}} \wedge (\texttt{coverage}(X,Y)^{\,3} > \alpha \times min_{\texttt{area}})$$

où, $min_{\texttt{area}}$ est le seuil d'aire minimale et α est un paramètre donné par l'utilisateur pour fixer le recouvrement minimal entre motifs locaux. Cette contrainte n-aire peut être étendue à k motifs ($k \geq 2$) comme suit :

$$synexpression([X_1, ..., X_k]) \equiv \begin{cases} \forall\ 1 \leq i < j \leq k, \\ \texttt{area}(X_i) > min_{\texttt{area}} \wedge \\ \texttt{area}(X_j) > min_{\texttt{area}} \wedge \\ \texttt{coverage}(X_i, X_j) > \alpha \times min_{\texttt{area}} \end{cases}$$

Dans la pratique, le biologiste est intéressé par les regroupements maximaux, ce qui correspond à la contrainte suivante :

$$synexpression([X_1, ..., X_k]) \equiv \begin{cases} \forall\ 1 \leq i < j \leq k, \\ \big(\texttt{area}(X_i) > min_{\texttt{area}} \wedge \\ \texttt{area}(X_j) > min_{\texttt{area}} \wedge \\ \texttt{coverage}(X_i, X_j) > \alpha \times min_{\texttt{area}}\big) \wedge \\ \forall \mathbf{Z} \textbf{ tel que } \texttt{freq}(\mathbf{Z}) > min_{\texttt{freq}} \textbf{ et } \texttt{area}(\mathbf{Z}) > min_{\texttt{area}}, \\ \exists \mathbf{i} \in [\mathbf{1..k}], \texttt{coverage}(\mathbf{Z}, \mathbf{X_i}) < \alpha \times min_{\texttt{area}} \end{cases}$$

2. Un concept formel associe un ensemble maximal d'items à un ensemble maximal de transactions. Un motif tolérant aux erreurs constitue une relaxation de cette association en cherchant à extraire des rectangles d'ensembles denses en valeur 1 mais acceptant aussi un nombre contrôlé de 0.

3. Le recouvrement entre deux motifs, dans ce cas, est défini par :
$\texttt{coverage}(X_i, X_j) = \texttt{freq}(X_i \cup X_j) \times \texttt{size}(X_i \cap X_j)$

2.2.4 Discussion

L'utilisation de contraintes n-aires permet de modéliser des motifs de plus haut niveau et de réduire ainsi considérablement la collection des motifs extraits. Néanmoins, les méthodes de la littérature sont dédiées à des classes particulières de contraintes (cf. section 2.2.2), voire dans certains cas à des requêtes précises (cf. section 2.2.3). Pour certaines requêtes telles que les motifs pics [Crémilleux et Soulet, 2008], il n'existe pas de méthode de résolution associée. Il n'existe pas non plus de méthodes génériques permettant de combiner des contraintes n-aires, ni des contraintes locales avec des contraintes n-aires. Ce constat forme une des motivations de notre travail.

2.3 Construction de modèles

2.3.1 Généralités

Les motifs extraits, notamment les motifs locaux, sont rarement utilisés comme produit final de l'extraction de motifs sous contraintes. Ces motifs forment généralement la source de méthodes de construction de modèles prédictifs (construction de classifieurs permettant de prédire la classe d'un nouvel objet non étiqueté) [Bringmann *et al.*, 2009], de modèle descriptifs (catégorisation des instances de la base de données appelée clustering ou classification non supervisée) [Durand et Cremilleux, 2002] ou encore la construction de résumés qui recense des techniques variées [Toivonen *et al.*, 1995, Mielikäinen et Mannila, 2003, Knobbe et Ho, 2006].

Il existe des cadres génériques pour la construction de modèles globaux à partir de motifs locaux [Knobbe *et al.*, 2008, Giacometti *et al.*, 2009], mais ces cadres ne proposent pas de méthode, ils permettent de mieux comparer et comprendre celles existantes. Dans ce contexte, E. Kahnjari Miyaneh a proposé un cadre pour la formalisation et la modélisation des modèles globaux fondés sur les motifs locaux (Pattern-Based Modeling framework, PBM) [Khanjari Miyaneh, 2009, Giacometti *et al.*, 2009]. Ce cadre permet de spécifier déclarativement la construction de modèles globaux de natures diverses (classifieurs, clustering ou résumés). Cette spécification est faite via les paramètres d'un algorithme générique appelé IGMA (Incremental Global Modeling Algorithm) qui ajoute itérativement le motif maximisant une mesure parmi un ensemble de candidats jusqu'à ce que le modèle couvre l'intégralité de la base de données.

2.3.2 Classification

2.3.2.a Classification non supervisée

La *classification non supervisée* ou *clustering* [Berkhin, 2006] consiste à faire émerger des groupes au sein d'un ensemble d'éléments sans aucune information à priori. Les groupes créés sont appelés clusters. L'objectif est l'identification de classes d'objets similaires au sein d'une collection d'objets donnée comme par exemple :

- Identification des clients qui ont des profils d'achat similaires,
- Identification de thèmes dans une collection de documents (applications en recherche d'information),
- etc.

Le clustering est une méthode non supervisée, car elle ne nécessite pas d'avoir identifié au préalable des classes correspondant aux différents clusters. Le clustering permet de

partitionner les données en sous-ensembles (ou groupes) tels que la similarité entre les données d'un même cluster et la dissimilarité les clusters soient les plus grandes possible.

Exemple 2.3.1.
Soit le jeu de données composé par les transactions suivantes :
$t_1 = \{A, B\}, t_2 = \{A, B\}, t_3 = \{A, C, D\}, t_4 = \{C, D\}$.
L'ensemble $\{AB, CD\}$ est un clustering couvrant l'intégralité des transactions du jeu de données et le divisant en deux groupes sans chevauchement.

2.3.2.b Classification supervisée

Lorsque les groupes (ou classes) existent à priori, on parle de classification supervisée. Les groupes sont étiquetés par une valeur de classe. L'objectif est alors de créer un modèle attribuant correctement la classe à de nouveaux éléments. La construction d'un tel modèle nécessite de disposer d'un ensemble d'exemples dont la classe est connue. Le modèle connu doit vérifier une capacité de généralisation, c'est à dire être capable de prédire correctement la classe de nouveaux exemples.

Dans ce manuscrit, nous nous intéressons particulièrement à la classification supervisée à base de règles d'association [Bringmann *et al.*, 2009]. Deux approches très classiques dans ce domaine sont `CBA` [Liu *et al.*, 1998] et `CMAR` [Li *et al.*, 2001]. Toutes deux procèdent selon le même schéma : tout d'abord l'ensemble des règles d'association valides (relativement à des seuils de fréquence et de confiance préalablement fixés) est extrait, puis les règles redondantes et les moins intéressantes sont retirées de cet ensemble. Enfin, le classifieur est formé par cet ensemble post-traité.

Exemple 2.3.2.
Soit le jeu de données composé par les transactions suivantes :
$t_1 = \{A, B, c_1\}, t_2 = \{A, B, c_1\}, t_3 = \{A, C, D, c_2\}, t_4 = \{C, D, c_2\}$.
Un classifieur à base règles possible est le suivant :
$\{AB \rightarrow c_1, ACD \rightarrow c_2, \ \rightarrow c_1\}$ *où la dernière règle permet de classifier les éléments ne correspondant à aucune des règles précédentes dans la classe par défaut c_1.*

2.4 Conclusion

Nous avons présenté dans ce chapitre les principales approches permettant de réduire le nombre de motifs extraits tout en gardant les plus pertinents.

Notons que d'autres travaux ont essayé de résumer 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]. À la différence de nombreuses approches d'extraction de motifs contraints, ces approches ne garantissent pas la complétude ni la correction de l'ensemble des motifs extraits. Pour cette raison, nous n'avons pas présenté ces approches dans ce chapitre.

Nous avons introduit dans la section 2.2 (page 33) la notion de contrainte n-aire, idée centrale des travaux réalisés dans cette thèse. Les contraintes n-aires permettent d'extraire des motifs de plus haut niveau en tirant profit de leur interdépendance. Remarquons que les motifs définis par les contraintes n-aires ne vérifient pas forcément des propriétés globales. Les règles d'exception sont un exemple typique de contrainte n-aire ne vérifiant pas de propriété globale sur la totalité du jeu de données.

Chapitre 3

Problèmes de Satisfaction de Contraintes

Sommaire

Un CSP (Problème de Satisfaction des Contraintes) est un triplet $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ où $\mathcal{X}$ est un ensemble de variables, $\mathcal{D}$ est l'ensemble de leurs domaines respectifs, et $\mathcal{C}$ est un ensemble de contraintes restreignant les valeurs que peuvent simultanément prendre les variables. Chaque variable $X_i \in \mathcal{X}$ possède un ensemble de valeurs possibles $D_i \in \mathcal{D}$ (domaine) qui est de cardinalité finie.

Résoudre un CSP consiste à déterminer s'il existe au moins une instanciation des variables de $\mathcal{X}$ (par les valeurs de leur domaine) qui satisfasse simultanément toutes les contraintes. Les CSP sont le plus souvent résolus par des méthodes de recherche arborescente où des sous-arbres sont élagués par filtrage. Le filtrage consiste à déterminer certaines valeurs ne pouvant apparaître dans une solution et à les retirer des domaines correspondants. Les sous-arbres associés ne seront alors pas développés. La grande majorité des méthodes de recherche arborescente pour les CSP sont sûres (on n'obtient que des solutions) et complètes (on obtient toutes les solutions).

Ce chapitre est dédié aux différents cadres CSP que nous avons utilisés pour modéliser l'extraction de motifs. Tout d'abord, nous présentons le cadre général des CSP en

introduisant les principales définitions et en portant une attention toute particulière au filtrage : nous étudierons l'Arc-Cohérence et la Cohérence aux Bornes. Puis, nous nous intéresserons au cadre particulier des CSPE (CSP Ensemblistes). Enfin, nous introduirons les QCSP (CSP Quantifiés) qui est une extension du cadre CSP.
Ce chapitre peut être consulté de dux manières différentes :

- Grille de lecture verticale : les CSP en général (Section 3.1), puis le cas particulier des CSPE (Section 3.2), et enfin les QCSP généralisation des CSP (Section 3.3).
- Grille de lecture horizontale : le cadre général des CSP sera utilisé dans tous les chapitres "contributions", i.e. les chapitres 4, 5, 6 et 7. Les CSPE seront utilisés dans le chapitre 5 décrivant l'approche *Hybride*. Les CSP Quantifiés seront utilisés dans le Chapitre 7.

3.1 Problèmes de Satisfaction de Contraintes

Dans cette section, nous définissons les principales notions relatives aux CSP. Puis, nous nous intéressons au filtrage par Arc-Cohérence et au filtrage par Cohérence aux bornes.

3.1.1 Définitions

Définition 3.1.1 (CSP). *Un CSP est un triplet $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ tel que :*

- $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ *est l'ensemble des* ***variables***,
- $\mathcal{D} = \{D_1, D_2, ..., D_n\}$ *est l'ensemble des* ***domaines des variables***. *Chaque domaine D_i est un ensemble fini contenant les valeurs possibles pour la variable x_i.*
- $\mathcal{C} = \{c_1, c_2, ..., c_e\}$ *est l'ensemble des* ***contraintes***.

Chaque contrainte c_i porte sur un sous-ensemble de variables de $\mathcal{X}$ appelé $var(c_i)$. L'arité (nombre de variables) de c_i est définie par $|var(c_i)|$.

Définition 3.1.2 (instanciation). *Une* ***instanciation élémentaire*** *est un couple variable-valeur (x_i, v) tel que $v \in D_i$. Une* ***instanciation*** *est un ensemble d'instanciations élémentaires. Si une instanciation porte sur toutes les variables alors on parle d'****instanciation complète****, autrement on parle d'****instanciation partielle****.*

Définition 3.1.3 (solution). *Une* ***solution*** *d'un CSP est une instanciation complète S telle que toutes les contraintes de P sont satisfaites par S.*

Exemple 3.1.1. *On considère le CSP suivant :*

- $\mathcal{X} = \{x_1, x_2, x_3\}$,
- $\mathcal{D} = \{D_1, D_2, D_3\}$ *avec* $D_1 = D_2 = D_3 = \{1, 2, 3\}$,
- $\mathcal{C} = \{c_1, c_2\}$ *avec* $c_1 = [x_1 < x_2]$ *et* $c_2 = [x_2 = x_3]$.

Alors :

$var(c_1) = \{x_1, x_2\}$

$I_1 = \{(x_1, 1), (x_2, 2)\}$, $I_2 = \{(x_1, 1), (x_2, 3)\}$ *et* $I_3 = \{(x_1, 2), (x_2, 3)\}$ *sont trois instanciations partielles satisfaisant c_1.*

$var(c_2) = \{x_2, x_3\}$

$I'_1 = \{(x_2, 1), (x_3, 1)\}$, $I'_2 = \{(x_2, 2), (x_3, 2)\}$ *et* $I'_3 = \{(x_2, 3), (x_3, 3)\}$ *sont trois instanciations partielles satisfaisant c_2.*

Il existe trois solutions (instanciations complètes satisfaisant c_1 et c_2) :

- $S_1 = \{(x_1, 1), (x_2, 2), (x_3, 2)\}$,
- $S_2 = \{(x_1, 1), (x_2, 3), (x_3, 3)\}$,
- $S_3 = \{(x_1, 2), (x_2, 3), (x_3, 3)\}$

3.1.2 Les contraintes réifiées

La réification d'une contrainte C correspond à la formule logique suivante :

$$(C \Longleftrightarrow x = 1) \quad avec \quad D_x = \{0, 1\} \quad et \quad x \notin var(C)$$

Les contraintes réifiées sont donc des contraintes associant une variable booléenne à la satisfaction d'une contrainte. La résolution s'effectue comme suit :

- si la contrainte $x = 1$ est satisfaite alors la contrainte C doit être satisfaite,
- si la contrainte $x = 0$ est satisfaite alors la contrainte $\neg C$ doit être satisfaite,
- si la contrainte C est satisfaite alors la contrainte $x = 1$ doit être satisfaite,
- si la contrainte C n'est pas satisfaite alors la contrainte $x = 0$ doit être satisfaite.

Ces contraintes offrent une très grande puissance de modélisation et la plupart des solveurs de CSP permettent de modéliser des problèmes à l'aide de ce type de contraintes.

Ces contraintes nous seront utiles pour l'approche *Pure-CP* (cf. chapitre 6).

3.1.3 Filtrage par Arc-Cohérence

Les CSP sont le plus souvent résolus par des méthodes de recherche arborescente où des sous-arbres sont élagués par filtrage. Le filtrage consiste à identifier certaines valeurs ne pouvant apparaître dans une solution et à les retirer des domaines correspondants. Les sous-arbres associés ne seront alors pas développés par l'algorithme de recherche. Ainsi, le filtrage permet d'obtenir un nouveau CSP équivalent au CSP de départ (ils possèdent exactement les mêmes solutions), mais dont la taille de l'arbre de recherche sera plus petite.

Enfin, les algorithmes de filtrage doivent être de complexité polynomiale faible. En effet, le temps gagné (en évitant de parcourir certains sous-arbres inutiles) doit être supérieur au temps nécessaire pour réaliser les filtrages successifs. Le filtrage le plus communément utilisé est le filtrage par Arc-Cohérence (AC). Sa complexité temporelle, dans le pire cas, est en $O(e \times d^2)$, avec e nombre de contraintes et d taille du plus grand domaine.

Définition 3.1.4 (support d'une valeur pour une contrainte)**.** *Soit c une contrainte binaire portant telle que : $var(c) = \{x_1, x_2\}$. La valeur (x_j, v_j) est un **support** de la valeur (x_i, v_i) pour la contrainte c ssi (v_i, v_j) satisfait c.*

Définition 3.1.5 (viabilité d'une valeur)**.** *Une valeur est **viable** ssi elle possède au moins un support pour chacune des contraintes portant sur elle.*

Définition 3.1.6 (Arc-Cohérence d'un CSP)**.** *Un CSP est arc-cohérent ssi aucun domaine n'est vide et si toutes les valeurs des domaines sont viables.*

Exemple 3.1.2. *Le filtrage par arc-cohérence du CSP de l'exemple 3.1.1 conduit aux domaines suivants :*

- $D_1 = \{1, 2\}$,
- $D_2 = \{2, 3\}$,
- $D_3 = \{2, 3\}$.

```
Fonction Revise(x_i, x_j, D_i) : booléen
    éliminé ← Faux ;
    Pour tout v ∈ D_i faire
        Si (∄ v_j ∈ D_j t.q. (v, v_j) satisfait c_{i,j}) Alors
            D_i ← D_i\{v} ;
            éliminé ← Vrai ;
        Fin Si
    Fin Pour
    Retourner éliminé ;
Fin
```

Algorithme 2: `Revise` (cas des contraintes binaires)

De nombreux algorithmes de filtrage ont été proposés pour rendre un réseau de contraintes arc-cohérent. `AC-3` fut l'une des premières méthodes proposées [Mackworth, 1977]. La fonction `Revise`(x_i, x_j, D_i) (cf. algorithme 2) retire du domaine D_i de la variable x_i les valeurs sans support pour la contrainte $c_{i,j}$. La complexité temporelle de `Revise` est en $O(d^2)$ dans le pire cas.

`AC-3` (cf. algorithme 3) gère un ensemble Q de contraintes à traiter. Initialement, Q contient tous les couples $(c_{i,j}, x_i)$. Tant que Q n'est pas vide, un même traitement est effectué : le premier couple $(c_{i,j}, x_i)$ de Q est sélectionné ; puis, on tente de réduire le domaine D_i de la variable x_i à l'aide de la fonction `Revise`. Si D_i a été modifié par `Revise`, alors tous les couples $(c_{i,k})$, avec $k \neq j$, non présents dans Q, sont ajoutés. Si le domaine d'une variable devient vide, alors `AC-3` retourne `Echec` signalant l'absence de solution. Lorsque Q devient vide, l'algorithme s'arrête et le réseau de contraintes est arc-cohérent. `AC-3` maintient la cohérence d'arc en $O(e \times d^3)$.

```
Procédure AC-3(X, D, C)
    Soit Q = {(c_{i,j}, x_i) t.q. c_{i,j} ∈ C, x_i ∈ X}
    Tant que (Q ≠ ∅) faire
        Extraire (c_{i,j}, x_i) de Q ;
        Si (Revise(x_i, x_j, D_i)) Alors
            Si (D_i = ∅) Alors                                  ;
                Retourner Echec
            Fin Si
            Q ← Q ∪ {(c_{k,i}, x_k) t.q. k ≠ j} ;
        Fin Si
    Fait
    Retourner Succes ;
Fin
```

Algorithme 3: `AC-3` (cas des contraintes binaires)

Plusieurs améliorations de `AC-3` ont été proposées :

- `AC-4` [Mohr et Henderson, 1986] a été la première méthode permettant d'établir l'AC en temps optimal dans le pire cas, i.e. en $O(e \times d^2)$. `AC-4` calcule, tout d'abord,

tous les supports de toutes les valeurs ($O(e \times d^2)$), puis effectue le filtrage proprement dit en $O(e \times d^2)$.
- `AC-6` [Bessière et Cordier, 1993] a permis de diminuer la complexité en moyenne en gérant de manière paresseuse les supports. Au lieu de calculer l'ensemble de tous les supports pour toutes les valeurs, `AC-6` s'assure seulement qu'il existe au moins un support pour chaque valeur. Lorsqu'un tel support est retiré `AC-6` ne recherche plus un nouveau support à partir du début du domaine, mais à partir de celui venant d'être retiré.
- `AC-2001` [Bessière et Régin, 2001] a encore amélioré la complexité en moyenne en stockant, dans une structure de données additionnelle, le dernier support obtenu et en testant, toujours en premier, la présence de celui-ci.

3.1.4 Filtrage par Cohérence aux Bornes

Lorsque les domaines des variables sont, par exemple, des intervalles de flottants, il n'est pas raisonnable de vouloir établir l'AC. En effet, les domaines étant de très grande taille, le temps de calcul nécessaire au filtrage par AC serait prohibitif. Dans ce cas, on préfère établir une propriété plus faible, la cohérence aux bornes [Lhomme, 1993].

Définition 3.1.7 (Cohérence aux Bornes d'une contrainte). *Une contrainte c d'arité k est cohérente aux bornes ssi*
Pour toute variable $x_i \in var(c)$ *de domaine* $D_i = [\mathbf{a}, \mathbf{b}]$ *:*
- $\exists a_1 \in D_1, ..., a_{i-1} \in D_{i-1}, a_{i+1} \in D_{i+1}, ..., a_k \in D_k$, *t.q.* $(a_1, ..., a_{i-1}, \mathbf{a}, a_{i+1}, ...a_k)$ *satisfait c,*
- $\exists b_1 \in D_1, ..., b_{i-1} \in D_{i-1}, b_{i+1} \in D_{i+1}, ..., b_k \in D_k$, *t.q.* $(b_1, ..., b_{i-1}, \mathbf{b}, b_{i+1}, ...b_k)$ *satisfait c.*

Définition 3.1.8 (Cohérence aux Bornes d'un CSP). *Un CSP* $(\mathcal{X},\mathcal{D},\mathcal{C})$ *est cohérent aux bornes ssi toute contrainte* $c \in \mathcal{C}$ *est cohérente aux bornes.*

3.1.4.a Exemples de contraintes binaires

Nous considérons deux exemples de contraintes binaires pour lesquelles nous exprimons le test de cohérence aux bornes ainsi que le filtrage associé.

1. Soit la contrainte $x_1 \leq x_2$ où x_1 et x_2 sont deux variables de domaines respectifs $D_1 = [lb_1 .. ub_1]$ et $D_2 = [lb_2 .. ub_2]$.
   Cette contrainte est cohérente aux bornes ssi $lb_1 \leq ub_2$
   Dans ce cas, après filtrage, on a :
   $D'_1 = [lb_1 .. min(ub_1, ub_2)]$ et $D'_2 = [max(lb_1, lb_2) .. ub_2]$

   Exemple 3.1.3.
   D_1=[1..5] *et* D_2=[0..7] *alors* D'_1=[1..5] *et* D'_2=[1..7]
2. Soit la contrainte $x_1 = x_2$ où x_1 et x_2 sont deux variables de domaines respectifs $D_1 = [lb_1 .. ub_1]$ et $D_2 = [lb_2 .. ub_2]$.
   Cette contrainte est cohérente aux bornes ssi $lb_1 \leq ub_2$.
   Dans ce cas, après filtrage, on a :
   $D'_1 = D'_2 = [max(lb_1, lb_2) .. min(ub_1, ub_2)]$.

Exemple 3.1.4.

D_1=[1..5] et D_2=[0..3] alors D'_1=D'_2=[1..3]

3.1.4.b Un exemple de contrainte ternaire

On utilise l'arithmétique des intervalles définie par Allen en 1983 [Allen, 1983] :
- $[a,b] + [c,d] = [a+c, b+d]$
- $[a,b] - [c,d] = [a-d, b-c]$
- $[a,b] \times [c,d] = [min(a.c, a.d, b.c, b.d), max(a.c, a.d, b.c, b.d)]$
- $[a,b]/[c,d] = [min(a/c, a/d, b/c, b/d), max(a/c, a/d, b/c, b/d)]$ si $0 \notin [c,d]$

Propriétés :

- Les opérations + et × sont commutatives et associatives.
- × est sous-distributive par rapport à + : soient I, J, K des intervalles, alors $I \times (J+K) \subset (I \times J) + (I \times K)$, la réciproque étant fausse.

Soit la contrainte $x_1 = x_2 + x_3$ où x_1, x_2 et x_3 sont trois variables de domaines respectifs $D_1 = [lb_1 .. ub_1]$, $D_2 = [lb_2 .. ub_2]$ et $D_3 = [lb_3 .. ub_3]$.
Considérons les trois projections :

- p_1 : $x_1 = x_2 + x_3$
- p_2 : $x_2 = x_1 - x_3$
- p_3 : $x_3 = x_1 - x_2$

En appliquant l'arithmétique des intervalles successivement sur p_1, p_2 et p_3, on obtient les domaines réduits suivants :

- $D'_1 = [lb_2 + lb_3 .. ub_2 + ub_3] \cap D_1$
- $D'_2 = [lb_1 - ub_3 .. ub_1 - lb_3] \cap D_2$
- $D'_3 = [lb_1 - ub_2 .. ub_1 - lb_2] \cap D_3$

Exemple 3.1.5.

Soient D_1=[4..8], D_2=[0..3] et D_3=[2..2]. Par filtrage aux bornes, on obtient :

- $D'_1 = [2..5] \cap [4..8] = [4..5]$
- $D'_2 = [2..6] \cap [0..3] = [2..3]$
- $D'_3 = [1..8] \cap [2..2] = [2..2]$

3.1.4.c Algorithme de filtrage

L'algorithme de filtrage par cohérence aux bornes est dû à Waltz en 1972 [Waltz, 1972].

```
Procédure BC(𝒳, 𝒟, 𝒞)
    Soit Q l'ensemble des contraintes
    Tant que (Q est non vide) faire
        extraire une contrainte c de Q ;
        VarModifiees := ReviseBC(c, D) ;
        Pour tout X_i ∈ VarModifiees faire
            Pour pour toute contrainte c' ≠ c tq X_i ∈ var(c') faire
                Q := Q ∪ {c'} ;
            Fin Pour
        Fin Pour
    Fait
Fin
```

Algorithme 4: `Cohérence aux bornes` (cas des contraintes n-aires)

```
Fonction ReviseBC(c, D) : booléen
    soit VarModifiees = ∅ ;
    Pour tout X_i ∈ var(c) faire
        D'_i := Narrowing(c, X_i, D_i) ;
        Si (D'_i est vide) Alors
            Retourner Echec
        Sinon
            Si (D_i ≠ D'_i) Alors
                VarModifiees := VarModifiees ∪{X_i} ;
            Fin Si
        Fin Si
    Fin Pour
    Retourner VarModifiees ;
Fin
```

Algorithme 5: `ReviseBC` (cas des contraintes n-aires)

```
Fonction Narrowing(c, X_i, D_i) : Domaine
    soit p_i la projection de de X − i sur X_1, ..., X_{i−1}, ..., X_{i+1}, ..., X_n ;
    D_i = p_i(D_1, ..., D_{i−1}, ..., D_{i+1}, ..., D_n) ∪ D_i ;
    Retourner D_i ;
Fin
```

Algorithme 6: `Fonction Narrowing` (cas des contraintes n-aires)

3.1.5 Résolution d'un CSP

Les méthodes de recherche arborescente reposent généralement sur le principe du *backtrack* [Golomb et Baumert, 1965] associé à des méthodes de filtrage permettant

d'élaguer l'arbre de recherche. De telles méthodes de recherche arborescente pour les CSP sont sûres (on n'obtient que des solutions) et complètes (on obtient toutes les solutions). Dans cette Section, nous rappelons le parcours basé sur le retour arrière chronologique ainsi que le parcours avec Maintien de l'Arc-Cohérence (MAC [Sabin et Freuder, 1994, Bessière et Régin, 1996]) qui réalise un filtrage par AC à chaque nœud de l'arbre.

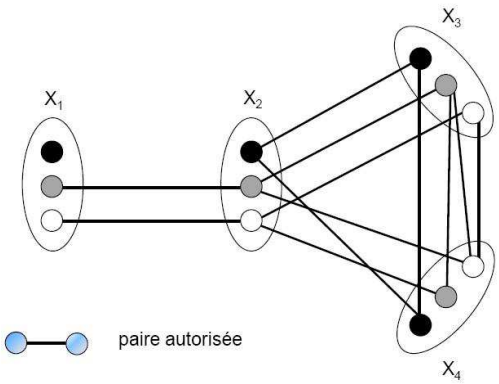

FIGURE 3.1 – Graphe de cohérence

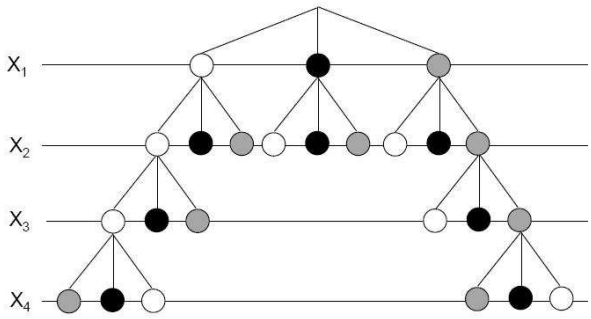

FIGURE 3.2 – Arbre de recherche parcouru par backtrack

3.1.5.a Backtrack chronologique

Les variables du problème sont instanciées avec les valeurs de leur domaine, dans un ordre prédéfini, jusqu'à ce que l'un de ces choix viole une contrainte. La dernière instanciation effectuée est alors remise en cause. Une nouvelle valeur est essayée pour la dernière variable instanciée (variable courante). Si toutes les valeurs du domaine de cette variable été visitées, on doit procéder à un retour en arrière. Pour cela, une nouvelle valeur est choisi pour la variable précédant immédiatement la variable courante. Ce processus est répété jusqu'à obtenir une solution, c'est à dire une instanciation de toutes les variables ne violant aucune contrainte. Si on a parcouru tout l'arbre de recherche sans en trouver, alors le problème ne possède aucune solution.

Exemple 3.1.6. *Soit le CSP décrit par son graphe de cohérence*[1] *à la figure 3.1*[2]*. L'arbre de recherche par Backtrack chronologique est décrit par la figure 3.2.*

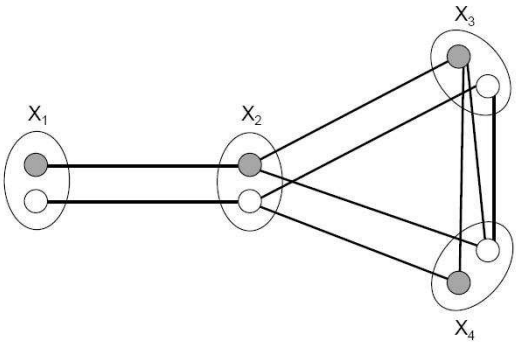

FIGURE 3.3 – Graphe de cohérence après filtrage par AC

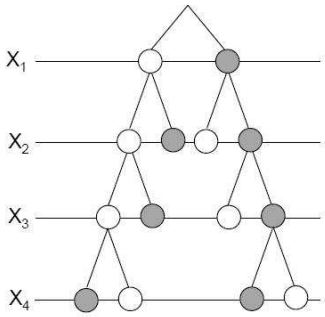

FIGURE 3.4 – Arbre de recherche parcouru par MAC

3.1.5.b Maintien d'arc-cohérence

Supposons qu'en cherchant une solution, Chronological Backtracking donne à une variable x_i une valeur v qui exclut toutes les valeurs possibles pour une variable x_j. L'algorithme Chronological Backtracking ne s'en rendra compte que lorsque x_j sera considérée. De plus, avec un retour arrière chronologique, avant que x_i ne soit reconsidérée, il se peut qu'une grande partie de l'espace de recherche soit explorée en vain. Ceci pourrait être évité en prenant en compte le fait que (x_i, v) ne pourra jamais faire partie d'une solution, puisqu' il n'y a aucune valeur de x_j qui soit compatible avec (x_i, v). Utiliser le filtrage permet d'éviter la situation décrite ci-dessus. Une valeur v n'est acceptée qu'après avoir vérifié les domaines des variables futures. S'il existe un domaine qui ne contient que des valeurs incompatibles avec v, alors v est éliminée. Par ailleurs, le filtrage peut être utilisé après chaque instanciation (x_i, v). En effet, les valeurs des variables futures qui ne sont pas compatibles avec (x_i, v) peuvent être éliminées sans perte de solutions.

1. le graphe de cohérence est un graphe dont les sommets sont les valeurs des domaines, et dont les arcs relient les sommets représentant des valeurs compatibles.

2. Ce CSP n'ayant qu'une unique solution (gris, gris, gris, blanc), on a choisi un ordre de parcours des valeurs de manière à explorer l'integralité de l'arbre de recherche pour trouver cette solution.

Les algorithmes les plus connus sont : Forward Checking [Haralick et Elliot, 1980] et Maintaining Arc Concistency (MAC) [Sabin et Freuder, 1994, Bessière et Régin, 1996]. L'algorithme de maintien d'arc cohérence MAC procède au au filtrage par AC après chaque instanciation de variable.

3.2 CSPs ensemblistes

3.2.1 Définitions

Nous présentons dans cette section un cadre particulier de CSPs, les CSPs ensemblistes ou CSPE. Un CSPE est un CSP dans lequel une variable peut être instanciée par un ensemble de valeurs. Dans le cas des CSPE, les domaines des variables sont représentés par des intervalles ensemblistes.

Définition 3.2.1 (Intervalle ensembliste)**.** *Soient lb et ub deux ensembles tels que* $lb \subseteq ub$*, l'intervalle* $[lb..ub] = \{E \text{ tel que } lb \subseteq E \text{ et } E \subseteq ub\}$ *est un intervalle ensembliste.*

Exemple 3.2.1.
$[\{1\}..\{1,2,3\}] = \{\{1\},\{1,2\},\{1,3\},\{1,2,3\}\}$
$[\{\}..\{1,2,3\}] = \{\{\},\{1\},\{1,2\},\{1,3\},\{2,3\},\{1,2,3\}\}$

Définition 3.2.2 (CSP ensembliste)**.** *Un CSP Ensembliste (CSPE) est un triplet* $(\mathcal{X},\mathcal{D},\mathcal{C})$ *tel que :*
- $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ *est l'ensemble des variables,*
- $\mathcal{D} = \{D_1, D_2, ..., D_n\}$ *est l'ensemble des domaines des variables. Chaque domaine* D_i *est* **un intervalle ensembliste**.
- $\mathcal{C} = \{c_1, c_2, ..., c_e\}$ *est l'ensemble des* ***contraintes***.

3.2.2 Exemple

Le problème consiste à affecter des fréquences radio à deux émetteurs t_1 et t_2. Les fréquences disponibles pour t_1 sont : $\{1,2,3,4\}$. Les fréquences disponibles pour t_2 sont : $\{3,4,5,6\}$. La distance entre deux fréquences est égale à la valeur absolue de la différence entre ces fréquences. Les contraintes suivantes doivent être satisfaites :

(c_1) à chaque émetteur doivent être affectées exactement deux fréquences,

($c2$) les deux émetteurs ne peuvent avoir de fréquence commune,

($c3$) les deux fréquences d'un même émetteur doivent avoir une distance supérieure ou égale à 2,

($c4$) la fréquence 3 doit être allouée à l'émetteur t_1,

($c5$) la fréquence 4 doit être allouée à l'émetteur t_2.

Ce problème est modélisé par le CSPE suivant :
- $\mathcal{X} = \{x_1, x_2\}$ où x_1 (resp. x_2) représente l'ensemble des fréquences allouées à l'émetteur t_1 (resp. t_2).
- $D_1 = [\emptyset \,..\, \{1,2,3,4\}]$ et $D_2 = [\emptyset \,..\, \{3,4,5,6\}]$.
- $\mathcal{C} = \{c_1, c_2, c_3, c_4, c_5\}$ avec :
  (c_1) $\quad \mid x_1 \mid = 2 \wedge \mid x_2 \mid = 2$
  (c_2) $\quad x_1 \cap x_2 = \emptyset$
  (c_3) $\quad \forall v_1, v_2 \in x_i, \quad \mid v_1 - v_2 \mid \geq 2 \quad i = 1,2$

(c_4) $\quad 3 \in x_1$
(c_5) $\quad 4 \in x_2$

Ce CSPE possède une unique solution : $\{x_1 = \{1,3\},\ x_2 = \{4,6\}\}$

3.2.3 Cohérence aux Bornes pour les CSPE

Les intervalles ensemblistes pouvant être de très grande taille, le filtrage réalisé établit la cohérence aux bornes [Gervet, 1997].

