Cadre interactif de fouille de motifs avec prise en compte des préférences de l'utilisateur

Lobnury Hien

Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité INFORMATIQUE

Préparée au sein de l'Université de Caen Normandie

Cadre interactif de fouille de motifs avec prise en compte des préférences de l'utilisateur

Présentée et soutenue par
LOBNURY HIEN

Thèse soutenue le 07/12/2022 devant le jury composé de		
MME THI-BICH-HAN DAO	Maître de conférences HDR, Université d'Orléans	Rapporteur du jury
MME CHRISTINE SOLNON	Professeur des universités, INST NAT SC APPLIQ DE LYON	Membre du jury
M. ARNAUD SOULET	Maître de conférences, UNIVERSITE DE TOURS	Membre du jury
M. ALBRECHT ZIMMERMANN	Maître de conférences, Université de Caen Normandie	Membre du jury
M. CHRISTOPHE LECOUTRE	Professeur des universités, Université d'Artois	Président du jury
M. SAMIR LOUDNI	Professeur des universités, MINES NANTES	Directeur de thèse
M. PATRICE BOIZUMAULT	Professeur des universités, Université de Caen Normandie	Co-directeur de thèse

Thèse dirigée par SAMIR LOUDNI (Groupe de recherche en informatique, image, automatique et instrumentation) et PATRICE BOIZUMAULT (Groupe de recherche en informatique, image, automatique et instrumentation)

UNIVERSITÉ CAEN NORMANDIE

MIIS

GREYC
Laboratoire de recherche en sciences du numérique

Table des matières

Table des figures **ix**

I Introduction générale

Introduction **3**

1 Fouille de données 3
2 Objectifs et contributions 5
3 Organisation du mémoire 7

II État de l'art : Fouille de motifs et apprentissage

1 Fouille de motifs **11**

1.1 Cadre formel et définitions 12
1.2 Extraction de motifs sous contraintes locales 14
1.2.1 Quelques préliminaires mathématiques 14
1.2.2 Contraintes 15
1.2.3 Espace de recherche 15
1.2.4 Mesures d'intérêt 17
1.2.5 Représentation condensée 20
1.2.6 Discussion 21
1.3 Extraction d'ensembles de motifs 21
1.3.1 Théorie des ensembles de motifs 21
1.3.2 Mesures d'intérêt et redondance 22
1.4 Diversification des solutions 24
1.4.1 Contrainte de redondance sur les paires de motifs 25
1.4.2 Contrainte sur les transactions redondantes 25
1.4.3 Discussion 25
1.5 La compression des jeux de données 26

1.5.1 MDL . . . 26
1.5.2 Réduction de la redondance par la compression . . . 26
1.6 Échantillonnage de motifs . . . 28
1.6.1 Échantillonnage par marche aléatoire . . . 29
1.6.2 Échantillonnage en deux étapes . . . 29
1.6.3 Échantillonnage par le formalisme SAT . . . 30
1.7 Approches de réduction de la redondance en fouille de motifs . . . 30
1.7.1 Approche exhaustive : PatternsTeam . . . 30
1.7.2 Approche par compression : Krimp . . . 31
1.7.3 Approche heuristique : Picker . . . 33
1.7.4 Approche d'échantillonnage par marche aléatoire : Gibbs . . . 35
1.7.5 Approche d'échantillonnage en deux étapes : CFTP . . . 36
1.7.6 Synthèse . . . 38
1.8 Conclusion . . . 39

2 Programmation par contraintes 41
2.1 Modélisation . . . 41
2.1.1 Formalisme CSP . . . 42
2.1.2 Résolution d'un CSP . . . 43
2.2 Consistance et filtrage . . . 43
2.2.1 Consistance . . . 44
2.2.2 Filtrage de domaines . . . 45
2.2.3 Méthodes de Filtrage . . . 45
2.2.4 Propagation de contraintes . . . 46
2.3 Stratégies de recherche . . . 46
2.3.1 Backtrack et maintient d'arc-cohérence . . . 47
2.3.2 Ordre de choix des variables et des valeurs . . . 47
2.3.3 Algorithme de résolution . . . 48
2.4 Modélisation des contraintes . . . 48
2.4.1 Contraintes Réifiées . . . 49
2.4.2 Contraintes Globales . . . 49
2.5 Modèles PPC pour l'extraction de motifs . . . 50
2.5.1 Modèle Réifié . . . 51
2.5.2 Modèle par contrainte globale : ClosedPatterns . . . 52
2.5.3 Échantillonnage de motifs et contraintes `XOR` . . . 54
2.6 Conclusion . . . 55

3 Fouille interactive de motifs et apprentissage 57
3.1 Introduction . . . 57
3.2 Cadre interactif pour la fouille de motifs . . . 58
3.3 Représentation des motifs . . . 59

3.4 Retours de l'utilisateur . . . 61
3.4.1 Feedback par rangement . . . 62
3.4.2 Feedback quantitatif . . . 62
3.4.3 Feedback implicite . . . 63
3.4.4 Synthèse . . . 64
3.5 Apprentissage des préférences de l'utilisateur . . . 64
3.5.1 Tâches d'apprentissage des préférences . . . 64
3.5.2 Apprentissage d'une fonction de poids . . . 65
3.6 Sélection des motifs . . . 68
3.7 Schéma général de la fouille interactive de motifs et instanciation . . . 69
3.7.1 Schéma général de la fouille interactive de motifs . . . 69
3.7.2 LetSIP . . . 69
3.8 Conclusions . . . 70

III Contributions : diversité, échantillonnage et apprentissage

4 Extraction de motifs diversifiés 73
4.1 Introduction . . . 73
4.2 Modèles FullCP pour la contrainte de diversité . . . 75
4.2.1 Encodage PPC pour la contrainte linéaire de diversité . . . 75
4.2.2 Contraintes linéaires pour la diversification des solutions . . . 76
4.2.3 Procédure de recherche . . . 77
4.3 Relaxations anti-monotones de la contrainte de diversité . . . 77
4.3.1 Motivation . . . 78
4.3.2 Problème relaxé . . . 79
4.3.3 Borne Inférieure de Jaccard . . . 80
4.3.4 Borne supérieure de Jaccard . . . 82
4.4 La contrainte globale ClosedDiversity . . . 84
4.5 Booster la recherche de motifs diversifiés . . . 86
4.5.1 Motif Témoin Positif . . . 86
4.5.2 Mincov : une heuristique de branchement basée sur les fréquences estimées . . . 87
4.5.3 Witness : une heuristique de branchement basée sur les motifs témoins positifs . . . 88
4.5.4 Stratégies d'exploration des sous-arbres témoins . . . 88
4.5.5 Évitement des motifs diversités faux positifs . . . 88
4.6 Échantillonnage de motifs diversifiés . . . 89
4.6.1 Adaptations de l'algorithme Weightgen . . . 89
4.6.2 Oracles de sélection des motifs diversifiés . . . 90
4.6.3 Estimation et contrôle du nombre de contraintes `XOR` . . . 91

4.6.4 Résolution et propagation des contraintes XOR 92
4.7 Conclusions 93

5 Évaluations des méthodes de réductions de la redondance 95
5.1 Étude expérimentale de ClosedDiversity 96
5.1.1 Jeux de données de FIMI et paramétrages 96
5.1.2 Choix des méthodes de l'état de l'art 96
5.1.3 Implantation et évaluation 98
5.2 Résultats et Discussions 99
5.2.1 Apports d'une borne plus resserrée 99
5.2.2 Évaluations de la qualité de la diversification 99
5.2.3 Caractéristiques des motifs extraits 104
5.2.4 Taille des ensembles de motifs 105
5.2.5 Temps d'exécution 106
5.2.6 Évaluation des heuristiques de choix de variables 110
5.2.7 Analyse qualitative des bornes 112
5.3 Étude expérimentale de SDivJaX 113
5.3.1 Protocole expérimental 113
5.3.2 Temps d'exécution de SDivJaX 113
5.3.3 Impact de l'étape d'estimation des `XOR` sur SDivJaX 115
5.3.4 Évaluation de la qualité de diversification 116
5.4 Conclusions 117

6 Motifs discriminants et Apprentissage 119
6.1 Introduction 119
6.2 Nouvelle représentation des motifs 120
6.2.1 Motifs discriminants 120
6.3 Motifs discriminants comme descripteurs 122
6.4 Apprentissage à partir des descripteurs discriminants 123
6.4.1 Fonction d'apprentissage des préférences de l'utilisateur 123
6.4.2 Exploitation des descripteurs discriminants 124
6.5 DiSPaLe : Un cadre interactif de fouille de motifs exploitant les discriminants et la diversité 127
6.6 Évaluation expérimentale 127
6.6.1 Protocole expérimental 128
6.6.2 Évaluation de différents paramètres de DiSPaLe 131
6.6.3 Comparaisons entre DiSPaLe et LetSIP 133
6.6.4 Apports de la diversité dans LetSIP et DiSPaLe 134
6.7 Conclusions 136

IV Conclusions et perspectives

7 Bilan et perspectives **141**

7.1 Bilan . . . 141
7.1.1 Méthodes d'extraction de motifs diversifiés . . . 141
7.1.2 Échantillonnage de motifs diversifiés . . . 142
7.1.3 Nouveaux descripteurs dynamique décrivant le rangement de l'utilisateur . . . 142
7.1.4 Apprentissage des préférences avec les descripteurs discriminants . . . 142
7.2 Perspectives . . . 142
7.2.1 Amélioration de la diversité des ensembles de motifs . . . 142
7.2.2 Nouvelles évaluations de la diversité des ensembles de motifs . . . 143
7.2.3 Amélioration des performances de SDivJaX . . . 144
7.2.4 De nouvelles formes retours utilisateur et de représentation des motifs 144

V Annexes **147**

A Réduction de la redondance **149**

A.1 Temps d'exécution de ClosedDiv-Mincov et de CFTP . . . 150
A.2 Comparaison de la diversité des paires de motifs de ClosedDiv, Flexics, CFTP et Gibbs . . . 151
A.3 Comparaison de la diversité des paires de motifs de ClosedDiv, Picker, PatternsTeam et FullCP-2 . . . 152
A.4 Analyse de la répartition des fréquences des motifs . . . 153
A.5 Analyse de la redondance dans les ensembles de motifs de ClosedDiv et CFTP . . . 154

B Apprentissage **157**

B.1 Résultats complémentaires sur le regret . . . 157
B.2 Résultats complémentaires sur les temps d'execution . . . 157

Bibliographie

Bibliographie

Résumé **174**

Abstract **174**

Liste des tableaux

1.1 Base de données transactionnelle 12
1.2 Jeu de données de classe 1 19
1.3 Exemple d'un jeu de données et de sa table de code 27
1.4 Probabilités de tirage des motifs de la théorie $\mathcal{T}h(\mathcal{L}_{\mathcal{I}}, \mathcal{D}, freq_{\theta \geq 2})$ 29
1.5 Jeu de données d'illustration. 31
1.6 Jaccards de toutes les paires de motifs de Picker, Krimp, Gibbs et CFTP. . . 39

2.1 Jeu de données d'illustration de ClosedPatterns. 52

3.1 Jeu de données d'illustration. 60
3.2 Exemple de feedback quantitatif 63
3.3 Exemple de feedback binaire 63

5.1 Descriptif des bases de transactions retenues pour nos expérimentations 96
5.2 Analyse comparative des deux bornes inférieures 99
5.3 Analyse comparative du nombres de motifs et du nombre de nœuds explorés par les différentes approches PPC 104
5.4 Analyse comparative des temps d'exécution de SDivJaX avec Flexics. 114
5.5 Analyse comparative des temps d'exécution de SDivJaX-CDiv vs SDivJaX-CP. 115

6.1 Caractéristiques des jeux de données utilisés par DiSPaLe 129
6.2 Évaluation des regrets de DiSPaLe-Exp et DiSPaLe-Lin pour chaque descripteur, $\eta = 0.17$, $k = 10$ et pour les deux mesures $Regret_{\max}$ et $Regret_{Avg}$ 132
6.3 Évaluation des regrets de DiSPaLe-Lin ($\eta = 0.17$) et LetSIP pour chaque descripteur, $k = 10$ et pour les deux mesures $Regret_{\max}$ et $Regret_{Avg}$ 133
6.4 Évaluation des regrets de LetSIP+Div et LetSIP pour chaque descripteur et pour différentes valeurs de J_{max} 135

Table des figures

1 Processus d'extraction de connaissance 4

1.1 Treillis de motifs du jeu de données de la table 1 14
1.2 Élagage des motifs de fréquence inférieure à 2 du jeu de données 1 16
1.3 Motifs fermés du jeu de données de la table 1 avec un seuil $\theta = 2$ 20
1.4 Exemple de tuiles 27
1.5 Jeu de données compressé 28
1.6 Table de codes de KRIMP 32
1.7 Jeu de données compressé 33
1.8 Partitions induites par des motifs. Quatre motifs binaires peuvent induire au plus 16 blocs 34
1.9 Motifs obtenu avec PICKER. 35
1.10 Motifs obtenus avec GIBBS à 1000 itérations avec $k = 4$ 36
1.11 Étapes d'échantillonnage de motifs par CFTP 38
1.12 Motifs obtenus avec CFTP avec 2 facteurs de fréquence et $k = 4$ 38

2.1 Espace de recherche de l'exemple 2.1.1 44
2.2 Deux modélisation de la contrainte de différence 50
2.3 Arbre de recherche des motifs fermés avec CLOSEDPATTERNS 53

3.1 Approche générale de fouille et d'apprentissage actif de motifs 58
3.2 Exemple de feedback implicite par clics 63

4.1 Treillis des motifs fermés et fréquents ($\theta = 7$) 78
4.2 Treillis des motifs fermés et fréquents ($\theta = 7$) 79
4.3 Exemple de propagation de contraintes `XOR` utilisant l'élimination de Gauss. . . . 93

5.1 Évaluation de la redondance globale des paires de motifs de CLOSEDDIV, PICKER, PATTERNSTEAM et FULLCP-2. 100
5.2 Évaluation de la redondance globale des paires de motifs de CLOSEDDIV-MINCOV et KRIMP 101
5.3 Évaluation de la redondance globale des paires de motifs de CLOSEDDIV-MINCOV, FLEXICS, GIBBS et CFTP. 102
5.4 Nuages de points des descriptions des motifs 103
5.5 Comparaison des temps d'exécutions de CLOSEDP, CLOSEDP+JACCARD, CLOSEDDIV-MINCOV, CLOSEDDIV-MINCOV+JACCARD et FULLCP-2 106
5.6 Comparaison des temps d'exécution de CLOSEDDIV-MINCOV, FLEXICS, GIBBS et CFTP pour différents seuils θ 108

5.7 Distribution de fréquences cumulatives des motifs 109
5.8 Comparaison des temps d'exécution de ClosedDiv-Mincov et ClosedDiv-Witness 110
5.9 Comparaison des nombres de motifs générés par ClosedDiv-Mincov et ClosedDiv-Witness 111
5.10 Analyse qualitative des relaxations LB et UB 112
5.11 Évolution du nombre de contraintes XOR utilisées par itérations par SDivJaX-CDiv et Flexics. 116
5.12 Indice de Jaccard moyen par itération pour les approches SDivJaX-CDiv. . . . 117
5.13 Évaluation de la redondance globale des paires de motifs de ClosedDiv-Mincov, Flexics, et SDivJaX-CDiv 118

6.1 Schéma général d'apprentissage des poids des descripteurs statiques. 124
6.2 Exploitation des descripteurs discriminants dans l'apprentissage des poids. 125
6.3 Courbes évolutives des poids pour les deux facteurs multiplicatifs. 126
6.4 Effet du paramètre de régularisation η sur le regret pour $\ell = 0$. 130
6.5 Effet du paramètre de régularisation η sur le regret pour $\ell = 1$. 131
6.6 Jeu de données Anneal : vue détaillée des comparaisons entre DiSPaLe-Lin et LetSIP (regret cumulatif et non cumulatif) pour différents descripteurs. $Regret_{\max}$, $k = 10$ et $\ell = 1$. 134
6.7 Nombre de XORs et de Motifs par itérations de LetSIP, LetSIP+Div, DiSPaLe et DiSPaLe +Div 136
6.8 Regrets cumulatifs et non cumulatifs de LetSIP et LetSIP+Div pour le jeu de données Chess 137
6.9 Regrets cumulatifs et non cumulatifs de LetSIP et LetSIP+Div pour le jeu de données Vote 138
6.10 Temps d'exécution en *sec.* de LetSIP, LetSIP+Div, DiSPaLe-Exp et DiSPaLe-E+Div pour la combinaison de descripteurs ILFT. $k = 10$ et $\ell = 1$. 138

A.1 Comparaison des temps d'exécution de ClosedDiv ($J_{max} = 0.5$) et CFTP. . . . 150
A.2 Comparaison de la diversité des paires de motifs de ClosedDiv-Mincov et des méthodes d'échantillonnages. 151
A.3 Comparaison de la diversité des paires de motifs de ClosedDiv, Picker, PatternsTeam et FullCP-2. 152
A.4 Distribution cumulative des fréquences de ClosedDiv et des méthodes d'échantillonnage 153
A.5 Analyse de la redondance dans les ensembles de motifs de ClosedDiv et CFTP 154
A.6 Analyse de la redondance dans les ensembles de motifs de ClosedDiv et CFTP 155

B.1 Regrets cumulatifs et non cumulatifs $Regret_{\max}$ pour $k = 10$ et $\ell = 0$ (Chess et German-credit). 158
B.2 Regrets cumulatifs et non cumulatifs $Regret_{\max}$ pour $k = 10$ et $\ell = 0$ (Heart-cleveland et Hepatitis). 159
B.3 Regrets cumulatifs et non cumulatifs $Regret_{\max}$ pour $k = 10$ et $\ell = 0$ (Kr-vs-kp et Mushroom). 160
B.4 Regrets cumulatifs et non cumulatifs $Regret_{\max}$ pour $k = 10$ et $\ell = 0$ (Soybean et Vote). 161
B.5 Comparaison des durées de chaque itération de LetSIP, LetSIP+Div, DiSPaLe et DiSPaLe +Div. 162

Première partie

Introduction générale

Introduction

Cette thèse porte sur la problématique de l'analyse et de l'extraction de connaissances pertinentes à partir des données. Plus précisément, nous nous intéressons à la prise en compte des préférences de l'utilisateur dans la construction de modèles de qualité. Nous présenterons donc dans ce chapitre une introduction à cette problématique et introduirons notre approche pour essayer de la résoudre.

1 Fouille de données

L'Extraction de Connaissances à partir des Données (ECD) s'est développée en exploitant les techniques de l'intelligence artificielle, des statistiques et des bases de données. Elle peut être définie comme une tâche d'identification de modèles de données pertinents, nouveaux ou surprenants [52]. L'extraction des connaissances se déroule alors selon un processus pouvant être résumé par quatre étapes (voir la figure 1 [1]) :

1. la sélection des données cibles à partir des sources de données brutes ;
2. les pré-traitements et transformations effectués sur les données cibles afin de corriger les incohérences et retenir les données nécessaires à la résolution d'un problème ;
3. la fouille de données pour trouver le modèle de données permettant de résoudre le problème ;
4. l'évaluation du modèle par des experts qui vont l'analyser afin d'en extraire des connaissances.

L'évaluation du modèle consiste à mesurer sa capacité à décrire les informations renfermées dans les ensembles de données ou de prédire leurs évolutions. Pour cela, elle nécessite souvent l'intervention d'experts dont les recommandations servent à mieux paramétrer la construction du modèle afin de le rendre pertinent. Ces recommandations sont présentées sous le terme de *préférences*. Différentes méthodes de fouilles ont été proposées afin de prendre en compte les préférences de l'utilisateur (expert) dans la construction des modèles. Dans le cadre de notre travail, nous nous intéresserons aux méthodes de fouille dédiées à l'extraction de motifs.