Définition 3.2.3 (Cohérence aux Bornes d'une contrainte ensembliste)**.** *Une contrainte ensembliste c d'arité k est cohérente aux bornes ssi*
Pour toute variable $x_i \in var(c)$ de domaine $D_i = [\mathbf{a}, \mathbf{b}]$:

- $\exists a_1 \in D_1, ..., a_{i-1} \in D_{i-1}, a_{i+1} \in D_{i+1}, ..., a_k \in D_k$, *t.q.* $(a_1, ..., a_{i-1}, \mathbf{a}, a_{i+1}, ...a_k)$ *satisfait c,*
- $\exists b_1 \in D_1, ..., b_{i-1} \in D_{i-1}, b_{i+1} \in D_{i+1}, ..., b_k \in D_k$, *t.q.* $(b_1, ..., b_{i-1}, \mathbf{b}, b_{i+1}, ...b_k)$ *satisfait c.*

Définition 3.2.4 (Cohérence aux Bornes d'un CSP Ensembliste)**.** *Un CSPE ($\mathcal{X}$,$\mathcal{D}$,$\mathcal{C}$) est cohérent aux bornes ssi toute contrainte $c \in \mathcal{C}$ est cohérente aux bornes.*

Ces deux définitions sont analogues à celles données pour les intervalles numériques (cf. section 3.1.4). Dans le cas des CSPE, le filtrage procède également par mise à jour des bornes inférieures et supérieures des intervalles, sachant que désormais : $min(a, b) = a \cap b$ et $max(a, b) = a \cup b$.

3.2.3.a Exemples de contraintes binaires

Nous considérons deux exemples de contraintes binaires ensemblistes pour lesquelles nous exprimons le test de cohérence aux bornes ainsi que le filtrage associé.

1. Soit la contrainte $x_1 \subseteq x_2$ où x_1 et x_2 sont deux variables ensemblistes de domaines respectifs $D_1 = [lb_1 .. ub_1]$ et $D_2 = [lb_2 .. ub_2]$.
   Cette contrainte est cohérente aux bornes ssi $lb_1 \subseteq ub_2$.
   Dans ce cas, après filtrage, on a :
   $D'_1 = [lb_1 .. ub_1 \cap ub_2]$ et $D'_2 = [lb_1 \cup lb_2 .. ub_2]$

   Exemple 3.2.2.
   $D_1 = [AB..ABCD], D_2 = [A..ABC]$ *alors* $D'_1 = [AB..ABC]$ *et* $D'_2 = [AB..ABC]$

2. Soit la contrainte $x_1 = x_2$ où x_1 et x_2 sont deux variables ensemblistes de domaines respectifs $D_1 = [lb_1 .. ub_1]$ et $D_2 = [lb_2 .. ub_2]$.
   Cette contrainte est cohérente aux bornes ssi $lb_1 \subseteq ub_2$.
   Dans ce cas, après filtrage, on a :
   $D'_1 = D'_2 = [lb_1 \cup lb_2 .. ub_1 \cap ub_2]$.

   Exemple 3.2.3. $D_1 = [AB..ABCD], D_2 = [A..ABC]$ *alors* $D'_1 = D'_2 = [AB..ABC]$

3.2.3.b Exemples de contraintes ternaires

1. $Z = X \cup Y$.
   Soit $D_x = [a_x, b_x]$, $D_y = [a_y, b_y]$ et $D_z = [a_z, b_z]$
   Cette contrainte est cohérente aux bornes ssi $(a_x \cup a_y) \subseteq b_z$
   Dans ce cas, après filtrage, on a :
   $D'_x = [a_x \cup (a_z \setminus b_y) \,..\, b_x \cap b_z]$
   $D'_y = [a_y \cup (a_z \setminus b_x) \,..\, b_y \cap b_z]$
   $D'_z = [a_z \cup a_x \cup a_y \,..\, b_z \cap (b_x \cup b_y)]$

   Exemple 3.2.4.
   Soient les variables X, Y et Z ayant les domaines suivants :
   $D_x = [A \,..\, ABCD]$
   $D_y = [B \,..\, ABEF]$
   $D_z = [D \,..\, ABDE]$

   et la contrainte $Z = X \cup Y$.
   Après filtrage, on obtient les domaines suivants :
   $D'_x = [AD \,..\, ABD]$
   $D'_y = [B \,..\, ABE]$
   $D'_z = [ABD \,..\, ABDE]$

2. $Z = X \cap Y$.
   Soit $D_x = [a_x, b_x]$, $D_y = [a_y, b_y]$ et $D_z = [a_z, b_z]$
   Cette contrainte est cohérente aux bornes ssi $(b_x \cap b_y) \subseteq b_z$ et $(b_x \cap b_y) \neq \emptyset$
   Dans ce cas, après filtrage, on a :
   $D'_x = [a_x \cup a_z \,..\, b_x \setminus ((b_x \cap a_y) \setminus b_z)]$
   $D'_y = [a_y \cup a_z \,..\, b_y \setminus ((b_y \cap a_x) \setminus b_z)]$
   $D'_z = [a_z \cup (a_x \cap a_y) \,..\, b_z \cap b_x \cap b_y]$

   Exemple 3.2.5.
   Soient les variables X, Y et Z ayant les domaines suivants :
   $D_x = [A \,..\, ABCD]$
   $D_y = [B \,..\, ABDEF]$
   $D_z = [D \,..\, ABDE]$

   et la contrainte $Z = X \cap Y$.
   Après filtrage, on obtient les domaines suivants :
   $D'_x = [AD \,..\, ABCD]$
   $D'_y = [BD \,..\, ABDEF]$
   $D'_z = [D \,..\, ABD]$

3. $Z = X \setminus Y$.
   Soit $D_x = [a_x, b_x]$, $D_y = [a_y, b_y]$ et $D_z = [a_z, b_z]$
   Cette contrainte est cohérente aux bornes ssi $(b_x \setminus a_y) \subseteq b_z$
   Dans ce cas, après filtrage, on a :

$D'_x = [a_x \cup a_z \;..\; b_x \setminus (b_x \setminus (b_z \cup b_y))]$
$D'_y = [a_y \;..\; b_y \setminus a_z]$
$D'_z = [a_z \cup (a_x \setminus b_y) \;..\; b_z \cap (b_x \setminus a_y)]$

Exemple 3.2.6.
Soient les variables X, Y et Z ayant les domaines suivants :
$D_x = [A \;..\; ABCD]$
$D_y = [B \;..\; ABDEF]$
$D_z = [D \;..\; BDE]$

et la contrainte $Z = X \setminus Y$.
Après filtrage, on obtient les domaines suivants :
$D'_x = [AD \;..\; ABD]$
$D'_y = [B \;..\; ABEF]$
$D'_z = [D \;..\; D]$

3.2.3.c Autres contraintes

1. $X \cap Y = \emptyset$.
   Soit $D_x = [a_x, b_x]$ et $D_y = [a_y, b_y]$
   Cette contrainte est cohérente aux bornes ssi $a_x \cap a_y = \emptyset$
   Dans ce cas, après filtrage, on a : $D'_x = [a_x \;..\; b_x \setminus a_y))]$ et $D'_y = [a_y \;..\; b_y \setminus a_x]$

2. $m \subseteq X$.
   Soit m un ensemble et $D_x = [a_x, b_x]$
   Cette contrainte est cohérente aux bornes ssi $m \subseteq b_x$
   Dans ce cas, après filtrage, on a : $D'_x = [a_x \cup m, b_x]$

   Exemple 3.2.7.
   $m = AC$ et $D_x = [AB..ABCD]$ alors $D'_x = [ABC, ABCD]$

3. $m \nsubseteq X$.
   Soit m un ensemble et $D_x = [a_x, b_x]$
   Cette contrainte est cohérente aux bornes ssi $m \nsubseteq a_x$
   Dans ce cas, après filtrage, on a : $D'_x = [a_x,\; b_x \setminus (m \setminus a_x)]$

   Exemple 3.2.8.
   $m = AC$ et $D_x = [AB..ABCD]$ alors $D'_x = [AB, ABD]$

3.2.4 Union ensembliste et enveloppe convexe

L'enveloppe convexe de 2 intervalles I_1 et I_2 est le plus petit intervalle contenant à la fois I_1 et I_2.

Définition 3.2.5 (Enveloppe convexe de deux intervalles ensemblistes)**.** *Soient deux intervalles $I_1 = [lb_1, ub_1]$ et $I_2 = [lb_2, ub_2]$.*
L'enveloppe convexe des intervalles I_1 et I_2 est l'intervalle ensembliste $[lb_1 \cap lb_2 \;..\; ub_1 \cup ub_2]$.

Définition 3.2.6 (Enveloppe convexe de deux intervalles numériques)**.** *Soient deux intervalles* $I_1 = [lb_1, ub_1]$ *et* $I_2 = [lb_2, ub_2]$.
L'enveloppe convexe des intervalles I_1 *et* I_2 *est l'intervalle numérique* $[min(lb_1, lb_2)$ .. $max(ub_1, ub_2)]$.

L'union de 2 intervalles I_1 et I_2, qu'ils soient numériques ou ensemblistes, n'est pas forcément un intervalle. Si I_1 et I_2 sont disjoints, alors leur union ne forme pas un intervalle. Dans ce cas, les solveurs de contraintes **approximent cette union par l'enveloppe convexe** de I_1 et I_2,
Exemple : soit $I_1 = [1..3]$ et $I_2 = [4..9]$ alors $I_1 \cup I_2 = [1..3] \cap [4..9]$ et leur enveloppe convexe est l'intervalle [1..9].

Approximer l'union de deux intervalles ensemblistes par leur enveloppe convexe sera au centre de la discussion autour des performances de l'une des approches d'extraction de motifs sous contraintes n-aires que nous proposons dans ce manuscrit (cf. section 5.6).

3.3 CSPs Quantifiés

3.3.1 Présentation

Les problèmes de satisfaction de contraintes permettent de modéliser et de résoudre des problèmes de la forme "Existe-t-il un moyen pour moi de satisfaire un ensemble de conditions ?". Or souvent, on doit résoudre des problèmes qui s'expriment de manière plus complexe. Un exemple simple de tels problèmes est issu de la théorie des jeux : déterminer si un jeu à deux joueurs possède une stratégie gagnante peut s'exprimer de la manière suivante :

```
Existe-il un coup tel que,
pour tout coup adverse,
 il existe un coup tel que,
     pour tout coup adverse,...    je gagne?
```

L'exemple précédent illustre le besoin des contraintes quantifiées, comme c'est le cas pour certaines contraintes traitées dans ce mémoire (cf. chapitre 7). Néanmoins, les QCSP sont considérablement plus difficiles à résoudre que les CSP. En effet, une solution d'un tel problème doit exprimer chaque coup à jouer en fonction de toutes les séries de coups possibles de l'adversaire : par exemple, même en supposant que ledit adversaire n'ait à chaque coup que deux possibilités d'action, et que le jeu se déroule en k tours : il faut alors connaître, un moyen de répondre à chacun des 2^k scénarios que cela engendre.

Plusieurs travaux se sont intéressés à ce problème dont certains ont proposé des généralisations des techniques connues pour les CSP tel que l'arc cohérence, etc. [Bordeaux et Monfroy, 2003, Benedetti *et al.*, 2006, Verger et Bessière, 2006].

3.3.2 QCSP

Partons de la formule logique associée à un CSP ayant pour variables $x_1, x_2, ..., x_n$:

$$\exists x_1 \in D_1, \exists x_2 \in D_2, ..., \exists x_n \in D_n \; c_1 \wedge c_2 \wedge ... \wedge c_k$$

Cette formule est en *forme normale prénexe*. Toutes les variables apparaissent ainsi quantifiées en début de formule, constituant le *préfixe*, chaque quantification liant toutes

les occurrences de la variable dans le reste de la formule. Les QCSP étendent les CSP en introduisant des quantificateurs universels (i.e., $\forall$) en lieu et place de certains des quantificateurs existentiels (i.e., $\exists$) présents, comme le montre la formule suivante :

$$\forall x_1 \in D_1, \exists x_2 \in D_2, \forall x_3 \in D_3, ..., \exists x_n \in D_n \; c_1 \wedge c_2 \wedge ... \wedge c_k$$

Cette formule reste en forme normale prénexe. Cependant, l'introduction de quantificateurs universels crée une relation d'ordre sur les variables qui n'était pas présente dans les CSP : la satisfiabilité de la formule logique précédente est susceptible de changer si l'on inverse dans le préfixe deux variables quantifiées de manière différente (par exemple x_1 et x_2 dans la formule précédente) ou même deux variables séparées par une alternance de quantificateurs (par exemple x_1 et x_3 dans une formule dont le préfixe est $\exists x_1 \forall x_2 \exists x_3$). Cette relation d'ordre sera donc aussi précisée dans la définition formelle d'un QCSP. Cependant, si deux variables qui se suivent dans le préfixe sont quantifiées de la même manière (par exemple $\exists x_1 \exists x_2$), leur ordre relatif n'influe aucunement sur la sémantique de la formule. On peut les échanger sans altérer le sens de la formule. Plus généralement, si on appelle *bloc de quantification* une sous-séquence du préfixe où toutes les variables sont quantifiées de la même manière, alors le fait d'échanger deux variables d'un même bloc de quantification dans le préfixe ne modifie pas la sémantique de la formule. Ces blocs de quantification sont appelés *qsets* [Vautard, 2010].

Définition 3.3.1 (qset)**.**
Un qset est un couple (qt, W) avec W un ensemble de variables et $qt \in \{\forall, \exists\}$ un quantificateur.

Le préfixe d'un QCSP peut donc être formellement défini comme suit :

Définition 3.3.2 (préfixe de QCSP)**.**
Un préfixe de QCSP est une séquence de qsets $((qt_1, W_1), ..., (qt_n, W_n))$ dont les ensembles de variables sont deux à deux disjoints $(\forall i, j, i \neq j \rightarrow (W_i \cap W_j = \emptyset))$.

Un QCSP se définit alors comme un préfixe accompagné d'un ensemble de contraintes dont les variables doivent faire partie du préfixe :

Définition 3.3.3 (QCSP)**.**
Un QCSP est un couple (P, G) avec $P = ((qt_1, W_1), ..., (qt_n, W_n))$ un préfixe et G un ensemble de contraintes appelé Goal tel que $var(G) \subseteq \bigcup_{i \in \{1,...,n\}} W_i$.

Un exemple de problème modélisé de manière naturelle par un QCSP est donné par le jeu suivant :

Exemple 3.3.1.
On considère un jeu à deux joueurs x et y jouant alternativement. En début de partie, on dispose d'un tas de 100 jetons. Chaque tour de jeu se déroule comme suit : le joueur dont c'est le tour choisit un nombre i entre 1 et 10 et retire i jetons du tas. Le jeu s'arrête lorsque le tas est vide : le joueur qui a retiré le dernier jeton a gagné. Le fait que x possède une stratégie gagnante, en 10 tours, peut être modélisé par la formule quantifiée suivante :

$$\exists x_0 \; \forall y_1 \; \exists x_1 \; ... \; \forall y_9 \; \exists x_9 \Big(x_0 + \textstyle\sum_{i \in \{1,...,9\}} (y_i + x_i) \Big) = 100$$

Lors de chaque tour i, chaque coup x_i retenu par le joueur x doit conduire à la victoire quel que soit le coup y_i effectué par le joueur y.
Cette formule est modélisée par le QCSP suivant :

$$\begin{cases} Q = ([(\exists, x_0), (\forall, y_1), (\exists, x_1), \ ... \ (\forall, y_9), (\exists, x_9)], \{x_0 + \sum_{i \in \{1,...,9\}} (y_i + x_i) = 100\}) \\ avec \ : \ D_{x_0} = ... = D_{x_9} = D_{y_1} = ... = D_{y_9} = \{1, ..., 10\}. \end{cases}$$

Une solution du problème (stratégie gagnante pour x) est illustrée par la figure 3.5

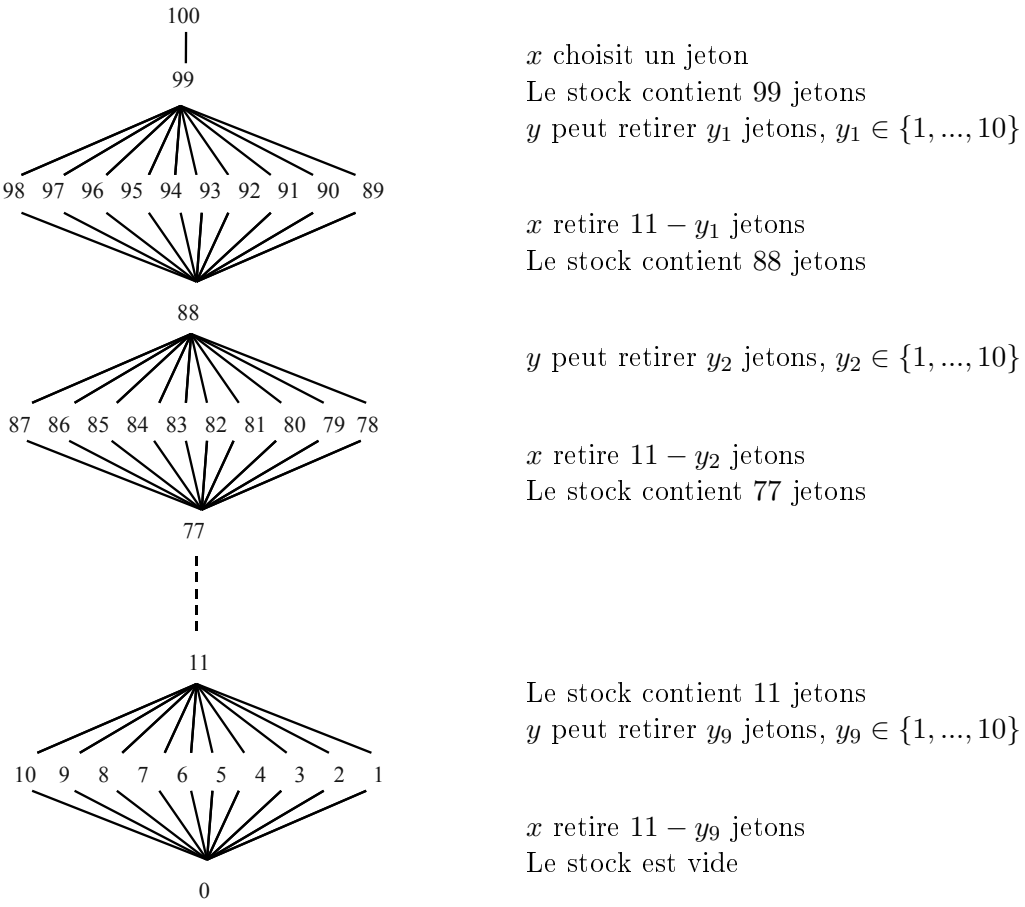

FIGURE 3.5 – La stratégie du joueur x gagnante pour le jeu à 100 jetons

3.3.3 QCSP$^+$

Le formalisme des QCSP est étendu dans [Benedetti *et al.*, 2007] pour permettre de modéliser explicitement des restrictions propres à chaque variable quantifiée. Chaque restriction est définie sous forme d'un ensemble de contraintes : le quantificateur ne porte alors que sur les tuples satisfaisant ces contraintes.

Un QCSP$^+$ doit donc intégrer des contraintes dont le but est de réduire la portée des quantificateurs. Ces contraintes sont donc ajoutées directement dans les qset.

Définition 3.3.4 (rqset)**.**
Un rqset est un triplet (qt, W, C) *avec* W *un ensemble de variables,* $qt \in \{\forall, \exists\}$ *un quantificateur et* C *un ensemble de contraintes.*

Le préfixe d'un QCSP$^+$ peut donc être défini comme suit :

Définition 3.3.5 (préfixe de QCSP$^+$)**.**
Un préfixe de QCSP est une séquence de rqsets $((qt_1, W_1, C_1), ..., (qt_n, W_n, C_n))$ *telle que :*
- $\forall i, j \in \{1, .., n\}, i \neq j \rightarrow W_i \cap W_j = \emptyset$
- $\forall i \in \{1, .., n\}, var(C_i) \subseteq (W_1 \cup ... \cup W_n)$

Le QCSP$^+$ se définit alors de manière analogue au QCSP comme suit :

Définition 3.3.6 (QCSP$^+$)**.**
Un QCSP$^+$ est un couple (P, G) *avec* $P = ((qt_1, W_1, C_1), ..., (qt_n, W_n, C_n))$ *un préfixe de QCSP$^+$ et* G *un ensemble de contraintes appelé Goal tel que* $var(G) \subseteq \bigcup_{i \in \{1,...,n\}} W_i$.

Exemple 3.3.2. *Le problème :*

$$\begin{array}{l} \exists X \in \{0,1,2,3\} \\ \quad \forall Y \in \{0,1,2\} \text{ tel que } Y \neq X, \\ \quad\quad \exists Z \in \{0,\ldots,6\} \text{ tel que } Z < 2 \times X, \\ \quad\quad\quad X + Y = Z \end{array}$$

est représenté par le QCSP$^+$ suivant :

$$\begin{cases} Q = ([(\exists, X), (\forall, Y, \{X \neq Y\}), (\exists, Z, \{Z < 2 \times X\})], Goal = \{X + Y = Z\}) \\ avec \ : \ D_X = \{0,1,2,3\}, \ D_Y = \{0,1,2\}, \ D_Z = \{0,...,6\}. \end{cases}$$

3.4 Conclusion

Nous avons présenté dans ce chapitre la cadre général des CSP ainsi que leurs principales méthodes de résolution et de filtrage Ce cadre général sera utilisé dans tous les chapitres de la partie Contributions. Nous avons aussi présenté le cadre particulier des CSPE ainsi que les QCSP$^+$, une généralisation des CSP permettant de résoudre les contraintes quantifiées. En effet, ces 2 familles de CSP sont utilisées pour décrire nos contributions aux chapitres 5 et 7 respectivement.

Troisième partie

Contributions

Chapitre 4

Un langage de requêtes à base de contraintes

Sommaire

Nous proposons dans ce chapitre, un langage de requêtes à base de contraintes permettant d'exprimer les requêtes de fouille de données. Ce langage comporte trois types de contraintes : des contraintes numériques sur des mesures telles que `freq`, `size`, des contraintes ensemblistes (définies à partir des opérateurs $\cap, \cup, \subset, ...$) et des contraintes plus spécifiques (couverture, fermeture, ...).

La modélisation et le langage proposés forment la base des approches d'extraction de motifs sous contraintes n-aires proposées dans les chapitres 5, 6 et 7.

Pour montrer l'intérêt et l'apport d'un tel langage ainsi que son adéquation pour modéliser les problèmes d'extraction de motifs (qu'ils soient locaux ou n-aires), nous l'illustrons sur trois catégories d'exemples :

1. Des exemples déjà résolus par des approches dédiées (règles d'exceptions, règles inattendues). Ces problèmes se caractérisent par une spécification complète. Notre apport consistera à les modéliser et les résoudre de façon générique (cf. section 4.3).
2. Des exemples concrets d'utilisation de requêtes n-aires pour l'extraction de motifs permettant la construction de modèles (groupes de synexpressions et conflits de classification) (cf. section 4.4).
3. Un exemple de construction de modèles descriptifs (catégorisation des instances de la base de données appelée clustering ou classification non supervisée) utilisant uniquement le langage de contraintes que nous proposons (cf. section 4.5).

Le dernier exemple permet aussi d'illustrer comment notre approche permet de raffiner les résultats obtenus en ajoutant des contraintes supplémentaires à la requête initiale au fur et à mesure qu'on veut imposer des propriétés supplémentaires aux motifs ciblés. Cette démarche est particulièrement utile lorsque les spécifications de l'utilisateur sont plus ou moins incomplètes, ou lorsqu'elles sont à préciser au cours du processus d'extraction.

Cette répartition en trois catégories est uniquement faite dans un but pédagogique. En effet, il n'y a pas vraiment de frontière entre ces trois catégories étant donné qu'on peut imaginer, par exemple, raffiner les règles d'exceptions obtenues en ajoutant des contraintes supplémentaires. Notons que ces différents exemples constituent un ensemble non exhaustif de requêtes qu'on peut modéliser à l'aide de notre langage de contraintes montrant l'étendue des domaines d'application de notre approche.

Le point fort de notre approche réside dans le fait que seulement avec un ensemble relativement petit de primitives, on arrive à couvrir une large panoplie de contraintes. Si on trouve dans la littérature d'autres travaux [Soulet, 2006, De Raedt *et al.*, 2008] proposant des cadres génériques d'extraction de motifs sous contraintes, à notre connaissance notre approche constitue le premier cadre générique permettant l'extraction de motifs définis par des contraintes n-aires.

Ce travail a donné lieu aux publications suivantes [Boizumault *et al.*, 2011, Métivier *et al.*, 2011, Métivier *et al.*, 2012].

4.1 Langage de contraintes proposé

4.2 Modélisation sous forme d'un CSP

Soit $\mathcal{T}$ l'ensemble des m transactions d'un jeu de données $\boldsymbol{r}$ et $\mathcal{I}$ l'ensemble de ses n items. La recherche de k motifs satisfaisant une conjonction/disjonction de contraintes $\mathcal{C}$ peut se modéliser à l'aide d'un CSP (cf. section 3.1) $\mathcal{P}$=$(\mathcal{X}, \mathcal{D}, \mathcal{C})$ où :

- $\mathcal{X}$=$\{X_1, ..., X_k\}$, où chaque variable X_i $(1 \leq i \leq k)$ représente un motif inconnu,
- $\mathcal{D}$=$\{D_{X_1}, ..., D_{X_k}\}$, le domaine initial D_{X_i} de chaque variable X_i est l'intervalle ensembliste $[\emptyset \,..\, \mathcal{I}] = 2^{\mathcal{I}}$,
- $\mathcal{C}$ est un ensemble de contraintes portant sur les X_i.

Les contraintes sont des relations portant sur des termes, ces termes étant construits en utilisant des constantes, des variables, des opérateurs et des symboles de fonctions. L'utilisateur peut définir de nouveaux symboles de fonctions aussi bien que de nouvelles contraintes. Nous décrivons enfin comment, l'utilisateur final peut, à partir du langage de contraintes prédéfinies proposé, formuler des requêtes ou bien définir de nouvelles contraintes.

4.2.1 Termes

Les termes sont construits à partir de :

1. **constantes** qui peuvent être des valeurs numériques (par exemple un seuil comme $min_{\texttt{freq}}$), des items (par exemple A) des motifs (par exemple $\{A, B\}$) ou des transactions (par exemple t_7) (cf. table 4.2).
2. **variables**, notées X_i, pour $1 \leq i \leq k$, représentant les motifs inconnus.
3. **opérateurs** :
   - opérateurs ensemblistes comme $\cap$ (intersection), $\cup$ (union), $\sqcup$ (union disjointe), $\setminus$ (différence), ...
   - opérateurs numériques comme $+, -, \times, /, \ldots$
4. **symboles de fonctions** portant sur un ou plusieurs motifs : `freq/1`, `size/1`, `support/1`, `ovelapItems/2`, `overlapTransactions/2`, ...

Exemple 4.2.1. *Exemples de termes :*
- $\mathit{freq}(X_1) \times \mathit{size}(X_1)$ *(i.e. l'aire du motif* X_1*)*
- $\mathit{freq}(X_1 \cup X_2) \times \mathit{size}(X_1 \cap X_2)$ *(i.e., le chevauchement entre l'aire de* X_1 *et l'aire de* X_2*),*
- $\mathit{freq}(X_1) - \mathit{freq}(X_2)$.

4.2.2 Symboles de fonctions prédéfinis

Le langage de contraintes que nous proposons contient un ensemble de fonctions prédéfinies :

- $\texttt{support}(X_i) = \{t \mid t \in \mathcal{T}, X_i \subseteq t\}$ le support du motif X_i.
- $\texttt{freq}(X_i) = \mid \{t \mid t \in \mathcal{T}, X_i \subseteq t\} \mid$ la fréquence du motif X_i.
- $\texttt{size}(X_i) = \mid \{j \mid j \in \mathcal{I}, j \in X_i\} \mid$ la cardinalité du motif X_i.
- $\texttt{ovelapItems}(X_i, X_j) = \mid X_i \cap X_j \mid$ le nombre d'items communs à X_i et X_j.
- $\texttt{overlapTransactions}(X_i, X_j) = \mid \texttt{support}(X_i) \cap \texttt{support}(X_j) \mid$ le nombre de transactions couvertes à la fois par X_i et X_j.

4.2.3 Symboles de fonctions définis par l'utilisateur

Notre langage de contraintes offre à l'utilisateur la possibilité de définir de nouveaux symboles de fonctions utilisant les différentes composantes du langage à savoir les constantes, les variables, les opérateurs et les symboles de fonctions (les prédéfinis mais aussi ceux préalablement définis par l'utilisateur).

Nous présentons dans ce qui suit quelques exemples de symboles de fonctions définis par l'utilisateur :

- $\texttt{area}(X_i) = \texttt{freq}(X_i) \times \texttt{size}(X_i)$
- $\texttt{coverage}(X_i, X_j) = \texttt{freq}(X_i \cup X_j) \times \texttt{size}(X_i \cap X_j)$
- Différentes mesures d'intérêt peuvent ainsi être définies. Considérons par exemple la mesure du *t*aux de croissance (growth rate) souvent utilisée dans la fouille de contrastes [Novak *et al.*, 2009b] et permettant de quantifier la valeur de l'émergence d'un motif entre deux parties d'une base de données. Soient D_1, $D_2 \subseteq \mathcal{T}$ deux

ensembles de transactions (i.e., classes) et $\texttt{freq'}(X_i, D_j)$ la fréquence de X_i dans D_j, le taux de croissance de X_i dans D_1 est

$$\texttt{growth-rate}(X_i)_{D_1} = \frac{\mid D_2 \mid \times \texttt{freq'}(X_i, D_1)}{\mid D_1 \mid \times \texttt{freq'}(X_i, D_2)}$$

4.2.4 Contraintes et requêtes

Les contraintes sont des relations portant sur des termes et pouvant être satisfaites ou non. Il y a trois types de contraintes prédéfinies :

1. **Contraintes numériques** comme : $<, \leq, =, \neq, \geq, >, \ldots$
   Exemples :
   - $\texttt{freq}(X_1) \leq 10$
   - $\texttt{size}(X_2) = 2 \times \texttt{size}(X_3)$
   - $\texttt{area}(X_1) < \texttt{size}(X_2) \times \texttt{freq}(X_3)$
2. **Contraintes ensemblistes** comme : $=, \neq, \in, \notin, \subset, \subseteq, \ldots$
   Exemples :
   - $A \in X_1$
   - $X_1 \cup X_2 \subset X_3$
   - $X_1 = X_2 \cap X_4$
3. **Contraintes spécifiques** comme :
   - $\texttt{closed}(X_i)$ est satisfaite ssi X_i est un motif fermé (cf. section 1.2.5).
   - $\texttt{coverTransactions}([X_1, ..., X_k])$ est satisfaite ssi chaque transaction est couverte par au moins un motif (i.e. $\bigcup_{1 \leq i \leq k} \texttt{support}(X_i) = \mathcal{T}$).
   - $\texttt{coverItems}([X_1, ..., X_k])$ est satisfaite ssi chaque item appartient à au moins un motif (i.e. $\bigcup_{1 \leq i \leq k} X_i = \mathcal{I}$).
   - $\texttt{noOverlapItems}([X_1, ..., X_k])$ [1] est satisfaite ssi les motifs X_i sont deux à deux disjoints.
   - $\texttt{noOverlapTransactions}([X_1, ..., X_k])$ [2] est satisfaite ssi les supports des motifs X_i sont deux à deux disjoints.
   - $\texttt{canonical}([X_1, ..., X_k])$ est satisfaite ssi pour chaque i tel que $1 \leq i < k$, le motif X_i est inférieur au motif X_{i+1} par rapport à l'ordre lexicographique.

Les requêtes sont ainsi des formules construites à partir de contraintes et des connecteurs logiques $\wedge$ (conjonction) et $\vee$ (disjonction).

4.3 Problèmes déjà résolus par approches dédiées

Nous montrons dans cette section, comment notre langage de requêtes permet de modéliser les règles d'exception et les règles inattendues. Ces deux problèmes ont déjà été résolus par des des approches dédiées. Nous décrivons dans les chapitres 5 et 6 comment ils peuvent être résolus par une approche générique.

1. Cette contrainte pourrait s'exprimer sous la forme : $\forall 1 \leq i < j \leq k$, $\texttt{ovelapItems}(X_i, X_j) = 0$, mais le filtrage serait moins performant.
2. Cette contrainte pourrait s'exprimer sous la forme : $\forall 1 \leq i < j \leq k$, $\texttt{overlapTransactions}(X_i, X_j) = 0$, mais le filtrage serait moins performant.

4.3.1 Règles d'exception

4.3.1.a Rappel de la formulation initiale

Les règles d'exception définies dans la section 2.2.3.b sont des règles qui expriment des situations déviant d'un comportement général modélisé par d'autres règles (appelées règles générales) : l'intérêt de cette définition est d'expliciter la nature d'une exception par rapport à un comportement général et admis. Plus formellement, les règles d'exception sont définies comme suit (I est un item, par exemple une valeur de classe, X et Y sont des motifs locaux) :

$$exception(X \rightarrow \neg I) \equiv \begin{cases} vrai & \text{si } \exists Y \in \mathcal{L}_{\mathcal{I}} \text{ tq } Y \subset X, \\ & (X \backslash Y \rightarrow I) \wedge (X \rightarrow \neg I) \\ faux & \text{sinon} \end{cases}$$

4.3.1.b Formulation à l'aide de notre langage de contraintes

Soient X et Y deux variables représentant les deux motifs inconnus, et I un item tel que I et $\neg I \in \mathcal{I}$ (I et $\neg I$ peuvent représenter deux classes présentes dans le jeu de données). Soient les seuils de fréquence $min_{\texttt{freq}}$ et $max_{\texttt{freq}}$ et les seuils de confiance δ_1 et δ_2. Les règles d'exception se modélisent comme suit dans notre langage de contraintes :

- $X \backslash Y \rightarrow I$ **doit être une règle fréquente de forte confiance :**
  $\texttt{freq}((X \backslash Y) \sqcup I) \geq min_{\texttt{freq}} \wedge \texttt{freq}(X \backslash Y) - \texttt{freq}((X \setminus Y) \sqcup I) \leq \delta_1$.

- $X \rightarrow \neg I$ **doit être une règle rare de forte confiance :**
  $\texttt{freq}(X \sqcup \neg I) \leq max_{\texttt{freq}} \wedge (\texttt{freq}(X) - \texttt{freq}(X \sqcup \neg I)) \leq \delta_2$.

D'où la requête :

$$exception(X, Y) \equiv \begin{cases} \texttt{freq}((X \setminus Y) \sqcup I) \geq min_{\texttt{freq}} \wedge \\ \texttt{freq}(X \setminus Y) - \texttt{freq}((X \setminus Y) \sqcup I) \leq \delta_1 \wedge \\ \texttt{freq}(X \sqcup \neg I) \leq max_{\texttt{freq}} \wedge \\ (\texttt{freq}(X) - \texttt{freq}(X \sqcup \neg I)) \leq \delta_2 \end{cases}$$

Remarque 4.3.1. *Étant donné que nous traitons dans cet exemple l'exemple particulier des règles rares, nous proposons de baser le calcul des confiances sur la définition des règles δ-fortes (cf.définition 1.1.9). Ceci nous permet de pouvoir contrôler avec plus de précision les valeurs des confiances des règles rares.*

4.3.2 Règles inattendues

4.3.2.a Rappel de la formulation initiale

Nous rappelons en premier lieu la formulation initiale des règles inattendues présentée dans la section 2.2.3.a. Une règle $X \rightarrow Y$ est *inattendue* par rapport à une croyance $U \rightarrow V$ (où U et V sont des motifs) ssi elle vérifie les contraintes suivantes :

1. $Y \wedge V$ n'est pas valide,
2. $X \wedge U$ est valide (XU est un motif fréquent),
3. $XU \rightarrow Y$ est valide ($XU \rightarrow Y$ est une règle fréquente et de confiance suffisante),

4. $XU \rightarrow V$ n'est pas valide (soit[3] $XU \rightarrow V$ n'est pas une règle fréquente, soit $XU \rightarrow V$ est une règle de faible confiance).

4.3.2.b Formulation à l'aide de notre langage de contraintes

Formulation initiale	Formulation avec notre langage de contraintes
$Y \wedge V$ n'est pas valide	$\texttt{freq}(Y \cup V) = 0$
$X \wedge U$ est valide (XU est un motif fréquent)	$\texttt{freq}(X \cup U) \geq min_{\texttt{freq1}}$
$XU \rightarrow Y$ est valide ($XU \rightarrow Y$ est une règle fréquente et de confiance suffisante)	$\texttt{freq}((X \cup U) \sqcup Y) \geq min_{\texttt{freq2}} \wedge$ $\texttt{freq}((X \cup U) \sqcup Y)/\texttt{freq}(X \cup U) \geq min_{\texttt{conf}}$
$XU \rightarrow V$ n'est pas valide (soit $XU \rightarrow V$ n'est pas une règle fréquente, soit $XU \rightarrow V$ est une règle de faible confiance)	$(\texttt{freq}((X \cup U) \sqcup V) < max_{\texttt{freq}} \vee$ $\texttt{freq}((X \cup U) \sqcup V)/\texttt{freq}(X \cup U) < max_{\texttt{conf}})$

TABLE 4.1 – Formulation des contraintes - Règles d'exception

D'où, étant donnée une croyance $U \rightarrow V$, une règle inattendue $un(X, Y)$ se modélise par la requête suivante :

$$un(X,Y) \equiv \begin{cases} \texttt{freq}(Y \cup V) = 0 \wedge \\ \texttt{freq}(X \cup U) \geq min_{\texttt{freq1}} \wedge \\ \texttt{freq}((X \cup U) \sqcup Y) \geq min_{\texttt{freq2}} \wedge \\ \texttt{freq}((X \cup U) \sqcup Y)/\texttt{freq}(X \cup U) \geq min_{conf} \wedge \\ (\texttt{freq}((X \cup U) \sqcup V) < max_{\texttt{freq}} \vee \\ \texttt{freq}((X \cup U) \sqcup V)/\texttt{freq}(X \cup U) < max_{conf}) \end{cases}$$

4.4 Vers la construction de modèles

Les motifs extraits sont souvent utilisés dans des méthodes pour la construction de modèles prédictifs (construction de classifieurs permettant de prédire la classe d'un nouvel objet non étiqueté), de modèle descriptifs (catégorisation des instances de la base de données appelée clustering ou classification non supervisée) (cf. section 2.3). Nous présentons dans ce qui suit deux exemples concrets d'utilisation des requêtes n-aires dans la phase d'extraction de motifs permettant par la suite la construction de modèles.

4.4.1 Conflits de classification

La combinaison des motifs locaux est un point clé de la qualité d'un classifieur à base d'associations. De tels classifieurs sont généralement construits à partir de règles fréquentes et de forte confiance. Typiquement, les paires de règles ayant un important

3. C'est la première fois que l'on rencontre une disjonction. Cela vaut le coup de s'y attarder un peu.

chevauchement entre leurs prémisses et concluant sur des classes distinctes sont particulièrement susceptibles de donner lieu à un conflit de classification. En effet, lorsqu'une règle d'une telle paire est déclenchée par un exemple à classer, l'autre règle concluant sur une autre valeur de classe est fortement susceptible d'être également déclenchée car les prémisses des deux règles sont relativement similaires. Ce double déclenchement conduira à un conflit de classification.

Soient $X \rightarrow c_1$ et $Y \rightarrow c_2$ deux règles fréquentes et de forte confiance, le fait que deux règles soient susceptibles de donner lieu à un conflit de classification se modélise par la requête suivante :

$$conflits\_classif(X,Y) \equiv \begin{cases} \texttt{freq}(X) \geq min_{\texttt{freq}} \wedge \\ \texttt{freq}(Y) \geq min_{\texttt{freq}} \wedge \\ (\texttt{freq}(X \sqcup \{c_1\})/\texttt{freq}(X)) \geq min_{\texttt{conf}} \wedge \\ (\texttt{freq}(Y \sqcup \{c_2\})/\texttt{freq}(Y)) \geq min_{\texttt{conf}} \wedge \\ \texttt{size}(X \cap Y) \geq (\texttt{size}(X) + \texttt{size}(Y))/4 \end{cases}$$

Les quatre premières inégalités portent sur la fréquence et la confiance des règles de classification. La dernière inégalité décrit le chevauchement souhaité : les deux règles doivent avoir en commun au moins la moitié des items de leurs prémisses.

Il est, bien entendu, possible de modifier les paramètres de la requête n-aire et/ou d'ajouter de nouvelles contraintes pour modéliser des types de conflits de classification plus spécifiques.

4.4.2 Groupes de synexpressions

La recherche de motifs candidats pour la construction de groupes de synexpressions (cf. section 2.2.3.d) à partir de k motifs locaux se modélise par la requête suivante :

$$synexpression([X_1,...,X_k]) \equiv \begin{cases} \wedge_{1 \leq i < j \leq k} \\ \big(\texttt{area}(X_i) > min_{\texttt{area}} \wedge \\ \texttt{area}(X_j) > min_{\texttt{area}} \wedge \\ \texttt{coverage}(X_i, X_j) > \alpha \times min_{\texttt{area}} \big) \end{cases}$$

Où $min_{\texttt{area}}$ est le seuil d'aire minimale et α est un paramètre donné par l'utilisateur pour fixer le recouvrement minimal entre motifs locaux (cf. section 2.2.3.d).

4.5 Clustering par affinements successifs

Un atout majeur des approches de résolution, associées à ce langage de requêtes est la flexibilité permettant à l'utilisateur de raffiner (respectivement généraliser) les résultats obtenus par ajout (respectivement retrait) de contraintes sans se soucier de la méthode de résolution. En pratique, l'utilisateur commence par formuler une première requête q_0 qu'il peut affiner (respectivement généraliser) par des opérations successives d'ajout (respectivement retrait) de contraintes (q_{i+1} est dérivée à partir de q_i) jusqu'à ce qu'il considère que l'information extraite est assez relevante.

Nous illustrons cette approche par affinements successifs avec l'exemple d'un problème de classification non supervisée (clustering). Le clustering consiste à faire émerger des groupes au sein d'un ensemble d'éléments sans aucune information a priori (cf. section 4.4.1). Les groupes créés sont appelés clusters. L'objectif étant de découvrir la structure sous-jacente des données pour en extraire de l'information [Fisher, 1987]. Le choix

Trans.	Items							
t_1	A			D		F		
t_2	A				E	F		
t_3	A				E		G	
t_4	A				E		G	
t_5		B			E		G	
t_6		B			E		G	
t_7			C		E		G	
t_8			C		E		G	
t_9			C		E			H
t_{10}			C		E			H
t_{11}			C			F	G	H

TABLE 4.2 – Jeu de données pour clustering $\mathcal{T}$

du clustering comme exemple explicatif de cette approche est justifié par l'importance et la popularité de cette tâche en fouille de données qui est aussi généralement effectuée à travers des requêtes jusqu'à l'obtention de solutions satisfaisantes. D'autant plus que notre approche permet naturellement d'intégrer les différentes contraintes requises pour les tâches de clustering à base de contraintes [Berkhin, 2006]. Nous considérons le jeu de données de la table 4.2 comme jeu de données exemple.

4.5.1 Modélisation du problème général

Les motifs fermés (cf. section 1.2.4) sont de bons candidats pour la recherche de clusterings à base d'association. Un problème de clustering peut être formulé comme suit : «trouver un ensemble de k motifs fermés $X_1, X_2, ..., X_k$ (i.e., clusters) couvrant toutes les transactions sans chevauchement entre les supports respectifs de ces motifs».

Notre langage de contraintes permet de modéliser ce problème de manière élégante et concise. la formulation précédemment citée. D'abord, la contrainte $\texttt{closed}(X_i)$ impose que chaque motif inconnu X_i soit fermé. La contrainte $\texttt{coverTransactions}([X_1, X_2, .., X_k])$ assure, quant à elle, que les motifs extraits couvrent la totalité de l'ensemble des transactions. Finalement, la contrainte $\texttt{noOverlapTransactions}([X_1, ..., X_k])$ permet d'éviter les chevauchements entre les supports des motifs extraits en imposant qu'ils soient deux à deux disjoints.

Nous obtenons ainsi la requête q_0 suivante modélisant le problème général de clustering :

$$\texttt{q}_0([X_1, ..., X_k]) \equiv \begin{cases} \wedge_{1 \leq i \leq k}\ \texttt{closed}(X_i) \wedge \\ \texttt{coverTransactions}([X_1, ..., X_k]) \wedge \\ \texttt{noOverlapTransactions}([X_1, ..., X_k]) \end{cases}$$

Exemple 4.5.1.

Considérons le jeu de données exemple de la table 4.2. Pour $k = 3$ motifs inconnus, nous obtenons 30 *solutions pour la requête q_0 (voir Table 4.3).*

Sol.	X_1	X_2	X_3
s_1	{C, F, G, H}	{E}	{A, D, F}
s_2	{C, F, G, H}	{A, D, F}	{E}
s_3	{A, D, F}	{C, F, G, H}	{E}
s_4	{A, D, F}	{E}	{C, F, G, H}
s_5	{E}	{C, F, G, H}	{A, D, F}
s_6	{E}	{A, D, F}	{C, F, G, H}
s_7	{A, F}	{C, H}	{E, G}
⋮	⋮	⋮	⋮
s_{13}	{C, E, H}	{E, G}	{F}
⋮	⋮	⋮	⋮
s_{19}	{A, F}	{C, E, H}	{G}
⋮	⋮	⋮	⋮
s_{25}	{A}	{B, E, G}	{C}
⋮	⋮	⋮	⋮
s_{30}	{C}	{B, E, G}	{A}

TABLE 4.3 – Ensemble des solutions de la requête q_0.

4.5.2 Affiner les requêtes

L'ensemble des solutions obtenues peut par la suite être réduit en affinant la requête q_0. En effet, l'utilisateur peut facilement ajouter des contraintes à la requête initialement posée afin d'obtenir des clusters plus spécifiques.

À titre d'exemple, à partir de la requête initiale q_0, nous dérivons la requête q_1 permettant d'éviter les solutions symétriques. Puis, nous affinons encore la requête afin d'éviter les clusters peu fréquents et les clusters de petite taille (requêtes q_2 et q_3).

4.5.2.a Éviter les solutions symétriques

Deux solutions s_i et s_j sont dites *symétriques* si et seulement si il existe une permutation σ, tel que $s_j = \sigma(s_i)$. Un problème de clustering, tel qu'il est posé dans la requête q_0, contient naturellement des solutions symétriques étant donné que les contraintes posées sont les mêmes pour chacun des motifs recherchés. Soit $s = (p_1, p_2, ..., p_k)$ une solution contenant k motifs p_i, toute permutation σ de ces k motifs $\sigma(s) = (p_{\sigma(1)}, p_{\sigma(2)}, ..., p_{\sigma(k)})$ est aussi une solution.

Exemple 4.5.2. *Par exemple, dans la table 4.3, les solutions de s_1 à s_6 sont symétriques et constituent un même clustering.*

La contrainte `canonical`$([X_1, ..., X_k])$ (cf. section 4.2.4) est alors utilisée pour éviter les solutions symétriques. Cette contrainte impose que pour tout i tel que. $1 \leq i < k$, le motif X_i est inférieur au motif X_{i+1} suivant l'ordre lexicographique. On obtient la

Sol.	X_1	X_2	X_3
s_1	{C, F, G, H}	{E}	{A, D, F}
s_7	{A, F}	{C, H}	{E, G}
s_{13}	{C, E, H}	{E, G}	{F}
s_{19}	{A, F}	{C, E, H}	{G}
s_{25}	{A}	{B, E, G}	{C}

TABLE 4.4 – Ensemble des solutions de la requête q_1.

requête q_1 en affinant la requête q_0 par ajout de la contrainte $\texttt{canonical}([X_1, ..., X_k])$:

$$\mathtt{q}_1([X_1, ..., X_k]) \equiv \begin{cases} \wedge_{1 \leq i \leq k} \texttt{closed}(X_i) \wedge \\ \texttt{coverTransactions}([X_1, ..., X_k]) \wedge \\ \texttt{noOverlapTransactions}([X_1, ..., X_k]) \wedge \\ \texttt{canonical}([X_1, ..., X_k]) \end{cases}$$

Exemple 4.5.3. *Pour le jeu de données exemple de la table 4.2, la requête q_1 possède seulement 5 solutions ($5 \times 3! = 30$) (cf. table 4.4).*

La contrainte $\texttt{canonical}([X_1, ..., X_k])$ joue un rôle très important dans l'écriture de requêtes de clustering. En effet, sans la contrainte $\texttt{canonical}([X_1, ..., X_k])$, chaque clustering à k motifs possible est représenté par $k!$ solutions constituant toutes les permutations possibles des motifs qui le composent. Du coup il est indispensable de casser cette symétrie avec la contrainte $\texttt{canonical}([X_1, ..., X_k])$ afin de pouvoir éliminer les solutions redondantes. D'autant plus que le filtrage associé à cette contrainte permet d'élaguer considérablement l'espace de recherche.

4.5.2.b Ignorer les solutions contenant des clusters peu fréquents

Quand un cluster extrait a une faible valeur de fréquence, il est considéré comme peu représentatif et par suite peu fiable. A partir de la requête q_1, on peut facilement ajouter des contraintes de seuil de fréquence sur les motifs imposant que chaque cluster obtenu soit suffisamment fréquent. Etant donné un seuil de fréquence minimale δ_1, la requête q_2 obtenue est la suivante :

$$\mathtt{q}_2([X_1, ..., X_k]) \equiv \begin{cases} \wedge_{1 \leq i \leq k} \texttt{closed}(X_i) \wedge \\ \texttt{coverTransactions}([X_1, ..., X_k]) \wedge \\ \texttt{noOverlapTransactions}([X_1, ..., X_k]) \wedge \\ \texttt{canonical}([X_1, ..., X_k]) \wedge \\ \wedge_{1 \leq i \leq k} \texttt{freq}(X_i) \geq \delta_1 \end{cases}$$

Exemple 4.5.4. *Pour une valeur de $\delta_1 = 2$ et le même jeu de données de la table 4.2, le motif $\{C, F, G, H\}$ appartenant à la solution s_1 est de fréquence 1 et est ainsi exclu. Pour la requête q_2, il reste 4 solutions (s_7, s_{13}, s_{19}, et s_{25}, voir Table 4.4).*

4.5.2.c Ignorer les motifs de petites tailles

Un clustering contenant au moins un motif X_i de petite taille [4] n'est pas considéré comme pertinent parce que X_i ne garantit pas suffisamment de similarité entre les transactions qui le supportent.

A partir de la requête q_2, on peut facilement ajouter des contraintes de cardinalité sur les motifs imposant que la taille de chaque motif solution soit supérieure ou égale à un seuil préalablement donné par l'utilisateur. Etant donné un seuil de cardinalité minimale δ_2, la requête q_3 obtenue est la suivante :

$$\mathtt{q}_3([X_1,...,X_k]) \equiv \begin{cases} \wedge_{1\leq i\leq k}\ \mathtt{closed}(X_i) \wedge \\ \mathtt{coverTransactions}([X_1,...,X_k]) \wedge \\ \mathtt{noOverlapTransactions}([X_1,...,X_k]) \wedge \\ \mathtt{canonical}([X_1,...,X_k]) \wedge \\ \wedge_{1\leq i\leq k}\ \mathtt{freq}(X_i) \geq \delta_1 \wedge \\ \wedge_{1\leq i\leq k}\ \mathtt{size}(X_i) \geq \delta_2 \end{cases}$$

Exemple 4.5.5. *Pour une valeur de $\delta_2 = 2$ et le même jeu de données de la table 4.2 et avec la requête q_3, il ne reste plus qu'une seule solution : $s7$ où $X_1 = \{A, F\}$, $X_2 = \{C, H\}$ et $X_3 = \{E, G\}$ voir Table 4.4).*

4.5.3 Autres problèmes de clustering

De la même manière, il est aussi possible de modéliser d'autres problèmes de clustering [Berkhin, 2006] comme le *soft clustering*, le *co-clustering*, et le *soft co-clustering* que nous détaillons dans la suite.

4.5.3.a Soft clustering

Ce problème n'est d'autre qu'une version relaxée du problème de clustering précédemment présenté dans laquelle on autorise un certain chevauchement entre les supports des motifs extraits (chevauchement inférieur à un seuil δ_T donné).

La requête q_4, qui est une version relaxée de la requête q_1, modélise ce problème :

$$\mathtt{q}_4([X_1,...,X_k]) \equiv \begin{cases} \wedge_{1\leq i\leq k}\ \mathtt{closed}(X_i) \wedge \\ \mathtt{coverTransactions}([X_1,...,X_k]) \wedge \\ \wedge_{1\leq i<j\leq k}\ \mathtt{overlapTransactions}(X_i, X_j) \leq \delta_T \wedge \\ \mathtt{canonical}([X_1,...,X_k]) \end{cases}$$

Exemple 4.5.6. *Pour une valeur de $k = 3$ et un chevauchement maximal entre les supports délimité par $\delta_T = 1$, la requête q_4 possède 13 solutions (cf. table 4.5). Si nous n'avions pas évité les solutions symétriques avec la contrainte $\mathtt{canonical}([X_1,...,X_k])$, nous aurions obtenu 78 (soit $3! \times 13$) solutions.*

Les solutions de la requête q_4 appliquée au jeu de données de la table 4.2, pour des valeurs de $k = 3$ et un chevauchement maximal entre les supports délimité par $\delta_T = 1$, sont présentées dans la table 4.5. Prenons par exemple la solution s'_1, un chevauchement existe entre les supports de X_1 et X_3 au niveau de la transaction t_{11}.

4. Les motifs de taille 1 par exemple, reflètent souvent des valeurs qui ne sont apparues que lors de la binarisation d'un attribut, comme par exemple A, B et C dans Table 4.2 et qui ne sont donc pas d'une grande utilité.

Sol.	X_1	X_2	X_3
s'_1	{C, F, G, H}	{E}	{F}
s'_2	{A, D, F}	{C, F, G, H}	{E}
s'_3	{A, F}	{C, F, G, H}	{E}
s'_4	{A, E, F}	{E}	{F}
s'_5	{A, D, F}	{E}	{F}
s'_6	{A}	{B, E, G}	{C}
s'_7	{A, F}	{C, E, H}	{G}
s'_8	{A, F}	{C, H}	{G}
s'_9	{A, F}	{C, H}	{E, G}
s'_{10}	{C, E, H}	{E, G}	{F}
s'_{11}	{C, H}	{E, G}	{F}
s'_{12}	{C, H}	{F}	{G}
s'_{13}	{C, E, H}	{F}	{G}

TABLE 4.5 – Ensemble des solutions de la requête q_4.

Ignorer les clusters peu fréquents (en fixant un seuil de fréquence minimale $\delta_1 = 2$) permet de n'obtenir que 8 solutions (de s'_6 à s'_{13}). En ajoutant une contrainte sur la cardinalité des clusters en fixant un seuil minimal $\delta_2 = 2$, on obtient une seule et unique solution qui est s'_9 (qui est aussi la solution s_7 du problème de clustering initial, voir Section 4.5.2.c).

4.5.3.b Co-clustering

Le co-clustering vise à trouver k clusters couvrant à la fois l'ensemble des transactions et l'ensemble des items, sans chevauchement ni entre les motifs ni entre leurs supports. Ce problème est décrit par la requête q_5 :

$$\mathtt{q}_5([X_1, ..., X_k]) \equiv \begin{cases} \wedge_{1 \leq i \leq k}\ \mathtt{closed}(X_i) \wedge \\ \mathtt{coverTransactions}([X_1, ..., X_k]) \wedge \\ \mathtt{noOverlapTransactions}([X_1, ..., X_k]) \wedge \\ \mathtt{coverItems}([X_1, ..., X_k]) \wedge \\ \mathtt{noOverlapItems}([X_1, ..., X_k]) \wedge \\ \mathtt{canonical}([X_1, ..., X_k]) \end{cases}$$

4.5.3.c Soft co-clustering

C'est une version relaxée du co-clustering dans laquelle on autorise un certain chevauchement entre les motifs (seuil δ_1) et un certain chevauchement entre leurs supports (seuil δ_2).

Ce problème est décrit par la requête q_6 (version relaxée de q_4 et q_5) :

$$\mathtt{q}_6([X_1, ..., X_k]) \equiv \begin{cases} \wedge_{1 \leq i \leq k}\ \mathtt{closed}(X_i) \wedge \\ \mathtt{coverTransactions}([X_1, ..., X_k]) \wedge \\ \wedge_{1 \leq i < j \leq k}\ \mathtt{overlapTransactions}(X_i, X_j) \leq \delta_T \wedge \\ \mathtt{coverItems}([X_1, ..., X_k]) \wedge \\ \wedge_{1 \leq i < j \leq k}\ \mathtt{ovelapItems}(X_i, X_j) \leq \delta_I \wedge \\ \mathtt{canonical}([X_1, ..., X_k]) \end{cases}$$

4.5.3.d Balanced clustering

Dans les problèmes de clustering, on a tendance à préférer généralement des solutions dans lesquelles les fréquences des différents clusters extraits sont proches les unes des autres. La requête q_7 décrit un problème de clustering dans lequel un équilibre entre les valeurs de fréquence des différents motifs doit être respecté. Pour chaque couple de clusters (X_i, X_j), leur différence de fréquences doit être inférieure ou égale au seuil $\Delta \times m$ où Δ est un pourcentage et m et le nombre de transactions de la base de données (cf. requête q_7).

$$\mathtt{q}_7([X_1, ..., X_k]) \equiv \begin{cases} \wedge_{1 \leq i \leq k} \ \mathtt{closed}(X_i) \wedge \\ \mathtt{coverTransactions}([X_1, ..., X_k]) \wedge \\ \mathtt{noOverlapTransactions}([X_1, ..., X_k]) \wedge \\ \mathtt{canonical}([X_1, ..., X_k]) \wedge \\ \wedge_{1 \leq i < j \leq k} \ \mid \mathtt{freq}(X_i) - \mathtt{freq}(X_j) \mid \leq \Delta \times m \end{cases}$$

D'une manière analogue, on peut aussi imposer un équilibre sur la taille des clusters extraits (cf. requête q_8).

$$\mathtt{q}_8([X_1, ..., X_k]) \equiv \begin{cases} \wedge_{1 \leq i \leq k} \ \mathtt{closed}(X_i) \wedge \\ \mathtt{coverTransactions}([X_1, ..., X_k]) \wedge \\ \mathtt{noOverlapTransactions}([X_1, ..., X_k]) \wedge \\ \mathtt{canonical}([X_1, ..., X_k]) \wedge \\ \wedge_{1 \leq i < j \leq k} \ \mid \mathtt{size}(X_i) - \mathtt{size}(X_j) \mid \leq \Delta \times n \end{cases}$$

4.6 Mise sous forme normale

Si l'utilisateur peut directement formuler ses requêtes sous la forme précédente, nous utiliserons par la suite une forme "normale" où les symboles de fonctions ne pourront porter que sur des variables. Cette transformation n'affecte en rien la sémantique de la contrainte mais nécessite l'ajout de variables intermédiaires. Enfin, c'est à partir de cette forme normale que travailleront les approches décrites dans les deux chapitres suivants.

Dans cette section, nous montrons la mise sous forme normale des requêtes modélisant les règles d'exception (cf. section 4.3.1) et les règles inattendues (cf. section 4.3.2). Le traitement des synexpressions (cf. section 4.4.2) et des conflits de classification (cf. section 4.4.1) s'effectue de manière similaire. Enfin, comme la modélisation du Clustering (cf. section 4.5) n'utilise aucun terme fonctionnel, elle est déjà sous forme normale.

4.6.1 Exemple des règles d'exception

Nous partons de la modélisation obtenue à la Section 4.3.1 puis nous effectuons, pas à pas, la transformation.

$$exception(X, Y) \equiv \begin{cases} \mathtt{freq}((X \setminus Y) \sqcup I) \geq min_{\mathtt{freq}} \wedge \\ \mathtt{freq}(X \setminus Y) - \mathtt{freq}((X \setminus Y) \sqcup I) \leq \delta_1 \wedge \\ \mathtt{freq}(X \sqcup \neg I) \leq max_{\mathtt{freq}} \wedge \\ \mathtt{freq}(X) - \mathtt{freq}(X \sqcup \neg I) \leq \delta_2 \end{cases}$$

Nous détaillons dans ce qui suit respectivement la mise sous forme normale de la règle générale et de la règle d'exception, et finalement les interdépendances entre les variables modélisant la requête.

Modélisation de la règle générale : la règle générale est définie par les deux contraintes suivantes :

- $\texttt{freq}((X \setminus Y) \sqcup I) \geq min_{\texttt{freq}}$ imposant que la règle générale soit fréquente,
- $\texttt{freq}(X \setminus Y) - \texttt{freq}((X \setminus Y) \sqcup I) \leq \delta_1$ imposant que la règle générale soit de forte confiance.

Nous avons donc besoin de connaître la fréquence du motif $X \setminus Y$ (ie. représentant la prémisse de la règle générale) et celle du motif $(X \setminus Y) \sqcup I$ (ie. représentant la règle générale). Ainsi nous introduisons les deux variables X_1 et X_2 telles que :

- X_1 représente $X \setminus Y$,
- X_2 représente $(X \setminus Y) \sqcup I$.

Remarque 4.6.1. *I et $\neg I$ sont deux constantes représentant deux items de classe.*

Modélisation de la règle d'exception : la règle d'exception est définie par les deux contraintes suivantes :

- $\texttt{freq}(X \sqcup \neg I) \leq max_{\texttt{freq}}$ imposant que la règle d'exception soit rare,
- $\texttt{freq}(X) - \texttt{freq}(X \sqcup \neg I) \leq \delta_2$ imposant que la règle d'exception soit de forte confiance.

Nous avons donc besoin de connaître la fréquence du motif X (ie. représentant la prémisse de la règle d'exception) et celle du motif $X \sqcup \neg I$ (ie. représentant la règle d'exception). Ainsi on introduit les deux variables X_3 et X_4 telles que :

- X_3 représente X,
- X_4 représente $X \sqcup \neg I$.

Interdépendances entre les variables : les interdépendances entre les différentes variables sont décrites par les contraintes suivantes :

- $X_1 \subset X_3$: la prémisse de la règle générale est incluse dans celle de la règle d'exception (cf. définition des règles d'exception section 2.2.3.b),
- $X_2 = X_1 \sqcup I$: la règle générale (X_2) est composée de la prémisse (X_1) et a comme conclusion l'item de classe I,
- $X_4 = X_3 \sqcup \neg I$: la règle d'exception (X_4) est composée de la prémisse (X_3) et a comme conclusion l'item de classe $\neg I$.

Notons que l'utilisation de l'opérateur d'union disjointe $\sqcup$ permet d'exclure toute les instanciations où un item de classe figure dans la prémisse d'une règle.

La table 4.6 résume la mise sous forme normale de la requête des règles d'exception.

CSP associé :

- $\mathcal{X} = \{X_1, X_2, X_3, X_4\}$.
- $\forall 1 \leq i \leq 4,\ D_{X_i} = [\emptyset..\mathcal{I}]$.
- $\mathcal{C} = \{X_1 \subset X_3,\ X_2 = X_1 \sqcup I,\ X_4 = X_3 \sqcup \neg I,\ \texttt{freq}(X_2) \geq min_{\texttt{freq}},$
  $\texttt{freq}(X_1) - \texttt{freq}(X_2) \leq \delta_1,\ \texttt{freq}(X_4) \leq max_{\texttt{freq}},\ \texttt{freq}(X_3) - \texttt{freq}(X_4) \leq \delta_2\}$.

Modélisation	Forme normale
interdépendances entre les motifs	$X_1 \subset X_3$ $\wedge \quad X_2 = X_1 \sqcup I$ $\wedge \quad X_4 = X_3 \sqcup \neg I$
$\texttt{freq}((X \setminus Y) \sqcup I) \geq min_{\texttt{freq}}$	$\texttt{freq}(X_2) \geq min_{\texttt{freq}}$
$\texttt{freq}(X \setminus Y) - \texttt{freq}((X \setminus Y) \sqcup I) \leq \delta_1$	$\texttt{freq}(X_1) - \texttt{freq}(X_2) \leq \delta_1$
$\texttt{freq}(X \sqcup \neg I) \leq max_{\texttt{freq}}$	$\texttt{freq}(X_4) \leq max_{\texttt{freq}}$
$\texttt{freq}(X) - \texttt{freq}(X \sqcup \neg I) \leq \delta_2$	$\texttt{freq}(X_3) - \texttt{freq}(X_4) \leq \delta_2$

TABLE 4.6 – Forme normale - Règles d'exception

La requête ainsi obtenue est la suivante :

$$exception([X_1, X_2, X_3, X_4]) \equiv \begin{cases} X_1 \subset X_3 \wedge \\ X_2 = X_1 \sqcup I \wedge \\ X_4 = X_3 \sqcup \neg I \wedge \\ \texttt{freq}(X_2) \geq min_{\texttt{freq}} \wedge \\ \texttt{freq}(X_1) - \texttt{freq}(X_2) \leq \delta_1 \wedge \\ \texttt{freq}(X_4) \leq max_{\texttt{freq}} \wedge \\ \texttt{freq}(X_3) - \texttt{freq}(X_4) \leq \delta_2 \end{cases}$$

4.6.2 Exemple des règles inattendues

Nous partons de la modélisation obtenue à la Section 4.3.2 puis nous effectuons, pas à pas, la transformation.

$$unexpected(X, Y) \equiv \begin{cases} \texttt{freq}(Y \cup V) = 0 \wedge \\ \texttt{freq}(X \cup U) \geq min_{\texttt{freq1}} \wedge \\ \texttt{freq}((X \cup U) \sqcup Y) \geq min_{\texttt{freq2}} \wedge \\ \texttt{freq}((X \cup U) \sqcup Y) / \texttt{freq}(X \cup U) \geq min_{\texttt{conf}} \wedge \\ (\texttt{freq}((X \cup U) \sqcup V) < max_{\texttt{freq}} \vee \\ \texttt{freq}((X \cup U) \sqcup V) / \texttt{freq}(X \cup U) < max_{\texttt{conf}}) \end{cases}$$

U et V étant des motifs donnés (*constantes*), on introduit les variables suivantes :
- X_1 représente X,
- X_2 représente Y,
- X_3 représente $Y \cup V$ d'où la contrainte $X_3 = X_2 \cup V$,
- X_4 représente $X \cup U$ d'où la contrainte $X4 = X_1 \cup U$,
- X_5 représente $(X \cup U) \sqcup Y$ d'où la contrainte $X5 = X_4 \sqcup X_2$,
- X_6 représente $(X \cup U) \sqcup V$ d'où la contrainte $X_6 = X_4 \sqcup V$.

CSP associé :
- $\mathcal{X} = \{X_1, X_2, X_3, X_4, X_5, X_6\}$.
- $\forall 1 \leq i \leq 6,\ D_{X_i} = [\emptyset..\mathcal{I}]$.

Modélisation	Forme normale
interdépendances entre les motifs	$X_3 = X_2 \cup V \ \wedge$ $X_4 = X_1 \cup U \ \wedge$ $X_5 = X_4 \sqcup X_2 \ \wedge$ $X_6 = X_4 \sqcup V$
$\texttt{freq}(Y \cup V) = 0$	$freq(X_3) = 0$
$\texttt{freq}(X \cup U) \ \geq min_{\texttt{freq1}}$	$\texttt{freq}(X_4) \ \geq min_{\texttt{freq1}}$
$\texttt{freq}(X \cup U \cup Y) \ \geq min_{\texttt{freq2}}$	$\texttt{freq}(X_5) \ \geq min_{\texttt{freq2}}$
$\texttt{freq}(X \cup U \cup Y) \ \geq min_{\texttt{conf}} \times \texttt{freq}(X \cup U)$	$\texttt{freq}(X_5) \ \geq min_{\texttt{conf}} \times \texttt{freq}(X_4)$
$\texttt{freq}(X \cup U \cup V) \ < max_{\texttt{freq}} \vee$ $\texttt{freq}(X \cup U \cup V) \ < max_{\texttt{conf}} \times \texttt{freq}(X \cup U)$	$\texttt{freq}(X_6) \ < max_{\texttt{freq}} \vee$ $\texttt{freq}(X_6) \ < max_{\texttt{conf}} \times \texttt{freq}(X_4)$

TABLE 4.7 – Formulation des contraintes - Règles inattendues

– $\mathcal{C} = \{X_3 = X_2 \cup V,\ X_4 = X_1 \cup U,\ X_5 = X_4 \sqcup X_2,\ X_6 = X_4 \sqcup V,$
$freq(X_3) = 0,\ \texttt{freq}(X_4) \ \geq min_{\texttt{freq1}},\ \texttt{freq}(X_5) \ \geq min_{\texttt{freq2}},$
$\texttt{freq}(X_5) \ \geq min_{\texttt{conf}} \times \texttt{freq}(X_4),$
$((\texttt{freq}(X_6) \ < max_{\texttt{freq}}) \ \vee \ (\texttt{freq}(X_6) \ < max_{\texttt{conf}} \times \texttt{freq}(X_4)))\}.$
La requête ainsi obtenue est la suivante :

$$unexpected([X_1, X_2, ..., X_6]) \equiv \begin{cases} X_3 = X_2 \cup V \ \wedge \\ X_4 = X_1 \cup U \ \wedge \\ X_5 = X_4 \sqcup X_2 \ \wedge \\ X_6 = X_4 \sqcup V \ \wedge \\ freq(X_3) = 0 \ \wedge \\ \texttt{freq}(X_4) \ \geq min_{\texttt{freq1}} \ \wedge \\ \texttt{freq}(X_5) \ \geq min_{\texttt{freq2}} \ \wedge \\ \texttt{freq}(X_5) \ \geq min_{\texttt{conf}} \times \texttt{freq}(X_4) \ \wedge \\ ((\texttt{freq}(X_6) \ < max_{\texttt{freq}}) \ \vee \ \wedge \\ (\texttt{freq}(X_6) \ < max_{\texttt{conf}} \times \texttt{freq}(X_4))) \end{cases}$$

4.7 Conclusion

Nous avons présenté dans ce chapitre un langage de requêtes à base de contraintes permettant d'exprimer les requêtes de fouille de données. Ce langage est très aisément extensible vu que de nouveaux symboles de fonction, ainsi que de nouvelles contraintes de différents niveaux de granularité, peuvent être définis à partir de ceux que nous proposons et être par la suite intégrés au langage. Le langage de requêtes proposé permet ainsi la modélisation et la résolution d'une très large panoplie de contraintes. A la différences d'autres cadres génériques pour définir d'une façon flexible les contraintes en se basant sur des primitives (*primitive-based-framework*), il s'agit du premier cadre permettant la modélisation et la résolution des contraintes n-aires.

Nous proposons dans l'annexe A (page 127) les tableaux récapitulatifs de l'ensemble des contraintes et symboles de fonctions composant notre langage.

Chapitre 5

Approche *Hybride*

Sommaire

Nous proposons dans ce chapitre une première approche de résolution de contraintes n-aires, le but étant de montrer la faisabilité de notre approche. Pour cela, nous avons fait coopérer un extracteur de motifs locaux (MUSIC-DFS, cf. section 1.4.1) et l'outil de programmation par contraintes ECL^iPS^e (cf. section C.1). Ce croisement de méthodes et outils venant de deux domaines différents explique le nom *Hybride* que nous avons donné à notre méthode. À notre connaissance, il s'agit de la première approche générique et flexible pour la résolution des contraintes n-aires en fouille de données.

Ce chapitre commence par donner une vue d'ensemble d'*Hybride* (section 5.1) puis détaille ses trois étapes (sections 5.2, 5.3 et 5.4) en considérant les règles d'exception comme exemple illustratif (cf. section 4.3.1). La section 5.5 présente une validation expérimentale de cette approche.

Cette approche est présentée dans [Khiari *et al.*, 2009, Khiari *et al.*, 2010a, Khiari *et al.*, 2010b, Khiari *et al.*, 2012].