Fouille de motifs sous contraintes locales. Le principal défi de la fouille de motifs est d'identifier les motifs qui sont nouveaux (ou intéressants) et utiles (ou exploitables). Silberschatz et Tuzhilin ont souligné que ces notions sont subjectives, c'est-à-dire qu'elles ne dépendent pas seulement des données disponibles, mais aussi des connaissances et des objectifs de l'analyste (l'utilisateur de l'algorithme) [126]. Les premiers travaux ont porté sur de l'extraction de motifs

1. extrait de la thèse de Mehdi Khiari [91]

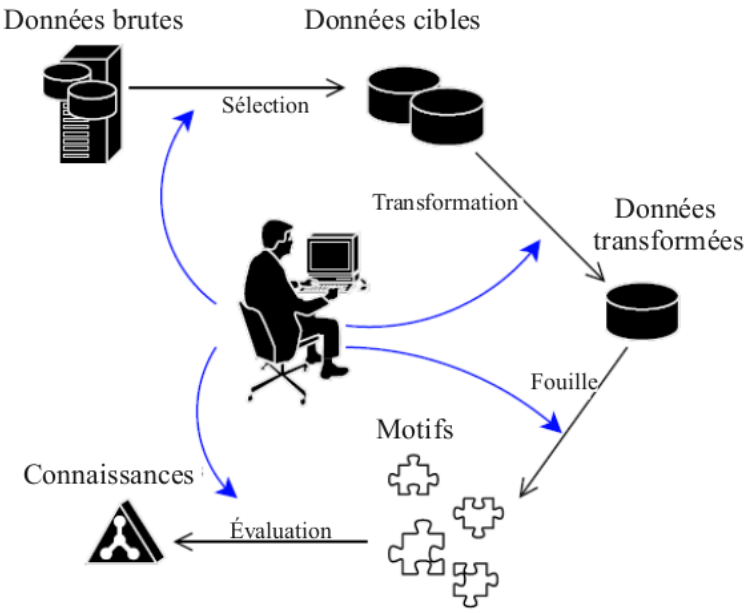


FIGURE 1 – Processus d'extraction de connaissance

sous contraintes locales [74]. Il s'agit d'attribuer grâce à une mesure d'intérêt des valeurs numériques aux motifs afin de représenter leur intérêt pour l'utilisateur. Un seuil fixé à l'avance permet alors de séparer l'ensemble des motifs en deux groupes : les motifs intéressants et les motifs non intéressants. Cela pourrait être vu comme un moyen de prendre en compte l'utilisateur dans le processus de fouille de motifs en incorporant un modèle formel des connaissances et des intérêts de l'utilisateur dans leurs algorithmes.

Parmi les premiers travaux, nous retrouvons ceux de Agrawal *et al.* [3, 1, 2]. Ceux-ci ont proposé des méthodes comme l'algorithme APRIORI [3] qui permet d'extraire des motifs fréquents d'un jeu de données. APRIORI utilise la fréquence comme mesure d'intérêt et sélectionne les motifs X vérifiant une contrainte de fréquence $\mathcal{C} : freq(X) \geq \theta$, avec θ un seuil de fréquence minimum. D'autres mesures d'intérêt ont été proposées parmi lesquelles on retrouve l'aire [59] ou l'entropie. Cependant, l'obtention de motifs intéressants avec ces méthodes traditionnelles d'extraction de motifs peut être un travail difficile et fastidieux. Parmi les difficultés rencontrées, on retrouve le problème de l'explosion de motifs (i.e. un grand nombre de motifs sont extraits dont beaucoup sont redondants), et la non prise en compte des connaissances du domaine.

L'explosion des motifs pose le problème de leur analyse car leur nombre rend cette tâche très difficile et souvent impossible. Les représentations condensées de motifs vérifiant une contrainte $\mathcal{C}$ ont alors été introduites afin de réduire le nombre de motifs générés. Elles s'appuient sur les propriétés des classes d'équivalence et exploitent des motifs pivots à partir desquelles il est possibles de régénérer tous les motifs vérifiant la contrainte $\mathcal{C}$. Les représentations condensées ont fait l'objet de nombreux travaux [109, 26, 31, 27, 131, 130, 129, 32] qui ont permis entre autres le développement des motifs fermées [109]. Toutefois, elles ne permettent pas de résoudre le problème de l'explosion des motifs et la redondance dans les motifs extraits demeure.

Extraction et échantillonnage d'ensemble de motifs. Pour réduire drastiquement le nombre de motifs extraits et réduire l'effort d'analyse, différents travaux se sont développés sur la fouille d'ensemble de motifs. La fouille d'ensemble de motifs permet de prendre en compte les relations entre les motifs et d'optimiser une mesure de qualité m. L'extraction des ensembles de motifs permet alors de réduire le nombre de motifs générés et d'améliorer la qualité des ensembles de motifs extraits. On retrouve ainsi l'approche des *Top-k* [55, 67] qui consiste à extraire les k meilleurs motifs par rapport à une mesure d'intérêt m. On retrouve également les approches par

compression [138], les motifs très informatifs [94], etc. Cependant, étant donné que ces approches procèdent par optimisation d'une mesure de qualité, elles sont limitées par la complexité et la durée des calculs qui les rendent souvent peu performantes.

Pour faire face à cette difficulté, l'échantillonnage de motif a été proposé. Il s'agit d'une approche permettant d'extraire des motifs en effectuant un tirage de ceux-ci proportionnellement à une mesure de qualité. Les méthodes d'échantillonnage permettent de contrôler le nombre de motifs produits et de garantir des temps de réponses raisonables. En effet, un des points forts de l'échantillonnage de motifs est d'offrir un accès direct à l'ensemble des motifs à faible coût tout en garantissant une très bonne diversité des motifs échantillonnés, grâce notamment à son caractère non exhaustif et à sa nature aléatoire. On retrouve entre autres la méthode Flexics [51] qui utilise des résultats récents en SAT pour réduire aléatoirement l'espace des solutions, puis de faire un tirage pondéré dans cet espace ou encore l'échantillonnage à deux étapes de Boley *et al.* [22]. Cependant, les approches d'échantillonnage peuvent être limitées par le fait qu'elles prennent difficilement en compte les préférences de l'utilisateur et n'utilisent qu'un nombre limité de mesures de qualité. Un nouveau paradigme s'est alors développé avec pour objectif de placer l'utilisateur au centre de la fouille.

Fouille interactive de motifs. Ces dernières années, la fouille de motifs a changé peu à peu de paradigme pour évoluer vers un modèle plus centré utilisateur, permettant d'impliquer l'utilisateur dans le processus de fouille de motifs de manière active. Dans ce nouveau paradigme, dénommé fouille interactive de motifs, les préférences de l'utilisateur sont prises en compte dans un processus interactif afin de guider la recherche vers des motifs plus intéressants pour lui. Cela est rendu possible par l'introduction de mécanismes de feedback qui permettent à l'utilisateur de spécifier ses préférences sur les motifs extraits. Les travaux de *Joachims* [83], de Bhuiyan *et al.* [20] et de *Dzyuba et Van Leeuwen* [47, 48] ont alors permis de proposer des outils intéressants de fouille interactive de motifs.

2 Objectifs et contributions

La question de la prise en compte des préférences explicites de l'utilisateur et la fourniture de recommandations fondées sur leurs préférences sont en rapport direct avec le développement de systèmes interactifs d'aide à la décision. **L'objectif de cette thèse** est de développer de nouvelles méthodes permettant la prise en compte explicites des préférences et/ou des connaissances de l'utilisateur dans le processus de fouille pour extraire des connaissances intéressantes. Nous avons retenu le cadre de la fouille interactive de motifs. L'idée centrale est d'alterner entre phases d'extraction de motifs et phases d'apprentissage sur les motifs intéressants selon le scénario suivant :

(1) à partir d'une requête initiale de l'utilisateur, le système extrait un premier lot de motifs d'une taille relativement petite, a priori pertinents ;

(2) l'utilisateur retourne son "feedback" en sélectionnant certains de ces motifs, les désignant comme réellement intéressants pour lui ;

(3) le système considère ces motifs comme des exemples des préférences de l'utilisateur et affine son "profil" (i.e. le système apprend les préférences de l'utilisateur) ;

(4) une nouvelle collection de motifs est extraite en utilisant ces préférences mises à jour, celle-ci est présentée à l'utilisateur, et retour à l'étape (1).

Une telle démarche soulève de nombreux défis méthodologiques à relever :

Q1 Comment assurer une production rapide de résultats (l'utilisateur ne doit pas attendre, sinon il décroche du système) et une *diversité* dans les résultats (sinon, l'utilisateur va se lasser) ?

Q2 Quels types de retours peut formuler un utilisateur et comment les capturer ?

Q3 Comment représenter les préférences des utilisateurs ? Comment exploiter ces préférences et les traduire en contraintes exploitables par les méthodes de fouille pour extraire des motifs plus intéressants ?

Dans cette thèse, nous nous concentrons sur les questions de recherche soulevées ci-dessous, qui concernent les étapes majeures de la fouille interactive de motifs.

Les principales contributions de cette thèse concernant **Q1** sont les suivantes :

— Un premier modèle (deux variantes) basé sur la programmation par contraintes, noté FullCP, pour extraire des motifs fréquents fermés et diversifiés. La diversité est contrôlée par une contrainte de seuil sur l'indice de Jaccard, une mesure de la similarité des couvertures des motifs. Notre idée est de formuler cette contrainte sous la forme d'une contrainte linéaire obtenue en décomposant l'indice de Jaccard en opérations simples sur des ensembles utilisant des variables booléennes intermédiaires représentant les couvertures des motifs.

— Un cadre générique qui exploite une première relaxation anti-monotone de l'indice de Jaccard, basée sur une borne inférieure, permettant de filtrer des motifs redondants et d'extraire des motifs fréquents, fermés et diversifiés [79, 77, 78]. Cette relaxation est intégrée à travers une nouvelle contrainte globale, dénommée ClosedDiversity, avec son algorithme de filtrage. Une seconde relaxation, basée sur une borne supérieure, est également exploitée via une nouvelle heuristique de branchement permettant de guider le processus de recherche vers des motifs diversifiés. Le grand attrait de cette approche est que nous sommes capables de déduire des propriétés anti-monotones qui peuvent être efficacement poussées dans notre cadre de programmation par contraintes, même si la contrainte Jaccard elle-même ne satisfait pas ces propriétés.

— Un nouvel algorithme d'échantillonnage de motifs, dénommé SDivJaX (pour "Sample Diverse pattern with Jaccard and Xor constraints") qui s'appuie sur les dernières avancées en matière d'échantillonnage pondéré dans SAT [35] et qui exploite notre contrainte globale ClosedDiversity, qui lui permet de garantir la diversité des motifs extraits et une production rapide des motifs.

— Une étude expérimentale exhaustive permettant de comparer plusieurs classes de méthodes de réduction de la redondance de la littérature : PatternsTeam [93], Picker [29], Flexics [51], CFTP [22], Gibbs [14] et Krimp [138]. Il s'agit de la toute première comparaison directe entre les résultats produits par ces méthodes, même si certaines de ces techniques ont été publiées il y a plus d'une décennie.

Notons que différentes méthodes de réduction de la redondance ont été proposées dans la littérature [51, 138, 29, 14, 22]. Cependant, il n'existe à notre connaissance aucune approche déclarative en fouille de motifs proposant de mesurer de façon explicite la redondance et d'utiliser une contrainte de diversité entre les motifs.

Concernant **Q3**, les principales contributions sont les suivantes :

— Une nouvelle classe de descripteurs de la qualité des motifs, exploitant les *motifs discriminants*, pour représenter les préférences de l'utilisateur. Les motifs discriminants peuvent être considérés comme étant des sous-motifs corrélés positivement ou négativement avec rangement de l'utilisateur. La recherche de ces motifs exploite les rangs numériques donnés aux motifs individuels comme des étiquettes numériques.

— Une nouvelle méthode d'apprentissage exploitant les motifs discriminants pour apprendre des poids sur les descripteurs issus des motifs discriminants, en exploitant la méthode stochastique de descente des coordonnées (SCD) [122]. Les poids appris sont ensuite

agrégés aux poids d'autres descripteurs statiques représentant certaines caractéristiques syntaxiques des motifs (comme les items, les transactions couvertes, la longueur et la fréquence des motifs). L'intérêt de l'utilisateur pour un motif X est alors obtenu grâce à une fonction de pondération ϕ qui utilise les poids appris et qui est mise à jour en fonction des retours de l'utilisateur.

— Une approche de fouille interactive de motifs, dénommée DiSPaLe (pour **Di**scriminating **S**ub-**Pa**ttern feature **Le**arning) ; elle exploite les motifs discriminants pour apprendre les poids d'une fonction de pondération des motifs. Par ailleurs, notre approche offre également la possibilité d'échantillionner des motifs diversifiés grâce à notre algorithme d'échantillonnage de motifs SDivJaX.

3 Organisation du mémoire

Ce manuscrit est organisé en deux parties :
— la première partie fait un état de l'art des méthodes de fouille et de réduction de la redondance dans les motifs ainsi que des méthodes d'apprentissage ;

— la deuxième partie présente nos différentes contributions sur la fouille de motifs diversifiés et sur l'apprentissage des préférences de l'utilisateur.

Plus précisément, le **chapitre 1** présente plus en détails la fouille de motifs et d'ensemble de motifs. Nous présenterons également différentes mesures de qualité et les méthodes de la littérature permettant de réduire la redondance dans les ensembles de motifs extraits.

Dans le **chapitre 2**, nous présenterons la programmation par contrainte et montrerons comment nous l'utiliserons pour l'extraction de motifs.

Au **chapitre 3**, nous introduirons la fouille interactive de motifs et détaillerons ses ingrédients essentiels comme les descripteurs de motifs, les formes de retours de l'utilisateur et les approches d'apprentissage.

Dans le **chapitre 4**, nous détaillerons nos contributions sur la fouille de motifs diversifiés : un premier modèle PPC (FullCP) pour l'extraction de motifs fréquents fermés et diversifiés ; un second modèle qui exploite la contrainte globale ClosedDiversity. Nous terminerons ce chapitre par la présentation de notre algorithme d'échantillonnage de motifs SDivJaX, qui exploite notre contrainte globale ClosedDiversity pour garantir la diversité des motifs extraits et une production rapide des motifs. Le chapitre est basé sur les recherches précédemment publiées dans les articles suivants :

— A. Hien, S. Loudni, N. Aribi, Y. Lebbah, M. El Amine Laghzaoui, A. Ouali, and A. Zimmermann. A relaxation-based approach for mining diverse closed patterns. *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2020 Proceedings*, Part I, volume 12457 of LNCS, pages 36–54. Springer, 2020.

— A. Hien, S. Loudni, N. Aribi, Y. Lebbah, M. Laghzaoui, A. Ouali, and A. Zimmermann. Fouille de motifs fermés et diversifiés basée sur la relaxation. *Conférence Internationale Francophone sur la Sciences des Données (CIFSD2021)*, Marseille, France, 2021. **Prix du 3 e meilleur article**.

Le **chapitre 5** est dédié à l'évaluation expérimentale de nos méthodes de fouille et d'échantillonnage de motifs diversifiés. Nous évaluerons les apports pratiques de notre contrainte globale ClosedDiversity par rapport aux méthodes complètes de réduction de la redondance (comme PatternsTeam et Picker) et aux méthodes d'échantillonnage de motifs (comme Flexics, CFTP et Gibbs). Nous discuterons les résultats obtenus en termes de diversité des solutions,

de temps de calcul et de quelques caractéristiques sur les motifs extraits. Enfin, nous évaluerons les apports de SDivJaX et ses différentes variantes pour l'échantillonnage de motifs diversifiés en termes de temps de calcul. Le chapitre est basé sur les deux articles journaux suivants :

— A. Hien, S. Loudni, N. Aribi, Y. Lebbah, A. Ouali, A. Zimmermann. Fouille de Motifs Diversifiés : Une approche Basée sur la Relaxation et l'échantillonnage. **Article soumis** au *Numéro spécial de la Revue des Nouvelles Technologies de l'Information (RNTI)*. pp.1-16, 2022.

— A. Hien, S. Loudni, N. Aribi, Y. Lebbah, M. Laghzaoui, A. Ouali, and A. Zimmermann. A Relaxation-based Approach for Diversity Pattern Set Mining. **Soumission en cours** à la revue *Constraints*, pages 1-65. Springer, 2022.

Le **chapitre 6** présente un algorithme interactif de découverte de motifs qui rassemble plusieurs briques algorithmiques dans certaines sont introduites dans les chapitres précédents. Il utilise les motifs discriminants comme nouvelle classe de descripteurs de la qualité des motifs, exploite ces nouveaux descripteurs pour l'apprentissage des préférences de l'utilisateur sous la forme d'une fonction de pondération logistique. La fonction ainsi apprise est ensuite exploitée dans un algorithme d'échantillonnage de motifs diversifiés. Le chapitre est basé sur l'article article suivant :

A. Hien, S. Loudni, N. Aribi, A. Ouali, A. Zimmermann. Exploiting complex pattern features for interactive pattern mining (2022). **Soumission à PAKDD 2023 en cours**.

En **conclusion**, nous ferons un bilan de nos différents travaux, puis nous discuterons des éventuelles perspectives.

Deuxième partie

État de l'art : Fouille de motifs et apprentissage

CHAPITRE 1

Fouille de motifs

Contents

1.1 Cadre formel et définitions 12
1.2 Extraction de motifs sous contraintes locales 14
1.2.1 Quelques préliminaires mathématiques 14
1.2.2 Contraintes 15
1.2.3 Espace de recherche 15
1.2.4 Mesures d'intérêt 17
1.2.5 Représentation condensée 20
1.2.6 Discussion 21
1.3 Extraction d'ensembles de motifs 21
1.3.1 Théorie des ensembles de motifs 21
1.3.2 Mesures d'intérêt et redondance 22
1.4 Diversification des solutions 24
1.4.1 Contrainte de redondance sur les paires de motifs 25
1.4.2 Contrainte sur les transactions redondantes 25
1.4.3 Discussion 25
1.5 La compression des jeux de données 26
1.5.1 MDL 26
1.5.2 Réduction de la redondance par la compression 26
1.6 Échantillonnage de motifs 28
1.6.1 Échantillonnage par marche aléatoire 29
1.6.2 Échantillonnage en deux étapes 29
1.6.3 Échantillonnage par le formalisme SAT 30
1.7 Approches de réduction de la redondance en fouille de motifs 30
1.7.1 Approche exhaustive : PatternsTeam 30
1.7.2 Approche par compression : Krimp 31
1.7.3 Approche heuristique : Picker 33
1.7.4 Approche d'échantillonnage par marche aléatoire : Gibbs 35
1.7.5 Approche d'échantillonnage en deux étapes : CFTP 36
1.7.6 Synthèse 38
1.8 Conclusion 39

Dans ce chapitre, nous introduirons la problématique de la fouille de motifs diversifiés. Nous introduirons les notions principales de la fouille de motifs et présenterons les contraintes de fouille ainsi que les mesures d'intérêt. Nous détaillerons le principe de la représentation condensée de motifs ainsi que les méthodes de résolution des problèmes de fouille. Nous terminerons par une

présentation des différentes mesures d'évaluation de la redondance, suivie par une description détaillée des méthodes de réduction (explicite ou implicite) de la redondance en fouille de motifs.

1.1 Cadre formel et définitions

Dans cette section, nous allons nous intéresser à la fouille de motifs ensemblistes. Il s'agit d'un domaine qui s'intéresse aux données sous forme d'une base de transactions, les données sont des transactions et chacune d'entre elle indique la présence d'objets/items. Cela peut s'apparenter à un supermarché qui enregistre ce qu'achète chaque client lors du passage à la caisse. À partir de cette base de données, l'objectif est de trouver des motifs intéressants.

1.1.1 Jeux de données

La version de base de l'extraction de motifs permet de faire la fouille dans une relation (table) d'une base de données transactionnelle dont les valeurs sont des booléens (indiquant la présence ou l'absence d'une propriété) :

Définition 1 (Jeu de données)**.**
Un jeu de données est un triplet $(\mathcal{I}, \mathcal{R}, \mathcal{T})$ où :
- *$\mathcal{I} = \{i_1, ..., i_n\}$ est l'ensemble des items : il s'agit de l'ensemble des littéraux (attributs)*
- *$\mathcal{T} = \{t_1, ..., t_m\}$ est l'ensemble des transactions : chaque transaction t_j est un sous-ensemble d'items : $t_j \subseteq \mathcal{I}$*
- *$\mathcal{R}$ est une relation binaire entre $\mathcal{I}$ et $\mathcal{T}$:*

$$\mathcal{R}(i,t) = \begin{cases} 1 \text{ si } i \text{ est présent dans } t : i\mathcal{R}t \\ 0 \text{ sinon} : \neg i\mathcal{R}t \end{cases}$$

Considérant la base transactionnelle $\mathcal{D}$ de la table 1.1, onze transactions étiquetées $t_1, ..., t_{11}$ sont décrites par 6 items A, B,..., F. Dans cet exemple, chaque transaction appartient à une seule classe et est étiquetée par un item de classe (c_1 ou c_2). On notera $\mathcal{T}_1$ l'ensemble des transactions de classe 1 et $\mathcal{T}_2$ celui de classe 2.