5.1 Aperçu général

L'idée clé de l'approche *Hybride* est de tirer profit des techniques efficaces d'extraction de motifs locaux et de laisser au solveur de CSP uniquement les contraintes non décomposables en contraintes locales. Plus précisément, *Hybride* repose sur les quatre constats suivants :

1. Une requête n-aire est une conjonction de contraintes locales (unaires) et de contraintes n-aires ($n > 1$).
2. Les contraintes locales peuvent être résolues par des méthodes classiques d'extraction de motifs locaux sous contraintes.
3. Les CSPEs offrent un cadre naturel et élégant pour modéliser les contraintes n-aires.

4. Nous disposons de ECL^iPS^e (cf. section C.1) qui dispose, entre autres, d'un solveur freeware permettant la résolution de CSPEs bâtis au-dessus de Prolog offrant ainsi la possibilité de développer rapidement un premier prototype.

La stratégie d'*Hybride* repose sur le partitionnement des contraintes en locales/n-aires, puis la résolution des des contraintes locales par un extracteur de motifs locaux (MUSIC-DFS) (cf. section 1.4.1, page 26), et la résolution des contraintes n-aires en utilisant ECL^iPS^e (cf. section C.1, page 131).

La figure 5.1 présente une vue d'ensemble des trois étapes de l'approche *Hybride* :

1. Modéliser la requête n-aire sous forme de CSP, puis distinguer les contraintes locales (portant sur un seul motif) des contraintes n-aires (portant sur plusieurs motifs).
2. Résoudre les contraintes locales à l'aide d'un extracteur de motifs locaux. Nous avons choisi MUSIC-DFS [1] [Soulet *et al.*, 2007] qui produit une représentation condensée par intervalles de tous les motifs satisfaisant les contraintes locales.
3. Résoudre les contraintes n-aires à l'aide des solveurs de contraintes d'ECL^iPS^e [2]. Le domaine de chaque variable résulte de la représentation condensée par intervalles calculée par la seconde étape.

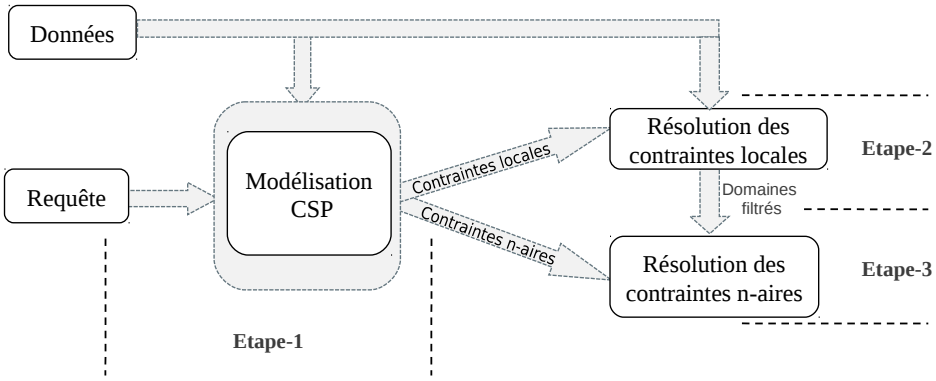

FIGURE 5.1 – Aperçu des 3 étapes de l'approche *Hybride*

5.2 Partitionner les contraintes

L'ensemble des contraintes composant la requête n-aire est décomposé en deux sous ensembles :

- $\mathcal{C}_{loc}$ est l'ensemble des contraintes locales à résoudre par MUSIC-DFS. Les solutions sont fournies sous forme d'une représentation condensée par intervalles selon la fermeture préfixée (cf. section 1.3.4).
- $\mathcal{C}_{n-aires}$ est l'ensemble des contraintes restantes à résoudre par ECL^iPS^e. Les domaines des variables X_i sont déduits de la représentation condensée par intervalles calculée dans la deuxième étape.

1. `http://www.info.univ-tours.fr/~soulet/music-dfs/music-dfs.html`
2. `http://www.eclipse-clp.org`

Exemple 5.2.1.

Nous illustrons tout au long de ce chapitre, l'approche Hybride par l'exemple des règles d'exceptions présenté dans la section 2.2.3.b et dont nous avons présenté la modélisation dans la section 4.6.1 obtenant la requête suivante :

$$exception([X_1, X_2, X_3, X_4]) \equiv \begin{cases} X_1 \subset X_3 \wedge \\ X_2 = X_1 \sqcup I \wedge \\ X_4 = X_3 \sqcup \neg I \wedge \\ \mathit{freq}(X_2) \geq min_{\mathit{freq}} \wedge \\ \mathit{freq}(X_1) - \mathit{freq}(X_2) \leq \delta_1 \wedge \\ \mathit{freq}(X_4) \leq max_{\mathit{freq}} \wedge \\ \mathit{freq}(X_3) - \mathit{freq}(X_4) \leq \delta_2 \end{cases}$$

La première étape consiste à partitionner l'ensemble des contraintes en contraintes locales $\mathcal{C}_{loc}$ et contraintes n-aires $\mathcal{C}_{n-aires}$ (cf. table 5.1). Cette partition s'effectue comme suit :

- $\mathcal{C}_{loc} = \{(\mathit{freq}(X_2) \geq min_{\mathit{freq}}),\ (\mathit{freq}(X_4) \leq max_{\mathit{freq}})\}$
- $\mathcal{C}_{n-aires} = \{(\mathit{freq}(X_1) - \mathit{freq}(X_2) \leq \delta_1),\ (X_2 = X_1 \sqcup I),\ (\mathit{freq}(X_3) - \mathit{freq}(X_4) \leq \delta_2),\ (X_4 = X_3 \sqcup \neg I),\ (X_1 \subset X_3)\}$

Forme normale	Locale	N-aire
$X_1 \subset X_3$		×
$X_2 = X_1 \sqcup I$		×
$X_4 = X_3 \sqcup \neg I$		×
$\texttt{freq}(X_2) \geq min_{\texttt{freq}}$	×	
$\texttt{freq}(X_1) - \texttt{freq}(X_2) \leq \delta_1$		×
$\texttt{freq}(X_4) \leq max_{\texttt{freq}}$	×	
$\texttt{freq}(X_3) - \texttt{freq}(X_4) \leq \delta_2$		×

TABLE 5.1 – Partition des contraintes en locales et n-aires

5.3 Résolution des contraintes locales

Les contraintes locales sont résolues avant et indépendamment des contraintes n-aires. Ainsi, l'espace de recherche des contraintes n-aires sera réduit à l'espace des solutions des contraintes locales. Ces solutions sont obtenues via MUSIC-DFS sous forme de représentation condensée par intervalles (cf. section 1.4.1).

L'ensemble des contraintes locales $\mathcal{C}_{loc}$ est composé d'une union disjointe de $\mathcal{C}_i$ (pour $i \in [1..n]$) où chaque $\mathcal{C}_i$ est l'ensemble des contraintes locales portant sur le motif X_i. Chaque $\mathcal{C}_i$ peut être résolu de manière séparée et indépendante. Soit CR_i la représentation condensée sous forme d'intervalles issue de la résolution de $\mathcal{C}_i$ par MUSIC-DFS, $CR_i = \bigcup_p (f_p, I_p)$ où chaque I_p est un intervalle ensembliste vérifiant : $\forall x \in I_p,\ \texttt{freq}(x) = f_p$. Les domaines des variables X_i sont définis par : $D_{X_i} = \bigcup_{I_p \in CR_i} I_p$

Exemple 5.3.1.

Reprenons l'exemple des règles d'exception en considérant le jeu de données de la table 5.2.

Trans.	Items					
t_1	A	B			c_1	
t_2	A	B			c_1	
t_3			C		c_1	
t_4			C		c_1	
t_5			C		c_1	
t_6	A	B	C	D		c_2
t_7			C	D		c_2
t_8			C			c_2
t_9				D		c_2

TABLE 5.2 – Jeu de données exemple

Considérons $\mathcal{C}_{loc}$ l'ensemble des contraintes locales relatives à la requête des règles d'exception, et prenons comme valeurs respectives de $(I, \neg I, min_{\mathit{freq}}, \delta_1, max_{\mathit{freq}}, \delta_2)$ les valeurs suivantes : $(c_1, c_2, 2, 1, 1, 0)$. L'ensemble des contraintes locales relatives à X_2, $\mathcal{C}_2 = \{\mathit{freq}(X_2) \geq 2\}$, est résolu par MUSIC-DFS *par la requête suivante :*

```
---------------
music-dfs data -q "freq(X2)>=2;"

Solutions sous forme de représentation condensée :

X2 in [Ac1..ABc1] U [CD..CDc2] U [Bc1..Bc1] -- freq(X2) = 2 ;
X2 in [A..AB] U [Cc1..Cc1] U [Cc2..Cc2] U
      [B..B] U [D..Dc2] -- freq(X2) = 3 ;
X2 in [c2..c2]  -- freq(X2) = 4 ;
X2 in [c1..c1]  -- freq(X2) = 5 ;
X2 in [C..C] -- freq(X2) = 6 ;
---------------
```

La contrainte $\mathit{freq}(X_2) \geq 2$ possède 15 solutions et sa représentation condensée comporte 11 intervalles. L'ensemble des motifs solution est ainsi le suivant :

- *Ac_1, ABc_1, CD, CDc_2 et Bc_1 de fréquence 2,*
- *A, AB, Cc_1, Cc_2, B, D et Dc_2 de fréquence 3,*
- *c_2 de fréquence 4,*
- *c_1 de fréquence 5,*
- *C de fréquence 6.*

À partir des contraintes n-aires, on peut parfois en déduire des contraintes locales redondantes (elles ne modifient pas l'ensemble des solutions du problème). Par contre, ajouter ces contraintes locales redondantes lors de la seconde étape (extraction des motifs locaux) va diminuer le nombre de solutions relatives aux contraintes locales obtenues par MUSIC-DFS *ce qui se traduit généralement par moins d'intervalles dans la représentation condensée obtenue. Par exemple, à $\mathcal{C}_2$, on peut ajouter la contrainte locale $c_1 \in X_2$. Cette contrainte est redondante car l'appartenance de c_1 à X_2 est aussi décrite dans la contrainte n-aire $(X_2 = X_1 \sqcup c_1)$. Cette requête possède 4 solutions et sa représentation condensée comporte 3 intervalle.*

```
---------------
music-dfs data -q "{c1} subset X2 and freq(X2)>=2;"

Solutions sous forme de représentation condensée :

X2 in [Ac1..ABc1] U [Bc1..Bc1] -- freq(X2) = 2 ;
X2 in [Cc1..Cc1] -- freq(X2) = 3 ;
---------------
```

Nous montrons dans la section 5.5 l'intérêt de la réduction du nombre d'intervalles produits par l'extracteur de motifs locaux.

5.4 Résolution des contraintes n-aires

Le domaine de chaque variable X_i est construit à partir de la représentation condensée CR_i extraite par MUSIC-DFS. La résolution des contraintes n-aires par les solveurs d'ECL^iPS^e permet alors d'obtenir l'ensemble de tous les motifs satisfaisant l'intégralité des contraintes.

Exemple 5.4.1. *Considérons $\mathcal{C}_{n-aires}$ l'ensemble des contraintes n-aires pour les règles d'exception, les valeurs respectives de $(I, \neg I, min_{\texttt{freq}}, \delta_1, maxfr, \delta_2)$ étant toujours les mêmes $(c_1, c_2, 2, 1, 1, 0)$. La session ECL^iPS^e ci-dessous illustre la requête permettant d'extraire toutes les paires de règles d'exception par backtracking :*

```
---------------
[eclipse 1]:
?- exceptionsRules(X1, X2, X3, X4).

Solutions :

X1={A,B}, X2={A,B,c1}, X3={A,B,C}, X4={A,B,C,c2};
X1={A,B}, X2={A,B,c1}, X3={A,B,D}, X4={A,B,D,c2};
X1={A,B}, X2={A,B,c1}, X3={A,B,C,D}, X4={A,B,C,D,c2};
.../...
---------------
```

Étant donné que X_2 représente la règle générale et X_4 son exception, la solution $X_1 = \{A, B\}, X_2 = \{A, B, c_1\}, X_3 = \{A, B, C\}, X_4 = \{A, B, C, c_2\}$ signifie que $AB \rightarrow c_1$ est une règle générale dont l'exception est $ABC \rightarrow c_2$.

5.5 Expérimentations

L'étude expérimentale porte sur le jeu de données *postoperative-patient-data* de UCI repository [3]. Ce jeu de données comporte $m = 90$ transactions décrites par $n = 23$ items et est caractérisé par deux classes (deux transactions appartenant à une troisième classe n'ont pas été considérées). *Hybride* est testée avec la requête des règles d'exception (dans ce qui suit, l'item I donné dans la définition des règles d'exception représente une des deux classes du jeu de données). Toutes les expériences ont été effectuées sur une machine dotée d'un processeur Intel Centrino Duo 2 GHz et de 2GB de mémoire RAM sous Linux.

Les figures 5.2 et 5.3 donnent respectivement le nombre de paires de règles d'exception en fonction de $min_{\texttt{freq}}$ et de δ_1. Plusieurs combinaisons de paramètres ont été testées.

3. www.ics.uci.edu/~mlearn/MLRepository.html

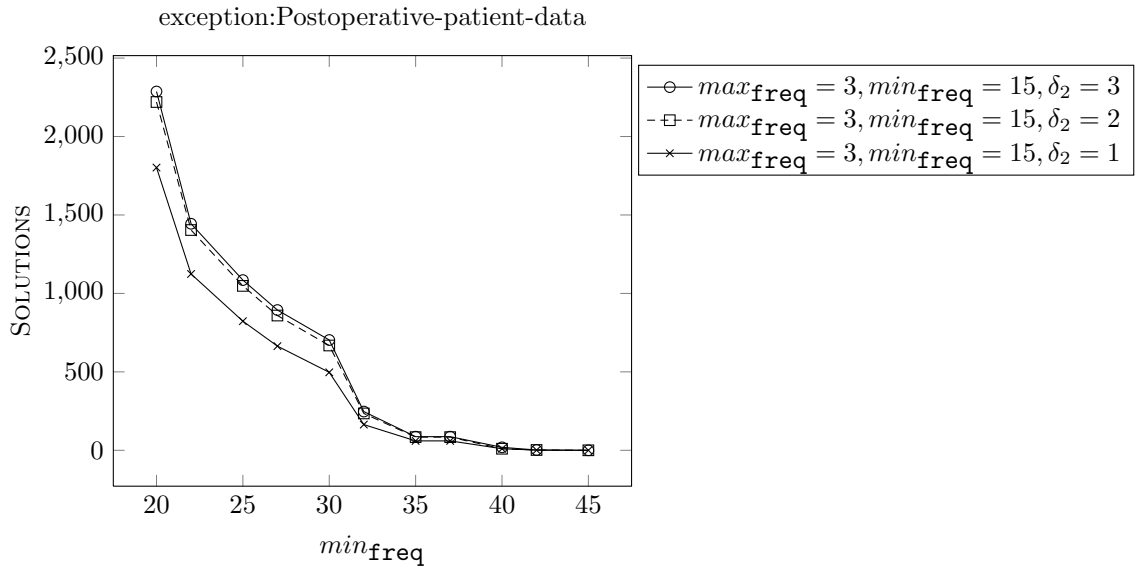

FIGURE 5.2 – Nombre de paires de règles en fonction de $min_{\texttt{freq}}$

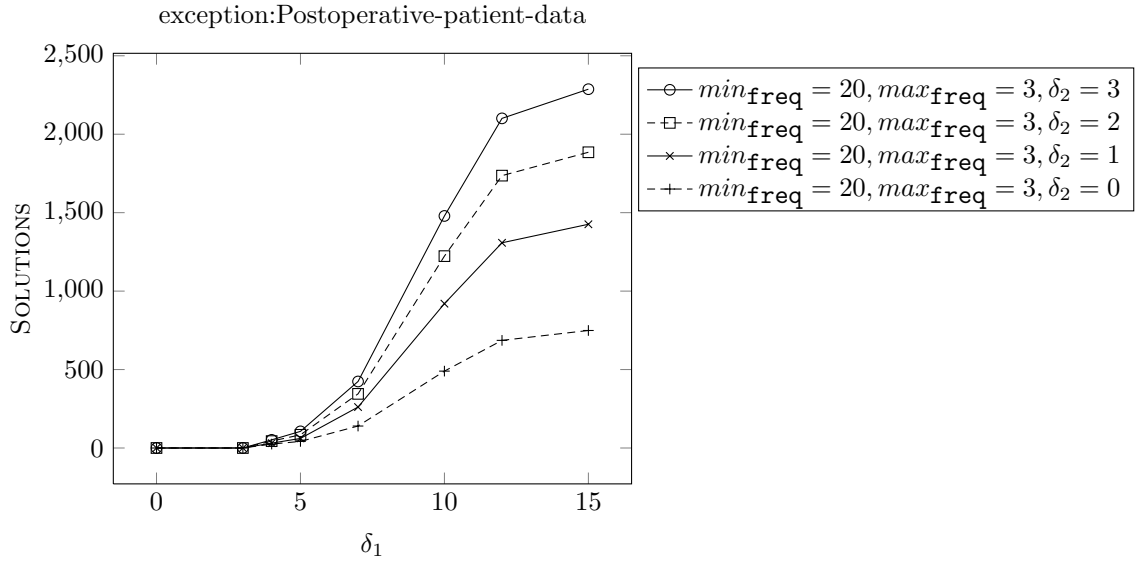

FIGURE 5.3 – Nombre de paires de règles en fonction de δ_1

Comme attendu, en diminuant la valeur de $min_{\texttt{freq}}$ on obtient un nombre plus élevé de paires de règles. On constate un comportement similaire en faisant varier δ_1 (cf. figure 5.3) : le nombre de paires de règles augmente en fonction de δ_1 (quand δ_1 croit, la valeur de la confiance des règles générales décroît, d'où l'obtention d'un plus grand nombre de paires).

Il est intéressant de noter que ces courbes mettent aussi en valeur la facilité de contrôle de la qualité des règles en modifiant les valeurs des paramètres. Par exemple,

pour $min_{\texttt{freq}}= 20, \delta_1 = 5$,$max_{\texttt{freq}}= 1, \delta_2 = 0$, on obtient 25 paires où la confiance de la règle générale est supérieure ou égale à 83% et des règles d'exception exactes (confiance égale à 100%). On constate que les paires de règles obtenues sont de très bonne qualité.

De plus, notre approche permet aisément l'ajout de nouvelles propriétés, toujours sous forme de contraintes, comme par exemple le contrôle de la taille de la règle générale relativement à celle de la règle d'exception (par exemple, au plus 3 items supplémentaires pour la règle d'exception par rapport à la règle générale).

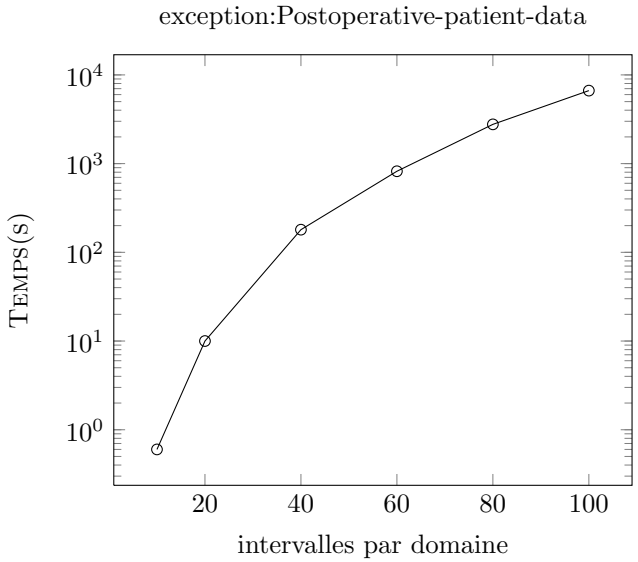

Figure 5.4 – Temps d'exécution en fonction du nombre d'intervalles par domaine

La figure 5.4 montre l'évolution du temps d'exécution en fonction du nombre d'intervalles par domaine. Pour chaque point de la courbe, les quatre variables ont le même domaine, et par conséquent le même nombre d'intervalles par domaine. Évidemment, le temps d'exécution augmente en fonction du nombre d'intervalles (on utilise une échelle algorithmique pour l'axe Y). Il est intéressant de noter que le temps d'exécution diminue quand on cherche des paires de règles de meilleure qualité. En effet, la recherche de règles générales de haute fréquence et des règles d'exception rares permet un meilleur élagage de l'espace de recherche et par suite un nombre d'intervalles moins important. Le tableau 5.3 décrit le nombre d'intervalles du domaine de la variable X_2 (voir section 4.6.1) par rapport à différentes contraintes locales.

Contrainte locale	Nombre d'intervalles dans D_{X_2}
-	3002
$I \in X_2$	1029
$I \in X_2 \wedge \texttt{freq}(X_2) >= 20$	52
$I \in X_2 \wedge \texttt{freq}(X_2) >= 25$	32

Table 5.3 – Nombre d'intervalles pour différentes contraintes locales (cas de D_{X_2})

5.6 Discussion

Les solveurs de CSP ensemblistes [Gervet, 1997] ont du mal à gérer les unions d'intervalles. Afin d'établir la consistance aux bornes, les solveurs de CSPE approximent l'union de deux intervalles par leur enveloppe convexe (cf. section 3.2.4). L'enveloppe convexe de $[lb_1 .. ub_1]$ et $[lb_2 .. ub_2]$ étant définie par $[lb_1 \cap lb_2 .. ub_1 \cup ub_2]$. Ainsi, si le filtrage s'applique de nombreuses fois sur le domaine d'une même variable, ce domaine peut rapidement être approximé par l'ensemble de tous les motifs possibles $[\emptyset .. \mathcal{I}]$. Afin de contourner ce problème, pour chaque variable X_i ayant pour représentation condensée $CR_i = \bigcup_p (f_p, I_p)$, la recherche s'effectue successivement sur chacun des I_p. Mais, si cette approche demeure correcte et complète, elle ne permet pas de profiter pleinement du filtrage car les retraits de valeurs non-viables se propagent uniquement sur les intervalles traités, et non pas sur l'intégralité des domaines. Ceci explique les résultats de la section 5.5 qui montrent que les temps d'exécution augmentent fortement en fonction du nombre d'intervalles. Cette constatation pourrait paraître rédhibitoire, mais le nombre d'intervalles ensemblistes décroît rapidement lors de la prise en compte des contraintes locales (cf. table 5.3).

Une approche alternative serait d'utiliser des représentations condensées non exactes afin de réduire le nombre d'intervalles ensemblistes par domaine, comme par exemple une représentation condensée reposant sur les motifs fréquents maximaux (cf. section 1.3.1). Dans ce cas, le nombre d'intervalles par domaine serait beaucoup plus petit, mais en raison de l'approximation de l'union de deux intervalles par leur enveloppe convexe, il serait nécessaire de gérer les valeurs non viables qui en résultent.

5.7 Conclusion

Nous avions présenté dans le chapitre 4 une modélisation générale sous forme d'un CSP d'un problème d'extraction de motifs sous contraintes, ainsi qu'une collection de primitives permettant de modéliser une large panoplie de requêtes n-aires.

Dans ce chapitre, nous avons proposé une première approche de résolution (l'approche *Hybride*) se basant sur l'utilisation conjointe d'un extracteur de motifs locaux et d'un solveur de contraintes. Le lien entre le système de contraintes et la base de données se fait à travers l'extracteur de motifs locaux utilisé. Ainsi, un prototype a pu être rapidement développé en utilisant ECL^iPS^e. À notre connaissance, il s'agit de la première approche générique permettant de modéliser et de résoudre une large panoplie de contraintes n-aires. De plus, cette approche permet de bénéficier des avancées récentes sur les extracteurs de motifs locaux.

Néanmoins, les outils utilisés avec *Hybride* n'offrent pas la possibilité de calculer simplement le support d'un motif donné, et il n'est pas alors possible de résoudre efficacement les contraintes portant sur les ensembles de transactions. D'où la nécessité de mettre en œuvre une approche permettant de gérer aussi bien les motifs que leurs supports, et offrant une meilleure efficacité de calcul. C'est, entre autre, l'objectif du chapitre suivant.

Chapitre 6

Approche *Pure-CP*

Sommaire

Dans ce chapitre, nous présentons la deuxième approche de résolution que nous avons proposée. Contrairement à *Hybride* (cf. chapitre 5), cette approche n'utilise pas d'extracteurs de motifs locaux, mais uniquement un solveur de contraintes, d'où son appellation *approche Pure-CP*.

Pour cela, des variables de décision, portant sur les motifs n-aires recherchés ainsi que sur leurs supports respectifs, sont tout d'abord introduites. Puis, chaque primitive (fonction ou contrainte) de notre langage (cf. chapitre 4) est reformulée en contraintes de plus bas niveau portant sur les variables de décision introduites. La nouvelle modélisation ainsi obtenue est alors soumise au solveur de contraintes *Gecode* (cf. C.2, page 132).

Pure-CP permet de couvrir une plus large collection de contraintes en offrant la possibilité de poser des contraintes sur les motifs aussi bien que sur les transactions étendant ainsi la panoplie des primitives traitées par *Hybride* (cf. section 5.7). De plus, *Pure-CP* permet une très nette amélioration des performances, en terme de temps de calcul, par rapport à *Hybride*.

La première section présente un aperçu général de l'approche *Pure-CP*. La section 6.3 donne la reformulation des contraintes composant notre langage de requêtes, étape in-

dispensable, pour pouvoir les résoudre avec l'approche *Pure-CP*. La section 6.4 présente l'étude expérimentale réalisée.

Cette approche est présentée dans [Khiari *et al.*, 2010c, Khiari *et al.*, 2010, Khiari *et al.*, 2011].

6.1 Aperçu général

L'approche *Pure-CP* est composée des étapes suivantes : Nous commençons par modéliser les k motifs recherchés à l'aide de variables de décision, puis nous montrons comment les liens sont établis, via des contraintes réifiées, entre les variables représentant les motifs, celles représentant les transactions et la base de données. Nous proposons une reformulation des principales contraintes numériques, ensemblistes et dédiées que nous avons présentées dans la section 4.1. Finalement, la résolution est effectuée par le solveur de CSP *Gecode* (cf. figure 6.1).

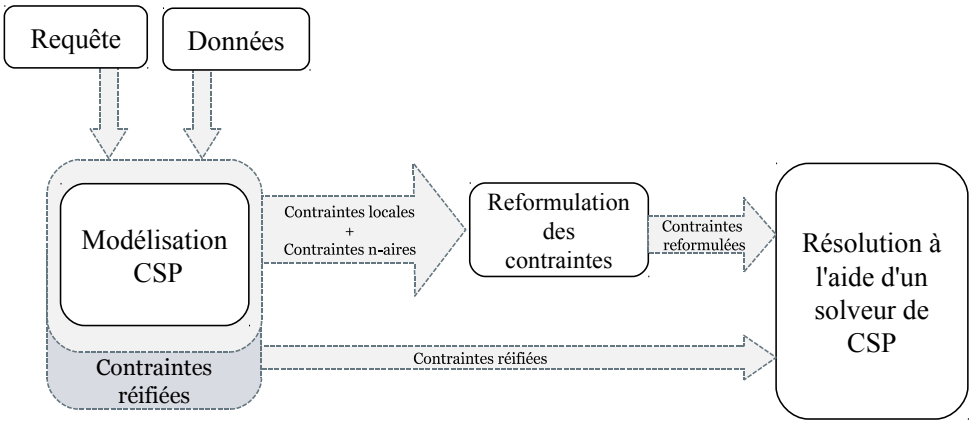

FIGURE 6.1 – Aperçu des étapes de l'approche *Pure-CP*

6.2 Modélisation des k motifs recherchés

Nous étendons dans ce qui suit la modélisation associée à FIM_CP pour les contraintes locales (cf. section 1.4.2, page 27) au cas des contraintes portant sur k motifs ($k \geq 1$).

Soit $\boldsymbol{r}$ un contexte transactionnel où $\mathcal{I}$ est l'ensemble de ses n items et $\mathcal{T}$ l'ensemble de ses m transactions. Soit $\boldsymbol{d}$ la matrice 0/1 de dimension (m, n) telle que $\forall t \in \mathcal{T}, \forall i \in \mathcal{I}$, $(d_{t,i} = 1)$ ssi $(i \in t)$. Soient $X_1, X_2, ..., X_k$ les k motifs recherchés.

6.2.1 Modélisation des motifs inconnus

Chaque motif inconnu X_j $(1 \leq j \leq k)$ est modélisé dans le CSP par n variables de décision $\{X_{1,j}, X_{2,j}, ..., X_{n,j}\}$ (de domaine $\{0, 1\}$) telles que $(X_{i,j} = 1)$ ssi l'item i appartient au motif X_j :

$$\forall i \in \mathcal{I}, (X_{i,j} = 1) \quad ssi \quad (i \in X_j) \tag{6.1}$$

Puis, le support T_{X_j} $(1 \leq j \leq k)$ de chaque motif X_j est représenté par m variables de décision $\{T_{1,j}, T_{2,j}, ..., T_{m,j}\}$ (de domaine $\{0,1\}$) qui sont associées à X_j comme suit : $(T_{t,j} = 1)$ ssi $(X_j \subseteq t)$:

$$\forall t \in \mathcal{T}, (T_{t,j} = 1) \;\; ssi \;\; (X_j \subseteq t) \tag{6.2}$$

Similairement au cas unaire, cette représentation permet de modéliser simplement des mesures de valeurs telles que :

- la fréquence d'un motif X_j : $\texttt{freq}(X_j) = \sum_{t \in \mathcal{T}} T_{t,j}$,
- la cardinalité d'un motif X_j : $\texttt{size}(X_j) = \sum_{i \in \mathcal{I}} X_{i,j}$.

La relation entre chaque motif recherché X_j $(1 \leq j \leq k)$, son support T_{X_j} et la base de données $\boldsymbol{r}$ est établie via des contraintes réifiées (cf. section 3.1.2, page 43) imposant que, pour chaque transaction t, $(T_{t,j} = 1)$ ssi $(X_j \subseteq t)$, ce qui se reformule en :

$$\forall j \in [1..k], \forall t \in \mathcal{T}, (T_{t,j} = 1) \Leftrightarrow \sum_{i \in \mathcal{I}} X_{i,j} \times (1 - d_{t,i}) = 0 \tag{6.3}$$

Cette équation vérifie la connexion de Galois g entre chaque motif X_j et son support T_{X_j} : $g(X_j) = T_{X_j}$ (cf. section 1.2.5.b).

Démonstration 6.2.1.

Un motif X_j $(1 \leq j \leq k)$, inclus dans une transaction t, est décrit par l'expression suivante :

$$\begin{aligned}
&\forall i \in \mathcal{I}, (i \in X_j) \Rightarrow (i \in t) \\
&\quad ssi \quad \forall i \in \mathcal{I}, (i \notin t) \Rightarrow (i \notin X_j) \\
&\quad ssi \quad \forall i \in \mathcal{I}, (d_{t,i} = 0) \Rightarrow (X_{i,j} = 0) \\
&\quad ssi \quad \forall i \in \mathcal{I}, (1 - d_{t,i} = 1) \Rightarrow (X_{i,j} = 0) \\
&\quad ssi \quad \forall i \in \mathcal{I}, X_{i,j} \times (1 - d_{t,i}) = 0 \\
&\quad ssi \quad \textstyle\sum_{i \in \mathcal{I}} X_{i,j} \times (1 - d_{t,i}) = 0
\end{aligned}$$

D'où le lien entre un motif X_j, son support T_{X_j} et la base de données $\boldsymbol{r}$ (décrite par la matrice $\boldsymbol{d}$), s'exprime par l'équivalence suivante :

$\forall t \in \mathcal{T}, (T_{t,j} = 1) \Leftrightarrow \sum_{i \in \mathcal{I}} X_{i,j} \times (1 - d_{t,i}) = 0$

En généralisant pour tout motif X_j tel que $j \in [1..k]$, on obtient :

$\forall j \in [1..k], \forall t \in \mathcal{T}, (T_{t,j} = 1) \Leftrightarrow \sum_{i \in \mathcal{I}} X_{i,j} \times (1 - d_{t,i}) = 0$

Remarque 6.2.1. *Pour une requête portant sur k motifs inconnus, il faut donc :*

- *$(k \times n) + (k \times m)$ variables de décision représentant respectivement les motifs inconnus et leur supports correspondants,*
- *$(k \times m)$ contraintes réifiées, sachant qu'une contrainte réifiée est d'arité au plus $(n+1)$.*

6.2.2 Exemple des règles d'exception

Exemple 6.2.1.

Considérons le jeu de données $\boldsymbol{r}$ (cf. tableau 6.1) décrit par l'ensemble de transactions $\mathcal{T} = \{t_1, t_2, \ldots, t_9\}$ et l'ensemble d'items $\mathcal{I} = [1..6]$ où les items A, B, C, D, c_1, c_2, sont numérotés de 1 à 6.

Trans.	Items				
t_1	A	B		c_1	
t_2	A	B		c_1	
t_3			C	c_1	
t_4			C	c_1	
t_5			C	c_1	
t_6	A	B	C	D	c_2
t_7			C	D	c_2
t_8			C		c_2
t_9				D	c_2

$\mathcal{T} \setminus \mathcal{I}$	1	2	3	4	5	6
t_1	1	1	0	0	1	0
t_2	1	1	0	0	1	0
t_3	0	0	1	0	1	0
t_4	0	0	1	0	1	0
t_5	0	0	1	0	1	0
t_6	1	1	1	1	0	1
t_7	0	0	1	1	0	1
t_8	0	0	1	0	0	1
t_9	0	0	0	1	0	1

TABLE 6.1 – Base de données transactionnelle $\boldsymbol{r}$.

Nous illustrons l'approche Pure-CP par l'exemple des règles d'exception :

$$exception([X_1, X_2, X_3, X_4]) \equiv \begin{cases} X_1 \subset X_3 \wedge \\ X_2 = X_1 \sqcup I \wedge \\ X_4 = X_3 \sqcup \neg I \wedge \\ \mathit{freq}(X_2) \geq min_{\mathit{freq}} \wedge \\ \mathit{freq}(X_1) - \mathit{freq}(X_2) \leq \delta_1 \wedge \\ \mathit{freq}(X_4) \leq max_{\mathit{freq}} \wedge \\ \mathit{freq}(X_3) - \mathit{freq}(X_4) \leq \delta_2 \end{cases}$$

A cette étape de l'écriture du modèle, nous ne nous intéressons pas encore aux contraintes composant la requête, mais uniquement au nombre k de motifs utilisés pour la modéliser. Nous avons aussi besoin des dimensions m (nombre de transactions) et n (nombre d'items) du jeu de données dans lequel sera effectuée l'extraction.

Les valeurs correspondantes à la requête $exception([X_1, X_2, X_3, X_4])$ appliquée au jeu de données $\boldsymbol{r}$ (cf. tableau 6.1) sont les suivantes :

- $k = 4$
- $n = 6$
- $m = 9$

À chaque motif X_j ($j \in [1..4]$), sont associées :

- *($n = 6$) variables $X_{1,j}, X_{2,j}, ..., X_{6,j}$ de domaines $\{0, 1\}$,*
- *($m = 9$) variables $T_{1,j}, T_{2,j}, ..., T_{6,j}$ de domaines $\{0, 1\}$ représentant les supports T_{X_j}.*

Ensuite, nous appliquons l'équation 6.3 pour générer l'ensemble de contraintes réifiées permettant d'établir le lien entre chaque motif X_j, son support T_{X_j} et la base de données associée ($1 \leq j \leq 4$).

L'ensemble des contraintes réifiées obtenu est le suivant :

- **Lien entre le motif X_1, son support T_{X_1} et $\boldsymbol{r}$:**
  $(T_{1,1} = 1) \Leftrightarrow (X_{3,1} = 0 \wedge X_{4,1} = 0 \wedge X_{6,1} = 0)$
  $(T_{2,1} = 1) \Leftrightarrow (X_{3,1} = 0 \wedge X_{4,1} = 0 \wedge X_{6,1} = 0)$
  $(T_{3,1} = 1) \Leftrightarrow (X_{1,1} = 0 \wedge X_{2,1} = 0 \wedge X_{4,1} = 0 \wedge X_{6,1} = 0)$
  $(T_{4,1} = 1) \Leftrightarrow (X_{1,1} = 0 \wedge X_{2,1} = 0 \wedge X_{4,1} = 0 \wedge X_{6,1} = 0)$

$(T_{5,1} = 1) \Leftrightarrow (X_{1,1} = 0 \wedge X_{2,1} = 0 \wedge X_{4,1} = 0 \wedge X_{6,1} = 0)$
$(T_{6,1} = 1) \Leftrightarrow (X_{5,1} = 0)$
$(T_{7,1} = 1) \Leftrightarrow (X_{1,1} = 0 \wedge X_{2,1} = 0 \wedge X_{5,1} = 0)$
$(T_{8,1} = 1) \Leftrightarrow (X_{1,1} = 0 \wedge X_{2,1} = 0 \wedge X_{4,1} = 0 \wedge X_{5,1} = 0)$
$(T_{9,1} = 1) \Leftrightarrow (X_{1,1} = 0 \wedge X_{2,1} = 0 \wedge X_{3,1} = 0 \wedge X_{5,1} = 0)$

- **Lien entre le motif X_2, son support T_{X_2} et r :**
$(T_{1,2} = 1) \Leftrightarrow (X_{3,2} = 0 \wedge X_{4,2} = 0 \wedge X_{6,2} = 0)$
$(T_{2,2} = 1) \Leftrightarrow (X_{3,2} = 0 \wedge X_{4,2} = 0 \wedge X_{6,2} = 0)$
$(T_{3,2} = 1) \Leftrightarrow (X_{1,2} = 0 \wedge X_{2,2} = 0 \wedge X_{4,2} = 0 \wedge X_{6,2} = 0)$
$(T_{4,2} = 1) \Leftrightarrow (X_{1,2} = 0 \wedge X_{2,2} = 0 \wedge X_{4,2} = 0 \wedge X_{6,2} = 0)$
$(T_{5,2} = 1) \Leftrightarrow (X_{1,2} = 0 \wedge X_{2,2} = 0 \wedge X_{4,2} = 0 \wedge X_{6,2} = 0)$
$(T_{6,2} = 1) \Leftrightarrow (X_{5,2} = 0)$
$(T_{7,2} = 1) \Leftrightarrow (X_{1,2} = 0 \wedge X_{2,2} = 0 \wedge X_{5,2} = 0)$
$(T_{8,2} = 1) \Leftrightarrow (X_{1,2} = 0 \wedge X_{2,2} = 0 \wedge X_{4,2} = 0 \wedge X_{5,2} = 0)$
$(T_{9,2} = 1) \Leftrightarrow (X_{1,2} = 0 \wedge X_{2,2} = 0 \wedge X_{3,2} = 0 \wedge X_{5,2} = 0)$

- *Les liens pour X_3 et X_4 sont établis de manière analogue.*

Nous présentons dans la section 6.3.5, la modélisation des différentes contraintes de la requête exception$([X_1, X_2, X_3, X_4])$.

6.3 Reformulation des contraintes

Une fois établi le lien entre les différentes variables de décision et la base de données, chaque contrainte de la requête à résoudre doit alors être modélisée à l'aide de ces variables de décision. Néanmoins, contrairement à *Hybride* où les variables du CSP à résoudre représentent directement les motifs, l'utilisation de variables de décision pour représenter les motifs nécessite de passer par une étape de reformulation afin de pouvoir résoudre les contraintes exprimées avec notre langage de requêtes.

La section suivante détaille cette étape de reformulation des contraintes.

6.3.1 Contraintes sur les motifs

Nous proposons, dans cette section, une reformulation des contraintes et symboles de fonction composant notre langage de requêtes.

6.3.1.a Les contraintes numériques

Considérons un opérateur $\texttt{op} \in \{<, \leq, >, \geq, =, \neq\}$; la table 6.2 récapitule les reformulation des contraintes numériques.

Contrainte	Reformulation
$\texttt{freq}(X_p)$ op α	$\sum_{t \in \mathcal{T}} T_{t,p}$ op α
$\texttt{size}(X_p)$ op α	$\sum_{i \in \mathcal{I}} X_{i,p}$ op α

TABLE 6.2 – Reformulation des contraintes numériques

Exemple 6.3.1.

Considérons la base de données de la table 6.1 et la requête $\mathit{freq}(X_1) \geq 3 \wedge \mathit{size}(X_1) \geq 2$*, une solution de cette requête est* $X_1 = \{1,1,0,0,0,0\}$ *(représentant le motif AB) ayant comme support* $T_{X_1} = \{1,1,0,0,0,1,0,0,0\}$.

Ainsi :

- size *(*X_1*)=2,*
- freq *(*X_1*)=3,*
- $\mathit{support}(X_1) = \{t_1, t_2, t_6\}$.

6.3.1.b Les contraintes ensemblistes

Les contraintes ensemblistes (telles que égalité, inclusion, appartenance,...) se reformulent directement à l'aide de contraintes linéaires. La table 6.3 récapitule ces reformulations.

Contrainte	**Reformulation**
$X_p = X_q$	$\forall i \in \mathcal{I}, X_{i,p} = X_{i,q}$
$X_p \subseteq X_q$	$\forall i \in \mathcal{I}, X_{i,p} \leq X_{i,q}$
$X_p \subset X_q$	$\forall i \in \mathcal{I}, X_{i,p} \leq X_{i,q} \ \wedge$ $\sum_{i\in\mathcal{I}} X_{i,p} < \sum_{i\in\mathcal{I}} X_{i,q}$
$i_o \in X_p$	$X_{i_0,p} = 1$
$X_p \cap X_q = \emptyset$	$\forall i \in \mathcal{I}, X_{i,p} + X_{i,q} \leq 1$
$X_p \cup X_q = X_r$	$\forall i \in \mathcal{I},$ $X_{i,p}+X_{i,q}-2\times X_{i,r} \geq -1 \wedge$ $X_{i,p} + X_{i,q} - 2 \times X_{i,r} \leq 0$
$X_p \cap X_q = X_r$	$\forall i \in \mathcal{I},$ $2\times X_{i,r}-X_{i,p}-X_{i,q} \geq -1 \wedge$ $2 \times X_{i,r} - X_{i,p} - X_{i,q} \leq 0$
$X_p \backslash X_q = X_r$	$\forall i \in \mathcal{I},$ $2\times X_{i,r}+X_{i,p}-X_{i,q} \geq -1 \wedge$ $2 \times X_{i,r} + X_{i,p} - X_{i,q} \leq 0$

TABLE 6.3 – Reformulation des contraintes ensemblistes sur les motifs

6.3.2 Contraintes sur les transactions

Une des forces de notre approche est la possibilité de pouvoir spécifier des contraintes portant aussi bien sur les motifs que sur les transactions les supportant. Les représentations des X_i et des T_{X_i} étant similaires, toutes les opérations sur les X_i utilisées pour formuler les contraintes ensemblistes précédemment présentées, sont applicables aussi sur les T_{X_i}. La table 6.4 récapitule les contraintes portant sur les supports.

Contrainte	Reformulation
$T_{X_p} = T_{X_q}$	$\forall t \in \mathcal{T}, T_{t,p} = T_{t,q}$
$T_{X_p} \subseteq T_{X_q}$	$\forall t \in \mathcal{T}, T_{t,p} \leq T_{t,q}$
$T_{X_p} \subset T_{X_q}$	$\forall t \in \mathcal{T}, T_{t,p} \leq T_{t,q} \ \wedge$ $\sum_{t\in\mathcal{T}} T_{t,p} < \sum_{t\in\mathcal{T}} T_{t,q}$
$t_o \in T_{X_p}$	$T_{t_0,p} = 1$
$T_{X_p} \cap T_{X_q} = \emptyset$	$\forall t \in \mathcal{T}, T_{t,p} + T_{t,q} \leq 1$
$T_{X_p} \cup T_{X_q} = T_{X_r}$	$\forall t \in \mathcal{T},$ $T_{t,p} + T_{t,q} - 2 \times T_{t,r} \geq -1 \wedge$ $T_{t,p} + T_{t,q} - 2 \times T_{t,r} \leq 0$
$T_{X_p} \cap T_{X_q} = T_{X_r}$	$\forall t \in \mathcal{T},$ $2 \times T_{t,r} - T_{t,p} - T_{t,q} \geq -1 \wedge$ $2 \times T_{t,r} - T_{t,p} - T_{t,q} \leq 0$
$T_{X_p} \backslash T_{X_q} = T_{X_r}$	$\forall t \in \mathcal{T},$ $2 \times T_{t,r} + T_{t,p} - T_{t,q} \geq -1 \wedge$ $2 \times T_{t,r} + T_{t,p} - T_{t,q} \leq 0$

TABLE 6.4 – Reformulation des contraintes ensemblistes sur les supports

6.3.3 Les contraintes dédiées

Cette section présente la reformulation d'un ensemble de contraintes dédiées à la fouille de données et présentées dans la section 4.2.4.

6.3.3.a `closed`(X_j)

- **Rappel :** La contrainte `closed`(X_j) est satisfaite ssi X_j est un motif fermé (cf. section 1.2.5)
- **Reformulation :**

> `closed`(X_j) est satisfaite
> *ssi*
> $\forall i \in \mathcal{I} : (X_{i,j} = 1) \Longleftrightarrow \sum_{t\in\mathcal{T}} T_{t,j}(1 - d_{t,i}) = 0$

- **Remarque :** Pour cette contrainte, comme pour toutes les autres, la reformulation proposée suppose que la contrainte de couverture définie par l'équation 6.3 (cf. page 89) est déjà imposée.

Démonstration 6.3.1.
D'après la définition 1.2.8, un motif X_j est fermé ssi $h(X_j) = X_j$.
Or $h(X_j) = (f \circ g)(X_j) = f(g(X_j))$
Nous avons déjà établi en utilisant l'équation 6.3 que $g(X_j) = T_{X_j}$
Ainsi, vérifier que X_j est un motif fermé revient à vérifier la conjonction suivante : $g(X_j) = T_{X_j} \ \wedge \ f(T_{X_j}) = X_j$

La première partie de la conjonction étant vérifiée par l'équation 6.3 (page 89), nous montrons dans ce qui suit que l'ensemble de tous les items communs aux transactions de T_{X_j} (son intention ou $f(T_{X_j})$ (cf. section 1.2.5.b)) est X_j.

Un item i de X_j $(1 \leq j \leq k)$, est supporté par toutes les transactions t de T_{X_j}, est décrit par l'expression suivante :

$$\forall t \in \mathcal{T}, (t \in T_{X_j}) \Rightarrow (i \in t)$$

$$ssi \quad \forall t \in \mathcal{T}, (i \notin t) \Rightarrow (t \notin T_{X_j})$$

$$ssi \quad \forall t \in \mathcal{T}, (d_{t,i} = 0) \Rightarrow (T_{t,j} = 0)$$

$$ssi \quad \forall t \in \mathcal{T}, (1 - d_{t,i} = 1) \Rightarrow (T_{t,j} = 0)$$

$$ssi \quad \forall t \in \mathcal{T}, T_{t,j} \times (1 - d_{t,i}) = 0$$

$$ssi \quad \sum_{t \in \mathcal{T}} T_{t,j} \times (1 - d_{t,i}) = 0$$

D'où, en généralisant, l'ensemble de tous les items communs à groupe de transactions T_{X_j} (ou $f(T_{X_j})$) doit vérifier l'équivalence suivante :

$$\forall i \in \mathcal{I}, (X_{i,j} = 1) \Longleftrightarrow \sum_{t \in \mathcal{T}} T_{t,j} \times (1 - d_{t,i}) = 0$$

6.3.3.b `coverTransactions`$([X_1, X_2, ..., X_k])$

- **Rappel :** La contrainte `coverTransactions`$([X_1, X_2, ..., X_k])$ impose que les motifs X_j $(1 \leq j \leq k)$ couvrent toutes les transactions t de $\mathcal{T}$.
- **Reformulation :** Quelle que soit la transaction $t \in \mathcal{T}$, il existe au moins un X_j couvrant t, or X_j couvre t ssi $(T_{t,j} = 1)$ d'où :

> `coverTransactions`$([X_1, X_2, ..., X_k])$ est satisfaite
> *ssi*
> $\forall t \in \mathcal{T}, \sum_{1 \leq j \leq k} T_{t,j} \geq 1$

.

- **Remarque :** Une transaction peut appartenir aux supports de plusieurs $X_j (1 \leq j \leq k)$.

6.3.3.c `noOverlapTransactions`$([X_1, X_2, ..., X_k])$

- **Rappel :** La contrainte `noOverlapTransactions`$([X_1, X_2, ..., X_k])$ impose que les supports des motifs X_j sont deux à deux disjoints.
- **Reformulation :** Quelle que soit la transaction $t \in \mathcal{T}$, il existe au plus un X_j couvrant t, or X_j couvre t ssi $(T_{t,j} = 1)$ d'où :

> `noOverlapTransactions`$([X_1, X_2, ..., X_k])$ est satisfaite
> *ssi*
> $\forall t \in \mathcal{T}, \sum_{1 \leq j \leq k} T_{t,j} \leq 1$

- **Remarque :** Certaines transactions peuvent ne pas être couvertes.

6.3.3.d `coverItems`$([X_1, X_2, ..., X_k])$

- **Rappel :** La contrainte `coverItems`$([X_1, X_2, ..., X_k])$ impose que les motifs X_j couvrent tous les items i de $\mathcal{I}$.
- **Reformulation :** Quel que soit l'item $i \in \mathcal{I}$, il existe au moins un motif X_j contenant i, or $i \in X_j$ ssi $(X_{i,j} = 1)$ d'où :

$$\texttt{coverItems}(X_1, X_2, ..., X_k) \text{ est satisfaite}$$
$$ssi$$
$$\forall i \in \mathcal{I}, \sum_{1 \leq j \leq k} X_{i,j} \geq 1.$$

- **Remarque :** Un item peut appartenir à plusieurs motifs.

6.3.3.e `noOverlapItems`$([X_1, X_2, ..., X_k])$

- **Rappel :** La contrainte `noOverlapItems`$([X_1, X_2, ..., X_k])$ impose que les motifs X_j $(1 \leq j \leq k)$ soient deux à deux disjoints.
- **Reformulation :** Quel que soit l'item $i \in \mathcal{I}$, chaque item i appartient à au plus un motif X_j, or $i \in X_j$ ssi $(X_{i,j} = 1)$ d'où :

$$\texttt{noOverlapItems}(X_1, X_2, ..., X_k) \text{ est satisfaite}$$
$$ssi$$
$$\forall i \in \mathcal{I}, \sum_{1 \leq j \leq k} X_{i,j} \leq 1$$

- **Remarque :** Certains items peuvent n'appartenir à aucun motif.

6.3.3.f `canonical`$([X_1, X_2, ..., X_k])$

- **Rappel :** La contrainte `canonical`$([X_1, X_2, ..., X_k])$ impose que les motifs X_j soient triés suivant l'ordre lexicographique croissant.
- **Reformulation :** Cette contrainte est modélisée en utilisant la contrainte prédéfinie "$<$" de *Gecode*.

$$\texttt{canonical}([X_1, X_2, ..., X_k]) \text{ est satisfaite}$$
$$ssi$$
$$\forall_{1 \leq j < k},\ X_j < X_{j+1}$$

6.3.4 Les symboles de fonction

6.3.4.a `support`

- **Rappel :** `support`(X_p) définit le support du motif X_p.
- **Reformulation :** Le support du motif X_p est représenté par le vecteur de variables de décision $(T_{t,p})_{t \in \mathcal{T}}$ (i.e., T_{X_p}).

6.3.4.b `freq`$'$

- **Rappel :** `freq`$'(X_p, D_j)$ représente la mesure de la fréquence d'un motif X_p dans une portion D_j de la base de données.

– **Reformulation :** $\texttt{freq}'(X_p, D_j) = \sum_{t \in \mathcal{T}_{D_j}} T_{t,p}$.

6.3.4.c `ovelapItems`

– **Rappel :** $\texttt{ovelapItems}(X_p, X_q)$ est le nombre d'items communs entre deux motifs X_p et X_q.
– **Reformulation :** $\texttt{ovelapItems}(X_p, X_q) = \texttt{size}(X_p \cap X_q)$: modélisé en utilisant les opérateurs `size` et $\cap$ précédemment définis.

6.3.4.d `overlapTransactions`

– **Rappel :** $\texttt{overlapTransactions}(X_p, X_q)$ est le nombre de transactions couvertes à la fois par X_p et X_q

– **Reformulation :**
$\texttt{overlapTransactions}(X_p, X_q) = \texttt{size}(\texttt{support}(X_p) \cap \texttt{support}(X_q))$
$= \texttt{size}(T_{X_p} \cap T_{X_q})$: modélisé en utilisant les opérateurs `size` et $\cap$ précédemment définis.

6.3.5 Exemple des règles d'exception

Exemple 6.3.2.

La table 6.5 récapitule la reformulation de toutes les contraintes de la requête d'extraction de règles d'exception (cf. section 6.2.2).

Forme normale	**Reformulation *Pure-CP***
$X_1 \subset X_3$	$\forall i \in \mathcal{I},\ X_{i,1} \leq X_{i,3} \wedge$ $(\sum_{i \in \mathcal{I}} X_{i,1} < \sum_{i \in \mathcal{I}} X_{i,3})$
$X_2 = X_1 \sqcup I$	$\forall i \in \mathcal{I} \setminus \{5\}, X_{i,2} = X_{i,1} \wedge$ $X_{5,2} = 1 \wedge$ $X_{5,1} = 0$
$X_4 = X_3 \sqcup \neg I$	$\forall i \in \mathcal{I} \setminus \{6\}, X_{i,4} = X_{i,3} \wedge$ $X_{6,4} = 1 \wedge$ $X_{6,3} = 0$
$\texttt{freq}(X_2) \geq min_{\texttt{freq}}$	$\sum_{t \in \mathcal{T}} T_{t,2} \geq min_{\texttt{freq}}$
$\texttt{freq}(X_1) - \texttt{freq}(X_2) \leq \delta_1$	$\sum_{t \in \mathcal{T}} (T_{t,1} - T_{t,2}) \leq \delta_1$
$\texttt{freq}(X_4) \leq max_{\texttt{freq}}$	$\sum_{t \in \mathcal{T}} T_{t,4} \leq max_{\texttt{freq}}$
$\texttt{freq}(X_3) - \texttt{freq}(X_4) \leq \delta_2$	$\sum_{t \in \mathcal{T}} (T_{t,3} - T_{t,4}) \leq \delta_2$

TABLE 6.5 – Modélisation *Pure-CP* des règles d'exception

6.4 Expérimentations

Cette section montre la faisabilité et les apports pratiques de l'approche *Pure-CP* présentée dans ce chapitre.

6.4.1 Protocole expérimental

Différentes expérimentations ont été menées sur plusieurs jeux de données de l'UCI *repository*[1] ainsi que sur un jeu de données réelles nommé `Meningitis` et provenant de l'Hôpital Universitaire de Grenoble (`Meningitis` regroupe, sur une période de quatre ans, l'ensemble des enfants atteints d'une méningite virale ou bactérienne). La table 6.6 résume les différentes caractéristiques de ces jeux de données.

Les expérimentations ont été menées sur plusieurs contraintes n-aires : règles d'exception, règles rares et conflits de classification. Nous présentons aussi les résultats relatifs à la résolution de la requête de clustering q_3 (cf. section 4.5.2.c) :

$$\mathtt{q}_3([X_1,...,X_k]) \equiv \begin{cases} \wedge_{1\leq i\leq k}\ \mathtt{closed}(X_i) \wedge \\ \mathtt{coverTransactions}([X_1,...,X_k]) \wedge \\ \mathtt{noOverlapTransactions}([X_1,...,X_k]) \wedge \\ \mathtt{canonical}([X_1,...,X_k]) \wedge \\ \wedge_{1\leq i\leq k}\ \mathtt{freq}(X_i) \geq min_{\mathtt{freq}} \wedge \\ \wedge_{1\leq i\leq k}\ \mathtt{size}(X_i) \geq min_{\mathtt{size}} \end{cases}$$

La machine utilisée est un PC 2.83 GHz Intel Core 2 Duo processor (4 GB de RAM) sous Ubuntu Linux.

Jeu de données	#trans	#items	densité
`Mushroom`	8142	117	0.18
`Australian`	690	55	0.25
`Meningitis`	329	84	0.27
`Soybean`	630	50	0.32
`Vote`	435	48	0.33
`Zoo`	101	36	0.44

TABLE 6.6 – Description des jeux de données

6.4.2 Faisabilité de l'approche

La figure 6.2 décrit l'évolution du nombre de règles d'exception en fonction des seuils $min_{\mathtt{freq}}$ et δ_1 pour les jeux de données `Mushroom` et `Meningitis`. La figure 6.3 décrit l'évolution du nombre de conflits de classification en fonction des seuils $min_{\mathtt{freq}}$ et $min_{\mathtt{conf}}$ pour les jeux de données `Mushroom` et `Australian`. Nous avons aussi testé d'autres valeurs de ces paramètres ainsi que d'autres jeux de données. Mais, comme les résultats obtenus sont similaires, ils ne sont pas indiqués ici.

Comme attendu, plus $min_{\mathtt{freq}}$ est petit, plus le nombre de règles d'exception est grand. Les résultats sont similaires quand δ_1 varie. Plus δ_1 est grand, plus le nombre de

1. `http://www.ics.uci.edu/~mlearn/MLRepository.html`

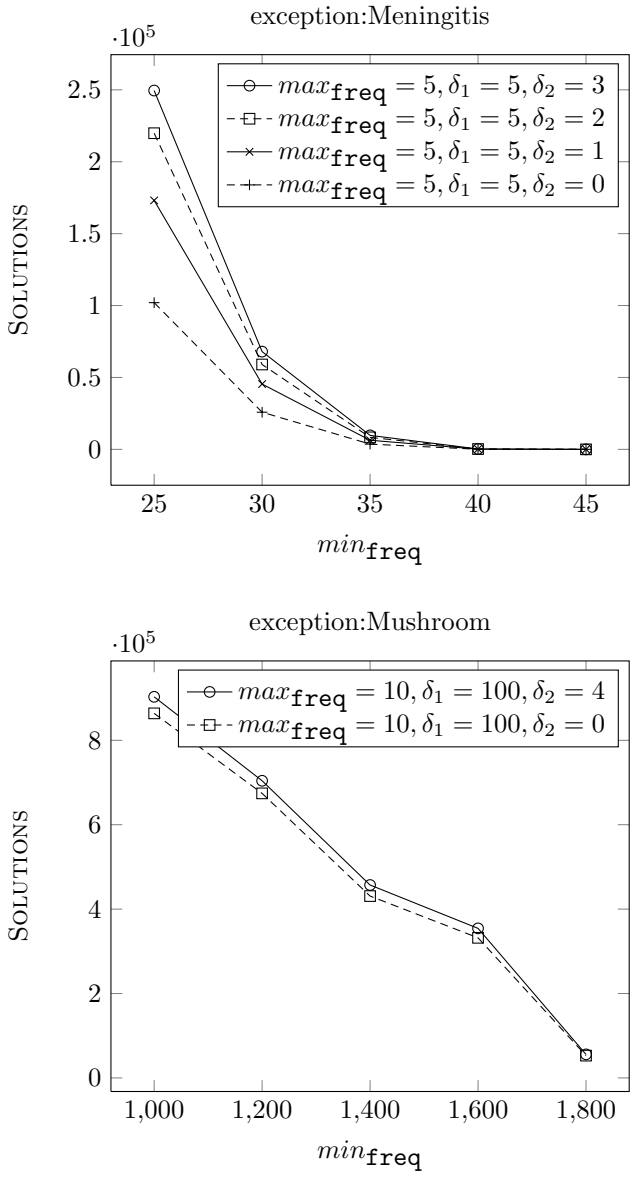

Figure 6.2 – Evolution du nombre de règles d'exception

règles d'exception augmente (quand δ_1 augmente, la confiance décroît et il y a donc un plus grand nombre de règles générales).

Finalement, la table 6.7 résume les résultats obtenus avec un exemple de requête de clutering (la requête q_3) montrant ainsi la faisabilité de ce la résolution de ce genre de requêtes à l'aide de l'approche *Pure-CP*.

Notons que comme la résolution effectuée par le solveur de contraintes est correcte et complète, notre approche permet d'extraire l'ensemble correct et complet des motifs solutions.

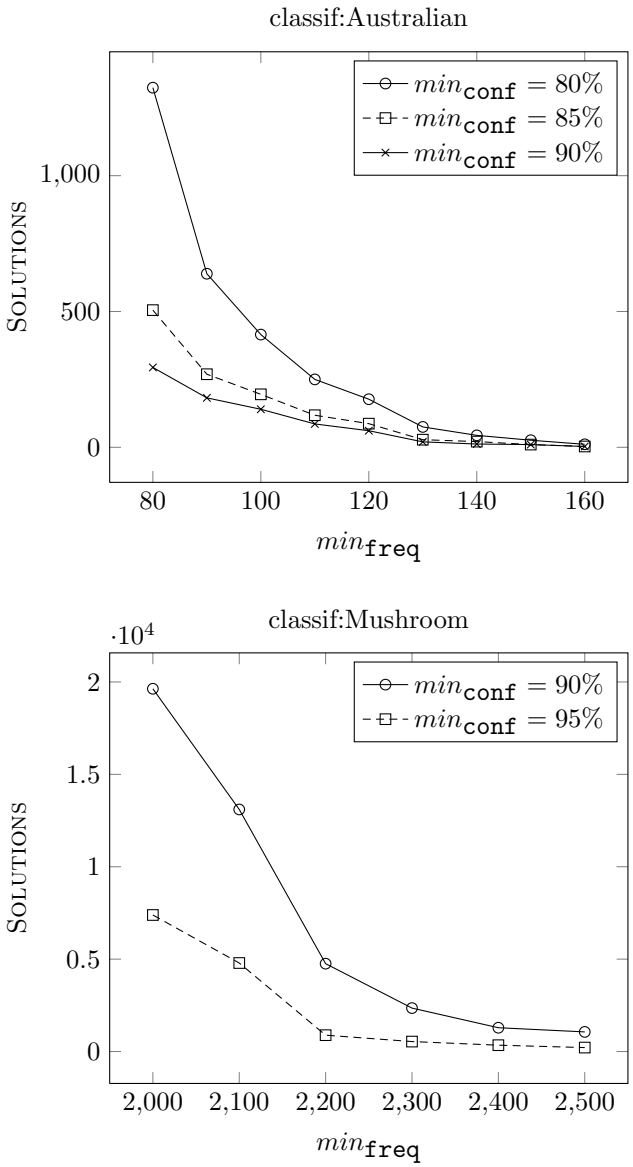

FIGURE 6.3 – Evolution du nombre de conflits de classification

6.4.3 Extraction de motifs pertinents

Les règles rares sont des règles de très faible fréquence. Ces règles peuvent être simplement de l'artefact comme elle peuvent aussi contenir des informations intéressantes. C'est en particulier le cas en médecine, où par exemple, il est important de repérer les symptômes non usuels ou les effets indésirables exceptionnels qui peuvent se déclarer chez un patient pour une pathologie ou un traitement donné. Les règles d'exception sont un exemple de règles rares. Mais, même s'il existe quelques travaux permettant d'extraire les règles rares [Szathmary *et al.*, 2007], il est impossible de distinguer les règles d'exception à partir de l'ensemble des règles rares. C'est une limitation forte car la plupart des règles rares sont peu fiables, d'où l'intérêt des règles d'exception et de leur extraction. La fi-

Dataset	nc	$min_{\texttt{size}}$	**Solutions**	**Temps**
Mushroom	2	1	4	8.16 s
		2	4	8.19 s
	3	1	14	$3h26mn10s$
		2	14	$3h26mn14s$
Soybean	2	1	3	0.08 s
		2	0	0.09 s
	3	1	16	$1mn25s$
		2	10	$1mn25s$
Zoo	2	1	14	0.01 s
		2	1	0.01 s
	3	1	190	6.87 s
		2	123	6.78 s
Meningite	2	1	8	0.1 s
		2	0	0.09 s
	3	1	54	$48mn11s$
		2	0	$47mn58s$
Vote	2	1	0	0.6 s
		2	0	0.6 s
	3	1	2	$7mn35s$
		2	0	$7mn35s$

TABLE 6.7 – Résultats expérimentaux pour la requête q_3 de clustering ($min_{\texttt{freq}} = 10\% \times m$)

gure 6.4 compare, sur le jeu de données `Meningitis`, le nombre de règles d'exception par rapport au nombre de règles rares en fonction du seuil de fréquence $min_{\texttt{freq}}$ (le nombre de règles rares correspond à la ligne en haut de la figure 6.4). Savoir rechercher directement les règles d'exception permet de réduire de manière drastique (plusieurs ordres de magnitude) le nombre de motifs obtenus (noter que l'axe des ordonnées suit une échelle logarithmique).

Les règles inattendues sont également sources d'informations utiles. Un exemple d'une telle règle découverte sur le jeu de données `Meningitis` est la règle :
`(pas de signes neurologiques` $\wedge$ `taux de polynucléaires immatures élevé)`
$\rightarrow$ `taux de polynucléaires neutrophiles normal`

Cette règle est inattendue par rapport à la croyance que des valeurs élevées à la fois de la numération des leucocytes et du taux de polynucléaires sont en forte faveur d'une méningite d'étiologie bactérienne. Les experts apprécient de disposer des contraintes n-aires pour la découverte de tels motifs.

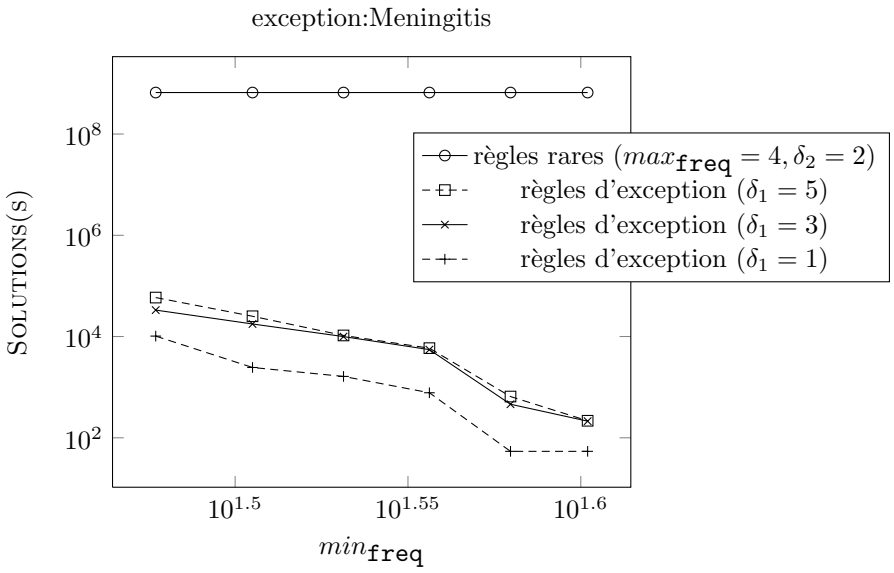

FIGURE 6.4 – Nombre de règles d'exception vs nombre de règles rares (`Meningitis`)

6.4.4 Efficacité

Ces expérimentations permettent aussi de quantifier les temps de calcul relatifs à notre approche. En pratique, les temps de calcul varient en fonction de la taille des jeux de données et de la dureté des contraintes[2]. C'est le cas des contraintes définies par des seuils de fréquence et de confiance élevés.

Pour `Meningitis` et `Australian` (cf. figures 6.5 et 6.6), l'ensemble de toutes les solutions est obtenu en quelques secondes (aux alentours d'une seconde dans la plupart des cas).

Pour `Mushroom`, les temps de calcul varient de quelques secondes pour les contraintes dures à environ 1 heure pour des seuils très faibles de fréquence et de confiance où le nombre de solutions obtenues est très grand. Les figures 6.5, 6.6, 6.7 et 6.8 décrivent les temps de calcul obtenus.

Un premier constat est que plus une contrainte est dure, plus les temps de calcul sont petits. Les temps de calcul dépendent aussi de la taille du jeu de données traité : plus il est grand, plus le nombre de contraintes réifiées (équation 6.3) est élevé (cf. section 6.5).

En ce qui concerne les résultats de nos expérimentations sur le clustering (cf. table 6.7), on constate que l'efficacité de la résolution est pénalisée lorsque le nombre de clusters recherché augmente. Ceci est dû au fait que plus le nombre de variables de décision croît linéairement en fonction du nombre k de clusters (cf. remarque 6.2.1, page 89).

Bien évidemment, notre approche pourrait être utilisée pour extraire des motifs locaux. Nous obtenons dans ce cas les mêmes temps de calcul que [De Raedt *et al.*, 2008] qui sont compétitifs vs les extracteurs de motifs locaux. Enfin, pour les règles d'exception, nous n'avons pas pu effectuer de comparaison avec la méthode ad hoc car les temps de calcul ne sont pas indiqués dans [Suzuki, 2002].

2. Une contrainte est dure si le ratio entre son nombre de solutions et la cardinalité du produit cartésien des domaines de ses variables est faible.

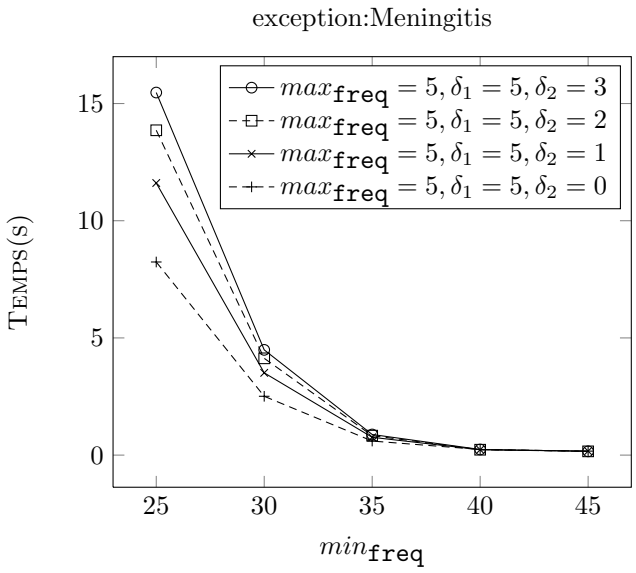

FIGURE 6.5 – Temps d'exécution-1

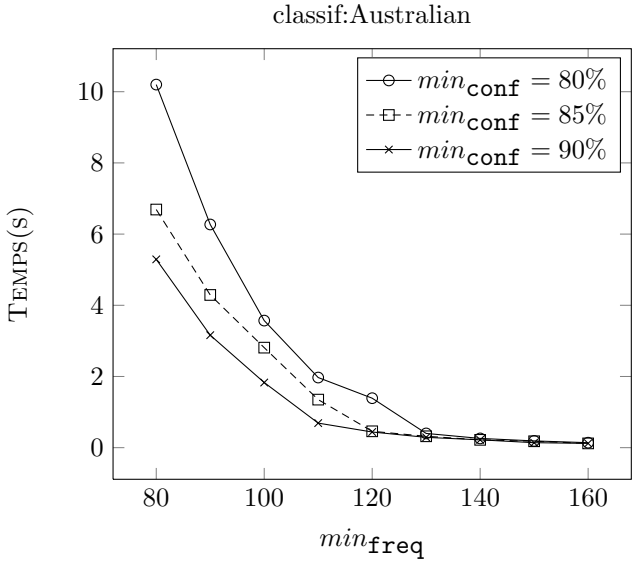

FIGURE 6.6 – Temps d'exécution-2

6.5 Comparaison entre *Hybride* et *Pure-CP*

Si l'approche *Hybride* (cf. chapitre 5) et l'approche *Pure-CP* présentée dans ce chapitre reposent sur une même modélisation générale, leurs mises en oeuvre diffèrent. *Hybride* utilise un extracteur de motifs locaux afin de construire la représentation condensée sous forme d'intervalles de tous les motifs satisfaisant les contraintes locales. Cette représentation condensée permet de construire les domaines des variables du CSP ensembliste

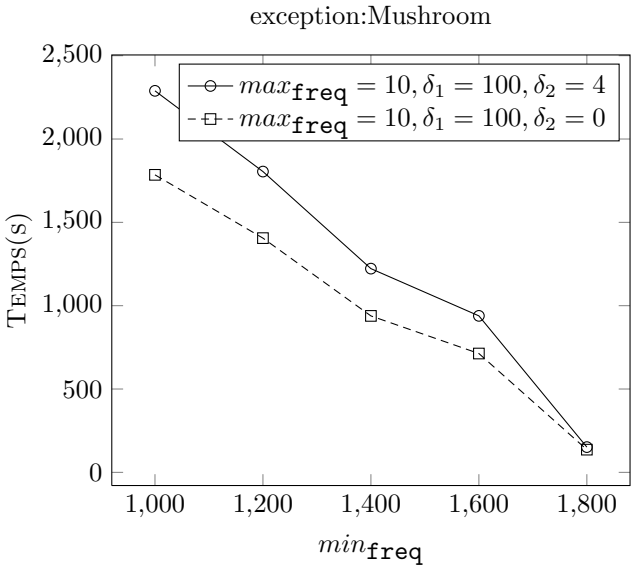

Figure 6.7 – Temps d'exécution-3

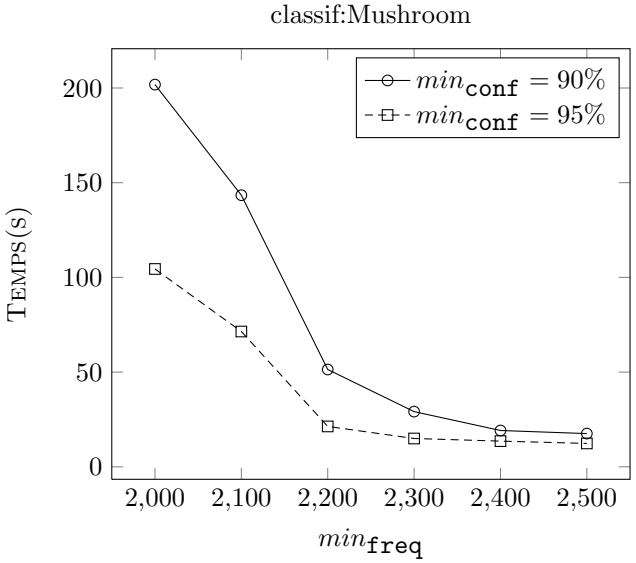

Figure 6.8 – Temps d'exécution-4

dont les variables représentent les motifs recherchés. Nous avons présenté une discussion sur les performances de l'approche *Hybride* dans la section 5.6.