$\mathcal{T}$	Classe	Items
t_1	c_1	A
t_2	c_1	A B D
t_3	c_1	A D E
t_4	c_1	A D E
t_5	c_1	B D E
t_6	c_1	B D E
t_7	c_2	C D E
t_8	c_2	C D E
t_9	c_2	C D F
t_{10}	c_2	C D F
t_{11}	c_2	C E F

Classe	$\mathcal{T}$ \ $\mathcal{I}$	A	B	C	D	E	F
c_1	t_1	1	0	0	0	0	0
	t_2	1	1	0	1	0	0
	t_3	1	0	0	1	1	0
	t_4	1	0	0	1	1	0
	t_5	0	1	0	1	1	0
	t_6	0	1	0	1	1	0
c_2	t_7	0	0	1	1	1	0
	t_8	0	0	1	1	1	0
	t_9	0	0	1	1	0	1
	t_{10}	0	0	1	1	0	1
	t_{11}	0	0	1	0	1	1

Table 1.1 – Base de données transactionnelle

Il existe différentes représentations pour les bases de données transactionnelles qui offrent chacune des avantages lors de l'extraction des motifs. En effet, les performances des algorithmes d'extraction de motifs peuvent être améliorées par la structure des données. Nous présentons quelques représentations rencontrées dans la littérature.

La représentation horizontale $\mathcal{H}_\mathcal{D}$. C'est la représentation la plus répandue. La base est stockée sous forme d'une liste de transactions, où chaque transaction est une liste des items qu'elle contient : $\forall t \in \mathcal{T}, \mathcal{H}_\mathcal{D}(t) = \{i \in \mathcal{I}, | i \in t\}$. Il s'agit de la représentation utilisée dans l'algorithme Apriori [3].

La représentation verticale $\mathcal{V}_\mathcal{D}$. Dans cette représentation, les items sont d'abord stockés dans une liste. Ensuite, pour chaque item, les transactions qui le contient sont listées. L'ensemble de ces transactions est noté $\mathcal{T}_\mathcal{D}(i)$. Cette représentation est utilisée par l'algorithme Eclat [146] et dans la contrainte globale ClosedPatterns [97]. C'est cette représentation que nous utiliserons dans la contrainte globale ClosedDiversity [79].

La représentation booléenne $\mathcal{B}_\mathcal{D}$ Les données transactionnelles sont représentées sous la forme d'une matrice d'incidence booléenne de taille $n \times m$ avec $\mathcal{D}_{ti} = 1$ si $i \in t$, 0 sinon (voir table 1.1). Il s'agit de la représentation utilisée dans [39] et [64].

La représentation hybride $\mathcal{VH}_\mathcal{D}$. Elle tire partie des représentations horizontale $\mathcal{H}_\mathcal{D}$ et verticale $\mathcal{V}_\mathcal{D}$ qu'elle maintient en parallèle. Chaque transaction t dans $\mathcal{V}_\mathcal{D}$ dispose d'un pointeur sur son contenu $\{i \mid i \in t\}$ dans $\mathcal{H}_\mathcal{D}$. Cela permet, en accédant à la couverture d'un item i, de connaître le contenu d'une transaction t (les items qu'elle comporte) de la couverture. La représentation hybride est efficace pour calculer la couverture des motifs en facilitant l'intersection des couvertures des items. Elle est utilisée dans l'algorithme LCM [135].

La représentation arborescente $\mathcal{A}_\mathcal{D}$. La représentation arborescente ou FP-Tree se présente sous forme d'une représentation horizontale compressée. Elle consiste à représenter un arbre préfixé, en fusionnant les transactions ayant les mêmes préfixes. Trier les items dans un ordre décroissant des fréquences, en commençant l'arbre par les items les plus fréquents, permet d'avoir un arbre plus compact. Cette représentation est utilisée dans l'algorithme FP-Growth [66].

Ces représentations sont exploitées pour extraire des relations utiles entre les objets de $\mathcal{T}$ qui sont décrits par des sous-ensembles d'items appelés *motifs*.

1.1.2 Motifs ensemblistes

Définition 2 (Motif).
Un motif ou itemset est un sous ensemble non vide de $\mathcal{I}$.

Par exemple, $\{A, E\}$, $\{B, D\}$, $\{C, E, F\}$ sont des motifs de la table 1.1. Par la suite, nous utiliserons des chaînes de caractères pour désigner les motifs. Ainsi, BD désignera le motif $\{B, D\}$.

Définition 3 (Taille et Couverture d'un motif).
Étant donné un motif X et un ensemble de transactions $\mathcal{T}$, la taille de X est le nombre d'items qu'il contient (i.e. sa cardinalité) : taille(X) = $|X|$. La couverture de X est l'ensemble des transactions qui le supportent. Une transaction $t \in \mathcal{T}$ supporte un motif X si et seulement si, $\forall i \in X$, $i\mathcal{R}t$ (ou $X \subseteq t$). Ainsi, la couverture de X vaut : $\mathcal{V}_\mathcal{D}(X) = \{t \in \mathcal{D} \mid X \subseteq t\}$

Définition 4 (Langage).
Le langage de motifs $\mathcal{L}_\mathcal{I}$ est l'espace de recherche des motifs construit sur l'alphabet $\mathcal{I}$, autrement dit, il s'agit de l'ensemble des parties de $\mathcal{I}$ (i.e. $2^{|\mathcal{I}|} \backslash \emptyset$) soit au total $2^{|\mathcal{I}|} - 1$ motifs au maximum.

La tâche d'extraction de motifs consiste alors à parcourir l'espace de recherche (qui comporte tous les motifs du langage $\mathcal{L}_\mathcal{I}$) afin de sélectionner un ou plusieurs motifs. Cet espace de recherche $\mathcal{L}_\mathcal{I}$ des motifs est appelé treillis (voir figure 1.1).

Exemple 1.1.1.
L'espace de recherche du jeu de données de la table 1.1 comporte 63 motifs que nous retrouvons dans le treillis de la figure 1.1 :

- *6 motifs de taille 1 : A, B, C, D, E et F*
- *15 motifs de taille 2 : AB, AC, AD, AE, AF, BC, BD, BE, BF, CD, CE, CF, DE, DF et EF*

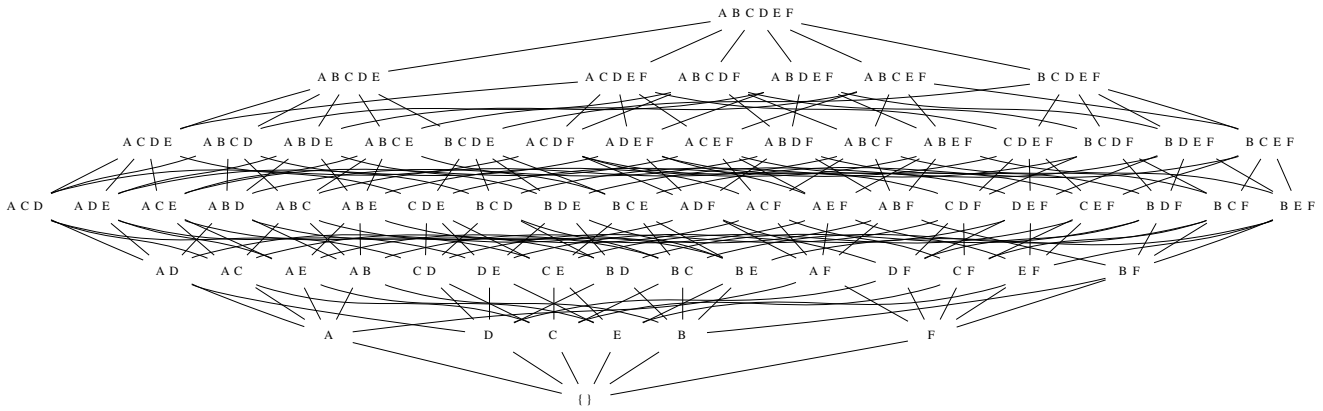


FIGURE 1.1 – Treillis de motifs du jeu de données de la table 1

— *20 motifs de taille 3 : ABC, ABD, ABE, ABF, ACD, ACE, ACF, ADE, ADF, AEF, BCD, BCE, BCF, BDE, BDF, BEF, CDE, CDF, CEF et DEF*
— *15 motifs de taille 4 : ABCD, ABCE, ABCF, ABDE, ABDF, ABEF, ACDE, ACDF, ACEF, ADEF, BCDE, BCDF, BCEF, BDEF, et CDEF*
— *6 motifs de taille 5 : ABCDE, ABCDF, ABCEF, ABDEF, ACDEF et BCDEF*
— *1 motif de taille 6 : ABCDEF*

1.2 Extraction de motifs sous contraintes locales

Le problème d'extraction de motifs peut être une tâche algorithmique difficile. En effet, le nombre de motifs potentiels pouvant être extraits est très grand $(2^{|\mathcal{I}|} - 1)$ et peut rendre la recherche très fastidieuse. Pour rendre cette tâche moins fastidieuse, il est possible de définir des contraintes sur le langage. Dans cette section, nous donnerons dans un premier lieu certains préliminaires mathématiques, puis nous présenterons les contraintes et aborderons l'exploration de l'espace de recherche.

1.2.1 Quelques préliminaires mathématiques

Une relation d'ordre est une relation binaire $\preceq$ dans un ensemble S permettant de comparer ses éléments entre eux.

— Un **ordre partiel** $\preceq$ dans S est une relation telle que $\forall a, b, c \in S$:
  — $\preceq$ est réflexive : $a \preceq a$
  — $\preceq$ est anti-symétrique : $a \preceq b \wedge b \preceq a \Rightarrow a = b$
  — $\preceq$ est transitif : $a \preceq b \wedge b \preceq c \Rightarrow a \preceq c$

— Une relation d'ordre $\preceq$ est symétrique lorsque : $\forall a, b \in S : a \preceq b \Rightarrow b \preceq a$.

— Un ensemble avec un ordre partiel est dit partiellement ordonné.

— Soit a et b deux éléments distincts d'un ensemble partiellement ordonné. Cet ensemble sera noté $(S, \preceq)$.
  — Si $a \preceq b$ ou $b \preceq a$ alors a et b sont dit comparables.
  — Si $a \npreceq b$ et $b \npreceq a$ alors a et b sont dit incomparables.

— Si toutes les paires d'éléments a et b de S sont comparables, l'ordre $\preceq$ est total.

Ces propriétés permettent de définir la notion d'équivalence dans une relation d'ordre.

Définition 5 (Relation et Classe d'équivalence)**.**
On appelle ***relation d'équivalence*** *toute relation d'ordre $\preceq$ dans un ensemble S qui est réflexive, symétrique et transitive. On la note alors $\sim$. La* ***classe d'équivalence*** *désigne pour chaque élément $X \in S$ l'ensemble $\{Y \in S \mid X \sim Y\}$.*

Dans un espace ordonné (ou partiellement ordonné), il est également possible de définir des fonctions. Ces fonctions possèdent des propriétés qu'il peut être intéressant d'étudier.

Propriété 1. *Une fonction $f : \mathbb{R} \to \mathbb{R}$ est monotone (respectivement anti-monotone) si $\forall x, y \in \mathbb{R}$:*

$$\begin{cases} x \leq y \Rightarrow f(x) \leq f(y) \\ x \geq y \Rightarrow f(x) \geq f(y) \end{cases}$$

La fonction f est anti-monotone si $\forall x, y \in \mathbb{R}$:

$$\begin{cases} x \leq y \Rightarrow f(y) \geq f(x) \\ x \geq y \Rightarrow f(y) \leq f(x) \end{cases}$$

Ces préliminaires mathématiques seront exploitées tout au long de ce chapitre, notamment dans les sections 1.2.3, 1.2.5 et 1.7.

1.2.2 Contraintes

Définition 6 (Contrainte)**.**
Une contrainte C est un prédicat défini sur un langage ou un jeu de données :

$$C : \mathcal{L}_{\mathcal{I}} \to \{vrai, faux\}$$

L'extraction de motifs sous contraintes consiste à extraire les motifs satisfaisant une contrainte C à partir d'un jeu de données. Les contraintes permettent ainsi d'évaluer l'intérêt d'un motif selon des critères fixés par l'utilisateur. Il est alors possible de formaliser l'ensemble des motifs satisfaisant ces critères.

Définition 7 (Théorie)**.**
Étant donné un langage $\mathcal{L}_{\mathcal{I}}$, un jeu de donné $\mathcal{D}$ et une contrainte C, la théorie $\mathcal{T}h(\mathcal{L}_{\mathcal{I}}, \mathcal{D}, C)$ est l'ensemble des motifs de $\mathcal{L}_{\mathcal{I}}$ satisfaisant C dans $\mathcal{D}$ [74].

Les contraintes peuvent ainsi servir à cibler des motifs bien précis, qui sont intéressants pour l'utilisateur, à ordonner (ranger) des motifs, ou à réduire le nombre de motifs pouvant être extraits.

Exemple 1.2.1.
Un exemple de contrainte sur les motifs serait de fixer leur taille, c'est à dire le nombre d'items qui composent le motif. Ainsi, si l'utilisateur est intéressé par des motifs de taille 3 dans le jeu de données de la table 1.1, on aura la théorie suivante :

$$\mathcal{T}h(\mathcal{L}_{\mathcal{I}}, \mathcal{D}, C_{taille}) = \{X \in \mathcal{L}_{\mathcal{I}} \mid taille(X) = 3\} =$$
$$\{ABC, ABD, ABE, ABF, ACD, ACE, ACF, ADE, ADF, AEF, BCD, BCE,$$
$$BCF, BDE, BDF, BEF, CDE, CDF, CEF \text{ et } DEF\}$$

1.2.3 Espace de recherche

La tâche d'extraction de motifs sous contraintes se fait de manière incrémentale. Les motifs sont construits en combinant progressivement les différents items du jeu de données et en vérifiant que le motif obtenu respecte la contrainte. Comme on peut le voir dans la figure 1.1, à chaque niveau, plusieurs nouvelles combinaisons d'items sont réalisées. Ainsi, le premier niveau correspond au motif vide $\emptyset$, puis le deuxième correspond au motif de taille 1 (A, B, C, D, E et F). On peut alors définir une relation d'ordre sur les motifs du langage $\mathcal{L}_{\mathcal{I}}$ et exploiter certaines propriétés mathématiques (définies plus haut) afin de faciliter l'extraction des motifs.

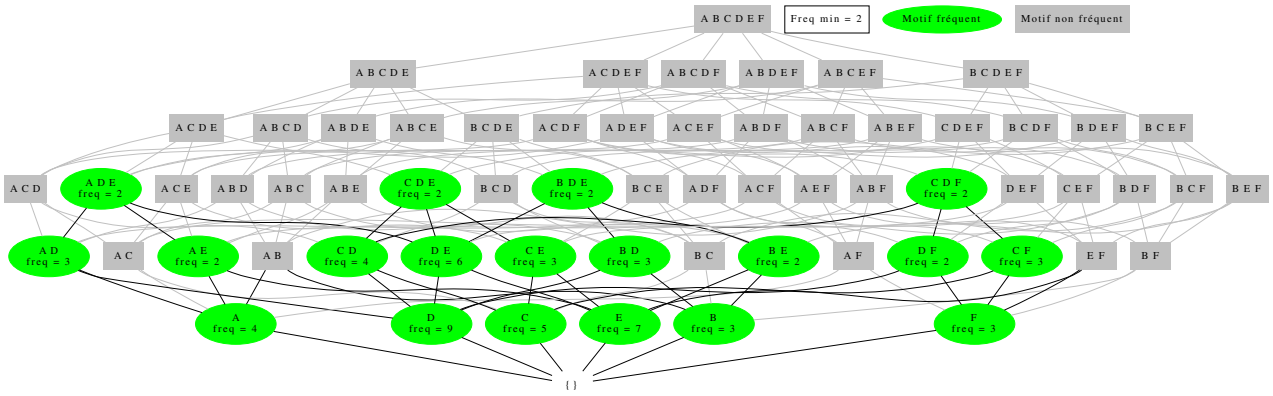


FIGURE 1.2 – Élagage des motifs de fréquence inférieure à 2 du jeu de données 1

Définition 8 (SPÉCIALISATION)**.**
Une relation de spécialisation $\preceq$ est une relation d'ordre partiel définie sur les motifs de $\mathcal{L}_\mathcal{I}$. Un motif X est plus spécifique (respectivement plus générique) qu'un motif Y si $Y \preceq X$ (respectivement $X \preceq Y$).

Un exemple de relation de spécialisation est l'inclusion ensembliste $\subseteq$ entre deux motifs. Si un motif X est inclus dans un motif Y ($X \subseteq Y$), on dit que Y est plus spécifique que X et que X est plus générique que Y. Ainsi, on peut dire que le motif BDE est plus spécifique que le motif D car $D \subseteq BDE$.

En exploitant la propriété 1, on peut exprimer la monotonie des contraintes et exploiter les propriétés qui en découlent dans l'extraction de motifs sous contraintes [25] [106].

Définition 9 (CONTRAINTE MONOTONE)**.**
Une contrainte C est monotone par rapport à une relation de spécialisation $\preceq$ si et seulement si $\forall X, Y \in \mathcal{L}_\mathcal{I}, X \preceq Y \Rightarrow (C(X) \Rightarrow C(Y))$

Définition 10 (CONTRAINTE ANTI-MONOTONE)**.**
Une contrainte C est anti-monotone par rapport à une relation de spécialisation $\preceq$ si et seulement si $\forall X, Y \in \mathcal{L}_\mathcal{I}, X \preceq Y \Rightarrow (C(Y) \Rightarrow C(X))$

Ainsi, pour deux motifs X et Y tels que $X \preceq Y$, si une contrainte monotone C est satisfaite sur X, alors elle l'est également sur Y. De même, si C est anti-monotone et satisfaite sur Y, alors, elle l'est aussi sur X.

Les propriétés de monotonie et d'anti-monotonie sont exploitées par les algorithmes d'extraction de motifs pour élaguer les motifs qui ne respectent pas les contraintes définies. En effet, si un motif ne vérifie pas une contrainte anti-monotone, alors, il est certain que toutes ses spécialisations ne satisfont pas cette contrainte. L'exploration de l'espace de recherche peut alors exclure les motifs ne vérifiant pas les contraintes et éviter de générer leurs spécialisations ce qui permet un gain de performance important.

Propriété 2. *Si un motif ne respecte pas une contrainte monotone (respectivement anti-monotone), alors toutes ses généralisations (respectivement spécialisations) ne la respectent pas.*

Exemple 1.2.2. *Dans la figure 1.2, nous illustrons l'extraction des motifs avec la contrainte de fréquence définies plus tard (voir définition 14) :* $freq(X) \geq 2, \forall X \in \mathcal{L}_\mathcal{I}$. *Les motifs marqués en vert représentent ceux dont la fréquence est supérieure ou égale à 2 et les autres ont une fréquence inférieure. Dans la construction itérative des motifs, la violation de la contrainte de fréquence sera détectée en premier dans les motifs dont la fréquence est inférieur au seuil (motifs en gris). Ceux-ci seront alors élagués, ce qui évitera d'explorer leurs spécialisations.*

Les propriétés de monotonie et d'anti-monotonie peuvent ainsi faciliter l'exploration de l'espace de recherche et permettre une extraction efficace des motifs des plus génériques au plus spécifiques ou inversement. Parallèlement, l'utilisation de contraintes non monotones ou anti-monotones peut rendre l'extraction des motifs plus difficile. Afin de faciliter l'utilisation de telles contraintes, Kifer *et al.* ont proposé dans [92] la notion de motif *témoin*. Un motif témoin est un motif (qui n'est pas nécessairement le plus général ou le plus spécifique) pour lequel on peut vérifier la satisfaction d'une contrainte afin de dériver des propriétés sur d'autres motifs.

Les motifs témoins apportent une nouvelle approche conceptuelle dans l'exploitation des contraintes : on passe d'une approche qui consiste à rechercher les propriétés pouvant être exploitées pour faciliter l'exploration à une approche qui consiste à identifier les motifs dont les propriétés permettent de déduire d'autres propriétés sur d'autres motifs.

Définition 11 (Motif témoin)**.**
Soit P et Q deux motifs, et $C : \mathcal{L}_{\mathcal{I}} \mapsto \{true, false\}$. Le motif W tel que $P \subseteq W \subseteq P \cup Q$ est appelé motif témoin positif (respectivement négatif) si et seulement si $\forall M$ tel que $P \subseteq M \subseteq P \cup Q : C(W) = true \Rightarrow C(M) = true$ (resp. $C(W) = false \Rightarrow C(M) = false$).

Une illustration des motifs témoins est donnée à l'exemple 1.2.3. Ainsi donc, la fouille de motifs sous contraintes permet d'extraire des motifs présentant un intérêt pour l'utilisateur. Cet intérêt peut être évalué avec des mesures sur des motifs pris individuellement ou par groupes. Dans la suite de cette section, nous aborderons donc la fouille de motifs intéressants et présenterons différentes mesures d'intérêts.