L'approche *Pure-CP* établit, quant à elle, directement le lien entre l'ensemble des transactions et les motifs recherchés à l'aide de contraintes réifiées. Le CSP obtenu est directement résolu avec un solveur de CSP (sans l'aide d'un extracteur de motifs locaux). De plus, le propagateur des contraintes réifiées implanté en *Gecode* est très efficace. Mais, le nombre total de contraintes peut être très élevé pour des jeux de données de

très grande taille. Considérons un jeu de données ayant n items, m transactions et une contrainte n-aire portant sur k motifs inconnus. Lier les transactions et les motifs nécessite $(k \times m)$ contraintes réifiées, chacune d'entre elles portant sur au plus $(n+1)$ variables (cf. équation 6.3 à la section 6.2.1). Ainsi, le nombre total de contraintes nécessaires à la modélisation de problèmes de très grande taille pourrait s'avérer prohibitif même si, en pratique, comme nous venons de le voir, *Pure-CP* permet de traiter des problèmes avec plusieurs milliers de transactions et quelques centaines d'items.

La différence fondamentale entre *Hybride* et *Pure-CP* réside dans leur façon de considérer l'ensemble des transactions. *Hybride* utilise un extracteur de motifs locaux et le CSP résultant possède un très petit nombre de contraintes et de variables, mais ces variables ont des domaines de très grande taille. À l'inverse, *Pure-CP* requiert un grand nombre de contraintes portant sur des variables de décision. Une voie médiane reste à trouver entre ces deux approches afin de pouvoir traiter des problèmes de très grande taille.

6.6 Conclusion

Nous avons proposé une approche *Pure-CP* permettant de modéliser et d'extraire des motifs décrits par des requêtes n-aires en fouille de données utilisant uniquement des techniques PPC. A notre connaissance, il s'agit de la première de ce genre. La modélisation décrite dans ce chapitre et les exemples traités illustrent la généricité et la flexibilité de notre approche permettant notamment de résoudre l'intégralité des contraintes écrites à l'aide du langage de requêtes présenté dans le chapitre 4. Les expérimentations menées (cf. section 6.4) montrent sa pertinence et sa faisabilité.

Néanmoins, ce cadre se base sur les CSPs où toutes les variables sont quantifiées existentiellement. Or certaines requêtes n-aires requièrent la quantification universelle pour être modélisées de manière concise et élégante, comme par exemple la requête *peak* (cf. section 2.2.3.c). Nous explorons dans le chapitre 7 une première approche pour résoudre de telles requêtes utilisant les CSPs Quantifiés (QCSP) [Benedetti *et al.*, 2008, Bordeaux et Monfroy, 2002].

Chapitre 7

Approche QCSP

Sommaire

Nous présentons dans ce chapitre une nouvelle approche de résolution, utilisant le langage de requêtes que nous avons proposé dans le chapitre 4, et étendant l'approche *Pure-CP* proposée dans le chapitre 6 en permettant de modéliser des requêtes contenant des quantifications universelles.

7.1 Motivations

Les variables d'un CSP sont toutes quantifiées existentiellement. Dans le cadre d'extraction de motifs sous contraintes utilisant une modélisation CSP, les seuls cas envisageables sont :

- **Existe-il** un motif vérifiant une propriété locale P (motifs locaux) ?
- **Existe-il** un motif vérifiant une propriété P' définie par rapport à un ensemble de motifs préalablement connus ou faisant partie des inconnues du problème (motifs n-aires) ?

Mais, dans les problèmes d'extraction de motifs sous contraintes, on trouve souvent des contraintes du type :

- **Existe-il** un motif X vérifiant une propriété P **quel que soit** un motif Y vérifiant une propriété P' (P' est généralement définie en fonction de X) ?

Pour la modélisation de telles requêtes, nous avons besoin d'utiliser la *quantification universelle* ($\forall$) que les CSPs classiques ne peuvent prendre en compte. D'où le recours à l'utilisation des Problèmes de Satisfaction de Contraintes Quantifiées (QCSP) qui sont certes plus difficiles à résoudre mais strictement plus expressifs que les CSPs (cf. section 3.3).

7.2 Exemples de requêtes

7.2.1 Motifs pics (*peak*)

Dans la section 2.2.3.c (page 36), nous avons présenté les motifs pics [Crémilleux et Soulet, 2008] comme exemple de requête n-aire définie à l'aide de la quantification universelle. Un motif pic est un motif qui possède une valeur, pour une mesure m donnée, assez grande par rapport à celles de tous ses voisins. Le voisinage d'un motif étant calculé par rapport à une distance d donnée. Le motif pic a ainsi un comportement exceptionnel par rapport à ses voisins.

Soit $\mathtt{m} : \mathcal{L} \rightarrow \mathbb{R}$ une mesure, δ un entier et ρ un réel, et soit d une distance, on a :

$$peak(X) \equiv \begin{cases} vrai & \text{si } \forall Y \in \mathcal{L}_{\mathcal{I}} \;\; t.q \;\; d(X,Y) < \delta, \mathtt{m}(X) \geq \rho \times \mathtt{m}(Y) \\ faux & \text{sinon} \end{cases}$$

Exemple 7.2.1 (pic d'émergence)**.**

Les motifs pics peuvent être recherchés relativement à diverses mesures comme l'aire (cf. exemple 2.2.2, page 37), le taux de croissance des motifs émergents, etc.

Les motifs émergents [Dong et Li, 1999] sont des motifs dont la fréquence varie fortement entre deux classes. Ils caractérisent les classes de manière quantitative et qualitative. De par leur capacité à faire ressortir les distinctions entre classes, les motifs émergents permettent de construire des classifieurs ou de proposer une aide au diagnostic. L'émergence d'un motif est quantifiée par la mesure de son taux de croissance (growth rate) entre deux classes.

Définition 7.2.1 (Taux de croissance d'un motif)**.**

Soient D_1et $D_2 \subseteq \mathcal{T}$ les ensembles de transactions correspondantes respectivement aux classes c_1 et c_2 et soit $\mathtt{freq'}(X, D_j)$ la fréquence du motif X dans D_j, le taux de croissance de X dans D_1 est :

$$\mathtt{growth\text{-}rate}_{D_1}(X) = \frac{|D_2| \times \mathtt{freq'}(X,D_1)}{|D_1| \times \mathtt{freq'}(X,D_2)}$$

Définition 7.2.2 (Motif émergent)**.**

Soient un taux de croissance minimum ρ et une base de données $\mathbf{r}$ partitionnée en deux sous ensembles D_1 et D_2. Le motif X est émergent de D_2 vers D_1 ssi $\mathtt{growth\text{-}rate}_{D_1}(X) \geq \rho$

Soit $m(X) = \mathtt{growth\text{-}rate}_{D_1}(X)$, $\delta = 1$, $\rho = 2$ et $d(X,Y) = |X \backslash Y| + |Y \backslash X|$, et considérons le jeu de données décrit par la table 7.1.

Le motif **AB** *est un pic puisque $\mathtt{growth\text{-}rate}_{D_1}(AB) = 3$ est au moins 2 fois plus grand que les taux de croissance de ses 4 voisins A, B, ABC et ABD qui sont respectivement* 1.5, 1.5, 0.75 *et* 1.5.

Trans.	Items					
t_1	A	B	C		c_1	
t_2	A	B		D	c_1	
t_3	A	B		D	c_1	
t_4	A	B			c_1	
t_5		B		D		c_2
t_6	A	B	C	D		c_2
t_7	A					c_2

TABLE 7.1 – Jeu de données exemple

7.2.2 Groupes de synexpression

Nous avons présenté une première version de requête aidant à découvrir les groupes de synexpression dans la section 4.4.2. La quantification universelle apporte l'attrait supplémentaire de pouvoir extraire les groupements maximaux, renforçant ainsi l'hypothèse que l'ensemble des motifs agglomérés forme un groupe de synexpression. La requête obtenue est la suivante :

$$synexpression([X_1,...,X_k]) \equiv \begin{cases} \forall\ 1 \leq i < j \leq k, \\ \big(\texttt{area}(X_i) > min_{\texttt{area}} \wedge \\ \texttt{area}(X_j) > min_{\texttt{area}} \wedge \\ \texttt{coverage}(X_i, X_j) > \alpha \times min_{\texttt{area}}\big) \wedge \\ \forall \mathbf{Z}\ \ \mathbf{t.q}\ \ \texttt{freq}(\mathbf{Z}) > min_{\texttt{freq}}\ \mathbf{et}\ \texttt{area}(\mathbf{Z}) > min_{\texttt{area}}, \\ \exists \mathbf{i} \in [\mathbf{1..k}], \texttt{coverage}(\mathbf{Z}, \mathbf{X_i}) < \alpha \times min_{\texttt{area}} \end{cases}$$

Pour l'écriture sous forme normale de cette requête, on a besoin des variables suivantes :

- $\{X_1, X_2, ..., X_k, Z\}$ variables représentant les motifs,
- $\{i_3, i_4, ..., i_k, i\}$ variables compteurs.

La requête des groupes de synexpression permettant l'extraction de groupements de taille k $Synexpression([X_1,...,X_k])$ peut ainsi être modélisée par la formule suivante :

$$\begin{cases} \exists X_1,\ [\texttt{freq}(X_1) > min_{\texttt{freq}}, \texttt{area}(X_1) > min_{\texttt{area}}] \\ \quad \exists X_2,\ [\texttt{freq}(X_2) > min_{\texttt{freq}} \wedge \texttt{area}(X_2) > min_{\texttt{area}} \wedge X_1 \neq X_2 \wedge \\ \qquad\qquad \texttt{coverage}(X_1, X_2) > \alpha \times min_{\texttt{area}}] \\ \quad\ \exists X_3\ [\texttt{freq}(X_3) > min_{\texttt{freq}} \wedge \texttt{area}(X_3) > min_{\texttt{area}} \wedge X_1 \neq X_3\ \wedge\ X_2 \neq X_3] \\ \quad\ \forall i_3 \in [1..2],\ [\texttt{coverage}(X_3, X_{i_3}) > \alpha \times min_{\texttt{area}}] \\ \quad\ ... \\ \quad\ \exists X_k\ [\texttt{freq}(X_k) > min_{\texttt{freq}} \wedge \texttt{area}(X_k) > min_{\texttt{area}} \wedge \\ \qquad\qquad X_1 \neq X_k\ \wedge\ ...\ \wedge\ X_{k-1} \neq X_k] \\ \quad\ \forall i_k \in [1..k-1],\ [\texttt{coverage}(X_k, X_{i_k}) > \alpha \times min_{\texttt{area}}] \\ \quad\ \forall Z,\ [\texttt{freq}(Z) > min_{\texttt{freq}} \wedge \texttt{area}(Z) > min_{\texttt{area}} \wedge \\ \qquad\qquad X_1 \neq Z\ \wedge\ ...\ \wedge\ X_k \neq Z] \\ \quad\ \exists i \in [1..k], \\ \qquad \texttt{coverage}(Z, X_i) < \alpha \times min_{\texttt{area}} \end{cases}$$

7.3 Approche QCSP

7.3.1 Exemples de modélisation

Dans cette section, nous montrons comment modéliser les contraintes de pic et de synexpression à l'aides des QCSP.

Exemple 7.3.1 (peak).

Cet exemple détaille la modélisation de la requête peak. Rappelons tout d'abord l'énoncé de cette contrainte. Soit $\mathtt{m} : \mathcal{L} \rightarrow \mathbb{R}$ *une mesure d'intérêt (taux d'émergence, aire,...),* δ *un entier et* ρ *un réel, et soit d une distance, on a :*

$$peak(X) \equiv \begin{cases} vrai & si\ \forall Y \in \mathcal{L}_{\mathcal{I}}\ tel\ que\ d(X,Y) \leq \delta,\ \mathtt{m}(X) \geq \rho \times \mathtt{m}(Y) \\ faux & sinon \end{cases}$$

Pour modéliser ce problème sous forme normale, on introduit les variables suivantes :

- X_1 *représentant le motif X quantifiée existentiellement*
- X_2 *représentant le motif Y représentant le voisinage de X et quantifiée universellement*

le requête peak est alors modélisée par le QCSP$^+$ *suivant :*

$Q = \Big(\big[(\exists, X_1, C_1),\ (\forall, X_2, C_2)\big],\ Goal = C_3 \Big)$

où :

- $C_1 = \emptyset$
- $C_2 = d(X_1, X_2) \leq \delta$
- $C_3 = \mathtt{m}(X_1) \geq \rho \times \mathtt{m}(X_2)$

Exemple 7.3.2 (Groupes de synexpression).

Nous considérons par exemple le cas où on veut spécifier un recouvrement composé de trois motifs X_1, X_2, X_3*. La modélisation de la requête des synexpressions dans ce cas demande l'utilisation des variables suivantes :*

- $\{X_1, X_2, X_3, Z\}$ *variables représentant les motifs.*
- $\{i_3, i\}$ *variables compteurs.*

La requête des groupes de synexpression dans le cas de la vérification d'un recouvrement composé de trois motifs s'écrit comme suit :

$$Syn() \equiv \left\{ \begin{array}{l} \exists X_1\ [\mathtt{freq}(X_1) > min_{\mathtt{freq}} \wedge\ \mathtt{area}(X_1) > min_{\mathtt{area}}] \\ \quad \exists X_2,\ [\mathtt{freq}(X_2) > min_{\mathtt{freq}} \wedge\ \mathtt{area}(X_2) > min_{\mathtt{area}} \wedge X_1 \neq X_2 \wedge \\ \quad \mathtt{coverage}(X_1, X_2) > \alpha \times min_{\mathtt{area}}] \\ \quad\ \exists X_3,\ [\mathtt{freq}(X_3) > min_{\mathtt{freq}} \wedge\ \mathtt{area}(X_3) > min_{\mathtt{area}} \wedge X_1 \neq X_3\ \wedge\ X_2 \neq X_3] \\ \quad\quad \forall i_3 \in [1..2],\ [\mathtt{coverage}(X_3, X_{i_3}) > \alpha \times min_{\mathtt{area}}] \\ \quad\quad\ \forall Z\ [\mathtt{freq}(Z) > min_{\mathtt{freq}} \wedge\ \mathtt{area}(Z) > min_{\mathtt{area}} \wedge X_1 \neq Z\ \wedge\ \ldots\ \wedge\ X_3 \neq Z] \\ \quad\quad\quad \exists i \in [1..3], \\ \quad\quad\quad\ \mathtt{coverage}(Z, X_i) < \alpha \times min_{\mathtt{area}} \end{array} \right.$$

La requête est finalement modélisée par le QCSP$^+$ suivant :

$Q = \Big(\big[(\exists, X_1, C_1), (\exists, (X_2, A_2), C_2),\ (\exists, (X_3), C_3), (\forall, (i_3 \in [1..2], A_3), C_4),$
$(\forall, (Z), C_5), (\exists, (i \in [1..3], B), C_6) \big],\ Goal = C_7 \Big)$

Où :

- $C_1 = \mathit{freq}(X_1) > min_{\mathit{freq}} \wedge \mathit{area}(X_1) > min_{\mathit{area}}$
- $C_2 = \mathit{freq}(X_2) > min_{\mathit{freq}} \wedge \mathit{area}(X_2) > min_{\mathit{area}} \wedge X_1 \neq X_2 \ \wedge \mathit{coverage}(X_1, X_2) > \alpha \times min_{\mathit{area}}$
- $C_3 = \mathit{freq}(X_3) > min_{\mathit{freq}} \wedge \mathit{area}(X_3) > min_{\mathit{area}} \wedge X_1 \neq X_2 \ \wedge X_2 \neq X_3$
- $C_4 = \mathit{coverage}(X_3, X_{i_3}) > \alpha \times min_{\mathit{area}}$
- $C_5 = \mathit{freq}(Z) > min_{\mathit{freq}} \wedge \mathit{area}(Z) > min_{\mathit{area}} \wedge X_1 \neq Z \ \wedge X_2 \neq Z \ \wedge X_3 \neq Z$
- $C_6 = \emptyset$
- $C_7 = \mathit{coverage}(Z, X_i) < \alpha \times min_{\mathit{area}}$

7.3.2 Modélisation des k motifs recherchés

L'approche QCSP repose sur la modélisation proposée pour *Pure-CP*. Elle utilise les mêmes contraintes réifiées pour établir le lien entre les motifs recherchés et les transactions les supportant (cf. section 6.2.1). Elle partage aussi la modélisation des motifs recherchés ainsi que l'implantation et la reformulation des contraintes (cf. section 6.3).

Ainsi, soit $\boldsymbol{r}$ un contexte transactionnel où $\mathcal{I}$ est l'ensemble de ses n items et $\mathcal{T}$ l'ensemble de ses m transactions et soient $X_1, X_2, ..., X_k$ les k motifs recherchés. Chaque motif inconnu X_j est modélisé par n variables de décision $\{X_{1,j}, X_{2,j}, ..., X_{n,j}\}$ et son support T_{X_j} est représenté par m variables de décision $\{T_{1,j}, T_{2,j}, ..., T_{m,j}\}$.

Si un motif X_J est quantifié existentiellement, alors toutes ses variables de décision le seront aussi (comme dans *Pure-CP*). Par contre, si un motif X_j est quantifié universellement, alors toutes ses variables de décision seront quantifiées universellement.

7.4 Expérimentations

Nous montrons, dans cette section, la faisabilité et les apports pratiques de l'approche basée sur les QCSPs présentée dans ce chapitre.

7.4.1 Protocole expérimental

Nous avons mené différentes expérimentations sur plusieurs jeux de données de l'UCI *repository*[1] ainsi que sur le jeu de données réelles `Meningitis` que nous avons introduit dans la section 6.4. La table 7.2 résume les différentes caractéristiques des jeux de données utilisés.

Les expérimentations ont été menées principalement sur la requête *peak* appliquée à la mesure du taux de croissance d'un motif émergent (`growth-rate`).

La machine utilisée est un PC 2.83 GHz Intel Core 2 Duo processor (4 GB de RAM) sous Ubuntu Linux.

Enfin nous avons utilisé le solveur de QCSP *QeCode* (cf. section C.3).

1. `http://www.ics.uci.edu/~mlearn/MLRepository.html`

Jeu de données	#trans	#items	densité
`Australian`	690	55	0.25
`German`	1000	76	0.28
`Meningitis`	329	84	0.27
`Postoperative-patient-data`	90	23	0.39

TABLE 7.2 – Description des jeux de données

7.4.2 Correction et complétude de l'approche

Nous utilisons le solveur de QCSP *QeCode* (cf. section C.3). Initialement, pour une instance QCSP donnée, *QeCode* retourne une seule stratégie gagnante (ou solution), si elle existe. Notre collaboration avec *Jérémie Vautard* [Vautard, 2010], un des développeurs de *QeCode*, nous a permis d'obtenir une version retournant l'intégralité des solutions.

Ainsi, notre approche permet d'extraire l'ensemble correct et complet des motifs satisfaisant les requêtes contenant des motifs quantifiés universellement.

7.4.3 Résultats expérimentaux

Les figures 7.1 et 7.2 décrivent la variation du nombre de motifs pics détectés par rapport à la mesure du taux croissance `growth-rate` en fonction des paramètres ρ (cf. exemple 7.3.1) et $minfr$ (seuil minimal de fréquence des motifs émergents) pour les jeux de données `German`, `Meningitis`, `Postoperative-patient-data` et `Australian`.

On constate que cette requête conduit à très peu de motifs (alors qu'il est bien commun que le nombre de motifs émergents soit très grand [Dong et Li, 1999]). Ces résultats mettent en valeur l'apport des contraintes contenant des quantifications universelles dans la réduction du nombre de motifs extraits en définissant des propriétés fortes sur les motifs recherchés. Ceci ouvre une piste intéressante vers la définition de requêtes permettant de cibler des motifs de haut niveau et en faible nombre.

7.4.4 Efficacité

Les expérimentations menées permettent aussi de quantifier les temps de calcul consommés par notre approche (cf. figures 7.3 et7.4). Naturellement, tout comme pour l'approche *Pure-CP*, les temps de calcul varient en fonction de la taille des jeux de données et de la dureté des contraintes.

Les résultats expérimentaux montrent aussi que la difficulté à fouiller les jeux de données augmente en fonction du nombre d'items composant le jeu de données. Ceci est expliqué par le fait que, pour les requêtes utilisées dans les expérimentations, la quantification universelle porte sur les variables représentant les motifs. La figure 7.5 met l'accent sur ce comportement que nous expliquons plus en détail dans la section suivante.

7.5 Discussion

Dans le cas des QCSP, une simple affectation de toutes les variables ne permet pas de vérifier la validité de la requête comme c'est le cas pour les CSP classiques. En effet, le *Goal* doit être vérifié pour toutes les valeurs possibles des variables universellement

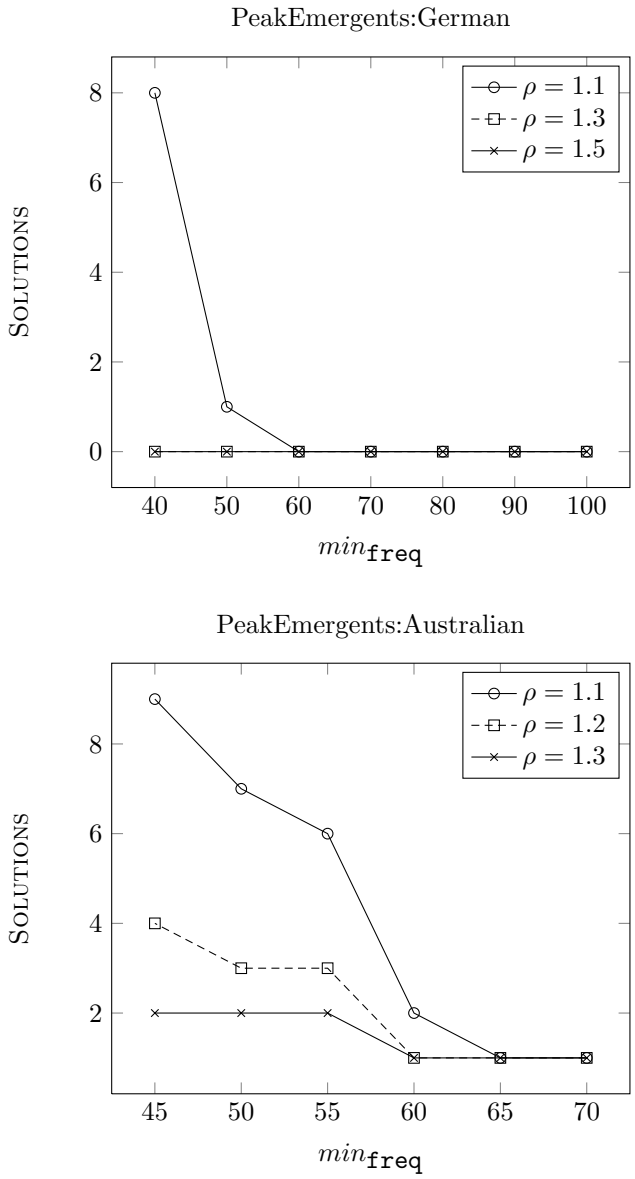

FIGURE 7.1 – Évolution du nombre de pics d'émergence extraits en fonction de $minfr$ (1/2)

quantifiées du problème. Une solution d'un QCSP doit donc exprimer tous ces cas possibles. De plus, la formule logique associée (qui est une formule du premier ordre sous forme prénexe) lie les variables dans un ordre donné par le préfixe (cf. section 3.3). Ainsi, la valeur d'une variable existentielle postérieure (selon l'ordre du préfixe) à une variable universelle sera fonction de la valeur de celle-ci, alors que dans l'ordre inverse, il est nécessaire d'avoir une valeur d'une variable existentielle consistante avec toutes les valeurs de la variable universelle placée postérieurement.

Ceci se traduit dans nos expérimentations par le fait qu'on a beaucoup plus de difficultés à effectuer l'extraction quand le nombre d'items augmente. Dans les requêtes

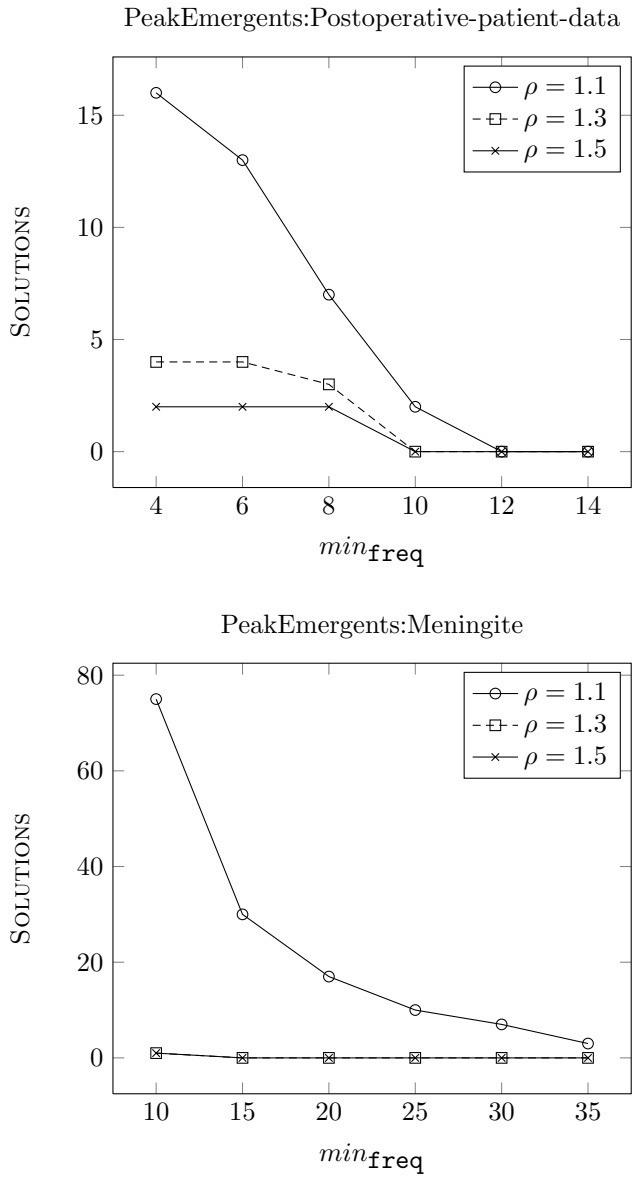

FIGURE 7.2 – Évolution du nombre de pics d'émergence extraits en fonction de $minfr$ (2/2)

présentées dans ce chapitre et celles utilisées dans les expérimentations, la quantification universelle porte sur les variables représentant les motifs. Il est ainsi plus coûteux de vérifier tous les motifs possibles quand le nombre d'items augmente.

7.6 Conclusion

Dans ce chapitre, nous avons montré l'apport des QCSPs pour modéliser de manière élégante et concise certaines requêtes en fouille de données. L'approche proposée accepte l'intégralité de l'ensemble des primitives présentées dans le chapitre 4. Elle ouvre la

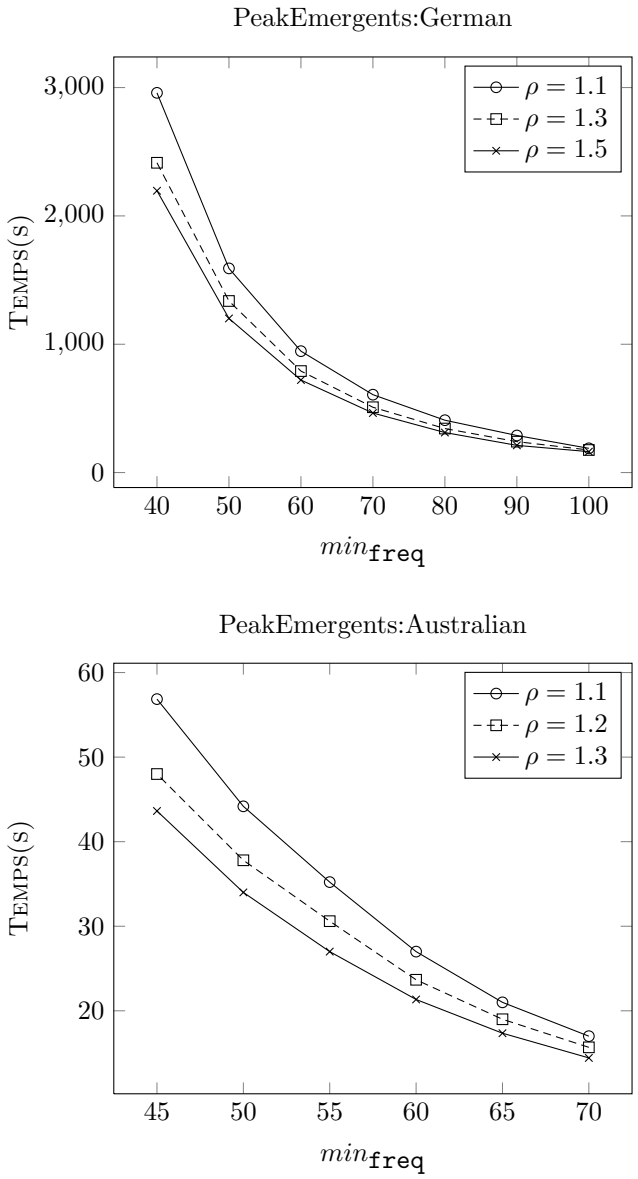

Figure 7.3 – Évolution du temps d'exécution de la requête d'extraction de pics d'émergence en fonction de $minfr$ (1/2)

porte vers la modélisation des requêtes cherchant à vérifier une propriété globale dans une base de données ou dans un ensemble de motifs vérifiant une propriété donnée (comme l'exemple du voisinage pour les *peak*). Les résultats expérimentaux ont montré la faisabilité de l'approche même s'il n'est pas possible pour le moment de traiter des bases de données de tailles réelles (i.e, plusieurs milliers d'items) pour la recherche de groupes de synexpression.

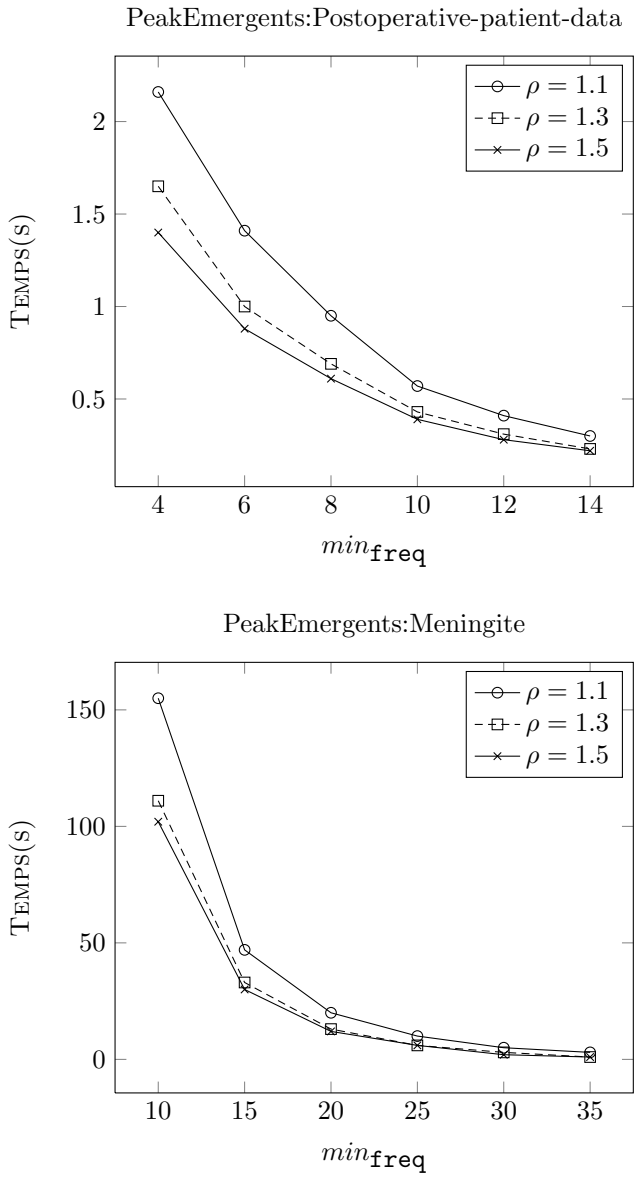

Figure 7.4 – Évolution du temps d'exécution de la requête d'extraction de pics d'émergence en fonction de $minfr$ (2/2)

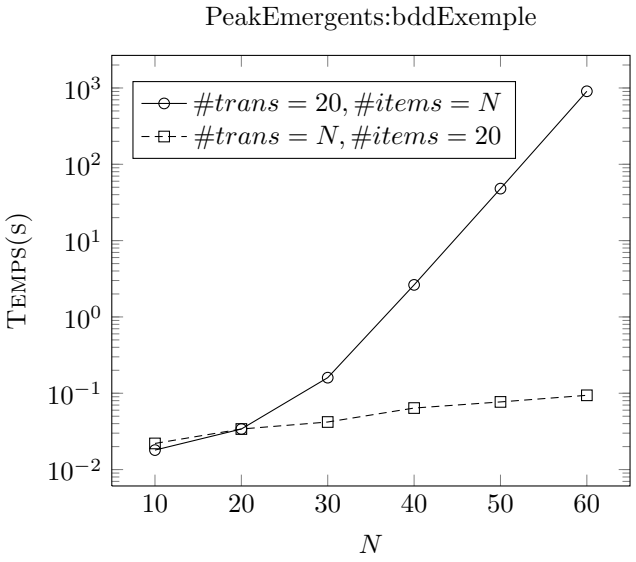

Figure 7.5 – Évolution du temps d'exécution de la requête d'extraction de pics d'émergence en fonction de la taille du jeu de données (densité fixée à 0.4)

Quatrième partie

Conclusion

Conclusion et perspectives

Conclusion

Cette thèse est l'une des toutes premières à s'intéresser aux liens entre la fouille de données et la PPC, et notamment aux apports de cette dernière à l'extraction de motifs sous contraintes. Notre motivation initiale découlait des constats suivants sur les limites de l'extraction de motifs sous contraintes :

- l'intégration d'une contrainte dans un extracteur de motifs locaux est liée au fait que l'on sache ou non lui associer un élagage performant,
- la difficulté de traiter des requêtes combinant des contraintes de natures différentes,
- l'absence de méthode générique pour l'extraction de motifs sous contraintes portant sur plusieurs motifs à la fois.

L'utilisateur devait se contenter alors d'un jeu de contraintes assez réduit ne permettant pas une extraction efficace de l'information que ce soit en terme de qualité ou de quantité. Ce manque d'expressivité est, entre autre, à l'origine d'un problème majeur dans les problèmes d'extraction de motifs locaux, à savoir le nombre conséquent de motifs produits, chose qui rend leur utilisation très difficile.

En pratique, l'utilisateur est souvent intéressé par la découverte de motifs de plus haut niveau que les classiques motifs locaux, ces motifs étant définis par des propriétés combinant plusieurs motifs à la fois. Mais, lorsqu'il s'agit de résoudre des contraintes portant sur plusieurs motifs, il est très rarement possible de trouver des propriétés d'élagage intéressantes et les méthodes de résolution existantes sont toutes ad hoc. Notre objectif était donc de proposer un cadre unifié permettant la modélisation et la résolution de contraintes portant sur plusieurs motifs.

Face à ce besoin en fouille de données de méthodes d'extraction génériques et flexibles associant puissance de modélisation et efficacité de résolution indépendamment des propriétés des contraintes et du nombre de motifs impliqués, la PPC offre le cadre approprié pour répondre à ce besoin : un cadre déclaratif associé à d'efficaces algorithmes de filtrage et de résolution.

Nos principales contributions sont les suivantes :

Un langage de contraintes pour l'extraction de motifs n-aires

Nous avons proposé dans cette thèse un cadre unifié pour modéliser et résoudre les contraintes portant sur plusieurs motifs. Nous avons proposé tout d'abord d'appeler *contrainte n-aire* une contrainte portant sur n motifs ($n \geq 1$). Puis, nous avons défini un langage de requêtes à base de contraintes de haut niveau permettant d'exprimer une large panoplie de contraintes n-aires. Ce langage est très aisément extensible vu que de nouveaux symboles de fonction, ainsi que de nouvelles contraintes ayant diffé-

rents niveaux de granularité, peuvent être ajoutés à partir de ceux proposés et être par la suite intégrés au langage. Il s'agit du premier cadre permettant la modélisation et la résolution des contraintes n-aires. Ces travaux ont donné lieu aux publications suivantes [Khiari *et al.*, 2009, Métivier *et al.*, 2011, Métivier *et al.*, 2012].

Approches génériques pour l'extraction de motifs sous contraintes n-aires

En premier lieu, nous avons développé *Hybride*, un prototype exploitant conjointement l'efficacité des extracteurs de motifs locaux ainsi que la puissance de modélisation et de résolution des CSP, et notamment les CSP Ensemblistes. Les expérimentations menées ont montré la faisabilité de *Hybride* malgré sa généricité, mais sur des bases de taille limitée. Ces travaux ont fait l'objet des publications [Khiari *et al.*, 2009, Khiari *et al.*, 2010a, Khiari *et al.*, 2010b, Khiari *et al.*, 2012].

Dans un second temps, nous avons proposé *Pure-CP* : une approche basée uniquement sur la PPC. Nous avons aussi proposé une reformulation des contraintes du langage que nous avons précédemment présenté en contraintes de plus bas niveau (portant sur des variables de décision) directement exploitables par un solveur de contraintes. *Pure-CP* a permis une nette amélioration des performances d'extraction par rapport à *Hybride*. De plus, une des forces de *Pure-CP* est la possibilité de pouvoir spécifier des contraintes aussi bien sur les motifs que les transactions les supportant. Cette opportunité est très intéressante lorsqu'il s'agit de vérifier des contraintes de couverture ou de recouvrement entre les supports souvent utilisées dans les requêtes de clustering par exemple. *Pure-CP* a permis ainsi de résoudre l'intégralité des contraintes proposées dans notre langage. Ces travaux ont fait l'objet des publications [Khiari *et al.*, 2010c, Khiari *et al.*, 2010, Khiari *et al.*, 2011].

Enfin, nous avons remarqué au cours de ce travail, que certaine requêtes n-aires requièrent la quantification universelle pour être modélisées de manière concise et élégante. *Hybride* et *Pure-CP* se basant sur le cadre des CSPs où toutes les variables sont quantifiées existentiellement, c'est pourquoi nous avons exploré dans le chapitre 7 la piste des QCSPs pour résoudre de tels problèmes. Nous avons alors proposé une première approche étendant *Pure-CP* et permettant de résoudre des contraintes quantifiées universellement. Les premiers résultats obtenus montrent qu'il s'agit d'une voie prometteuse.

Perspectives

Nous détaillons dans ce qui suit quelques perspectives de la suite de nos travaux à court et à long terme.

Développer un solveur dédié

Si l'approche *Pure-CP* permet de résoudre l'intégralité des contraintes écrites à l'aide du langage de requêtes présenté dans le chapitre 4, sa mise en œuvre actuelle reste néanmoins confrontée à quelques limites. Nous pointons dans ce qui suit les principales améliorations possibles dans cette mise en œuvre pour permettre la prise en charge de jeux de données réels de très grande taille.

1. *Représentation des données* : la représentation des données sous forme de matrice binaire telle que celle utilisée par *Pure-CP* nécessite de stocker en mémoire l'équi-

valent d'une représentation positive des données et une autre négative (correspondant respectivement aux 1 et aux 0). Soient n, m et d ($0 \leq d \leq 1$) respectivement le nombre d'items, le nombre de transactions et la densité d'un jeu de données $\boldsymbol{r}$. La représentation des données par matrice binaire est de taille $(m \times n)$ alors que la représentation positive est de taille $(n \times m \times d)$. Une deuxième perte d'espace mémoire provient de l'utilisation de contraintes prédéfinies dans *Gecode* qui impose de représenter la base de données sous forme de tableaux d'arguments passés directement aux contraintes. Ces tableaux contiennent une représentation du contenu de chaque ligne et de chaque colonne de la matrice, soit deux fois la taille initiale de la matrice binaire. Pour faire face à ces inconvénients, une représentation alternative des données associée à une réécriture des contraintes réifiées de l'équation 6.3 (page 89) permettrait une réduction de l'ordre de $\frac{2}{d}$ fois la taille de l'espace mémoire nécessaire pour représenter la base de données. Par exemple, sur les jeux de données utilisés dans le chapitre 6, d varie entre 0.18 et 0.44 donc la réduction espérée est de l'ordre d'environ 4 à 10 fois la taille des données.

2. *Propagations des contraintes* : dans un cadre PPC, les contraintes sont indépendantes les unes des autres. Considérons à titre d'exemple, la requête $\texttt{freq}(X) \geq min_{\texttt{freq}} \wedge \texttt{size}(X) \leq max_{size}$, et une instanciation vérifiant au cours du processus de résolution la contrainte de fréquence. Si la vérification ensuite de la contrainte de cardinalité change le domaine de l'une des variables impliquées, ceci déclenche la vérification à nouveau de la contrainte de fréquence. Cette dernière vérification est inutile si le nouveau domaine définit un motif inclus dans X, parce que la généralisation d'un motif fréquent est forcément fréquente. Il serait alors intéressant de mettre en place un mécanisme évitant ce test.

Notre objectif est de développer un solveur CSP dédié à l'extraction des motifs en fouille de données résolvant ces problèmes liés à la représentation des données et la propagation des contraintes.

Adapter le solveur dédié aux QCSP

Dans le cas de l'approche QCSP (cf. chapitre 7), la tâche d'extraction de motifs est encore plus complexe. En effet, une simple affectation de toutes les variables ne permet pas de vérifier la satisfaction des contraintes comme c'est le cas pour les CSP classiques : le *Goal* doit être vérifié pour toutes les valeurs possibles des variables universellement quantifiées du problème (cf. section 7.5, page 110). L'actuelle mise en œuvre de cette approche est une extension de *Pure-CP*. Il serait alors judicieux, de réimplémenter dans le nouveau solveur dédié à *Pure-CP* (cf. section précédente) les procédures de recherche pour les QCSP$^+$ proposées dans [Vautard, 2010] et implémentées dans *QeCode*.

Extraction de motifs sous contraintes souples

Les contraintes n-aires, et encore plus les contraintes n-aires quantifiées, permettent de modéliser de manière succincte les motifs ciblés, ce qui conduit à réduire considérablement le nombre de motifs extraits. Néanmoins, dans la pratique, fixer une valeur de seuil pour les contraintes sur les mesures (`freq`, `size`, `growth-rate`,...) est un problème difficile et ce choix relève souvent de l'arbitraire. Ainsi, il est possible que des motifs en réalité intéressants ne soient pas extraits car ratant de peu le seuil. Le problème de fixer les seuils est encore plus ardu lorsque plusieurs seuils doivent

être fixés simultanément dans une même requête. Si on trouve dans la littérature des approches des relaxation de contraintes dans le cas des extracteurs des motifs locaux [Bistarelli et Bonchi, 2005, Bistarelli et Bonchi, 2007], il n'en existe pas dans le cas de l'extraction de motifs dans un cadre CSP. Ce domaine de recherche fait l'objet d'une thèse démarrée cette année au GREYC et menée par Willy Ugarte.

Langage de contraintes de plus haut niveau

Une autre perspective porte sur l'extension du langage de contraintes que nous avons proposé dans le chapitre 4. Ce langage présente des contraintes et des symboles de fonction de haut niveau permettant à l'utilisateur final de modéliser aisément une large panoplie de requêtes n-aires. Néanmoins, pour certaines requêtes, la modélisation peut être complexe pour l'utilisateur. À titre d'exemple, supposons que l'utilisateur souhaite décrire le problème de clustering (cf. section 4.5.1, page 68) vérifiant les propriétés suivantes :

- couvrir tous les items ainsi que toutes les transactions de la base de données,
- éviter les clusters peu fréquents et les clusters de petite taille,
- autoriser un certain chevauchement entre les motifs ainsi qu'entre leurs supports respectifs. Ces chevauchement sont contrôlés par les seuils δ_I pour les motifs et δ_T pour les transactions,

Ceci se traduit par la requête suivante :

$$\mathtt{q}_{cl}([X_1,...,X_k]) \equiv \begin{cases} \wedge_{1\leq i\leq k}\ \mathtt{closed}(X_i) \wedge \\ \mathtt{coverTransactions}([X_1,...,X_k]) \wedge \\ \wedge_{1\leq i<j\leq k}\ \mathtt{overlapTransactions}(X_i,X_j) \leq \delta_T \wedge \\ \mathtt{coverItems}([X_1,...,X_k]) \wedge \\ \wedge_{1\leq i<j\leq k}\ \mathtt{ovelapItems}(X_i,X_j) \leq \delta_I \wedge \\ \mathtt{canonical}([X_1,...,X_k]) \\ \wedge_{1\leq i\leq k}\ \mathtt{freq}(X_i) \geq \delta_1 \wedge \\ \wedge_{1\leq i\leq k}\ \mathtt{size}(X_i) \geq \delta_2 \end{cases}$$

La définition de $\mathtt{q}_{cl}([X_1,...,X_k])$ comme nouvelle contrainte de haut niveau est d'ores et déjà possible dans notre cadre (cf. section 4.2.3, page 63). La principale difficulté rencontrée par l'utilisateur pour l'écriture d'une telle requête est de fixer les seuils δ_T, δ_I, δ_1 et δ_2. Nous pensons qu'exploiter les caractéristiques du problème et des données ainsi qu'utiliser des seuils relatifs sont des façons de proposer des valeurs par défaut aux seuils. Par exemple, une telle valeur pour le seuil bornant le recouvrement sur les transactions peut être issue du nombre total de transactions et du nombre de clusters. Typiquement, cette valeur peut correspondre à un pourcentage du nombre total de transactions pondéré par le nombre de clusters. De même, le recouvrement maximal sur les items peut être inféré à partir du nombre total d'items et du nombre de clusters. Comme il est naturel de chercher des clusters équilibrés, le rapport du nombre de transactions par le nombre de clusters suggère une valeur moyenne pour la fréquence d'un cluster, à partir de laquelle les fréquences minimale et maximale sont déduites.

À plus long terme, il s'agit de définir des primitives avec idéalement un seul paramètre correspondant à une mesure d'intérêt de l'utilisateur et à partir de laquelle les paramètres définissant la primitive sont inférés. Par exemple, pour un clustering, l'utilisateur peut souhaiter utiliser *l'index de Rand* [Rand, 1971] ou le nombre total d'items des transactions couvertes par les clusters. Pour un classifieur, une telle mesure est par exemple le taux de bien classés. Une perspective plus lointaine est de produire les combinaisons de paramètres des primitives optimisant ces mesures.

Extension aux valeurs manquantes et à d'autres applications

Notre cadre porte sur les jeux de données ne contenant pas de valeurs manquantes. En réalité, les bases de données réelles sont souvent entachées de valeurs manquantes [Grzymala-Busse et Hu, 2000, Rioult et Crémilleux, 2007]. Une collaboration a récemment démarré avec Jean-Philippe Métivier et François Rioult du GREYC autour de ce sujet. L'objectif est d'étudier la faisabilité de l'extraction de motifs dans un cadre PPC dans le cas de données incomplètes et notamment l'extraction de motifs définis à l'aide de contraintes n-aires.

Nous souhaitons aussi montrer comment notre approche permet de modéliser et de découvrir de façon déclarative d'autres motifs utilisés en fouille de données tels que les motifs contextuels [Rabatel *et al.*, 2010] ou encore les *skypatterns* [Soulet *et al.*, 2011].

Notre travail s'est intéressé uniquement à l'extraction de motifs ensemblistes. D'autres langages de motifs restent à explorer tels les motifs séquentiels ou les graphes. Un premier travail utilisant des techniques PPC pour l'extraction de motifs séquentiels fréquents est proposé dans [Coquery *et al.*, 2011]. Il présente une approche PPC utilisant un solveur SAT pour l'énumération de motifs dans une séquence. Ce travail s'inscrit dans le contexte plus large du projet ANR-DAG [2] qui s'intéresse entre autres à l'étude des liens entre l'extraction de motifs et les techniques CSP/SAT dans le but d'extraire de façon déclarative et générique les motifs recherchés.

2. `http://liris.cnrs.fr/dag/`

Cinquième partie

Annexes

Annexe A

Notre langage de requêtes

Les tableaux A.1 et A.2 récapitulent les symboles de fonction utilisés au chapitre 4. Le tableau A.3 recense les différentes contraintes dédiées offertes par notre langage de requêtes. Pour les contraintes numériques et les contraintes ensemblistes, se reporter à la section 4.2.4 (page 64).

A.1 Symboles de fonction prédéfinis

Symb. de fonction	**Arité**	**Définition**
`support`	1	`support`$(X_i) = \{t \mid t \in \mathcal{T}, X_i \subseteq t\}$ Le support d'un motif X_i
`freq`	1	`freq`$(X_i) = \mid \{t \mid t \in \mathcal{T}, X_i \subseteq t\} \mid$ ou encore `freq` $(X_i) =\mid$ `support`$(X_i) \mid$ La mesure de la fréquence d'un motif X_i
`freq'`	2	`freq'`$(X_i, D_j) = \mid \{t \mid t \in D_j, X_i \subseteq t\} \mid$ La mesure de la fréquence d'un motif X_i dans une portion D_j de la base de données
`size`	1	`size` $(X_i) = \mid \{j \mid j \in \mathcal{I}, j \in X_i\} \mid$ La mesure de la cardinalité d'un motif (X_i)
`ovelapItems`	2	`ovelapItems`$(X_i, X_j) =\mid X_i \cap X_j \mid$ Nombre d'items en commun entre deux motifs X_i et X_j
`overlapTransactions`	2	`overlapTransactions`$(X_i, X_j) =$ $\mid$`support`$(X_i) \cap$ `support`$(X_j) \mid$ Nombre de transactions couvertes à la fois par X_i et X_j

TABLE A.1 – Symboles de fonction prédéfinis

A.2 Symboles de fonction définis par l'utilisateur

Symb. de fonction	Arité	Définition
`area`	1	$\texttt{area}(X_i) = \texttt{freq}\ (X_i) \times \texttt{size}\ (X_i)$ L'aire d'un motif X_i
`coverage`	2	$\texttt{coverage}(X_i, X_j) =$ $\texttt{freq}(X_i \cup X_j) \times \texttt{size}(X_i \cap X_j)$ Le recouvrement entre $\texttt{area}(X_i)$ et $\texttt{area}(X_j)$
`growth-rate`	1	$\texttt{growth-rate}(X_i)_{D_1} = \frac{\lvert D_2 \rvert \times \texttt{freq'}(X_i, D_1)}{\lvert D_1 \rvert \times \texttt{freq'}(X_i, D_2)}$ Le taux de croissance d'un motif X_i entre deux portions de la base de donnée D_1 et D_2

TABLE A.2 – Quelques exemples de symboles de fonction définis par l'utilisateur

A.3 Contraintes prédéfinies dédiées à la fouille

Contrainte	Condition de satisfaction
$\texttt{closed}(X_i)$	$\forall X_j \supset X_i,\ \texttt{freq}(X_j) < \texttt{freq}(X_i)$ X_i est un motif fermé
$\texttt{canonical}([X_1, ..., X_k])$	$\forall_{1 \leq i < k}\ X_i < X_{i+1}$ Les motifs $X_1, ..., X_k$ respectent l'ordre lexicographique croissant
$\texttt{coverTransactions}([X_1, ..., X_k])$	$\bigcup_{1 \leq i \leq k} \texttt{support}(X_i) = \mathcal{T}$ Les motifs $X_1, ..., X_k$ couvrent toutes les transactions de la base de données
$\texttt{noOverlapTransactions}([X_1, ..., X_k])$	$\forall_{1 \leq i < j \leq k}\ \texttt{support}(X_i) \cap \texttt{support}(X_j) = \emptyset$ Les supports des motifs $X_1, ..., X_k$ sont deux à deux disjoints
$\texttt{coverItems}([X_1, ..., X_k])$	$\bigcup_{1 \leq i \leq k}\ X_i = \mathcal{I}$ Les motifs $X_1, ..., X_k$ couvrent tous les items de la base de données
$\texttt{noOverlapItems}([X_1, ..., X_k])$	$\forall_{1 \leq i < j \leq k} X_i \cap X_j = \emptyset$ Les motifs $X_1, ..., X_k$ sont deux à deux disjoints

TABLE A.3 – Contraintes prédéfinies dédiées à la fouille

Annexe B

Pure-CP : tableaux récapitulatifs des reformulations

Nous récapitulons dans ce qui suit la reformulation des symboles de fonctions prédéfinis et les contraintes élémentaires pour l'approche *Pure-CP*.

Les tableaux B.1, B.2, B.3 et B.4 résument l'ensemble en question.

Contrainte	**Reformulation**
$X_p = X_q$	$\forall i \in \mathcal{I}, X_{i,p} = X_{i,q}$
$X_p \subseteq X_q$	$\forall i \in \mathcal{I}, X_{i,p} \leq X_{i,q}$
$X_p \subset X_q$	$\forall i \in \mathcal{I}, X_{i,p} \leq X_{i,q} \ \wedge$ $\sum_{i \in \mathcal{I}} X_{i,p} < \sum_{i \in \mathcal{I}} X_{i,q}$
$i_o \in X_p$	$X_{i_0,p} = 1$
$X_p \cap X_q = \emptyset$	$\forall i \in \mathcal{I}, X_{i,p} + X_{i,q} \leq 1$
$X_p \cup X_q = X_r$	$\forall i \in \mathcal{I},$ $X_{i,p}+X_{i,q}-2\times X_{i,r} \geq -1 \wedge$ $X_{i,p} + X_{i,q} - 2 \times X_{i,r} \leq 0$
$X_p \cap X_q = X_r$	$\forall i \in \mathcal{I},$ $2\times X_{i,r}-X_{i,p}-X_{i,q} \geq -1 \wedge$ $2 \times X_{i,r} - X_{i,p} - X_{i,q} \leq 0$
$X_p \backslash X_q = X_r$	$\forall i \in \mathcal{I},$ $2\times X_{i,r}+X_{i,p}-X_{i,q} \geq -1 \wedge$ $2 \times X_{i,r} + X_{i,p} - X_{i,q} \leq 0$

TABLE B.1 – Reformulation des contraintes ensemblistes sur les motifs.

Contrainte	**Reformulation**
$T_{X_p} = T_{X_q}$	$\forall t \in \mathcal{T}, T_{t,p} = T_{t,q}$
$T_{X_p} \subseteq T_{X_q}$	$\forall t \in \mathcal{T}, T_{t,p} \leq T_{t,q}$
$T_{X_p} \subset T_{X_q}$	$\forall t \in \mathcal{T}, T_{t,p} \leq T_{t,q} \ \wedge$ $\sum_{t\in\mathcal{T}} T_{t,p} < \sum_{t\in\mathcal{T}} T_{t,q}$
$t_o \in T_{X_p}$	$T_{t_0,p} = 1$
$T_{X_p} \cap T_{X_q} = \emptyset$	$\forall t \in \mathcal{T}, T_{t,p} + T_{t,q} \leq 1$
$T_{X_p} \cup T_{X_q} = T_{X_r}$	$\forall t \in \mathcal{T},$ $T_{t,p} + T_{t,q} - 2 \times T_{t,r} \geq -1 \wedge$ $T_{t,p} + T_{t,q} - 2 \times T_{t,r} \leq 0$
$T_{X_p} \cap T_{X_q} = T_{X_r}$	$\forall t \in \mathcal{T},$ $2 \times T_{t,r} - T_{t,p} - T_{t,q} \geq -1 \wedge$ $2 \times T_{t,r} - T_{t,p} - T_{t,q} \leq 0$
$T_{X_p} \backslash T_{X_q} = T_{X_r}$	$\forall t \in \mathcal{T},$ $2 \times T_{t,r} + T_{t,p} - T_{t,q} \geq -1 \wedge$ $2 \times T_{t,r} + T_{t,p} - T_{t,q} \leq 0$

TABLE B.2 – Reformulation des contraintes ensemblistes sur les supports.

Symb. de fonction	**Arité**	**Reformulation**
`support`	1	$\texttt{support}(X_p) = T_{X_p}$
`freq`	1	$\texttt{freq}\ (X_p) = \sum_{t\in\mathcal{T}} T_{t,p}$
`freq'`	2	$\texttt{freq'}(X_p, D_j) = \sum_{t\in\mathcal{T}_{D_j}} T_{t,p}$
`size`	1	$\texttt{size}\ (X_p) = \sum_{i\in\mathcal{I}} X_{i,p}$
`ovelapItems`	2	$\texttt{ovelapItems}(X_p, X_q) =$ $\texttt{size}(X_p \cap X_q)$
`overlapTransactions`	2	$\texttt{overlapTransactions}(X_p, X_q) =$ $\texttt{size}(\texttt{support}(X_p) \cap \texttt{support}(X_q)) =$ $\texttt{size}(T_{X_p} \cap T_{X_q})$

TABLE B.3 – Reformulation des symboles de fonctions prédéfinis.

Contrainte	**Reformulation**
$\texttt{closed}(X_p)$	$\forall i \in \mathcal{I},$ $X_{i,p} = 1 \leftrightarrow \sum_{t\in\mathcal{T}} T_{t,p}(1 - d_{t,i}) = 0$
$\texttt{canonical}([X_1, ..., X_k])$	$\bigwedge_{1\leq j\leq k-1} X_j < X_{j+1}$
$\texttt{coverTransactions}([X_1, ..., X_k])$	$\forall t \in \mathcal{T}, \sum_{1\leq j\leq k} T_{t,j} \geq 1$
$\texttt{noOverlapTransactions}([X_1, ..., X_k])$	$\forall t \in \mathcal{T}, \sum_{1\leq j\leq k} T_{t,j} \leq 1$
$\texttt{coverItems}([X_1, ..., X_k])$	$\forall i \in \mathcal{I}, \sum_{1\leq j\leq k} X_{i,j} \geq 1$
$\texttt{noOverlapItems}([X_1, ..., X_k])$	$\forall i \in \mathcal{I}, \sum_{1\leq j\leq k} X_{i,j} \leq 1$

TABLE B.4 – Contraintes prédéfinies dédiées fouille de données.

Annexe C

Outils PPC utilisés

Ce chapitre présente les différents solveurs et outils de PPC que nous avons utilisés pour la mise en œuvre des différentes approches proposées dans la partie Contributions.

C.1 ECL^iPS^e

ECL^iPS^e [1] [Apt et Wallace, 2007] est un outil libre de PPC basé sur le langage de programmation logique *Prolog*. Il comprend des librairies spécifiques à chaque type de variables comme `lib(ic)` pour les entiers et les réels, `lib(ic_sets)` pour les ensembles d'entiers, `lib(sd)` pour les symboliques et la `lib(fd)` pour les variables à domaines finis.

La facilité d'utilisation d'ECL^iPS^e, ainsi que la richesse des types de données pris en charge, sont à l'origine du choix d'ECL^iPS^e pour mettre en œuvre notre premier prototype d'extraction de motifs sous contraintes n-aires *Hybride* (cf. chapitre 5).

Exemple C.1.1. *Nous illustrons la modélisation d'un CSP en ECL^iPS^e par un exemple célèbre dans la communauté de la PPC. Il s'agit de résoudre l'équation $SEND + MORE = MONEY$ où deux lettres différentes correspondent à deux chiffres distincts.*

Le problème $SEND + MORE = MONEY$ est modélisé par le CSP suivant :

- $\mathcal{X} = \{S, E, N, D, M, O, R, Y\}$
- $\mathcal{D} = \{D_S = D_E = D_N = D_D = D_M = D_O = D_R = D_Y = \{0, 1, ..., 9\}\}$
-

$$\mathcal{C} = \begin{cases} \mathit{alldifferent}([S, E, N, D, M, O, R, Y]) \wedge \\ S \neq 0 \wedge \\ M \neq 0 \wedge \\ (1000 \times S) + (100 \times E) + (10 \times N) + D + \\ (1000 \times M) + (100 \times O) + (10 \times R) + E = \\ (10000 \times M) + (1000 \times O) + (100 \times N) + (10 \times E) + Y \end{cases}$$

La figure C.1 présente le programme ECL^iPS^e commenté correspondant au problème $SEND + MORE = MONEY$.

1. `http://www.eclipseclp.org`

```
1  SendMoreMoney(Digits) :-
2       Digits = [S,E,N,D,M,O,R,Y],
3       Digits :: [0 .. 9],
4
5       M #\= 0,
6       S #\= 0,
7
8                  1000*S + 100*E + 10*N + D +
9                  1000*M + 100*O + 10*R + E #=
10      10000*M + 1000*O + 100*N + 10*E + Y,
11
12      alldifferent(Digits),
13
14      labeling(Digits).
```

l. 1 : entête du modèle,

l. 2-3 : déclarations des variables, entières, et de leurs domaines initiaux,

l. 5-6 : ajout des contraintes empêchant S et M de valoir 0,

l. 8-10 : ajout de la contrainte arithmétique spécifiant la relation de somme entre les digits,

l. 12 : ajout d'une contrainte globale `alldifferent` sur les variables du problème,

l. 14 : lancement de l'exploration de l'espace de recherche en énumérant les domaines des variables.

FIGURE C.1 – Programme ECL^iPS^e pour le problème $SEND + MORE = MONEY$

C.2 *Gecode*

Gecode [2] fait partie de la famille d'outils de résolution de CSP qui sont fournis sous la forme de bibliothèques d'objets et de fonctions, écrits dans un langage générique de programmation orientée objet. De tels outils requièrent de la part de l'utilisateur, pour résoudre un problème de contraintes donné, d'écrire un programme et de le compiler comme un programme traditionnel. *Gecode* est une bibliothèque de résolution de problèmes sous contraintes écrite en C++.

Gecode est principalement :

- Ouvert : *Gecode* fournit un solveur de contraintes avec l'état de l'art des performances tout en étant modulaire et extensible. *Gecode* est radicalement ouvert pour la programmation : il peut facilement être interfacé avec d'autres systèmes. Il est possible d'y programmer de nouveaux propagateurs où d'y mettre en œuvre de nouvelles contraintes, de nouvelles stratégies de branchement ...
- Libre : *Gecode* est distrbibué sous une licence MIT [3] et est remensé comme logiciel gratuit par le FSF [4].

2. `http://www.gecode.org`
3. Massachusetts Institute of Technology
4. Free Software Foundation

```
1  class Money : public Script {
2  protected:
3     static const int nl = 8;     /// Number of letters
4     IntVarArray le;     /// Array of letters
5  public:
6     Money(const Options& opt) : le(*this,nl,0,9) {
7      IntVar
8        s(le[0]), e(le[1]), n(le[2]), d(le[3]),
9        m(le[4]), o(le[5]), r(le[6]), y(le[7]);
10
11     rel(*this, s, IRT_NQ, 0);
12     rel(*this, m, IRT_NQ, 0);
13
14     distinct(*this, le, opt.icl());
15
16     rel(*this,              1000*s+100*e+10*n+d
17                  +          1000*m+100*o+10*r+e
18                 == 10000*m+1000*o+100*n+10*e+y,
19         opt.icl());
20
21     branch(*this, le, INT_VAR_SIZE_MIN, INT_VAL_MIN);
22   }
23
24   /// Constructor for cloning \a s
25   Money(bool share, Money& s) : Script(share,s) {
26     le.update(*this, share, s.le);
27   }
28   /// Copy during cloning
29   virtual Space*
30   copy(bool share) {
31     return new Money(share,*this);
32   }
33 };
34
35 int main(int argc, char* argv[]) {
36   Options opt("SEND+MORE=MONEY");
37   opt.solutions(0);
38   opt.iterations(20000);
39   opt.parse(argc,argv);
40   Script::run<Money,DFS,Options>(opt);
41   return 0;
42 }
```

- l. 1-9 : déclaration d'une classe C++ dédiée au problème, instanciation du paramètre *nl* (nombre de lettres) et déclaration et instanciation des variables,
- l.11-12 : ajout des contraintes empêchant S et M de valoir 0,
- l. 14 : ajout d'une contrainte `alldifferent` portant sur l'ensemble des variables,
- l. 16-19 : ajout de la contrainte arithmétique spécifiant la relation de somme entre les variables,
- l. 21 : lancement de la recherche et définition des heuristiques,
- l. 25-32 : clonage de l'état courant de l'arbre de recherche,
- l. 35-42 : initialisation et lancement de la résolution.

FIGURE C.2 – Programme *Gecode* pour le problème $SEND + MORE = MONEY$

- Portable : *Gecode* est implémenté en C++ en suivant rigoureusement la norme C++, il fonctionne ainsi sur différents types de machines et de systèmes d'exploitation.
- Accessible : *Gecode* dispose d'une décumentation assez riche et détaillée et en développement permanent. *Gecode* ainsi que sa documentation sont disponibles pour le téléchargement sur `http://www.gecode.org/`.
- Efficace : Les performances de *Gecode* sont très intéressantes en temps d'exécution. A titre d'exemple, *Gecode* à gagné toutes les éditions du challenge MinZinc entre 2008 et 2011.

Exemple C.2.1.

La figure C.2 donne un modèle Gecode commenté du problème $SEND + MORE = MONEY$. Il s'agit d'une classe C++ et d'une fonction main. Dans la première, les variables et les contraintes sont déclarées et instanciées, et la résolution paramétrée. Dans la deuxième, on lance la méthode de résolution associée à la classe qu'on vient de définir.

C.3 *QeCode*

QeCode[5] est un solveur de QCSP (cf. section 3.3, page 54) écrit en C++ et bâti sur *Gecode* (cf. section précédente). Il intègre aussi les méthodes de résolution des QCSP$^+$ proposées dans [Vautard, 2010]. L'ensemble des contraintes implémentées dans *Gecode* (y compris par les utilisateurs de *Gecode* permettant l'utilisation de propagateurs personnalisés) peut donc être utilisé dans *QeCode*. Ceci est à l'origine du choix de *QeCode* pour la mise en oeuvre de l'approche d'extraction de motifs sous contraintes quantifiées que nous proposons dans le chapitre 7.

Exemple C.3.1.

Soit le problème quantifié suivant :

$$\begin{array}{l} \exists X \in \{0,1,2,3\} \\ \qquad \forall Y \in \{0,1,2\} \textit{ tel que } Y \neq X, \\ \qquad\qquad \exists Z \in \{0,\ldots,6\} \textit{ tel que } Z < 2 \times X, \\ \qquad\qquad\qquad X + Y = Z \end{array}$$

Ce problème est représenté par le QCSP$^+$ suivant :

$$Q = ([(\exists, X), (\forall, Y, \{X \neq Y\}), (\exists, Z, \{Z < 2 \times X\})], Goal = \{X + Y = Z\})$$

La figure C.3 donne le programme QeCode associé à ce programme.

5. `http://www.univ-orleans.fr/lifo/software/qecode/QeCode.html`

```
1  int main() {
2    int* scopeSize = new int[3];
3    scopeSize[0] = 1;
4    scopeSize[1] = 1;
5    scopeSize[2] = 1;
6
7    bool* qtScope = new bool[3];
8    qtScope[0] = false;
9    qtScope[1] = true;
10   qtScope[2] = false;
11
12   Qcop p(3,qtScope,scopeSize);
13   p.QIntVar(0,0,3);
14   p.QIntVar(1,0,2);
15   p.QIntVar(2,0,6);
16
17   branch(*(p.space()),p.space()->getIntVars(1),INT_VAR_SIZE_MIN,↩
         INT_VAL_MIN);
18   p.nextScope();
19
20   rel(*(p.space()), p.var(1)!=p.var(0));
21   branch(*(p.space()),p.space()->getIntVars(2),INT_VAR_SIZE_MIN,↩
         INT_VAL_RND);
22   p.nextScope();
23
24   rel(*(p.space()), p.var(2) <(2*p.var(0)));
25   branch(*(p.space()),p.space()->getIntVars(3),INT_VAR_SIZE_MIN,↩
         INT_VAL_MIN);
26   p.nextScope();
27
28   rel(*(p.space()), p.var(0)+p.var(1)==p.var(2));
29   branch(*(p.space()),p.space()->getIntVars(3),INT_VAR_SIZE_MIN,↩
         INT_VAL_RND);
30
31   p.makeStructure();
32
33   QCSP_Solver s(&p);
34   unsigned long int nodes=0;
35   s.solve(nodes,3);
36   return 0;
37 }
```

- l. 2 : tableau pour stocker le nombre de variables dans chaque niveau de quantification (ou scope),
- l. 3-5 : nombre de variables dans chaque scope,
- l. 7 : tableau pour stocker le type de quantification de chaque scope,
- l. 8-10 : définition du type de quantification de chaque scope (false : $\exists$, true : $\forall$),
- l. 12 : déclaration du problème,
- l. 13-15 : initialisation des variables du problème,
- l. 17-18 : contraintes du premier niveau de quantification,
- l. 20-22 : contraintes du deuxième niveau de quantification,
- l. 24-26 : contraintes du troisième niveau de quantification,
- l. 28-29 : contraintes du *Goal*,
- l 33-35 : résolution du problème.

FIGURE C.3 – Programme *QeCode* pour l'exemple C.3.1

Bibliographie

[Agrawal et Srikant, 1994] Agrawal, R. et Srikant, R. (1994). Fast Algorithms for Mining Association Rules. *In Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. (Cité pages 16, 17, et 18.)

[Allen, 1983] Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Commun. ACM*, 26:832–843. (Cité page 46.)