1.2.4 Mesures d'intérêt

Comme nous l'avons vu à la section précédente, l'utilisation de contraintes permet de réduire l'espace de recherche et de sélectionner des motifs selon leur intérêt. Cet intérêt peut être évalué en utilisant des mesures [60] qui mettent l'accent sur la couverture [3], la fiabilité [107, 132], la particularité [95, 147], la diversité [80], la surprise [100, 7], l'utilité [144, 36] des motifs.

Définition 12 (Mesure d'intérêt)**.**
Étant donné un jeu de donné $\mathcal{D}$, une mesure m est une fonction qui associe une valeur à un motif : $m : \mathcal{L}_{\mathcal{I}} \mapsto \mathbb{R}$

L'expression de la mesure évolue donc en fonction de l'intérêt recherché. Nous verrons dans la suite trois mesures d'intérêt ainsi que les contraintes qui les exploitent.

A. Fréquence

La fréquence est l'une des mesures de base utilisée en fouille de motifs. Elle permet de comptabiliser le nombre d'apparition d'un motif dans un jeu de données.

Définition 13 (Fréquence d'un motif)**.**
La fréquence d'un motif X est égale au cardinal de sa couverture : $freq(X) = |\mathcal{V}_{\mathcal{D}}(X)| = |\{t \in \mathcal{D} \mid X \subseteq t\}|$.

La fréquence est donc une mesure qui utilise la couverture pour mesurer l'intérêt des motifs. Cette mesure permet alors d'identifier les récurrences dans le jeu de données. Ces récurrences, appelés motifs fréquents, sont des ensembles d'items qui ont un nombre d'apparition minimal supérieur à un seuil donné par l'utilisateur. Plus un motif couvre de transactions, plus il peut être alors susceptible d'être intéressant [60]. L'utilisateur peut ainsi exprimer une préférence sur la fréquence des motifs à extraire.

L'extraction de motifs fréquents peut être réalisée en définissant une contrainte de fréquence permettant de sélectionner les motifs dont la fréquence est supérieure à un seuil θ donné.

Définition 14 (Contrainte de fréquence minimale)**.**
Un motif X est fréquent si et seulement si il vérifie la contrainte : $freq(X) \geq \theta$

L'extraction de motifs fréquents consiste alors à extraire les motifs de la théorie :

$$\mathcal{T}h(\mathcal{L}_\mathcal{I}, \mathcal{D}, freq_\theta) = \{X \in \mathcal{L}_\mathcal{I} | freq(X) \geq \theta\}$$

L'extraction de motifs fréquents est facilitée par l'exploitation de l'anti-monotonie de la contrainte de fréquence qui permet d'élaguer efficacement les motifs ayant une fréquence inférieure au seuil θ.

Exemple 1.2.3.
Pour un seuil $\theta = 2$, nous pouvons extraire les motifs suivants : A, B, C, D, E, F, AD, AE, BD, BE, CD, CE, CF, DE, DF, ADE, BDE, CDE, CDF.
Dans ce cas, le motif AB est un témoin négatif, par contre, ABD est un témoin positif.

La valeur de la fréquence peut être indiquée de deux manières :
— la fréquence absolue dont la valeur est un entier (entre 1 et $|\mathcal{T}|$) qui indique la couverture du motif ;
— la fréquence relative, sous forme de pourcentage, indique la proportion de transactions couvertes par le motif.

B. Aire

Définition 15 (Aire d'un motif)**.**
L'aire d'un motif [59] X est définie par : $aire(X) = freq(X) \times taille(X)$.

Utilisée dans une contrainte avec un seuil d'aire minimale, cette mesure permet de rechercher des motifs à la fois « suffisamment » fréquents et longs. Sur le jeu de données dans la table 1, onze motifs satisfont la contrainte $aire(X) \geq 6$. Il s'agit des motifs D « 9 », E « 6 », AD « 6 », BD « 6 », CD « 8 », CF « 6 », DE « 12 », ADE « 6 », BDE « 6 », CDE « 6 », CDF « 6 » (les valeurs entre « », sont les valeurs de l'aire).

C. Entropie

L'entropie est une mesure qui permet de quantifier la quantité d'information que représente un motif dans un jeu de données. Pour cela, elle évalue le nombre de bits nécessaires pour représenter le motif dans le jeu de données en calculant la projection binaire de ce motifs dans les différentes transactions.

Définition 16 (Projection binaire)**.**
La projection binaire (ou simplement la projection) B d'un motif X dans une transaction t est une représentation binaire de taille $|X|$ matérialisant la présence ou l'absence de chaque item de X dans t : $B = (b_1, ..., b_k) \in \{0, 1\}^k$, avec $k = |X|$. Chaque item i présent est représenté par $b_i = 1$ et les items absents sont représentés par un $b_i = 0$. Étant donné un jeu de données $\mathcal{D}$ avec un ensemble de transactions $\mathcal{T}$, la projection d'un motif X dans $\mathcal{D}$ se note $\mathcal{D}(\mathcal{T}_X)$ et représente l'ensemble des projection de X dans les transactions $t \in \mathcal{T}$ du jeu de données $\mathcal{D}$.

Exemple 1.2.4.
Considérons les motifs de classe 1 du jeu de données de la table 1.1. On peut s'intéresser aux motifs de ce nouveau jeu de données, notamment les motifs A et DE.
— *Le motif A est présent dans quatre transactions et absent des deux restantes. On peut donc matérialiser sa présence avec un seul bit qui vaudra 1 si une transaction contient le motif et 0 si non.*

$\mathcal{T}$	Items			
t_1	A			
t_2	A	B	D	
t_3	A		D	E
t_4	A		D	E
t_5		B	D	E
t_6		B	D	E

TABLE 1.2 – Jeu de données de classe 1

— *Le motif DE est également présent dans quatre transactions (t_3, t_4, t_5, t_6) et absent des deux autres (t_1, t_2). Par contre, une partie de DE (l'item D) est présente dans la transaction t_2. La projection binaire du motif DE dans les transactions du jeu de données donne ainsi trois représentations possibles :*

1. $(1, 1)$ *: dans ce cas, le motif DE est entièrement présent dans la transaction (cas de t_3, t_4, t_5, t_6) ;*
2. $(0, 0)$ *: DE est totalement absent de la transaction (cas de t_1) ;*
3. $1, 0$ *: seul D est présent dans la transaction t_2.*

On aura donc besoin de plus d'un bit pour représenter le motif. Pour évaluer le nombre de bits évoqué plus haut, il est possible de passer par le calcul de l'entropie du motif. L'entropie d'un motif se définie alors comme suit :

Définition 17 (ENTROPIE D'UN MOTIF)**.**
L'entropie d'un motif X dans un jeu de données $\mathcal{D}$ mesure la proportion d'informations que représente ce motif. Elle s'exprime comme suit :

$$H(\mathcal{T}, X) = - \sum_{B \in \{0,1\}^k} \frac{|freq(x_1=b_1,...,x_k=b_k)|}{|\mathcal{T}|} log_2(\frac{|freq(x_1=b_1,...,x_k=b_k)|}{|\mathcal{T}|}),$$

avec $X = \{x_1, ..., x_k\}$ un motif de taille k, et $B = (b_1, ..., b_k) \in \{0, 1\}^k$ un tuple de valeurs binaires représentant les différentes projections de X dans le jeu de données $\mathcal{D}$.

Une valeur d'entropie élevée indique que le motif présente plusieurs projections binaires dans le jeu de données et nécessite donc plus de bits. Ce motif peut cependant être intéressant comme le montre la proposition 1.

Proposition 1. *Soit un motif $X = \{x_1, ..., x_k\}$. Alors : $H(\mathcal{T}, X) \leq \sum_{i \in [1,k]} H(\mathcal{T}, x_i)$*

La proposition 1 établie qu'un motif X de taille k a une entropie inférieure à la somme des entropies de ses généralisations de taille 1. Le motif X occupera donc moins de bits que l'ensemble de ses généralisations. Ainsi, l'entropie peut être utilisée pour extraire des ensembles de motifs permettant de compresser le jeu de données [138]. En effet, une entropie faible indique que le nombre de projections du motif est faible. Cela se traduit alors par le fait que les items du motif apparaissent très souvent dans les mêmes sous-ensembles de transactions. On rencontre cette situation dans les généralisations des motifs, c'est-à-dire, les motifs de taille réduite. Dans le cadre d'une compression de jeu de données, privilégier ces motifs avec une entropie faible peut conduire au risque de devoir augmenter le nombre de motifs nécessaire pour couvrir toutes les transactions du jeu de données, ce qui aura pour effet d'augmenter la taille du jeu de données compressée. Par contre, une entropie élevée indique qu'il existe de nombreuses transactions que le motif ne couvre que partiellement. Il suffira alors de sélectionner les motifs de telle sorte que leurs projections binaires soient complémentaires, ce qui aura pour effet de réduire le nombre de motifs nécessaire à la compression et de réduire la taille du jeu de données compressée.

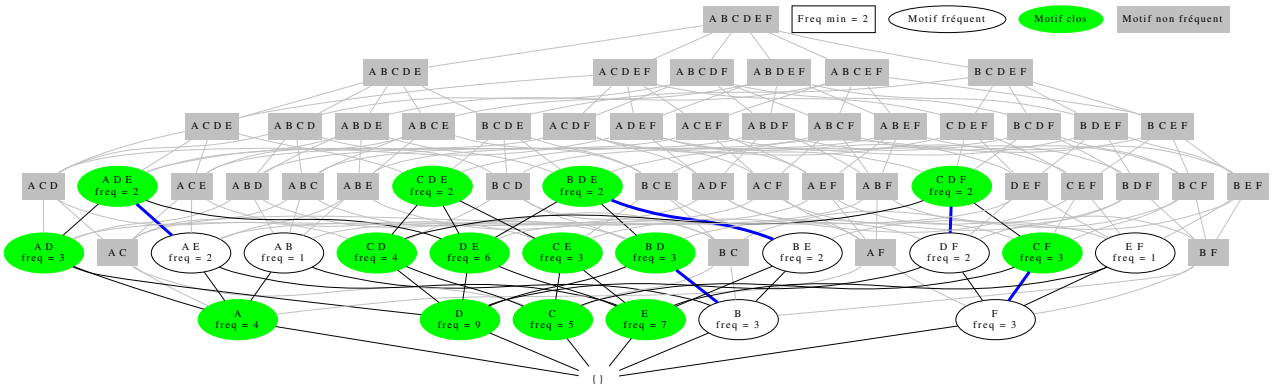


FIGURE 1.3 – Motifs fermés du jeu de données de la table 1 avec un seuil $\theta = 2$

1.2.5 Représentation condensée

Comme nous l'avons vu à la définition 4, l'extraction des motifs peut générer un nombre important de motifs. En plus du problème d'explosion des motifs, les motifs extraits peuvent présenter de nombreuses redondances entre eux. Les redondances font référence au fait que plusieurs motifs décrivent les mêmes parties du jeu de données. Afin de réduire l'ensemble des motifs fréquents et obtenir un ensemble moins large de motifs, il est possible d'utiliser les **représentations condensées exactes**. L'idée est alors 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 d'une théorie $\mathcal{T}h(\mathcal{L}_\mathcal{I}, \mathcal{D}, C)$. D'autres représentations condensées existent notamment les représentations condensées dites approximatives qui se différencient des premières par les fréquences régénérées des motifs qui sont approximatives tandis que les autres fréquences sont exactes.

Les représentations condensées exactes de motifs satisfaisant une contrainte C permettent d'augmenter l'efficacité des algorithmes d'extraction et réduisent également le nombre de motifs générés. Elles reposent sur le principe des classes d'équivalence : l'idée centrale est de n'extraire qu'un nombre réduit de motifs à partir desquels il est possible de régénérer, si on le souhaite, tous les motifs vérifiant C. Dans la suite de ce manuscrit, nous nous intéresserons à une représentation condensée de motifs, à savoir les motifs fermés [109]. Il existe toutefois comme autre représentation condensée, les motifs libres [28].

Définition 18 (MOTIFS FERMÉS)**.**
Un motif $X \in \mathcal{L}_\mathcal{I}$ est fermé par rapport à la fréquence si et seulement si toutes ses spécialisations strictes ont une fréquence strictement inférieure à celle de X :

$$fermé(X) \iff \forall Y \supset X,\ freq(Y) < freq(X).$$

Définition 19 (MOTIFS LIBRES)**.**
Un motif $X \in \mathcal{L}_\mathcal{I}$ est libre si et seulement si toutes ses généralisations strictes ont une fréquence strictement supérieure à celle de X :

$$libre(X) \iff \forall Y \subset X,\ freq(Y) > freq(X).$$

Les motifs libres et fermés structurent le treillis des motifs en classes d'équivalence. En effet, une classe d'équivalence est délimitée par un ou plusieurs motifs libres qui constituent ses éléments minimaux et par un seul motif fermé qui constitue son élément maximal (minimaux et maximal s'entendent au sens de la taille des motifs). Tous les motifs d'une même classe d'équivalence ont ainsi la même couverture. Pour chaque classe d'équivalence, seul le plus grand motif au sens de l'inclusion est retenu (c'est le motif fermé).

Dans la figure 1.3, nous retrouvons les motifs fermés du jeu de données de la table 1. Les motifs fermés sont représentés en vert et les motifs fréquents non fermés dans des nœuds sans

couleur. Les autres motifs ne sont pas fréquents ($\theta = 2$). Nous avons également relié par un arc de couleur bleu les motifs appartenant à la même classe d'équivalence. Ainsi, DF et CDF sont dans la même classe (CDF étant fermé) ; il en est de même pour AE et ADE. Nous retrouvons également des classes d'équivalence contenant un seul motif. C'est le cas des classes d'équivalence contenant les motifs $C, D, CE, DE, etc.$ Ainsi, pour $\theta = 2$, on retrouve au total 14 classes d'équivalence contenant chacune un motif fermé et éventuellement d'autres motifs (non fermés). Les motifs fermées sont des représentations condensées qui permettent de réduire le nombre de motifs générés et de réduire la redondance entre les motifs.

1.2.6 Discussion

Une des limites de l'extraction de motifs locaux se trouve dans le choix de la mesure d'intérêt. En effet, il peut être difficile d'exprimer sous forme de fonction un intérêt sur les motifs. Par ailleurs, il n'est pas toujours possible d'exploiter la monotonie ou l'anti-monotonie afin de rendre la recherche plus efficace. De plus, l'intérêt d'un motif local ne dépend pas des autres motifs extraits. En effet, la réponse d'une requête d'extraction étant un ensemble de motifs, l'utilisateur désire souvent découvrir des motifs de plus haut niveau reposant sur des caractéristiques qui impliquent plusieurs motifs locaux et donnant au final un sens global à l'ensemble de motifs qui est retourné. Pour cela, l'intérêt d'un motif ne doit pas être considéré de façon indépendante des autres motifs extraits mais il est nécessaire de comparer les motifs entre eux.

Enfin, la réduction de la redondance avec les motifs fermés ne s'effectue qu'au niveau des motifs qui sont membre de la même classe d'équivalence, c'est-à-dire les motifs qui ont la même couverture que leur spécialisation et leur généralisation. Or, il existe également des redondances entre des motifs de classe d'équivalence différente. Par ailleurs, l'utilisateur pourrait être intéressé par des motifs dont la redondance ne dépasse pas un seuil. Dans ce cas, l'utilisation de mesures de diversité peut être utile.

1.3 Extraction d'ensembles de motifs

Les motifs locaux décrivent des phénomènes locaux et permettent de ce fait d'avoir une compréhension du jeu de données uniquement limitée à quelques sous-ensembles. Par ailleurs, les méthodes d'extraction de motifs locaux peuvent générer énormément de motifs difficiles à analyser par l'utilisateur à cause de leur nombre mais également à cause des nombreuses redondances très souvent rencontrées.

Pour mieux modéliser et décrire l'ensemble du jeu de données, l'on pourrait s'intéresser aux ensembles de motifs. L'objectif est de sélectionner des sous-ensembles de motifs locaux qui permettent de décrire l'ensemble du jeu de données et qui ont l'avantage d'être moins nombreux que les motifs locaux. C'est motifs qui ont un intérêt global peuvent alors être plus pertinent pour l'utilisateur.

1.3.1 Théorie des ensembles de motifs

Une des limites des contraintes locales vient de leur incapacité à modéliser les relations entre différents motifs. En effet, l'utilisateur peut être intéressé par des motifs dont l'extraction tient compte de l'évaluation d'autres motifs. Les contraintes sur les motifs ainsi que les mesures d'intérêt ne sont alors plus focalisées sur un seul motif local mais sur des ensembles de motifs.

Définition 20 (Contrainte d'ensemble de motifs)**.**
Une contrainte d'ensemble de motifs est un prédicat $\mathbb{C} : 2^{\mathcal{L}_\mathcal{I}} \rightarrow \{vrai, faux\}$ *[148].*

Avec les ensembles de motifs, l'intérêt exprimé par l'utilisateur porte sur un ensemble de motifs. Afin de définir les types d'ensembles de motifs pouvant être recherchés, De Raedt et al. [40]

ont défini des primitives (support, taille, redondance, représentativité). Ces primitives peuvent être combinées avec des agrégats (comme la moyenne, min, max, $\sum$) portant sur un ensemble de motifs afin d'obtenir des contraintes sur les ensembles de motifs.

L'extraction d'ensemble de motifs s'effectue en deux phases : une première phase d'extraction de motifs sous contraintes locales et une deuxième phase de construction des ensembles de motifs à partir des motifs extraits à la première phase. Ainsi, on a la théorie des ensembles de motifs suivantes :

$$\mathcal{T}h(2^{\mathcal{L}_\mathcal{I}}, \mathcal{D}, \mathbb{C}) = \{\mathbb{X} \in 2^{\mathcal{L}_\mathcal{I}} | \mathbb{C}(\mathbb{X}) \text{ est vrai}\}$$

Pour l'extraction d'ensembles de motifs, la contrainte $\mathbb{C}$ doit être satisfaite par chaque ensemble de motifs produit c'est-à-dire chaque sous-ensemble de $2^{\mathcal{L}_\mathcal{I}}$. Pour cela, il est possible d'adapter et exploiter les propriétés de monotonie et d'anti-monotonie comme le montrent les auteurs de [40].

Les méthodes d'extraction d'ensembles de motifs utilisent des approches qui sont très souvent relatives à un problème donné. Différents types de motifs peuvent alors être définis avec des contraintes d'ensembles de motifs. Nous retrouvons par exemple les *Pattern teams* [93], les motifs construits par relation d'équivalence [29], les motifs utilisés pour la compression [138], les motifs très informatifs [94], etc.

1.3.2 Mesures d'intérêt et redondance

Contrairement aux contraintes locales sur les motifs qui permettent de déterminer l'intérêt individuels des motifs, l'évaluation des ensembles de motifs prend en compte les relations entre les différents motifs de l'ensemble. Les mesures et les contraintes sur les ensembles peuvent alors être réalisées de deux manières :

- mesures sur les paires de motifs : chaque motif de l'ensemble est évalué par rapport à tous les autres motifs pris individuellement ;
- mesures sur tous les motifs de l'ensemble : l'évaluation est réalisée sur tous les motifs en même temps.

Définition 21 (Mesures sur des paires de motifs)**.**
Une mesure m sur des paires des motifs est définie comme suit :

$$m : \mathcal{L}_\mathcal{I} \times \mathcal{L}_\mathcal{I} \rightarrow \mathbb{R}.$$

Définition 22 (Mesures sur les ensembles de motifs)**.**
Les mesures d'ensemble M sont définies comme suit :

$$M : 2^{\mathcal{L}_\mathcal{I}} \rightarrow \mathbb{R}.$$

Différentes mesures d'intérêt peuvent être définies sur les ensembles de motifs en fonction des besoins de l'utilisateur. Ainsi, on pourra s'intéresser aux mesures permettant de résoudre le problème d'explosion des motifs observé dans la fouille de motifs sous contraintes locales. En effet, ces motifs présentent beaucoup de redondance entre eux du fait de la non prise en compte des relations entre eux. Il est alors important de pouvoir connaître la similarité entre les motifs. Cette similarité peut être mesurée sur les items ou sur les couvertures des motifs. Dans notre travail, nous traiterons uniquement la diversité sur les couvertures.

A. Similarité entre paires de motifs

L'évaluation de la redondance dans un ensemble de motifs peut s'effectuer en considérant les différentes paires de motifs de l'ensemble. Différentes mesures permettent d'effectuer cette évaluation.

A1) Redondance et chevauchement.

Définition 23 (Redondance et chevauchement)**.**
Considérons deux motifs P_1 et P_2. Leur chevauchement est défini par :

$$ovlp(P_1, P_2) = \mathcal{V}_{\mathcal{D}}(P_1) \cap \mathcal{V}_{\mathcal{D}}(P_2)$$

On définit alors la redondance entre les deux motifs en quantifiant leur chevauchement :

$$chevauchement(P_1, P_2) = |ovlp(P_1, P_2)|$$

L'évaluation du chevauchement permet de mesurer la similarité d'une paire de motifs et permet ainsi de connaître le degré de redondance entre les deux motif. Elle peut être évaluée de façon relative en divisant sa valeur par $|\mathcal{T}|$.

A2) Différence symétrique
Une autre méthode d'évaluation de la similarité consiste à calculer la différence symétrique.

Définition 24 (Différence symétrique)**.**
Considérons deux motifs P_1 et P_2. L'ensemble défini par :

$$diff(P_1, P_2) = \mathcal{V}_{\mathcal{D}}(P_1) \cup \mathcal{V}_{\mathcal{D}}(P_2) - ovlp(P_1, P_2)$$

est appelé différence symétrique de P_1 et P_2.

La différence symétrique permet d'évaluer la proportion de transactions couvertes exclusivement par chacun des deux motifs. Contrairement à la mesure de chevauchement qui mesure la similarité entre les motifs, elle permet d'évaluer leur dissimilarité.

A3) Indice de Jaccard
L'indice de Jaccard, lui, peut être utilisé pour mesurer à la fois la similarité et la dissimilarité dans une paire de motifs.

Définition 25 (Indice de Jaccard)**.**
Soient deux motifs X et Y, l'indice de Jaccard mesure la proportion de chevauchement entre les couvertures des deux motifs : $Jac(X, Y) = \frac{|\mathcal{V}_{\mathcal{D}}(X) \cap \mathcal{V}_{\mathcal{D}}(Y)|}{|\mathcal{V}_{\mathcal{D}}(X) \cup \mathcal{V}_{\mathcal{D}}(Y)|}$.

La mesure de la similarité s'effectue sur la proportion de transactions partagées par les deux motifs. La dissimilarité peut alors être obtenue en effectuant le calcul $Jac(X, Y)_{dis} = 1 - Jac(X, Y)$.

B. Similarité dans les ensembles de motifs

L'évaluation de la redondance dans les ensembles de motifs peut également s'effectuer de façon globale sur l'ensemble des motifs de l'ensemble. Une des pistes pour cette approche consiste à comptabiliser les transactions redondantes. Il s'agit des transactions qui sont couvertes par au moins deux motifs d'un ensemble $\mathbb{X}$. La redondance peut alors être modélisée par des variables booléennes $u_t, t \in \mathcal{T}$ tel que $(u_t = 1)$ si et seulement si la transaction t est couverte par au moins deux motifs [108]. Le nombre de transactions redondantes est ainsi modélisé par :

$$trans\_red(\mathbb{X}, \mathcal{T}) = \frac{\sum_{t \in \mathcal{T}} u_t}{|\mathcal{T}|}$$

C. Discussions

Les deux approches présentées précédemment présentent chacune quelques avantages et des limites. L'évaluation de la redondance entre les paires de motifs est intéressante car elle permet d'identifier de façon précise les sous-ensembles de motifs ayant apporté beaucoup de redondance à

l'ensemble. Cependant, cette approche procède de façon locale sur chaque motif et ne permet pas d'évaluer la redondance de façon globale sur tout l'ensemble de motifs. Toutefois, en utilisant des agrégateurs, il est possible d'avoir une vision globale de la redondance dans l'ensemble. Comme agrégateurs, on peut utiliser la somme $\sum$, le maximum max, le minimum min ou la moyenne Avg. Enfin, étant donné que l'évaluation s'effectue sur les différentes paires, cette approche peut ne pas passer à l'échelle notamment lorsque la taille de l'ensemble de motifs devient très grande.

L'approche d'évaluation de la redondance sur l'ensemble des motifs, elle, permet d'évaluer la redondance de façon globale et prend en compte tous les motifs de l'ensemble. Elle est donc à priori meilleure que l'approche d'évaluation des paires. Toutefois, cette approche ne permet pas d'identifier de façon précise les motifs redondants. De ce fait, ses bons résultats peuvent être facilités par un effet de noyade qui permet aux transactions non redondantes de masquer l'existence de quelques transactions très redondantes.

Exemple 1.3.1.
Soit l'ensemble de motifs $\mathbb{X} = \{AB, DE, CE, F\}$ *suivant avec leurs couvertures :*

- AB *avec* $\mathcal{V}_{\mathcal{D}}(AB) = \{t_1, t_2, t_3\}$
- DE *avec* $\mathcal{V}_{\mathcal{D}}(DE) = \{t_1, t_2, t_4, t_5\}$
- CE *avec* $\mathcal{V}_{\mathcal{D}}(CE) = \{t_1, t_2, t_6, t_7\}$
- F *avec* $\mathcal{V}_{\mathcal{D}}(F) = \{t_1, t_2, t_8\}$

En utilisant la mesure l'indice de Jaccard et comme agrégateur la moyenne Avg, on obtient : $Jaccard_{\mathbb{X}} = Avg_{X,Y \in \mathbb{X}, X \neq Y} Jac(X, Y) = 0.41$. *En utilisant la mesure red_trans, on obtient* $red\_trans(\mathbb{X}) = 0.25$.

Dans l'exemple 1.3.1, *red_trans* permet de comprendre qu'il y a une redondance sur 25% des transactions. Cependant, étant donné que cette mesure n'arrive pas à identifier efficacement les redondances entre les paires de motifs, elle n'arrive pas à identifier le fait que 50% de la couvertures de chaque motif est partagée par les autres motifs.

Dans le chapitre 4, nous utiliserons donc l'indice de Jaccard comme mesure d'évaluation de la redondance entre les motifs car il permet d'évaluer la proportion de redondance dans une paire de motifs et permet l'utilisation de seuil de redondance.

1.4 Diversification des solutions

Dans la section précédente, nous avons présenté différentes mesures d'évaluation de la similarité dans les ensembles de motifs. La réduction de la redondance consiste alors à trouver un ensemble $\mathbb{X}$ qui optimise une mesure de similarité *red*. Le problème de la diversité des solutions peut ainsi se présenter comme suit :

Définition 26 (Problème de la diversification des solutions)**.**
Soit $k \geq 2$ *un entier, et* $\mathcal{S}$ *un ensemble de motifs. Le problème de diversification consiste à trouver un sous-ensemble* $\mathbb{X} \subseteq \mathcal{S}$ *de taille* k *tel que*

$$redondance(\mathbb{X}) \leq redondance(\mathbb{Y}) \ \forall \ \mathbb{Y} \subseteq \mathcal{S}.$$

Il s'agit de trouver un ensemble $\mathbb{X}$ minimisant la redondance entre ses motifs. Pour le résoudre, nous pouvons utiliser des contraintes de redondances sur les motifs. Les différentes mesures de redondances définies précédemment établissent différentes manières d'évaluer la redondance dans les ensembles de motifs. Les contraintes de redondance peuvent alors s'exprimer de différentes façons.

1.4.1 Contrainte de redondance sur les paires de motifs

Le problème de la diversité entre les paires de motifs a été étudié par Hebrard *et al.* dans [72] et peut se présenter comme suit :

Définition 27 (Problème de la diversification entre des paires)**.**
Soit $k \geq 2$ un entier, et $\mathcal{S}$ un ensemble de motifs. Le problème MaxDiverseKSet(k) *consiste à trouver un sous-ensemble $\mathbb{X} \subseteq \mathcal{S}$ de taille k tel que pour tout $\mathbb{Y} \subseteq \mathcal{S}$, $|\mathbb{Y}| = k$:*

$$\max_{\substack{X_1, X_2 \in \mathbb{X} \\ X_1 \neq X_2}} red(X_1, X_2) \leq \max_{\substack{Y_1, Y_2 \in \mathbb{Y} \\ Y_1 \neq Y_2}} red(Y_1, Y_2).$$

Étant donné S un ensemble de motifs déjà extraits (un historique de motifs), la problème MostDistant(S) *consiste à trouver un motif $X \in \mathbb{X}$ tel que $\forall\ Y \in \mathbb{X}$:*

$$\max_{Z \in S} red(X, Z) \leq \max_{Z \in S} red(Y, Z)$$

MaxDiverseKSet(k) permet ainsi de trouver un ensemble $\mathbb{X}$ de k motifs minimisant une mesure de redondance red. Pour trouver cet ensemble, il est possible d'utiliser MostDistant(S) qui est une méthode gloutonne permettant de trouver le motif X qui minimise le mieux la redondance avec un ensemble S d'autres motifs. On peut donc approximer MaxDiverseKSet(k) en résolvant MostDistant(S) k fois. En considérant $X_i \in \mathbb{X} = \{X_1, \ldots, X_k\}$ comme le i^e motif trouvé par MostDistant(S), on peut écrire :

$$X_i = \text{MostDistant}(S),\ S = \{X_1, \ldots, X_{i-1}\}.$$

La diversité dans $\mathbb{X}$ est alors assurée en agrégeant les diversités de toutes les paires de motifs [40] :

$$\bigotimes_{X_i, X_j \in \mathbb{X}, X_i \neq X_j} red(X_i, X_j) \lesseqgtr \theta_{red}$$

où θ_{redd} représente un seuil de redondance.

1.4.2 Contrainte sur les transactions redondantes

L'évaluation de la redondance est effectuée sur les différentes transactions du jeu de données. Pour cela, on utilise une variable booléenne T_t qui vaut 1 si et seulement s'il existe au moins un motif $X \in \mathbb{X}$ qui couvre t ($X \subseteq t$). On utilise également une matrice binaire a de $|\mathcal{T}|$ lignes et $|\mathbb{X}|$ colonnes tel que $a_{t,i}$ vaut 1 si et seulement si X_i couvre la transaction t : $X_i \subseteq t$. La contrainte de redondance s'exprime alors comme suit [108] :

$$2u_t \leq \sum_{i \in [0, |\mathbb{X}|]} a_{t,i} X_i \leq y_t + |\mathbb{X}|.u_t \qquad \forall t \in \mathcal{T}$$

1.4.3 Discussion

L'extraction d'ensembles de motifs constitue une tâche importante en fouille de données permettant, d'une part, de contrôler le nombre parfois énorme de motifs locaux extraits, et d'autre part, de découvrir des corrélations entre ces motifs. Cependant, la modélisation du problème d'extraction d'ensemble de motifs peut être complexe à cause de la difficulté d'exprimer des relations entre les motifs d'un même ensemble. Par ailleurs, les méthodes d'extractions sont souvent peu efficaces et nécessitent pour certaines une procédure en deux étapes passant par une génération coûteuse de motifs locaux. Il peut alors être intéressant d'exploiter les techniques d'échantillonnage de motifs.

Dans la section 1.7, nous présenterons quelques méthodes de fouille d'ensembles de motifs permettant de réduire la redondance entre les motifs de l'ensemble.

1.5 La compression des jeux de données

La compression de données est une tâche qui consiste à décrire les jeux de données avec des ensembles d'items permettant d'obtenir des représentations de tailles plus réduites que celles des jeux de données initiaux. Elle peut s'apparenter à la fouille de motifs dans laquelle les ensembles d'items utilisés pour la compression forment un ensemble de motifs respectant une contrainte sur la taille de la description du jeu de données en entrée. Cette taille peut alors être évaluée par une mesure qui calcule le gain obtenu en représentant le jeu de données par un ensemble de motifs.

La compression consiste ainsi à trouver l'ensemble de motifs qui décrit le mieux le jeu de données tout en minimisant la taille de la description.

1.5.1 MDL

L'un des principes exploité pour trouver l'ensemble de motifs R est celui de MDL (Minimum Description Length ou Longueur de Description Minimale) [115]. Le principe de MDL est de trouver l'ensemble de motifs qui donne la plus courte description des données. Plus précisément, étant donné l'ensemble de motifs de la théorie $\mathcal{T}h(\mathcal{L}_\mathcal{I}, \mathcal{D}, \mathcal{C})$, MDL recherche le sous-ensemble de motifs $\mathbb{X} = \{X_1, X_2, \dots\} \subseteq \mathcal{T}h(\mathcal{L}_\mathcal{I}, \mathcal{D}, \mathcal{C})$ tel que

$$\nexists \mathbb{Y} \subseteq \mathcal{T}h(\mathcal{L}_\mathcal{I}, \mathcal{D}, \mathcal{C}) \text{ avec } L(\mathbb{Y}, \mathcal{D}) < L(\mathbb{X}, \mathcal{D})$$

avec

- $L(\mathbb{X}, \mathcal{D}) = L(\mathbb{X}) + L(\mathcal{D} \mid \mathbb{X})$, la longueur totale du jeu de données compressé et de l'ensemble de motifs utilisé ;
- $L(\mathbb{X})$ la longueur en bits de la description $\mathbb{X}$;
- $L(\mathcal{D} \mid \mathbb{X})$ la longueur en bits de la description du jeu de données lorsqu'il est codé avec $\mathbb{X}$.

Pour minimiser la taille totale $L(\mathbb{X}, \mathcal{D})$ du jeu de données compressé, il est important de minimiser celles de $L(\mathbb{X})$ et de $L(\mathcal{D} \mid \mathbb{X})$. Pour cela, les différentes approches utilisant MDL [73, 124, 125, 127, 133, 138] associent à chaque motif $X \in \mathcal{T}h(\mathcal{L}_\mathcal{I}, \mathcal{D}, \mathcal{C})$ un code c_X de telle sorte que toutes les transactions $t \in \mathcal{T}$ soient décrites par un ensemble de code $C(t)$. Une table de correspondance permet alors d'associer chaque motif X à un code c_X et à la liste des transactions dans lesquelles X est utilisée. Dans cette table, seules les motifs $X \in \mathbb{X}$ seront ainsi associés à une transaction. Cette table des codes (ou « code table ») CT est alors exploitée pour réaliser la compression et minimiser la taille du jeu de données compressé.

Exemple 1.5.1.
Soit le jeu de données suivant et sa table de code correspondant. Le motif C a été mis en gris (voir la table 1.3b) car il ne sera pas utilisé pour la compression ; sa couverture est alors vide.

1.5.2 Réduction de la redondance par la compression

Comme nous l'avons vu dans la section précédente, la compression consiste à trouver un ensemble de motifs permettant de décrire le mieux le jeu de données avec une taille de description réduite. Certaines méthodes exploitant MDL pour la compression [138, 102] annoncent une réduction de la redondance dans les ensembles de motifs trouvés. Cependant, cette réduction de la redondance doit être relativisée et replacée dans son contexte. En effet, les motifs extraits doivent être considérés comme des tuiles [59] qui associent un ensemble d'items et un ensemble de transactions.

Définition 28 (Tuiles [14])**.**
Une **tuile** $\tau = (I, T)$ *se définie comme une paire ordonnée d'items $I \subseteq \mathcal{I}$ et de transactions $T \subseteq \mathcal{T}$ tel que*

$$\tau(I, T) = \{(t, i) \mid t \in T, i \in I\}$$

	A	B	C
t_1	×	×	×
t_2	×	×	×
t_3	×	×	×
t_4	×	×	×
t_5	×	×	×
t_6	×	×	
t_7	×		
t_8		×	

(a) Exemple de jeu de données

⇒

Motifs			Codes	Couvertures
A	B	C	C_{ABC}	$\{t_1, t_2, t_3, t_4, t_5\}$
A	B		C_{AB}	$\{t_6\}$
A			C_A	$\{t_7\}$
	B		C_B	$\{t_8\}$
		C	C_c	$\emptyset$

(b) Table de code correspondant

TABLE 1.3 – Exemple d'un jeu de données et de sa table de code

L'ensemble d'items I de la tuile peut être considéré comme un motif tel que $T \subseteq \mathcal{V}_{\mathcal{D}}(I)$. Il est alors possible de construire un ensemble de tuiles τ_i présentant peu de chevauchement étant donné que les ensembles de transactions associées peuvent être plus réduites que les couvertures des motifs.

Exemple 1.5.2.
Dans la figure 1.4a, les tuiles $\langle AB, \{t_6\}\rangle$ *et* $\langle ABC, \{t_1, t_2, t_3, t_4, t_5\}\rangle$ *ne couvrent pas les mêmes transactions. Par contre,* $\langle AB, \{t_4, t_5, t_6\}\rangle$ *et* $\langle BC, \{t_2, t_3, t_4, t_5\}\rangle$ *(figure 1.4b) présentent des chevauchements sur l'items* B *et sur les transactions* t_4 *et* t_5.

	A	B	C
t_1	×	×	×
t_2	×	×	×
t_3	×	×	×
t_4	×	×	×
t_5	×	×	×
t_6	×	×	
t_7	×		
t_8		×	

(a) $\langle AB, \{t_6\}\rangle$ et $\langle ABC, \{t_1, t_2, t_3, t_4, t_5\}\rangle$

⇒

	A	B	C
t_1	×	×	×
t_2	×	×	×
t_3	×	×	×
t_4	×	×	×
t_5	×	×	×
t_6	×	×	
t_7	×		
t_8		×	

(b) $\langle AB, \{t_4, t_5, t_6\}\rangle$ et $\langle BC, \{t_2, t_3, t_4, t_5\}\rangle$

FIGURE 1.4 – Exemple de tuiles

Les méthodes de compression exploitant MDL permettent ainsi de décrire différentes tuiles des jeux de données. Les motifs extraits présentent alors peu (ou pas) de redondances étant donné qu'ils décrivent différentes parties du jeu de données. Leur recouvrement est donc faible. Dans la figure 1.5, nous illustrons une compression du jeu de données de la table 1.3a par quatre tuiles de la table de code 1.3b. Nous constatons alors que les motifs utilisés permettent de décrire le jeu de données sans recouvrement entre eux.

Cependant, en évaluant la similarité des motifs entre eux avec l'indice de Jaccard, nous constatons qu'il existe plus de redondance entre les motifs. L'indice de Jaccard $Jac(AB, ABC)$ des motifs AB et ABC vaut ainsi 0.83. De même, $Jac(A, B) = 0.75$.

La compression permet donc de réduire la redondance lorsqu'on considère comme langage des motifs les tuiles. En considérant d'autres formes de motifs, comme les motifs ensemblistes, la diversité est relativement moins importante. Dans la section 1.7.2, nous présenterons la méthode de compression KRIMP et discuterons son apport à la diversité.

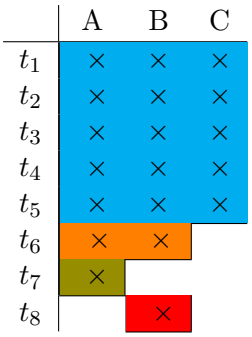


FIGURE 1.5 – Jeu de données compressé

1.6 Échantillonnage de motifs

L'échantillonnage de motifs [71] consiste à générer un sous-ensemble de motifs parmi ceux qui auraient été extraits par une méthode d'extraction de motifs sous contraintes. Il s'agit d'une méthode non exhaustive d'extraction de motifs assurant une bonne interaction avec l'espace des motifs et offrant des garanties statistiques fortes grâce à sa nature aléatoire.

L'échantillonnage de motifs est une technique de fouille de motifs permettant d'accéder aux motifs du langage $\mathcal{L}_\mathcal{I}$ par une procédure simulant une distribution $\pi : \mathcal{L}_\mathcal{I} \Rightarrow [0,1]$. Cette procédure est définie par rapport à une mesure d'intérêt $m : \pi(.) = \frac{m(.)}{Z}$, où Z est une constante de normalisation définie par : $Z = \sum_{X \in \mathcal{L}_\mathcal{I}} m(X, \mathcal{D})$.

Définition 29 (Échantillonnage de motifs)**.**
L'échantillonnage de motifs du langage $\mathcal{L}_\mathcal{I}$ selon une distribution proportionnelle à la mesure d'intérêt m dans un jeu de données $\mathcal{D}$ peut être formulé par l'opérateur suivant :

$$Echantillonner_k(\mathcal{L}_\mathcal{I}, \mathcal{D}, m) = \bigcup_{i=1}^{k} X_i \sim m(\mathcal{L}_\mathcal{I}, \mathcal{D}) \ [45]$$

avec k représentant le nombre d'échantillons de motifs à extraire et $X \sim m(\mathcal{L}_\mathcal{I}, \mathcal{D})$ signifiant que le motif X est tiré avec une probabilité proportionnelle à la mesure d'intérêt m :

$$X \sim m(\mathcal{L}_\mathcal{I}, \mathcal{D}) \iff \pi(X) = \frac{m(X,\mathcal{D})}{Z}$$

L'objectif principal des méthodes d'échantillonnage en sortie est de permettre l'extraction rapide d'échantillons de motifs représentatifs de l'ensemble des motifs qui peuvent être extraits du jeu de données. Si par exemple, un motif X_1 a un intérêt deux fois plus élevé que celui d'un motif X_2 selon le choix de l'utilisateur, alors X_1 a deux fois plus de chance d'être dans l'échantillon que le motif X_2.