[Apt et Wallace, 2007] Apt, K. R. et Wallace, M. (2007). *Constraint Logic Programming using Eclipse*. Cambridge University Press, New York, NY, USA. (Cité page 131.)

[Arimura et Uno, 2009] Arimura, H. et Uno, T. (2009). Polynomial-delay and polynomial-space algorithms for mining closed sequences, graphs, and pictures in accessible set systems. *In SDM*, pages 1087–1098. SIAM. (Cité page 26.)

[Azé, 2003] Azé, J. (2003). *Extractin de connaissances à partir de données numériques et textuelles*. Thèse de doctorat en informatique, Université de Paris-Sud. (Cité page 4.)

[Bastide *et al.*, 2002] Bastide, Y., Taouil, R., Pasquier, N. ans Stumme, G. et Lakhal, L. (2002). Pascal : un algorithme d'extraction des motifs fréquents. *In Technique et science informatiques*, volume 21 - n1/2002, pages 65–95. (Cité page 21.)

[Bastide *et al.*, 2000] Bastide, Y., Taouil, R., Pasquier, N., Stumme, G. et Lakhal, L. (2000). Mining Frequent Patterns with Counting Inference. *SIGKDD Explorations, Special Issue on Scalable Algorithms*, 2(2):71–80. (Cité page 21.)

[Benedetti *et al.*, 2006] Benedetti, M., Lallouet, A. et Vautard, J. (2006). Reusing CSP Propagators for QCSPs. *In* Azevedo, F., Barahona, P., Fages, F. et Rossi, F., éditeurs : *CSCLP*, volume 4651 de *Lecture Notes in Computer Science*, pages 63–77. Springer. (Cité page 54.)

[Benedetti *et al.*, 2007] Benedetti, M., Lallouet, A. et Vautard, J. (2007). Qcsp made practical by virtue of restricted quantification. *In* Veloso, M. M., éditeur : *IJCAI*, pages 38–43. (Cité page 56.)

[Benedetti *et al.*, 2008] Benedetti, M., Lallouet, A. et Vautard, J. (2008). Quantified constraint optimization. *In* Stuckey, P. J., éditeur : *CP*, volume 5202 de *Lecture Notes in Computer Science*, pages 463–477. Springer. (Cité page 104.)

[Berkhin, 2006] Berkhin, P. (2006). A survey of clustering data mining techniques. *Grouping Multidimensional Data*, Cl(c):25–71. (Cité pages 5, 38, 68, et 71.)

[Bessière et Cordier, 1993] Bessière, C. et Cordier, M.-O. (1993). Arc-consistency and arc-consistency again. *In AAAI*, pages 108–113. (Cité page 45.)

[Bessière et Régin, 1996] Bessière, C. et Régin, J.-C. (1996). MAC and Combined Heuristics : Two Reasons to Forsake FC (and CBJ ?) on Hard Problems. *In CP*, pages 61–75. (Cité pages 48 et 50.)

[Bessière et Régin, 2001] Bessière, C. et Régin, J.-C. (2001). Refining the basic constraint propagation algorithm. *In IJCAI*, pages 309–315. (Cité page 45.)

[Besson *et al.*, 2006] Besson, J., Robardet, C. et Boulicaut, J.-F. (2006). Mining a new fault-tolerant pattern type as an alternative to formal concept discovery. *In ICCS'06*, pages 144–157, Aalborg, Denmark. (Cité page 37.)

[Birkhoff, 1967] Birkhoff, G. (1967). Lattice theory. *In Colloquium Publications*, volume 25. American Mathematical Society, 3. édition. (Cité page 22.)

[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. (Cité page 122.)

[Bistarelli et Bonchi, 2007] Bistarelli, S. et Bonchi, F. (2007). Soft constraint based pattern mining. *Data Knowl. Eng.*, 62(1):118–137. (Cité page 122.)

[Boizumault *et al.*, 2011] Boizumault, P., Crémilleux, B., Khiari, M., Loudni, S. et Métivier, J.-P. (2011). Discovering knowledge using a constraint-based language. *CoRR*, abs/1107.3407. (Cité page 62.)

[Boley *et al.*, 2010] Boley, M., Horváth, T., Poigné, A. et Wrobel, S. (2010). Listing closed sets of strongly accessible set systems with applications to data mining. *Theor. Comput. Sci.*, 411(3):691–700. (Cité page 26.)

[Bonchi *et al.*, 2009] Bonchi, F., Giannotti, F., Lucchese, C., Orlando, S., Perego, R. et Trasarti, R. (2009). A constraint-based querying system for exploratory pattern discovery. *Inf. Syst.*, 34(1):3–27. (Cité page 5.)

[Bonchi et Lucchese, 2007] Bonchi, F. et Lucchese, C. (2007). Extending the state-of-the-art of constraint-based pattern discovery. *Data Knowl. Eng.*, 60:377–399. (Cité page 17.)

[Bordeaux et Monfroy, 2002] Bordeaux, L. et Monfroy, E. (2002). Beyond np : Arc-consistency for quantified constraints. *In* Hentenryck, P. V., éditeur : *CP*, volume 2470 de *Lecture Notes in Computer Science*, pages 371–386. Springer. (Cité page 104.)

[Bordeaux et Monfroy, 2003] Bordeaux, L. et Monfroy, E. (2003). Résolution de contraintes quantifiées sur les domaines finis par arc-consistance quantifiée. *Journal électronique d'intelligence artificielle*, 2(23). (Cité page 54.)

[Boulicaut et Bykowski, 2000] Boulicaut, J.-F. et Bykowski, A. (2000). Frequent Closures as a Concise Representation for Binary Data Mining. *In PADKK '00 : Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications*, pages 62–73, London, UK. Springer-Verlag. (Cité pages 23 et 24.)

[Boulicaut *et al.*, 2000] Boulicaut, J.-F., Bykowski, A. et Rigotti, C. (2000). Approximation of frequency queries by means of free-sets. *In PKDD '00 : Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 75–85, London, UK. Springer-Verlag. (Cité pages 16 et 22.)

[Boulicaut *et al.*, 2003] Boulicaut, J.-F., Bykowski, A. et Rigotti, C. (2003). Free-Sets : A Condensed Representation of Boolean Data for the Approximation of Frequency Queries. *Data Min. Knowl. Discov.*, 7(1):5–22. (Cité pages 23 et 25.)

[Bringmann *et al.*, 2009] Bringmann, B., Nijssen, S. et Zimmermann, A. (2009). Pattern-based classification : A unifying perspective. *In From Local Patterns to Global Models (LeGo-09), ECML/PKDD-09 Workshop.* (Cité pages 38 et 39.)

[Bringmann et Zimmermann, 2007] Bringmann, B. et Zimmermann, A. (2007). The chosen few : On identifying valuable patterns. *In ICDM*, pages 63–72. IEEE Computer Society. (Cité page 32.)

[Bringmann et Zimmermann, 2009] Bringmann, B. et Zimmermann, A. (2009). One in a million : picking the right patterns. *Knowl. Inf. Syst.*, 18(1):61–81. (Cité page 32.)

[Burdick *et al.*, 2003a] Burdick, D., Calimlim, M., Flannick, J., Gehrke, J. et Yiu, T. (2003a). Mafia : A performance study of mining maximal frequent itemsets. *In* Goethals, B. et Zaki, M. J., éditeurs : *FIMI*, volume 90 de *CEUR Workshop Proceedings*. CEUR-WS.org. (Cité page 5.)

[Burdick *et al.*, 2003b] Burdick, D., Calimlim, M., Flannick, J., Gehrke, J. et Yiu, T. (2003b). MAFIA : A Performance Study of Mining Maximal Frequent Itemsets. *In* Goethals, B. et Zaki, M. J., éditeurs : *FIMI*, volume 90 de *CEUR Workshop Proceedings*. CEUR-WS.org. (Cité page 24.)

[Burdick *et al.*, 2005] Burdick, D., Calimlim, M., Flannick, J., Gehrke, J. et Yiu, T. (2005). Mafia : A maximal frequent itemset algorithm. *IEEE Trans. Knowl. Data Eng.*, 17(11):1490–1504. (Cité page 23.)

[Calders et Goethals, 2003] Calders, T. et Goethals, B. (2003). Minimal -Free Representations of Frequent Sets. *In* Lavrac, N., Gamberger, D., Blockeel, H. et Todorovski, L., éditeurs : *PKDD*, volume 2838 de *Lecture Notes in Computer Science*, pages 71–82. Springer. (Cité page 23.)

[Calders et Goethals, 2007] Calders, T. et Goethals, B. (2007). Non-derivable itemset mining. *Data Min. Knowl. Discov.*, 14(1):171–206. (Cité page 23.)

[Calders *et al.*, 2005] Calders, T., Rigotti, C. et françois Boulicaut, J. (2005). A survey on condensed representations for frequent sets. *In In : Constraint Based Mining and Inductive Databases, Springer-Verlag, LNAI*, pages 64–80. Springer. (Cité page 20.)

[Cerf, 2010] Cerf, L. (2010). *Constraint-Based Mining of Closed Patterns in Noisy n-ary Relations*. Thèse de doctorat en informatique, INSA-Lyon. (Cité page 27.)

[Coquery *et al.*, 2011] Coquery, E., Jabbour, S. et Sais, L. (2011). A constraint programming approach for enumerating motifs in a sequence. *In* Spiliopoulou, M., Wang, H., Cook, D. J., Pei, J., Wang, W., Zaïane, O. R. et Wu, X., éditeurs : *ICDM Workshops*, pages 1091–1097. IEEE. (Cité page 123.)

[Crémilleux et Soulet, 2008] Crémilleux, B. et Soulet, A. (2008). Discovering knowledge from local patterns with global constraints. *In* Gervasi, O., Murgante, B., Laganà, A., Taniar, D., Mun, Y. et Gavrilova, M. L., éditeurs : *ICCSA (2)*, volume 5073 de *Lecture Notes in Computer Science*, pages 1242–1257. Springer. (Cité pages 33, 36, 37, 38, et 106.)

[De Raedt *et al.*, 2008] De Raedt, L., Guns, T. et Nijssen, S. (2008). Constraint Programming for Itemset Mining. *In ACM SIGKDD International Conference on Knowledge Discovery and Data Mining edition :14 location*, Las Vegas, Nevada, USA. (Cité pages 5, 7, 8, 9, 27, 28, 62, et 101.)

[Dong et Li, 1999] Dong, G. et Li, J. (1999). Efficient mining of emerging patterns : Discovering trends and differences. *In KDD*, pages 43–52. (Cité pages 106 et 110.)

[Durand et Cremilleux, 2002] Durand, N. et Cremilleux, B. (2002). Ecclat : a new approach of clusters discovery in categorical data. *In In the 22nd Int. Conf. on Know-*

ledge Based Systems and Applied Arti Intelligence (ES'02), pages 177–190. Springer. (Cité pages 20, 21, et 38.)

[Duval *et al.*, 2007] Duval, B., Salleb, A. et Vrain, C. (2007). On the discovery of exception rules : A survey. *In* Guillet, F. et Hamilton, H. J., éditeurs : *Quality Measures in Data Mining*, volume 43 de *Studies in Computational Intelligence*, pages 77–98. Springer. (Cité page 33.)

[El-Hajj et Zaïane, 2005] El-Hajj, M. et Zaïane, O. R. (2005). Finding All Frequent Patterns Starting from the Closure. *In ADMA*, pages 67–74. (Cité page 23.)

[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. (Cité page 4.)

[Fisher, 1987] Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172. (Cité page 67.)

[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. (Cité page 32.)

[Ganter et Wille, 1997] Ganter, B. et Wille, R. (1997). Applied Lattice Theory : Formal Concept Analysis. (Cité page 24.)

[Geerts *et al.*, 2004] Geerts, F., Goethals, B. et Mielikäinen, T. (2004). Tiling databases. *In Discovery Science*, volume 3245 de *Lecture Notes in Computer Science*, pages 278–289. Springer. (Cité pages 5 et 15.)

[Geng et Hamilton, 2006] Geng, L. et Hamilton, H. J. (2006). Interestingness measures for data mining : A survey. *ACM Comput. Surv.*, 38. (Cité page 15.)

[Gervet, 1997] Gervet, C. (1997). Interval Propagation to Reason about Sets : Definition and Implementation of a Practical Language. *Constraints*, 1(3):191–244. (Cité pages 51 et 86.)

[Giacometti *et al.*, 2011] Giacometti, A., Marcel, P. et Soulet, A. (2011). A relational view of pattern discovery. *In* Yu, J. X., Kim, M.-H. et Unland, R., éditeurs : *DASFAA (1)*, volume 6587 de *Lecture Notes in Computer Science*, pages 153–167. Springer. (Cité page 33.)

[Giacometti *et al.*, 2009] Giacometti, A., Miyaneh, E. K., Marcel, P. et Soulet, A. (2009). A framework for pattern-based global models. *In* Corchado, E. et Yin, H., éditeurs : *IDEAL*, volume 5788 de *Lecture Notes in Computer Science*, pages 433–440. Springer. (Cité pages 29 et 38.)

[Golomb et Baumert, 1965] Golomb, S. W. et Baumert, L. D. (1965). Backtrack programming. *J. ACM*, 12(4):516–524. (Cité page 47.)

[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. (Cité pages 23 et 24.)

[Grzymala-Busse et Hu, 2000] Grzymala-Busse, J. W. et Hu, M. (2000). A comparison of several approaches to missing attribute values in data mining. *In* Ziarko, W. et Yao, Y. Y., éditeurs : *Rough Sets and Current Trends in Computing*, volume 2005 de *Lecture Notes in Computer Science*, pages 378–385. Springer. (Cité page 123.)

[Guns *et al.*, 2011a] Guns, T., Nijssen, S. et De Raedt, L. (2011a). k-Pattern set mining under constraints. *IEEE Transactions on Knowledge and Data Engineering*. (Cité page 8.)

[Guns *et al.*, 2011b] Guns, T., Nijssen, S. et Raedt, L. D. (2011b). Evaluating pattern set mining strategies in a constraint programming framework. *In* Huang, J. Z., Cao, L. et Srivastava, J., éditeurs : *PAKDD (2)*, volume 6635 de *Lecture Notes in Computer Science*, pages 382–394. Springer. (Cité page 8.)

[Guns *et al.*, 2011c] Guns, T., Nijssen, S. et Raedt, L. D. (2011c). Itemset mining : A constraint programming perspective. *Artif. Intell.*, 175(12-13):1951–1983. (Cité page 8.)

[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. (Cité page 32.)

[Hand, 2002] Hand, D. J. (2002). Pattern detection and discovery. *In* Hand, D. J., Adams, N. M. et Bolton, R. J., éditeurs : *Pattern Detection and Discovery*, volume 2447 de *Lecture Notes in Computer Science*, pages 1–12. Springer. (Cité pages 5 et 15.)

[Haralick et Elliot, 1980] Haralick, R. et Elliot, G. (1980). Increasing Tree Search Efficiency for Constraint Satisfaction Problems. *Artificial Intelligence*, 14(3):263–313. (Cité page 50.)

[Hentenryck *et al.*, 1992] Hentenryck, P. V., Deville, Y. et Teng, C.-M. (1992). A generic arc-consistency algorithm and its specializations. *Artif. Intell.*, 57(2-3):291–321. (Cité page 8.)

[Khanjari Miyaneh, 2009] Khanjari Miyaneh, E. (2009). *Un cadre générique pour les modèles globaux fondés sur les motifs locaux.* Thèse de doctorat en informatique, Université de François Rabelais Tours. (Cité page 38.)

[Khiari *et al.*, 2009] Khiari, M., Boizumault, P. et Crémilleux, B. (2009). Local constraint-based mining and set constraint programming for pattern discovery. *In From Local Patterns to Global Models (LeGo-09), ECML/PKDD-09 Workshop*, pages 61–76, Bled, Slovenia. (Cité pages 7, 33, 79, et 120.)

[Khiari *et al.*, 2010a] Khiari, M., Boizumault, P. et Crémilleux, B. (2010a). Allier csps et motifs locaux pour la découverte de motifs sous contraintes n-aires. *In* Yahia, S. B. et Petit, J.-M., éditeurs : *EGC*, volume RNTI-E-19 de *Revue des Nouvelles Technologies de l'Information*, pages 199–210. Cépaduès-Éditions. (Cité pages 7, 79, et 120.)

[Khiari *et al.*, 2010b] Khiari, M., Boizumault, P. et Crémilleux, B. (2010b). Combining csp and constraint-based mining for pattern discovery. *In* Taniar, D., Gervasi, O., Murgante, B., Pardede, E. et Apduhan, B. O., éditeurs : *ICCSA (2)*, volume 6017 de *Lecture Notes in Computer Science*, pages 432–447. Springer. (Cité pages 7, 29, 79, et 120.)

[Khiari *et al.*, 2010c] Khiari, M., Boizumault, P. et Crémilleux, B. (2010c). Constraint programming for mining n-ary patterns. *In* Cohen, D., éditeur : *CP*, volume 6308 de *Lecture Notes in Computer Science*, pages 552–567. Springer. (Cité pages 8, 33, 88, et 120.)

[Khiari *et al.*, 2010] Khiari, M., Boizumault, P. et Crémilleux, B. (2010). Extraction de motifs n-aires utilisant la PPC. *In 6-èmes Journées Francophones de Programmation par Contraintes (JFPC'10)*, pages 167–176, Caen. (Cité pages 8, 88, et 120.)

[Khiari *et al.*, 2011] Khiari, M., Boizumault, P. et Crémilleux, B. (2011). A generic approach for modeling n-ary patterns. *In 19th International Symposium on Methodologies for Intelligent Systems (ISMIS 2011)*, volume 6804 de *LNAI*, pages 300–305, Warsaw, Poland. Springer-Verlag. (Cité pages 8, 88, et 120.)

[Khiari *et al.*, 2012] Khiari, M., Boizumault, P. et Crémilleux, B. (2012). Combining CSP and constraint-based mining for pattern discovery. *In Advances in Knowledge Discovery and Management 2 (Post-EGC'10 Selected Papers)*, Studies in Computational Intelligence, Vol. 398. Springer. 20 pages. (Cité pages 7, 79, et 120.)

[Kléma *et al.*, 2008] Kléma, J., Blachon, S., Soulet, A., Crémilleux, B. et Gandrillon, O. (2008). Constraint-based knowledge discovery from sage data. *In Silico Biology*, 8(0014). (Cité page 37.)

[Knobbe *et al.*, 2008] Knobbe, A., Crémilleux, B., Fürnkranz, J. et Scholz, M. (2008). From local patterns to global models : The lego approach to data mining. *In Int. Workshop LeGo co-located with ECML/PKDD'08*, pages 1–16, Antwerp, Belgium. (Cité pages 29 et 38.)

[Knobbe et Ho, 2006] Knobbe, A. J. et Ho, E. K. Y. (2006). Pattern teams. *In* Fürnkranz, J., Scheffer, T. et Spiliopoulou, M., éditeurs : *PKDD*, volume 4213 de *Lecture Notes in Computer Science*, pages 577–584. Springer. (Cité pages 32, 33, et 38.)

[Kramer et De Raedt, 2001] Kramer, S. et De Raedt, L. (2001). Feature Construction with Version Spaces for Biochemical Applications. *In* Brodley, C. E. et Danyluk, A. P., éditeurs : *ICML*, pages 258–265. Morgan Kaufmann. (Cité page 4.)

[Kuramochi et Karypis, 2004] Kuramochi, M. et Karypis, G. (2004). Finding Frequent Patterns in a Large Sparse Graph. *In* Berry, M. W., Dayal, U., Kamath, C. et Skillicorn, D. B., éditeurs : *SDM*. SIAM. (Cité page 4.)

[Lakshmanan *et al.*, 1999] Lakshmanan, L. V. S., Ng, R. T., Han, J. et Pang, A. (1999). Optimization of constrained frequent set queries with 2-variable constraints. *In* Delis, A., Faloutsos, C. et Ghandeharizadeh, S., éditeurs : *SIGMOD Conference*, pages 157–168. ACM Press. (Cité pages 5 et 34.)

[Lan *et al.*, 2005] Lan, Y., Janssens, D., Wets, G. et Chen, G. (2005). Improving associative classification by incorporating novel interestingness measures. *In* Lau, F. C. M., Lei, H., Meng, X. et Wang, M., éditeurs : *ICEBE*, pages 282–288. IEEE Computer Society. (Cité page 5.)

[Lhomme, 1993] Lhomme, O. (1993). Consistency Techniques for Numerical CSPs. *In Proceedings of IJCAI*, pages 232–238. (Cité page 45.)

[Li *et al.*, 2007] Li, J., Liu, G. et Wong, L. (2007). Mining statistically important equivalence classes and delta-discriminative emerging patterns. *In KDD '07 : Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 430–439, New York, NY, USA. ACM. (Cité page 21.)

[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 : *ICDM*, pages 369–376. IEEE Computer Society. (Cité pages 5 et 39.)

[Liu *et al.*, 1998] Liu, B., Hsu, W. et Ma, Y. (1998). Integrating classification and association rule mining. *In Proceedings of the 4th international conference on Knowledge Discovery and Data mining (KDD'98)*, pages 80–86. AAAI Press. (Cité pages 5 et 39.)

[Mackworth, 1977] Mackworth, A. K. (1977). Consistency in networks of relations. *Artif. Intell.*, 8(1):99–118. (Cité page 44.)

[Mannila et Toivonen, 1996] Mannila, H. et Toivonen, H. (1996). Multiple uses of frequent sets and condensed representations. *In In Proc. KDD Int. Conf. Knowledge Discovery in Databases*, pages 189–194. AAAI Press. (Cité page 23.)

[Mannila et Toivonen, 1997] Mannila, H. et Toivonen, H. (1997). Levelwise search and borders of theories in knowledgediscovery. *Data Min. Knowl. Discov.*, 1:241–258. (Cité pages 5, 14, 15, 17, et 19.)

[Mannila *et al.*, 1997] Mannila, H., Toivonen, H. et Verkamo, I. A. (1997). Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289. (Cité page 4.)

[Métivier *et al.*, 2011] Métivier, J.-P., Boizumault, P., Crémilleux, B., Khiari, M. et Loudni, S. (2011). A constraint-based language for declarative pattern discovery. *In Workshop on Declarative Pattern Mining, ICDM 2011 : The 11th IEEE International Conference on Data Mining*, pages 1112–1119, Vancouver, Canada. (Cité pages 8, 62, et 120.)

[Métivier *et al.*, 2012] Métivier, J.-P., Boizumault, P., Crémilleux, B., Khiari, M. et Loudni, S. (2012). A constraint-based language for declarative pattern discovery. *In 27th annual ACM Symposium on Applied Computing (SAC'12)*, pages 1–7, Riva del Garde (Trento), Italy. (Cité pages 8, 62, et 120.)

[Mielikäinen, 2004] Mielikäinen, T. (2004). Separating structure from interestingness. *In* Dai, H., Srikant, R. et Zhang, C., éditeurs : *PAKDD*, volume 3056 de *Lecture Notes in Computer Science*, pages 476–485. Springer. (Cité page 39.)

[Mielikäinen et Mannila, 2003] Mielikäinen, T. et Mannila, H. (2003). The pattern ordering problem. *In* Lavrac, N., Gamberger, D., Blockeel, H. et Todorovski, L., éditeurs : *PKDD*, volume 2838 de *Lecture Notes in Computer Science*, pages 327–338. Springer. (Cité page 38.)

[Mitchell, 1981] Mitchell, T. M. (1981). Generalization as Search. *In* Webber, B. L. et Nilsson, N. J., éditeurs : *Readings in Artificial Intelligence*, pages 517–542. Morgan Kaufmann, Los Altos. (Cité pages 17 et 19.)

[Mohr et Henderson, 1986] Mohr, R. et Henderson, T. C. (1986). Arc and path consistency revisited. *Artif. Intell.*, 28(2):225–233. (Cité page 44.)

[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. (Cité pages 5 et 15.)

[Negrevergne, 2011] Negrevergne, B. (2011). *A Generic and Parallel Pattern Mining Algorithm for Multi-Core Architectures*. Thèse de doctorat, University of Grenoble. (Cité page 26.)

[Ng *et al.*, 1998] Ng, R. T., Lakshmanan, V. S., Han, J. et Pang, A. (1998). Exploratory mining and pruning optimizations of constrained associations rules. *In proceedings of ACM SIGMOD'98*, pages 13–24. ACM Press. (Cité page 17.)

[Nijssen et Guns, 2010] Nijssen, S. et Guns, T. (2010). Integrating constraint programming and itemset mining. *In* Balcázar, J. L., Bonchi, F., Gionis, A. et Sebag, M., éditeurs : *ECML/PKDD (2)*, volume 6322 de *Lecture Notes in Computer Science*, pages 467–482. Springer. (Cité page 8.)

[Nijssen *et al.*, 2009] Nijssen, S., Guns, T. et De Raedt, L. (2009). Correlated itemset mining in roc space : a constraint programming approach. *In KDD*, pages 647–656. (Cité page 7.)

[Novak *et al.*, 2009a] Novak, P. K., Lavrac, N. et Webb, G. I. (2009a). Supervised descriptive rule discovery : A unifying survey of contrast set, emerging pattern and subgroup mining. *Journal of Machine Learning Research*, 10:377–403. (Cité page 20.)

[Novak *et al.*, 2009b] Novak, P. K., Lavrac, N. et Webb, G. I. (2009b). Supervised descriptive rule discovery : A unifying survey of contrast set, emerging pattern and subgroup mining. *Journal of Machine Learning Research*, 10:377–403. (Cité page 63.)

[Padmanabhan et Tuzhilin, 1998] Padmanabhan, B. et Tuzhilin, A. (1998). A belief-driven method for discovering unexpected patterns. *In KDD*, pages 94–100. (Cité pages 5 et 35.)

[Padmanabhan et Tuzhilin, 1999] Padmanabhan, B. et Tuzhilin, A. (1999). Unexpectedness as a measure of interestingness in knowledge discovery. *In In Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 275–281. AAAI Press. (Cité page 35.)

[Pasquier *et al.*, 1999] Pasquier, N., Bastide, Y., Taouil, R. et Lakhal, L. (1999). Discovering frequent closed itemsets for association rules. *In Proceedings of the 7th biennial International Conference on Database Theory (ICDT'99)*, Lecture Notes in Computer Science, Vol. 1540, pages 398–416. Springer-Verlag. (Cité pages 21, 23, et 24.)

[Rabatel *et al.*, 2010] Rabatel, J., Bringay, S. et Poncelet, P. (2010). Contextual sequential pattern mining. *In* Fan, W., Hsu, W., Webb, G. I., Liu, B., Zhang, C., Gunopulos, D. et Wu, X., éditeurs : *ICDM Workshops*, pages 981–988. IEEE Computer Society. (Cité page 123.)

[Raedt et Zimmermann, 2007] Raedt, L. D. et Zimmermann, A. (2007). Constraint-based pattern set mining. *In SDM*. SIAM. (Cité pages 29, 33, et 34.)

[Rand, 1971] Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850. (Cité page 122.)

[Rioult, 2005] Rioult, F. (2005). *Extraction de connaissances dans les bases de données comportant des valeurs manquantes ou un grand nombre d'attributs.* Thèse de doctorat, Université de Caen Basse-Normandie. (Cité page 5.)

[Rioult et Crémilleux, 2007] Rioult, F. et Crémilleux, B. (2007). Mining correct properties in incomplete databases. *In Proceedings of the 5th international conference on Knowledge discovery in inductive databases*, KDID'06, pages 208–222, Berlin, Heidelberg. Springer-Verlag. (Cité page 123.)

[Roberto J. Bayardo, 1998] Roberto J. Bayardo, J. (1998). Efficiently mining long patterns from databases. *SIGMOD Rec.*, 27(2):85–93. (Cité page 24.)

[Sabin et Freuder, 1994] Sabin, D. et Freuder, E. C. (1994). Contradicting Conventional Wisdom in Constraint Satisfaction. *In* Borning, A., éditeur : *PPCP*, volume 874 de *Lecture Notes in Computer Science*, pages 10–20. Springer. (Cité pages 48 et 50.)

[Siebes *et al.*, 2006] Siebes, A., Vreeken, J. et van Leeuwen, M. (2006). Item sets that compress. *In* Ghosh, J., Lambert, D., Skillicorn, D. B. et Srivastava, J., éditeurs : *SDM*. SIAM. (Cité page 39.)

[Soulet, 2006] Soulet, A. (2006). *Un cadre générique de découverte de motifs sous contraintes basées sur des primitives.* Thèse de doctorat en informatique, Université de Caen Basse-Normandie. (Cité pages 5, 23, 25, 26, 27, 33, et 62.)

[Soulet *et al.*, 2004a] Soulet, A., Crémilleux, B. et Rioult, F. (2004a). Condensed representation of emerging patterns. *In* Dai, H., Srikant, R. et Zhang, C., éditeurs : *PAKDD*, volume 3056 de *Lecture Notes in Computer Science*, pages 127–132. Springer. (Cité page 21.)

[Soulet et Crémilleux, 2005] Soulet, A. et Crémilleux, B. (2005). An efficient framework for mining flexible constraints. *In Proceedings of the 9th PAKDD'05*, volume 3518 de *LNAI*, pages 661–671. (Cité pages 5 et 26.)

[Soulet et Crémilleux, 2008] Soulet, A. et Crémilleux, B. (2008). Adequate condensed representations of patterns. *In* Daelemans, W., Goethals, B. et Morik, K., éditeurs : *ECML/PKDD (1)*, volume 5211 de *Lecture Notes in Computer Science*, pages 20–21. Springer. (Cité page 23.)

[Soulet *et al.*, 2004b] Soulet, A., Crémilleux, B. et Rioult, F. (2004b). Représentation condensée de motifs émergents. *In* Hébrail, G., Lebart, L. et Petit, J.-M., éditeurs : *EGC*, volume RNTI-E-2 de *Revue des Nouvelles Technologies de l'Information*, pages 265–276. Cépaduès-Éditions. (Cité page 23.)

[Soulet *et al.*, 2007] Soulet, A., Klema, J. et Crémilleux, B. (2007). *Efficient Mining under Rich Constraints Derived from Various Datasets*, volume 4747 de *LNCS*, pages 223–239. Springer. (Cité pages 26, 27, et 80.)

[Soulet *et al.*, 2011] Soulet, A., Raïssi, C., Plantevit, M. et Crémilleux, B. (2011). Mining dominant patterns in the sky. *In* Cook, D. J., Pei, J., Wang, W., Zaïane, O. R. et Wu, X., éditeurs : *ICDM*, pages 655–664. IEEE. (Cité page 123.)

[Stadler et Stadler, 2002] Stadler, A. M. R. et Stadler, P. F. (2002). Basic Properties of Filter Convergence Spaces. (Cité page 22.)

[Suzuki, 2002] Suzuki, E. (2002). Undirected Discovery of Interesting Exception Rules. *IJPRAI*, 16(8):1065–1086. (Cité pages 5, 33, 35, 36, et 101.)

[Szathmary *et al.*, 2007] Szathmary, L., Napoli, A. et Valtchev, P. (2007). Towards rare itemset mining. *In ICTAI (1)*, pages 305–312. IEEE Computer Society. (Cité page 99.)

[Termier, 2004] Termier, A. (2004). *Extractin d'arbres fréquents dans un corpus hétérogène de données semi-structurées : application à la fouille de documents XML.* Thèse de doctorat en informatique, Université de Paris-Sud. (Cité page 4.)

[The et Padmanabhan, 2000] The, B. P. et Padmanabhan, B. (2000). Small is beautiful : Discovering the minimal set of unexpected patterns. *In In Proceedings of the Sixth SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 54–63. ACM Press. (Cité page 35.)

[Toivonen *et al.*, 1995] Toivonen, H., Klemettinen, M., Ronkainen, P., Hatonen, K. et Mannila, H. (1995). Pruning and grouping discovered association rules. *In ECML'95 MLnet workshop on statistics, machine learning, and knowledge discovery in databases*, pages 47–52. (Cité page 38.)

[Vautard, 2010] Vautard, J. (2010). *Modélisation et résolution de problèmes de décision et d'optimisation hiérarchiques en utilisant des contraintes quantifiées.* These, Université d'Orléans. (Cité pages 55, 110, 121, et 134.)

[Verger et Bessière, 2006] Verger, G. et Bessière, C. (2006). : A bottom-up approach for solving quantified csps. *In* Benhamou, F., éditeur : *CP*, volume 4204 de *Lecture Notes in Computer Science*, pages 635–649. Springer. (Cité page 54.)

[Waltz, 1972] Waltz, D. L. (1972). Generating semantic descriptions from drawings of scenes with shadows. Rapport technique, Cambridge, MA, USA. (Cité page 46.)

[Wille, 1982] Wille, R. (1982). Restructuring lattice theory : An approach based on hierarchies of concepts. *In* Rival, I., éditeur : *Ordered Sets*, pages 445–470. Reidel, Dordrecht-Boston. (Cité page 22.)

[Xin *et al.*, 2006] Xin, D., Cheng, H., Yan, X. et Han, J. (2006). Extracting redundancy-aware top-k patterns. *In* Eliassi-Rad, T., Ungar, L. H., Craven, M. et Gunopulos, D., éditeurs : *KDD*, pages 444–453. ACM. (Cité pages 32 et 33.)

[Yin et Han, 2003] Yin, X. et Han, J. (2003). CPAR : classification based on predictive association rules. *In proceedings of the 2003 SIAM Int. Conf. on Data Mining (SDM'03)*, San Fransisco, CA. (Cité page 33.)

Résumé

Titre : Découverte de motifs n-aires utilisant la programmation par contraintes.

La fouille de données et la Programmation Par Contraintes (PPC) sont deux domaines de l'informatique qui ont eu, jusqu'à très récemment, des destins séparés. Cette thèse est l'une des toutes premières à s'intéresser aux liens entre la fouille de données et la PPC, et notamment aux apports de cette dernière à l'extraction de motifs sous contraintes.

Différentes méthodes génériques pour la découverte de motifs locaux ont été proposées. Mais, ces méthodes ne prennent pas en considération le fait que l'intérêt d'un motif dépend souvent d'autres motifs. Un tel motif est appelé motif n-aire. Très peu de travaux concernant l'extraction de motifs n-aires ont été menés et les méthodes développées sont toutes ad hoc. Cette thèse propose un cadre unifié pour modéliser et résoudre les contraintes n-aires en fouille de données.

Tout d'abord, l'extraction de motifs n-aires est modélisée sous forme de problème de satisfaction de contraintes (CSP). Puis, un langage de requêtes à base de contraintes de haut niveau est proposé. Ce langage permet d'exprimer une large panoplie de contraintes n-aires. Plusieurs méthodes de résolution sont développées et comparées.

Les apports principaux de ce cadre sont sa déclarativité et sa généricité. Il s'agit du premier cadre générique et flexible permettant la modélisation et la résolution de contraintes n-aires en fouille de données.

Abstract

Title : Constraint Programming for mining n-ary patterns.

Until recently, data mining and Constraint Programming have been developed separately one from the other. This thesis is one of the first to address the relationships between these two areas of computer science, in particular using constraint programming techniques for constraint-based mining.

The data mining community has proposed generic approaches to discover local patterns under constraints, and this issue is rather well-mastered. However, these approaches do not take into consideration that the interest of a pattern often depends on the other patterns. Such a pattern is called n-ary pattern or pattern set. Few works on mining n-ary patterns were conducted and the proposed approaches are ad hoc. This thesis proposes an unified framework for modeling and solving n-ary constraints in data mining.

First, the n-ary pattern extraction problem is modeled as a Constraint Satisfaction Problem (CSP). Then, a high-level declarative language for mining n-ary patterns is proposed. This language allows to express a wide range of n-ary constraints. Several solving methods are developed and compared.

The main advantages of this framework are its declarative and generic sides. To the best of our knowledge, it is the first generic and flexible framework for modeling and mining n-ary patterns.

Mots-clés indexation RAMEAU : Exploration de données, Programmetion par contraintes, Contraintes (intelligence artificielle), Bases de données.

Discipline : Informatique et applications.

Laboratoire : GREYC CNRS UMR 6072, Campus Côte de Nacre, Boulevard du Maréchal Juin BP 5186 - 14032 Caen CEDEX.