Une méthode naïve d'échantillonnage de motifs consiste à énumérer en premier lieu l'ensemble de tous les motifs intéressants à l'aide des algorithmes d'extraction exhaustive de motifs tels que APRIORI [3] ou FP-GROWTH [66]. En second lieu, un échantillon de motifs est construit à partir de l'ensemble de motifs extraits tel que chaque motif soit tiré avec une probabilité proportionnelle à son intérêt dans le jeu de données.

Exemple 1.6.1.
En considérant le jeu de données de la table 1 et en considérant comme mesure d'intérêt la fréquence des motifs, l'échantillonnage de motifs ayant une fréquence minimale $\theta = 2$ s'écrit :

$$Echantillonner_k(\mathcal{T}h(\mathcal{L}_\mathcal{I}, \mathcal{D}, freq(X,\mathcal{D}) \geq 2), \mathcal{D}, freq)$$

avec un facteur de normalisation $Z = 67$. La probabilité de tirer le motif CF est égale à $\pi(CF) = \frac{3}{67} = 0.045$ (voir table 1.4). La probabilité de tirer un échantillon de trois motifs contenant AD,

E et CF est égale au produit des probabilité d'échantillonnage de ces trois motifs : $\pi(AD) \times \pi(E) \times \pi(CF) = \frac{3}{67} \times \frac{6}{67} \times \frac{3}{67} = 0.00018$.

Motifs	Probabilités	Motifs	Probabilités
A	0.060	AE	0.030
B	0.044	BD	0.045
C	0.075	DE	0.090
D	0.134	CF	0.045
E	0.105	BDE	0.030
CD	0.060	ADE	0.030
BE	0.030	CDF	0.030

TABLE 1.4 – Probabilités de tirage des motifs de la théorie $\mathcal{T}h(\mathcal{L}_{\mathcal{I}}, \mathcal{D}, freq_{\theta \geq 2})$

Cependant, cette méthode naïve est potentiellement très coûteuse notamment pour les grands jeux de données pour lesquelles le problème d'explosion des motifs complexifie l'étape d'extraction des motifs. Les méthodes d'échantillonnage développées ont alors consisté à tirer les différents motifs directement à partir du jeu de données. Pour cela différentes méthodes d'échantillonnage ont été proposées. Nous décrirons dans les sections suivantes trois approches d'échantillonnage de motifs qui ont fait l'objet d'une étude comparative dans [45].

1.6.1 Échantillonnage par marche aléatoire

Il s'agit d'une approche qui utilise des marches aléatoires dans l'espace de recherche pour échantillonner des motifs intéressants. L'idée est de simuler une distribution d'une loi de probabilité et de tirer les motifs proportionnellement à cette distribution. Cette approche permet ainsi d'éviter l'étape coûteuse de génération exhaustive des motifs avant leur tirage.

La plupart des méthodes d'échantillonnage par marche aléatoire utilisent des Chaînes de Markov par Monte Carlo (MCMC). Une chaîne d'états finis $v_0, v_1, ..., v_n$ est dite de Markov si la probabilité d'atteindre un état v_i ne dépend que de l'état v_{i-1} et pas des autres. L'idée derrière cette approche est alors de construire une chaîne de Markov simulant une distribution d'une loi de probabilité. Chaque état v_i simulant une distribution, l'échantillonnage consiste à poursuivre la simulation jusqu'à obtenir la distribution probabiliste correspondante à la mesure d'intérêt m. On pourra alors tirer des motifs proportionnellement à cette distribution. Différentes méthodes d'échantillonnage exploitant les chaînes de Markov ont été proposées :

- *Al Hasan et Zaki* [71] ont proposé une méthode pour échantillonner des sous-graphes fréquents dans un jeu de données de graphes.
- Boley *et al.* [23] ont développé un algorithme basé sur les MCMC pour échantillonner des motifs ensemblistes en prenant en compte toute mesure d'intérêt strictement positive.
- Bhuiyan *et al.* [20] ont proposé une méthode pour la découverte interactive de motifs.
- Bendimerad *et al.* [14] ont proposé une méthode utilisant une mesure d'intérêt subjectif afin d'échantillonner des tuiles.

Dans la section 1.7, nous présenterons la méthode d'échantillonnage de [14] (GIBBS).

1.6.2 Échantillonnage en deux étapes

Il s'agit d'une approche qui effectue l'échantillonnage grâce à deux étapes successives (voir figure 1.11) :

1. tirer une transaction du jeu de données proportionnellement au nombre de motifs qu'elle contient ;
2. tirer un motif de cette transaction proportionnellement à son intérêt.

Cette approche est adaptée aux mesures d'intérêt m pouvant s'exprimer sous forme d'une somme d'utilité de motifs. Une fonction d'utilité u permet d'assigner un poids $u(X,t)$ à chaque motif X d'une transaction t de manière à ce que les motifs les plus préférés par l'utilisateur reçoivent les poids plus élevés. Ainsi, l'intérêt de X dans le jeu de donné s'exprime alors comme suit :

$$m(X,\mathcal{D}) = \sum_{t\in\mathcal{D}\wedge X\subseteq t} u(X,t)$$

Boley *et al.* ont proposé dans [24] et [22] deux méthodes permettant d'échantillonner des motifs en une procédure à deux étapes. Ils montrent également que les motifs sont tirés proportionnellement à la mesure d'intérêt m de l'utilisateur. Dans la section 1.7, nous présenterons la méthode CFTP [22] et montrerons comment elle peut être utilisée pour la fouille de motifs diversifiés.

1.6.3 Échantillonnage par le formalisme SAT

L'échantillonnage de motifs peut également exploiter le formalisme SAT. Cette approche a été mise en œuvre par Dzyuba *et al.* dans [51]. Il s'agit d'utiliser des contraintes `XOR` aléatoires (voir le chapitre 2) afin de partitionner l'espace de recherche en des cellules plus réduites et de tirer un motif dans les différentes cellules proportionnellement à leur poids.

Dzyuba *et al.* ont proposé une méthode utilisant WeightGen [35] afin d'estimer le nombre de contraintes `XOR` à utiliser pour partitionner l'espace de recherche. Il suffit ensuite d'énumérer tous les motifs de la cellule et de tirer un motif proportionnellement à la mesure d'intérêt. Cette approche évite donc l'énumération naïve de tous les motifs de la théorie. Les auteurs ont montré dans [51] la capacité de leur méthode à échantillonner des motifs proportionnellement à une mesure d'intérêt quelconque. Nous donnerons plus de détails sur cette méthode dans la section 2.5.3.

Dans la section suivante, nous présenterons quelques méthodes d'extraction et d'échantillonnage de motifs pouvant être utilisées pour réduire la redondance entre les motifs. Ces méthodes sont exploitées dans le chapitre 5 dans le cadre de l'évaluation de notre méthode.

1.7 Approches de réduction de la redondance en fouille de motifs

Dans cette section, nous présenterons différentes méthodes d'extraction de motifs permettant de réduire la redondance entre les motifs selon les différentes mesures d'intérêt introduites précédemment. Nous exploiterons comme exemple d'illustration le jeu de données de la table 1.1 que nous reprenons dans la table 1.5.

1.7.1 Approche exhaustive : PatternsTeam

Knobbe et Ho ont proposé dans [93] une méthode d'extraction d'ensemble de motifs dans les jeux de données. Leur approche consiste alors à trouver le sous-ensemble de motifs de taille k qui optimise une mesure de qualité Φ.

Théorie. Pour trouver le meilleur sous-ensemble de motifs optimisant Φ, *Knobbe et Ho* proposent le framework PatternsTeam qui commence par extraire un ensemble de motifs $\mathcal{X}$ du langage en utilisant différentes mesures d'intérêt (comme la fréquence) $\{\phi_1,\ldots,\phi_l\}$ auxquelles sont associés différents seuils de valeurs $\{\sigma_1,\ldots,\sigma_l\}$. PatternsTeam sélectionne alors dans $\mathcal{X}$ le sous-ensemble de motifs $\mathbb{X}$ de taille k de la théorie suivante :

$\mathcal{T}$	Items					
t_1	A					
t_2	A	B		D		
t_3	A			D	E	
t_4	A			D	E	
t_5		B		D	E	
t_6		B		D	E	
t_7			C	D	E	
t_8			C	D	E	
t_9			C	D		F
t_{10}			C	D		F
t_{11}			C		E	F

TABLE 1.5 – Jeu de données d'illustration.

$$\mathcal{T}h(2^{\mathcal{L}_\mathcal{I}}, \mathcal{D}, \mathbb{C}) = \{\mathbb{X} \in 2^{\mathcal{L}_\mathcal{I}} \mid \underbrace{taille(\mathbb{X}) = k \ \wedge \ (\nexists \mathbb{X}' \in 2^{\mathcal{L}_\mathcal{I}} \ tq \ taille(\mathbb{X}') = k \wedge \Phi(\mathbb{X}') \geq \Phi(\mathbb{X}))}_{\mathbb{C}}\}$$

Quatre mesures de qualité Φ sont proposées par les auteurs de PATTERNSTEAM parmi lesquelles on retrouve l'entropie. Pour évaluer l'entropie, *Knobbe et Ho* commencent par considérer chaque motif X_i de $\mathbb{X} = \{X_1, \ldots, X_k\}$ comme un item. Ils associent ensuite à chaque ensemble $\mathbb{X}$ une projection binaire

$$S(\mathbb{X}, c) = \{\langle c_1 = \mathcal{R}(X_1, t), \ldots c_k = \mathcal{R}(X_k, t)\rangle, \forall t \in \mathcal{D}\}.$$

Puis, l'entropie de $\mathbb{X}$ est calculé en utilisant ces différentes projection binaires $c \in \{0,1\}^k$:

$$H(\mathcal{T}, \mathbb{X}) = -\sum_{c \in \{0,1\}^k} \frac{|freq(X_1 = c_1, \ldots, X_k = c_k)|}{|\mathcal{T}|} log_2(\frac{|freq(X_1 = c_1, \ldots, X_k = c_k)|}{|\mathcal{T}|}).$$

Discussion. L'utilisation de l'entropie dans PATTERNSTEAM permet d'obtenir une mesure de l'uniformité de la distribution des motifs. En effet, en transformant l'ensemble de motifs $\mathbb{X}$ en un motif où les X_i sont considérés comme des items, il est possible de connaître les différents chevauchements entre les motifs en calculant la projection binaire de $\mathbb{X}$. Ainsi, en trouvant l'ensemble de motifs qui optimise l'entropie, PATTERNSTEAM permet d'obtenir l'ensemble de motifs qui réduits le mieux la redondance. Toutefois, cette méthode est fortement handicapée par la complexité de l'évaluation des entropies des motifs. En effet, étant donné le nombre $M = |\mathcal{X}|$ de motifs fournis au départ, la recherche de sous-ensemble de motifs $\mathbb{X}$ de taille k optimisant l'entropie est très coûteuse et ne passe pas à l'échelle.

1.7.2 Approche par compression : Krimp

Comme nous l'avons présenté à la section 1.5.2, il est possible d'obtenir des motifs diversifiés en utilisant une approche par compression. En effet, la compression comme nous l'avons vu avec la représentation condensée permet d'obtenir un ensemble assez réduit de motifs qui décrivent le jeu de données et qui permettent de retrouver tous les autres motifs de la théorie.

Vreeken *et al.* ont proposé dans [138] une méthode exploitant le principe de MDL [115] et qui permet d'extraire des ensembles de motifs qui compressent le jeu de données. Cette méthode prend en entrée un jeu de données et un ensemble de motifs préalablement extrait (les motifs

Algorithm 1: KRIMP

1 **Entrées** : $\mathcal{D}$: jeu de données, $\mathcal{F}$: ensemble de motifs candidats ;
2 **Sortie** : CT : table de code ;
3 **Début**
4 $CT \leftarrow$ **Table de code Standard** ;
5 $\mathcal{F}_o \leftarrow \mathcal{F}$ **ordonné selon l'ordre de Sélection de Krimp** ;
6 **Pour chaque** $F \in \mathcal{F}_o \backslash \mathcal{I}$ **faire**
7 $CT_c \leftarrow CT \cup F$;
8 **Si** $L(CT_c, \mathcal{D}) < L(CT, \mathcal{D})$ **alors**
9 $CT \leftarrow CT_c$;
10 **retourne** CT;

fréquents ou fermés par exemple) et essaie de trouver le sous-ensemble de motifs qui compresse le mieux le jeu de données.

L'algorithme 1 illustre la méthode de compression de KRIMP. L'algorithme commence par initialiser la table des codes avec les motifs singletons (contenant un seul item) (voir ligne 4). Puis, les motifs candidats $\mathcal{F}$ sont ordonnés par KRIMP (ligne 5) afin de pouvoir déterminer l'ordre de sélection de ceux-ci.

KRIMP parcours ainsi les différents motifs de $\mathcal{F}_o$ (ligne 6) et les sélectionne selon l'ordre fixé à la ligne 5 (ligne 7). Les tailles de description induites par les motifs des ensembles $CT \cup F$ et CT sont alors comparées (ligne 8) et le nouvel ensemble $CT \cup F$ est retenu s'il améliore la compression (ligne 9). À la fin de l'algorithme, KRIMP retourne alors la table des codes finale (ligne 10). Cette table des codes comporte l'ensemble de motifs $\mathbb{X}$ de la théorie suivante :

$$\mathcal{T}h(2^{\mathcal{L}_\mathcal{I}}, \mathcal{D}, \mathbb{C}) = (\forall X \ \in \mathbb{X},\ code_{\mathbb{X}}(X) \text{ est optimal}) \wedge (\forall \mathbb{X}' \in 2^{\mathcal{L}_\mathcal{I}}, L(\mathbb{X}', \mathcal{D}) < L(\mathbb{X}, \mathcal{D}))$$

Exemple 1.7.1.
En utilisant le jeu de données de la table 1.5 ainsi qu'un support $\theta = 2$, KRIMP nous retourne la table de code de la table 1.6. Cette table des codes représente les motifs utilisés par KRIMP pour la compression, avec leur code couleur et les transactions dans lesquelles ils sont utilisés.

Motifs	Transactions associées	Code couleur
D	$\{t_2, t_3, t_4, t_5, t_6, t_9, t_{10}\}$	
E	$\{t_3, t_4, t_5, t_6, t_{11}\}$	
A	$\{t_1, t_2, t_3, t_4\}$	
B	$\{t_2, t_5, t_6\}$	
CF	$\{t_9, t_{10}, t_{11}\}$	
CDE	$\{t_7, t_8\}$	

FIGURE 1.6 – Table de codes de KRIMP

L'utilisation des motifs de la figure 1.6 permet d'obtenir le jeu de données compressé de la figure 1.7.

Discussion. La compression proposée par les auteurs de KRIMP permet d'obtenir une diversité implicite entre les motifs de l'ensemble solution. En effet, pour la description de chaque transaction t, KRIMP utilise un sous-ensemble $S = cover(\mathbb{X}, t)$ de motifs tel que :

$$\bigcup_{X \in S} X = t \ \wedge \ \forall X, Y \in S, \text{ soit on a } X = Y, \text{soit } X \cap Y = \emptyset$$

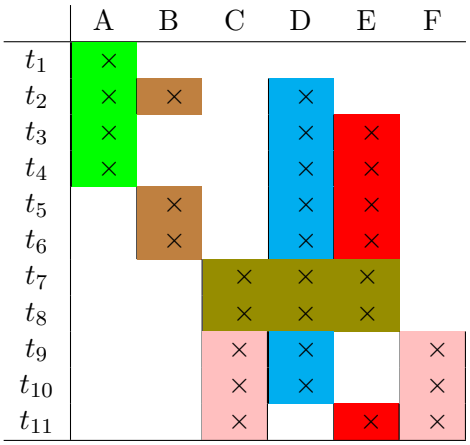


FIGURE 1.7 – Jeu de données compressé

Cette démarche permet de s'assurer que deux motifs similaires ne seront pas utilisés pour la compression car leurs contributions seraient potentiellement redondantes. Par contre, deux motifs présentant plus de différence pourraient être utilisées étant donné que leurs contribution à la compression seraient complémentaires. Cependant, il n'est pas très évident d'obtenir une faible redondance entre différentes paires de motifs extraits par KRIMP étant donné que l'objectif des auteurs n'est pas explicitement la diversité. Par ailleurs, et comme conséquence, il n'est pas possible de contrôler la redondance entre les motifs de KRIMP.

1.7.3 Approche heuristique : Picker

Le processus d'extraction de l'ensemble de motifs avec PICKER se fait en deux phases :

1. extraire des motifs sous contraintes locales
2. utiliser ces motifs pour construire l'ensemble de motifs $\mathbb{X}$.

Théorie

Les partitions exploitées par PICKER pour construire les ensembles de motifs peuvent être définies comme suit :

Définition 30 (Partition)**.**
Une partition d'un jeu de données $\mathcal{D}$ est un ensemble $\mathcal{A} = \{D_1, ..., D_a\}$ tel que $\forall D_i \in \mathcal{A}, D_i \subset \mathcal{D}$. Chaque élément de $\mathcal{A}$ est une cellule de la partition, les différentes cellules étant deux à deux disjointes et leur union égale à $\mathcal{D}$:

$$\forall D_i, D_j \in \mathcal{A}, D_i \neq D_j : D_i \cap D_j = \emptyset \wedge \bigcup_{D_i \in \mathcal{A}} D_i = \mathcal{D}$$

Pour former ces partitions, Zimmermann *et al.* [29] définissent des relations d'équivalence entre les couvertures des motifs. Ces relations d'équivalence vont alors permettre de former des sous-ensembles T_i sur l'ensemble des transactions $\mathcal{T}$.

Définition 31 (Relation d'équivalence dans les ensembles de motifs)**.**
Étant donné un ensemble $\mathbb{X}$ de motifs, la relation d'équivalence $\sim_S$ sur l'ensemble des transactions $\mathcal{T}$ se définit comme suit :

$$\sim_{\mathbb{X}} = \{(t_1, t_2) \in \mathcal{T} \times \mathcal{T} | \forall P \in \mathbb{X}, P(t_1) = P(t_2)\}$$

avec $P(t) = 1$ si $t \in \mathcal{V}_{\mathcal{D}}(P)$ et 0 sinon

Deux transactions $t_1, t_2 \in \mathcal{T}$ sont alors dites équivalentes sous $\mathbb{X}$ si et seulement si elles contiennent exactement les mêmes motifs. Cette relation d'équivalence nous permet ainsi de définir une partition de $\mathcal{T}$ sur $\mathbb{X}$:

	t_1	t_3	t_4	t_5
P_1	x	x	x	
P_2	x		x	x
P_3	x		x	
P_4		x		

(a) Couverture et occurrence des motifs

P_1	$\{t_1\ t_4$	t_3	t_5	$t_2\}$
P_2	$\{t_1\ t_4$	$t_3\}$	$\{t_5$	$t_2\}$
P_3	$\{t_1\ t_4\}$	$\{t_3\}$	$\{t_5\}$	$\{t_2\}$
P_4	$\{t_1\ t_4\}$	$\{t_3\}$	$\{t_5\}$	$\{t_2\}$
	$\{t_1\ t_4\}$	$\{t_3\}$	$\{t_5\}$	$\{t_2\}$

(b) Partitions induites par l'ensemble $\mathbb{X} = \{P_1, P_2, P_3, P_4\}$

FIGURE 1.8 – Partitions induites par des motifs. Quatre motifs binaires peuvent induire au plus 16 blocs

$$\mathcal{T}/\sim_{\mathbb{X}} = \bigcup_{t \in \mathcal{T}} t' \in \mathcal{T} \mid t' \sim_{\mathbb{X}} t$$

Les ensembles de motifs étant construits de façon itérative, les différents motifs P_i vont progressivement induire des partitions sur $\mathcal{T}$ en subdivisant les blocs D_{i-1} (définition 30) en un maximum de deux sous-blocs $D_i = \{D_{i_1}, D_{i_2}\}$:

— D_{i_1} qui regroupe un sous-ensemble de transactions couvertes par le motif

— D_{i_2} qui regroupe un sous-ensemble de transactions non couvertes par le motif

Exemple 1.7.2.
Dans la figure 1.8, nous avons en 1.8a une base transactionnelle représentant les couverture de chaque motif P_i. Ainsi : $\mathcal{V}_{\mathcal{D}}(P_1) = \{t_1, t_3, t_4\}$ et $\mathcal{V}_{\mathcal{D}}(P_4) = \{t_3\}$. Dans la figure 1.8b, un arbre des différentes partitions induites par l'ajout de chaque motif P_i à $\mathbb{X}$ est construit. On constate ainsi que P_1 induit une partition avec deux blocs $D_1 = \{D_{1_1}, D_{1_2}\}$, avec $D_{1_1} = \{t_1, t_3, t_4\}$ et $D_{1_2} = \{t_2, t_5\}$. L'ajout de P_2 à $\mathbb{X}$ va induire une nouvelle partition avec quatre blocs. Par contre, l'ajout de P_3 et P_4 ne créera aucun nouveau bloc. Ces deux motifs ne décrivent donc aucune partition supplémentaire du jeu de données et ne sont pas nécessaires pour décrire tous le jeu de données.

L'ajout de nouveaux motifs dans l'ensemble $\mathbb{X}$ n'est alors possible que si les motifs candidats P_i induisent de nouvelles partitions sur $\mathcal{T}$. En effet, lorsque l'ajout d'un motif ne modifie pas $\mathcal{T}/\mathbb{X}$, cela signifie que le nouveau motif est dans la même classe d'équivalence qu'un motif ajouté précédemment. Cela nous permet d'exprimer la contrainte de redondance suivante :

$$C(\mathbb{X}) \text{ est vrai } \equiv \nexists P \in \mathbb{X} : |\mathcal{T}/\sim_{\mathbb{X}}| = |\mathcal{T}/\sim_{\mathbb{X}-\{P\}}|$$

Le rejet du motif candidat permet ainsi de garantir une diversité implicite dans l'ensemble de motifs $\mathbb{X}$. Par ailleurs, l'ajout de chaque motif est conditionné par la satisfaction d'un mesure de qualité Φ qui permet de sélectionner les motifs afin d'obtenir le meilleur ensemble possible.

Exemple 1.7.3.
En exécutant PICKER sur le jeu de données 1.5, on obtient l'ensemble constitué des motifs suivants : $X_1 = C, X_2 = DE, X_3 = B, X_4 = E$. Ces motifs sont représentés par la figure 1.9

Discussion. L'approche proposée par Zimmermann *et al.* permet de construire des ensembles de motifs permettant à la fois de décrire le jeu de données et de minimiser le nombre de motifs utilisés. PICKER n'exprime donc pas de façon explicite une contrainte de diversité entre les motifs de l'ensemble. Cependant, l'exploitation qu'il fait des classes d'équivalence permet de réduire la redondance entre les motifs étant donné que deux motifs de la même classe d'équivalence ne peuvent être ajoutés à l'ensemble solution $\mathbb{X}$. Cette approche présente donc un intérêt comme méthode de réduction de la redondance. Cependant, elle possède quelques limites. En effet, il s'agit d'une méthode de post-traitement qui nécessite une phase d'extraction de motifs potentiellement coûteuse. Par ailleurs, elle utilise une matrice pour représenter l'absence ou la présence des motifs dans les différentes transactions, ce qui rajoute un coût à l'extraction de l'ensemble

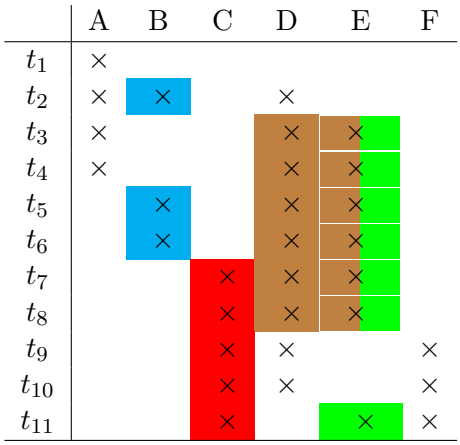


FIGURE 1.9 – Motifs obtenu avec PICKER.

solution et rend difficile son passage à l'échelle. Enfin, il n'est pas possible de contrôler le niveau de redondance entre les motifs. En effet, il n'est pas possible d'ajouter un seuil de redondance maximum entre les motifs de l'ensemble.

Nous proposons à la section suivante une méthode itérative d'extraction des motifs qui essaie de surmonter ces différentes limites.

1.7.4 Approche d'échantillonnage par marche aléatoire : Gibbs

Les méthodes d'échantillonnage par marche aléatoire exploitent très souvent les Chaînes de Markov par Monte Carlo [71, 23, 20, 14]. L'objectif de cette classe de méthodes est de simuler une distribution d'une loi de probabilité et de tirer des motifs proportionnellement à la distribution simulée. Bendimerad *et al.* ont proposé une méthode (que nous noterons GIBBS) [14] permettant d'échantillonner des motifs X proportionnellement à une mesure d'intérêt subjective SI :

$$SI(X) = \frac{IC(X)}{L(X)}$$

Les motifs échantillonnés par GIBBS sont des tuiles [59] qui peuvent être représentées par un ensemble d'items I et par un ensemble de transactions T couvrant I : $X = (I, T)$. SI a été introduite dans [21] et permet de mesurer la qualité d'une tuile en terme de quantité d'information IC et de longueur de la description. La quantité d'information est calculée en évaluant la probabilité de tirer une tuile $X = (I, T)$ dans le jeu de données :

$$IC(X) = -log(P(X \in \mathcal{D})) = \sum_{i \in I, t \in T} -log(p_{i,t})$$

Avec cette formulation, les tuiles les moins fréquentes ne seront pas défavorisées. En effet, même avec une valeur de fréquence faible, on obtient pour ces motifs une valeur IC positive et relativement élevée. De ce fait, ces tuiles, présentées par Bendimerad *et al.* comme des tuiles « surprenantes » et potentiellement intéressantes pour un utilisateur donné, ne sont pas pénalisées par leurs fréquences. Ainsi, plus la probabilité de tirer une tuile est petite, plus son IC est grand.

La mesure L évalue la difficulté pour un utilisateur d'assimiler un motif (i.e., une tuile). En effet, si la tuile comporte beaucoup d'items et de transactions, elle peut être potentiellement plus difficile à mémoriser. Cette mesure est évaluée comme suit :

$$L(X) = a + b(|I| + |T|)$$

où a et b sont deux constantes permettant de mesurer l'importance de la contribution de $|I|$ et $|T|$.

Les tuiles échantillonnées sont alors celles qui optimisent la quantité d'information IC et la longueur de la description L. Pour échantillonner les différentes tuiles, Bendimerad *et al.* proposent alors de simuler la mesure SI par une procédure à p itérations. Chaque itération k comporte deux étapes :

— tirer un ensemble d'items $I^{(k)}$ proportionnellement à l'ensemble de transactions $T^{(k-1)}$:

$$I^{(k)} \sim P(\mathbf{I} = I \mid \mathbf{T} = T^{(k-1)})$$

— tirer un ensemble de transactions T^k proportionnellement à l'ensemble I^k :

$$T^{(k)} \sim P(\mathbf{T} = T \mid \mathbf{I} = I^{(k)})$$

Les auteurs de [14] garantissent que la tuile $X = (I, T)^{(k)}$ obtenue est tirée proportionnellement à la mesure SI.

Exemple 1.7.4.
En exécutant GIBBS *sur le jeu de données 1.5 et en fixant le nombre k de motifs à* 4 *et le nombre d'itération à* 1000*, on obtient les motifs suivants de la figure 1.10 :* $X_1 = \langle DE, \{t_3, t_5, t_7\}\rangle$ *(en marron),* $X_2 = \langle D, \{t_5, t_9\}\rangle$ *(en rose),* $X_3 = \langle E, \{t_3, t_8, t_{11}\}\rangle$ *(en rouge) et* $X_4 = \langle AD, \{t_3\}\rangle$ *(en vert).*

Motifs	Couleur
$X_1 = \langle DE, \{t_3, t_5, t_7\}\rangle$	
$X_2 = \langle D, \{t_5, t_9\}\rangle$	
$X_3 = \langle E, \{t_3, t_8, t_{11}\}\rangle$	
$X_4 = \langle AD, \{t_3\}\rangle$	

(a) Couleurs associés aux motifs

	A	B	C	D	E	F
t_1	×					
t_2	×	×		×		
t_3	×			×	×	
t_4	×			×	×	
t_5		×		×	×	
t_6		×		×	×	
t_7			×	×	×	
t_8			×	×	×	
t_9			×	×		×
t_{10}			×	×		×
t_{11}			×		×	×

(b) Représentation des motifs dans le jeu de données.

FIGURE 1.10 – Motifs obtenus avec GIBBS à 1000 itérations avec $k = 4$

Discussion. GIBBS [14] exploite une mesure d'intérêt subjective [21] qui est mise à jour à chaque découverte de motifs afin de garantir que chaque nouveau motif extrait décrive de nouvelle information comparativement aux précédents motifs. Ce procédé est sémantiquement similaire à celui utilisé par CLOSEDDIVERSITY qui maintient un historique de solutions déjà extraits afin de garantir que les nouveaux motifs soient diversifiés par rapport à chaque motif de l'historique. Cette mesure d'intérêt est exploitée par la méthode GIBBS afin de réduire les temps d'exécution.

1.7.5 Approche d'échantillonnage en deux étapes : CFTP

Comme nous l'avons présenté à la section 1.6.2, l'échantillonnage en deux étapes introduit par Boley *et al.* [24] consiste en des tirages à deux étapes :

1. la 1^{re} étape consiste à tirer un tuple γ du jeu de données proportionnellement à son poids (la somme des intérêts des motifs qui le composent). Cette étape correspond à la ligne 4 de l'algorithme 2 ;
2. la 2^{e} étape consiste à tirer un motif du tuple proportionnellement à son intérêt dans le tuple (ligne 6 de l'algorithme 2).

Algorithm 2: Échantillonnage en deux étapes avec CFTP

Entrées : $\mathcal{D}$: jeu de données, m : mesure d'intérêt
Sortie : un motif tiré aléatoirement $X \sim m(\mathcal{L}, \mathcal{D}) : w(X) = \frac{m(X,\mathcal{D})}{Z}$;
Début
  ▷ **Tirage d'un tuple γ proportionnellement à son poids**
  $\gamma \sim w_m(\mathcal{D})$
  ▷ **Tirage d'un motif X proportionnellement à son utilité dans γ**
  $X \sim u(X : X \in \gamma)$
  retourne X;

Pour réaliser l'étape 1, il est nécessaire de calculer le poids de chaque tuple γ. En effet, les tuples sont tirés proportionnellement à leur poids qui est égal à la somme des utilités des motifs qui les composent. En considérant le cas où chaque motif a la même utilité et que celle-ci vaut 1, alors, le poids d'un tuple est égal au nombre de motifs qu'il comporte. Le tirage des tuples aura ainsi tendance à favoriser ceux qui comportent le plus d'items. Par ailleurs, le choix de la fréquence comme mesure d'intérêt des motifs aura pour effet de favoriser les motifs les plus fréquents. Or, la fréquence du motif peut ne pas être une bonne mesure de l'intérêt de l'utilisateur. À contrario, l'utilisateur peut être intéressé par des motifs ayant un support peu élevé. Par ailleurs, il peut exister beaucoup de redondance dans la couverture des motifs ayant un grand support.

Boley *et al.* introduisent alors dans [22] la notion de **facteur de fréquence**. Cette nouvelle approche leur permet de sélectionner à l'étape 1 un sous-ensemble de c transactions du jeu de données ($c \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$) puis de former les motifs de l'étape 2 à partir des items communs aux c transactions. Les facteurs de fréquence permet alors de considérer la fréquence comme une mesure :

— positive : dans ce cas, les motifs ayant un grand support seront privilégiés ;
— négative : dans ce cas, ce sont les motifs ayant un petit support qui seront privilégiés.

Ils proposent ainsi d'échantillonner des motifs qui combinent ces deux facteurs de fréquence $q \in \{supp, \overline{supp}\}$. Pour cela, chaque tuple sera échantillonné à partir d'un sous-ensemble de c transactions. Un facteur de fréquence est alors associé à chaque transaction. L'échantillonnage se fait selon la distribution $\mathcal{F} : \mathcal{L}(\mathcal{D}) \to [0, 1]$ avec

$$\mathcal{F}(X) = w_\star(X) \prod_{i=1}^{c} \frac{q_i(\mathcal{D}_i, X)}{Z}$$

, où $q_i \in \{supp, \overline{supp}\}$, $w_\star(X) = \bigstar_{x \in X} w(x)$ et $\bigstar \in \{\prod, \sum\}$ ($\prod$ (respectivement $\sum$) si le motif est tiré proportionnellement à la fréquence (respectivement l'aire)).

Exemple 1.7.5.
En exécutant CFTP sur le jeu de données 1.5 et en fixant le nombre k de motifs à tirer à 4, le facteur de fréquence c à 2, on obtient les motifs de la figure 1.12 tirés selon la distribution de fréquence : $X_1 = CDE$ (en rose), $X_2 = CD$ (en vert), $X_3 = CF$ (en jaune) et $X_4 = ABD$ (en bleu). Pour tirer le motif $X_3 = CF$, CFTP commence par tirer la paire de transactions $\{t_9, t_{10}\}$ qui comporte les items communs C, D et F. Le poids de cette paires de motifs est égale au nombre de motifs pouvant être formés avec les items communs aux deux transactions, soit 7. CFTP tire alors un motif $X \in \{C, D, F, CD, CF, DF, CDF\}$ et l'ajoute à l'ensemble $\mathbb{X}$ des motifs à échantillonner.

Discussion. Cette procédure permet d'introduire de la diversité dans les motifs échantillonnés. En effet, en privilégiant les motifs avec une grande fréquence, on augmente les possibilités de redondance. Cependant, l'utilisation des facteurs de fréquence (positif et négatif) permet de

$\mathcal{T}$	Items					
t_1	A					
t_2	A	B		D		
t_3	A			D	E	
t_4	A			D	E	
t_5		B		D	E	
t_6		B		D	E	
t_7			C	D	E	
t_8			C	D	E	
t_9			C	D		F
t_{10}			C	D		F
t_{11}			C		E	F

(a) Jeu de données

→

Paires	Motifs possibles	Poids
$\{t_1, t_2\}$, $\{t_1, t_3\}$ $\{t_1, t_4\}$	A	1
$\{t_2, t_3\}$, $\{t_2, t_4\}$	A, D, AD	3
$\{t_3, t_4\}$	A, D, E, AD, AE, DE, ADE	7
⋮	⋮	⋮
$\{t_9, t_{10}\}$	C, D, F, CD, CF, DF, CDF	3
$\{t_9, t_{11}\}$	C, F, CF	3
$\{t_10, t_{11}\}$	C, F, CF	3

(b) Tirage d'une paire de transactions (c=2) proportionnellement à son poids

→

Motifs candidats
C
D
F
CD
CF
DF
CDF

(c) Tirage du motif CF

FIGURE 1.11 – Étapes d'échantillonnage du motif CF par CFTP

Motifs	Couleur
$X_1 = ABD$	
$X_2 = CF$	
$X_3 = CDE$	
$X_4 = CD$	

(a) Couleurs associés aux motifs

	A	B	C	D	E	F
t_1	×					
t_2	×	×		×		
t_3	×			×	×	
t_4	×			×	×	
t_5		×		×	×	
t_6		×		×	×	
t_7			×	×	×	
t_8			×	×	×	
t_9			×	×		×
t_{10}			×	×		×
t_{11}			×		×	×

(b) Représentation des motifs dans le jeu de données.

FIGURE 1.12 – Motifs obtenus avec CFTP avec 2 facteurs de fréquence et $k = 4$

mettre en avant des motifs ayant une répartition plus clairsemée dans le jeu de données, ce qui permet de réduire la redondance. Toutefois, les méthodes proposées par Boley *et al.* [24, 22] ne permettent pas l'ajout de contraintes supplémentaires sur les motifs. Cette limite a pour conséquence de favoriser les motifs très peu fréquents (et donc très diversifiés) lorsque la méthode est utilisée avec des facteurs de fréquence très bas ($c \in \{1, 2, 3, 4\}$). Pour augmenter la probabilité de tirer des motifs plus fréquents, il est nécessaire d'utiliser des facteurs de fréquence plus grand ($c \geq 5$), ce qui pénalise les performance de CFTP.

1.7.6 Synthèse

Les approches présentées précédemment abordent la diversité de différentes manières. KRIMP et GIBBS introduisent la diversité dans leurs motifs en raisonnant sur les tuiles. En effet, dans KRIMP, chaque motif est associé à un sous-ensemble de transactions et non à toutes les transactions de sa couverture. Par ailleurs, les motifs de GIBBS sont tirés proportionnellement à une mesure d'intérêt qui s'exprime sur un sous-ensemble d'items et de transactions formant une tuile. Ces deux méthodes peuvent ainsi minimiser les chevauchements entre les différentes tuiles grâce à la taille relativement réduites des transactions. Cependant, en considérant uniquement les items de chaque tuile et en leur associant leur couverture dans le jeu de données, on observe que la redondance est plus importante.

De la même manière, PICKER exploite les partitions induites par les motifs pour construire ses ensembles. CFTP lui utilise un processus de Markov pour tirer ses motifs. Cependant, comme nous le verrons dans le chapitre 5, PICKER ne passe pas toujours à l'échelle et CFTP souffre du problème de la longue traîne qui favorise les motifs longs (avec beaucoup d'items) et peu fréquents notamment lorsqu'on le paramètre avec des facteurs de fréquences bas ($c \leq 5$). Ce phénomène permet ainsi à CFTP d'avoir les meilleures résultats en terme de diversité.

Pour illustrer ce comportement, nous avons calculé les indices de Jaccard de toutes les paires de motifs extraits par les quatre méthodes Picker, Krimp, Gibbs et CFTP (voir les exemples précédents) avec le jeu de données de la table 1.5. Nous avons ensuite reportés dans la table 1.6 les valeurs minimales, maximales et moyennes de ces indices de Jaccard.

Méthodes	MIN	MAX	Moyenne π	Écart-type σ
Picker	0.0	0.86	0.32	0.26
Krimp	0.0	0.5	0.16	0.18
Gibbs	0.0.25	0.86	0.5	0.21
CFTP	0.0	0.5	0.15	0.21

Table 1.6 – Jaccards de toutes les paires de motifs de Picker, Krimp, Gibbs et CFTP.

On remarque ainsi que pour le même jeu de données (table 1.5) Krimp et CFTP obtiennent les meilleurs résultats en terme de diversité. Cependant, pour chacune de ses méthodes, il n'est pas possible de contrôler la diversité de façon explicite avec une mesure et un seuil. Au contraire, la diversité est obtenue grâce des heuristiques et est gérée de façon implicite.

Nous proposerons alors au chapitre 4 une méthode permettant d'approximer le problème MaxDiverseKSet en exploitant le modèle de MostDistant. Cette méthode, ClosedDiversity, utilisera un historique des motifs extraits, l'indice de Jaccard (pour mesurer la redondance) et un seuil afin d'extraire des ensembles de motifs diversifiés respectivement au seuil fixé.

1.8 Conclusion

Dans ce chapitre, nous avons présenté les problématiques de l'extraction de motifs sous contraintes locales et de l'extraction d'ensembles de motifs. La fouille de motifs est l'un des domaines les plus actifs de la fouille de données et différents travaux ont ainsi proposé des mesures permettant d'obtenir des motifs intéressants pour l'utilisateur.

Cependant, le problème de la redondance dans les motifs extraits a continué de se poser. Étant donné que la recherche de l'ensemble de motifs minimisant le mieux la redondance est NP-complète, plusieurs méthodes approchées ont permis d'approximer le problème en proposant un ensemble de motifs moins redondant que l'ensemble des motifs fréquents. Ces méthodes ont proposé différentes approches parmi lesquelles la compression [138] et les approches par classe d'équivalence [29]. On retrouve également les approches par échantillonnage de motifs permettant d'augmenter les performances des méthodes d'extraction en proposant des techniques permettant de tirer rapidement des motifs de bonne qualité. Dzyuba *et al.* (Flexics, [51]) ainsi que Boley *et al.* (CFTP, [22]) et Bendimerad *et al.* (Gibbs, [14]) ont ainsi proposé des méthodes d'échantillonnage performantes exploitant les contraintes XOR et la simulation markovienne. Toutefois, aucune de ces méthodes n'exploite de mesures permettant de quantifier la redondance. La diversité qu'elles intègrent dans les ensembles de motifs est alors implicite car il n'est pas possible de la personnaliser en définissant des seuils de redondances.

Nous proposons donc dans le chapitre 4 d'exploiter l'indice de Jaccard afin de mesurer le degré de redondance dans les motifs. Nous exploiterons alors cette mesure afin de filtrer les motifs non diversifiés.

CHAPITRE 2

Programmation par contraintes

Contents

2.1 Modélisation 41
2.1.1 Formalisme CSP 42
2.1.2 Résolution d'un CSP 43
2.2 Consistance et filtrage 43
2.2.1 Consistance 44
2.2.2 Filtrage de domaines 45
2.2.3 Méthodes de Filtrage 45
2.2.4 Propagation de contraintes 46
2.3 Stratégies de recherche 46
2.3.1 Backtrack et maintient d'arc-cohérence 47
2.3.2 Ordre de choix des variables et des valeurs 47
2.3.3 Algorithme de résolution 48
2.4 Modélisation des contraintes 48
2.4.1 Contraintes Réifiées 49
2.4.2 Contraintes Globales 49
2.5 Modèles PPC pour l'extraction de motifs 50
2.5.1 Modèle Réifié 51
2.5.2 Modèle par contrainte globale : ClosedPatterns 52
2.5.3 Échantillonnage de motifs et contraintes `XOR` 54
2.6 Conclusion 55

Dans ce chapitre, nous introduirons la programmation par contraintes et les différentes notions que nous utiliserons dans la suite de ce manuscrit. Pour des informations plus approfondies, il est possible de se référer au livre de Rossi *et al.* [116].

Nous introduirons ainsi le formalisme CSP, les notions de filtrage et de cohérence de domaine. Nous présenterons ensuite le schéma de résolution des CSPs et quelques types de contraintes. Puis, nous détaillerons deux approches PPC pour l'extraction de motifs.

2.1 Modélisation

La programmation par contraintes (PPC) est un formalisme permettant de modéliser et résoudre des problèmes combinatoires [116]. Elle offre un cadre déclaratif pour modéliser et résoudre des problèmes complexes de l'intelligence artificielle, de la logique et de la recherche

opérationnelle par un ensemble de relations logiques entre les variables, les contraintes, décrivant des propriétés sur les solutions à satisfaire. L'utilisateur a alors la possibilité d'exprimer ses besoins de manière déclarative en fonction des solutions qu'il souhaite obtenir.

2.1.1 Formalisme CSP

Pour la résolution des problèmes, la PPC exploite un formalisme appelé problème de satisfaction de contraintes ou CSP (*Constraint Satisfaction Problem*). Ce formalisme qui étend le modèle *SAT* utilise des techniques de la programmation linéaire, de la théorie des graphes et de la recherche opérationnelle. Il a la particularité de séparer l'étape de résolution des problèmes de l'étape de modélisation, permettant ainsi la possibilité de développer des méthodes de résolution génériques.

Définition 32 (CSP)**.**
Un CSP ou réseau de contraintes $\mathcal{P}$ est défini par un triplet $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ tel que :

- *$\mathcal{X}$ est un ensemble fini de n variables $\mathcal{X} = \langle X_1, ..., X_n \rangle$;*
- *$\mathcal{D} = \langle \mathcal{D}_1, ..., \mathcal{D}_n \rangle$ est un n-uplet des domaines finis des variables, $\mathcal{D}_i$ étant l'ensemble contenant les valeurs de X_i ;*
- *$\mathcal{C} = \langle C_1, .., C_m \rangle$ est l'ensemble des contraintes. Chaque contrainte C_i porte sur un sous-ensemble de variables $var(C_i)$ de $\mathcal{X}$. Ce sous-ensemble est appelé la portée de la contrainte. L'arité d'une contrainte C_i désigne le nombre de variables sur lesquelles C_i est définie c'est-à-dire $|var(C_i)|$.*

Exemple 2.1.1.
Comme illustration, nous pouvons considérer le CSP $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ défini par :
- $\mathcal{X} = \{X_1, X_2, X_3\}$
- $\mathcal{D} = \{D_1, D_2, D_3\}$, avec $D_1 = \{1, 2\}$, $D_2 = \{0, 1, 2, 3, 4\}$, $D_3 = \{2, 3\}$
- $\mathcal{C} = \{C_1, C_2\}$ avec $C_1 : X_1 < X_2$ et $C_2 : X1 \leq X2 - X3$

La résolution d'un CSP consiste alors à trouver l'ensemble des tuples $A = \langle a_1, ..a_n \rangle \in \langle D_1, ..., D_n \rangle$ tels que $\forall i \in \overline{1...m}$, C_i est vraie. Pour cela, il faut instancier chaque variable X_i avec une valeur de son domaine D_i.

Définition 33 (Instanciation)**.**
Une instanciation $A = \langle a_1, ..a_k \rangle$ est un ensemble d'affectations de valeurs a_i à un ensemble de variables $\mathcal{Y} = \{X_1, \dots X_k\}$ telle que $a_i \in D_i$. Lorsque l'instanciation porte sur toutes les variables X_i de $\mathcal{X}$, c'est-à-dire $\mathcal{Y} = \mathcal{X}$, alors elle est dite complète. Dans le cas contraire (ie $\mathcal{Y} \subset \mathcal{X}$), elle est dite partielle.

Nous désignerons par $A[X_{i_1}, \dots X_{i_k}]$ la projection de l'instanciation A sur les variables $X_{i_1}, \dots X_{i_k}$ c'est à dire $\langle a_1, ..a_k \rangle$.

Exemple 2.1.2.
Considérons le CSP de l'exemple 2.1.1. Comme instanciation, nous pouvons avoir :
- $A_1 = \langle X_2 = 4, X_3 = 3 \rangle$ qui est une instanciation partielle ;
- $A_2 = \langle X_1 = 2, X_2 = 0, X_3 = 3 \rangle$ qui est une instanciation totale.

L'affectation de valeurs aux variables peut-être une tâche fastidieuse dans le cas où le CSP comporte un grand nombre de variables ayant chacune des domaines de grandes taille. En effet, pour réaliser une instanciation complète des n variables X_i, en supposant que chaque variable a un domaine contenant d valeurs, il faudrait choisir l'une des d^n instanciations. Or, toutes ces instanciations ne sont pas nécessairement réalisables étant donné les valeurs à affecter à chaque variable. La faisabilité d'une instanciation est alors vérifiée par des relations entre les valeurs que peuvent prendre les variables du problèmes.

Définition 34 (Contraintes)**.**
Une contrainte est définie par une relation R_C qui détermine les affectations autorisées pour un sous-ensemble de variables $X_C \subseteq X$: $R_C \subseteq \prod_{X_i \in X_C} D_i$

Exemple 2.1.3.
Considérons le CSP de l'exemple 2.1.1. Les tuples de valeurs respectant la contrainte C_1 sont : $(1,2,2)$, $(1,2,3)$, $(1,3,2)$, $(1,3,3)$, $(1,4,2)$, $(1,4,3)$, $(2,3,2)$, $(2,3,3)$, $(2,4,2)$, $(2,4,3)$, $(1,2,2)$. *Ainsi, on constate que l'instanciation complète $A_2 = \langle X_1 = 2, X_2 = 0, X_3 = 3\rangle$ donnée précédemment ne respecte pas la contrainte C_1 car on a $X_1 > X_2$.*

Lorsqu'une instanciation complète satisfait toutes les contraintes de $\mathcal{C}$, alors, elle constitue une solution.

Définition 35 (Solution)**.**
Une solution est une affectation complète telle que toutes les contraintes de $\mathcal{C}$ sont satisfaites.

Par la suite, nous noterons $A \in C_i$ lorsqu'une instanciation quelconque A satisfait une contrainte C_i. Ainsi, en reprenant l'exemple de CSP défini précédemment, il existe quatre solutions : $(1,3,2)$, $(1,4,2)$, $(1,4,3)$ et $(2,4,2)$.

2.1.2 Résolution d'un CSP

Comme nous l'avons vu précédemment, la recherche des solutions d'un CSP consiste à parcourir l'espace de recherche pour trouver les différentes instanciations de valeurs qui satisfont les différentes contraintes. Dans de nombreux cas, il n'est pas évident de trouver les solutions du CSP. Les raisons de cette difficulté peuvent être liées au nombre de variables souvent trop grand ou aux domaines de ces variables qui sont trop vastes. Le nombre d'instanciations totales des variables d'un CSP est alors égale à $\prod_{1 \leq i \leq n} |D_i|$.

Pour trouver les solutions du CSP, une énumération exhaustive de toutes les instanciations n'est donc pas possible. Les méthodes de résolution procèdent alors en deux étapes : l'instanciation des variables à une des valeurs de leur domaine et le filtrage des valeurs inconsistantes des domaines des variables.

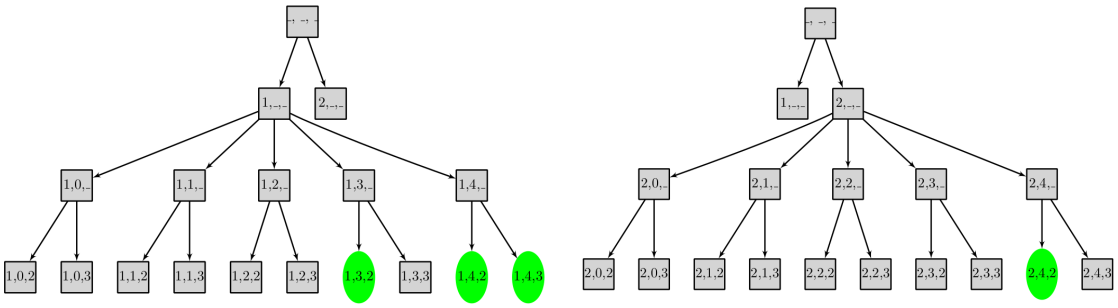
- Le filtrage permet de réduire la taille de l'espace de recherche grâce à la propagation de contraintes ;
- l'instanciation des variables permet également une réduction de l'espace de recherche en éliminant, grâce à la propagation de contraintes et des techniques d'exploration, de sous-espaces non solutions.

Ces deux tâches sont fréquemment associées afin d'améliorer la performance des algorithmes de résolution des CSP. La résolution peut ainsi être décrite par l'algorithme 3.

2.2 Consistance et filtrage

Le parcours de l'espace de recherche des CSPs se fait de façon arborescente et induit une construction itérative des solutions. Les variables du problème sont en effet instanciées au fur et à mesure jusqu'à obtenir une instanciation complète qui satisfait toutes les contraintes $\mathcal{C}$.

Un parcours naïf de l'espace de recherche consisterait à instancier chaque variable du problème avec chaque valeur de son domaine, puis tester la satisfaction des contraintes lorsqu'on a une instanciation totale. Cette méthode conduit à l'énumération de toutes combinaisons de valeurs possibles et peut être représentée par une structure arborescente dont les nœuds correspondent aux instanciations partielles ou totales. La recherche des solutions consiste alors à explorer tous les nœuds de l'arbre de recherche. Pour l'exemple 2.1.1, l'espace de recherche correspond ainsi à l'arbre de la figure 2.1 où les solutions sont affichées en vert.



(a) Sous-arbre avec $X_1 = 1$

(b) Sous-arbre avec $X_1 = 2$

FIGURE 2.1 – Espace de recherche de l'exemple 2.1.1

Cette exploration naïve peut conduire à des phénomènes de *trashing (déchets)* [101] qui consistent en l'exploration répétée de sous-arbres qui ne donnerons jamais de solutions.

Exemple 2.2.1.
Dans la figure 2.1b, on constate que le sous-arbre ayant pour racine $\langle 2, \_, \_ \rangle$ *ne contient qu'une seule solution :* $\langle 2, 4, 2 \rangle$*. Un autre sous-arbre ayant pour racine* $\langle 2, 1, \_ \rangle$ *ne contient aucune solution car les instanciations de* X_1 *à* 2 *et* X_2 *à* 1 *ne permettent pas de trouver de valeur* $v \in D_3$ *du domaine de* D_3 *permettant de satisfaire la contrainte* C_2*. On parle alors d'***inconsistance**. *Une exploration naïve de l'arbre de recherche passera pourtant deux fois par le nœud* $\langle 2, 1, \_ \rangle$*.*

2.2.1 Consistance

Définition 36 (Consistance locale)**.**
Une instanciation A *sur* $\mathcal{Y}$ *viole une contrainte* C_i *si et seulement si* $var(C_i) \subseteq \mathcal{Y}$ *et* $A[var(C_i)] \notin C_i$*. Une instanciation* A *est dite localement consistante, si elle ne viole aucune des contraintes qui impliquent les variables de* $\mathcal{Y}$*. Toutes les valeurs* a_i *de* A *sont alors dites localement inconsistantes.*

Définition 37 (Support d'une valeur pour une contrainte)**.** *Étant donné une contrainte binaire* C *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 *si et seulement si* (v_i, v_j) *satisfait* C*.*

Exemple 2.2.2.
En reprenant le CSP de l'exemple 2.1.1 ainsi que la figure 2.1, on peut dire que $A_1[\mathcal{Y}] = \langle 1, 4 \rangle$*, avec* $\mathcal{Y} = \{X_1, X_2\}$*, est localement consistante car* $var(C_1) = \{X_1, X_2\} \subseteq \mathcal{Y}$*.*
Par contre, $A_2[\mathcal{Y}] = \langle 1, 0 \rangle$*, avec* $\mathcal{Y} = \{X_1, X_2\}$*, est localement inconsistante puisque* $var(C_1) \subseteq \mathcal{Y}$ *et* C_1 *n'est pas satisfaite.*

Une solution d'un CSP est donc une instanciation complète sur $\mathcal{X}$ localement consistante.

Définition 38 (Consistance globale d'une instanciation partielle)**.**
Une instanciation partielle A *sur* $\mathcal{Y}$ *est globalement consistante si elle peut être étendue à une solution. En d'autres termes, il existe une solution* S *telle que* $A[\mathcal{Y}] = S[\mathcal{Y}]$

Exemple 2.2.3.
Considérons le CSP de l'exemple 2.1.1.

- *l'instanciation* $A_1[\mathcal{Y}] = \langle 2, 3 \rangle$ *est localement consistante et globalement inconsistante car elle ne peut être étendue à une solution ;*
- *l'instanciation* $A_2[\mathcal{Y}] = \langle 1, 4 \rangle$ *est localement et globalement consistante car elle peut être étendue aux solutions* $\langle 1, 4, 2 \rangle$ *et* $\langle 1, 4, 3 \rangle$*.*

Définition 39 (Réseau globalement consistant)**.**
Un réseau $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ est globalement consistant, si et seulement si pour toute instanciation partielle A sur $\mathcal{Y}$, telle que $\mathcal{Y} \subseteq \mathcal{X}$, A est globalement consistante.

Afin de réduire la taille de l'espace de recherche et faciliter la découverte de solutions, on a recours au **filtrage**.

2.2.2 Filtrage de domaines

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 explorés et 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.

L'un des filtrage les plus communément utilisé pour détecter les valeurs inconsistantes est le filtrage par *Consistance de Domaine* (Dans la littérature l'appellation arc-consistance (AC) est aussi répandue).

Définition 40 (Consistance de domaine)**.** *Une contrainte C_j vérifie la consistante de domaine sur les variables $var(C_j)$ si et seulement si pour toute variable $X_i \in var(C_j)$ et pour toute valeur $a \in D_i$, il existe une instanciation A localement consistante avec $A[X_i] = v$ et $A[var(C_i)] \in C_j$.*

Exemple 2.2.4.
En appliquant le filtrage par consistance de domaine sur la contrainte C_2 (de l'exemple 2.1.1), le domaine de chaque variable devient :

$$D_1 = \{1, 2\}, D_2 = \{3, 4\}, D_3 = \{2, 3\}$$

Toutes les valeurs ne pouvant pas former une instanciation localement consistante, tenant compte des contraintes $\mathcal{C}$ sont filtrées.

Lorsque l'ensemble des contraintes $\mathcal{C}$ d'un réseau $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ vérifient la consistance de domaine, on dit que le réseau est domaine-consistant (ou arc-consistant), mais cela ne garantit pas que le réseau soit globalement consistant, puisque la consistance est une notion purement locale.

2.2.3 Méthodes de Filtrage

La propriété de consistance de domaine permet d'avoir des algorithmes de filtrage assurant un bon compromis entre le temps gagné (en éliminant certaines valeurs) et celui nécessaire pour réaliser ce filtrage. Plusieurs algorithmes de filtrage ont été proposés pour rendre les CSPs domaine-consistants. L'algorithme le plus connu est AC-3 [101, 139] qui a été proposé dans un premier temps pour les cas des contraintes binaires. La complexité de AC-3 dans un réseau de contraintes est en $O(er^3d^{r+1})$ en temps et $O(er)$ en espace, où $e = |\mathcal{C}|$, d la cardinalité du plus grand domaine et r la plus grande arité parmi les contraintes $\mathcal{C}$.

Par la suite, plusieurs extensions de AC-3 ont été proposées pour améliorer la complexité : en insistant particulièrement sur les informations que peut stocker la fonction *Revise* pour améliorer le filtrage :

- *AC*4 [103] : Contrairement à AC-3, AC-4 stocke un certain nombre d'informations concernant les supports des valeurs dans une pré-étape. Le but est de réduire les appels de la fonction *Revise* sur les mêmes contraintes. AC-4 maintient l'AC en une complexité temporelle optimale $O(erd^r)$ et $O(ed^2)$ en espace.
- AC-6 [15] : AC-6 est un compromis entre AC-3 et AC-4. En effet, l'idée est de conserver la complexité optimale dans le pire des cas de AC-4 et d'arrêter la recherche de support pour une valeur le plutôt possible (dès que le premier support est trouvé, comme le fait AC-3). AC-6 maintient également une structure de données plus légère que AC-4. AC-6 a une complexité temporelle en $O(erd^r)$ et $O(ed)$ en espace.

— AC-2001 [17] : améliore la complexité en moyenne, en améliorant la façon de stocker les supports. AC-2001 utilise des pointeurs sur les supports au lieu des listes. AC-2001 maintient l'AC avec une complexité temporelle optimale en $O(er^2d^r)$ et $O(ed^2)$ en espace.

D'autres consistances moins fortes que la consistance de domaine peuvent être appliquées [41], dans le cas où le maintien de la consistance de domaine se révèle coûteux sur de larges domaines. Une des consistances moins fortes est la *consistance de bornes* [99] adoptée dans les CSPs avec des domaines de valeurs continues.

2.2.4 Propagation de contraintes

A l'issue d'un filtrage opéré sur le domaine d'une variable X_i impliquée dans une contrainte C_k, il peut s'avérer intéressant d'examiner si ce changement de domaine affecte d'autres variables. Pour cela, on réveille toutes les contraintes C_l impliquant X_i. En effet, il est possible qu'une valeur $v \in D_j$ telle que la variable $X_j \in var(C_l)$, étant au départ localement consistante, devienne inconsistante, suite au filtrage d'une valeur dans D_i. En d'autres termes, la valeur filtrée de D_i représentait un support pour une ou plusieurs valeurs de X_j. Ce mécanisme est appelé **propagation**. Il permet de réduire successivement les domaines, grâce aux appels de filtrage sur plusieurs variables. Ceci augmente considérablement la puissance du filtrage, particulièrement si une variable est impliquée dans beaucoup de contraintes (la propagation touchera de plus en plus de variables).

En complément aux méthodes de filtrages, la résolution des CSP exploite des stratégies d'exploration de l'espace de recherche afin d'obtenir des solutions.

2.3 Stratégies de recherche

Une fois la propagation des contraintes terminée, il n'est pas garanti que toutes les variables auront été instanciées : certains domaines peuvent encore contenir plusieurs valeurs. Il est alors nécessaire d'instancier directement chacune des variables correspondantes avec une des valeurs de leur domaine jusqu'à obtenir une solution. Les différentes instanciations de variables se font les unes après les autres, permettant une exploration arborescente de l'arbre de recherche dans lequel chaque nœud correspond à une instanciation. Ce processus se répète ainsi jusqu'à l'obtention de toutes les solutions du CSP :

1. on sélectionne chaque variable $X_i \in \mathcal{X}$ non encore instanciée ($|D_i| > 1$) ;
2. on sélectionne chaque valeur $v \in D_i$ pour instancier X_i ;
3. on réalise une propagation de contraintes : l'instanciation d'une variable X_i avec v équivaut au filtrage de toutes les autres valeurs v' du domaine D_i. La propagation permet alors de vérifier que le CSP est toujours domaine-consistant étant donné la nouvelle instanciation et le filtrage ;
4. si la consistance de domaine n'est plus respectée, on déclenche un Échec afin d'éviter le *trashing* et l'exploration de la branche est abandonnée. On revient alors à l'étape 2 et on sélectionne une autre valeur. Si la consistance est respectée, on retourne à l'étape 2 pour tester une autre valeur ou à l'étape 1 pour sélectionner une nouvelle variable.

La résolution d'un CSP réside donc dans l'établissement d'une consistance globale. Les méthodes de recherche arborescente reposent alors sur le principe du Backtrack [62] 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 obtient que des solutions) et complètes (on obtient toutes les solutions).

Dans la suite de cette section, nous présenterons la méthode par *retour-arrière* (Backtrack) ainsi que le parcours avec Maintien de l'Arc-Consistance [118, 16] qui réalise un filtrage par AC à chaque nœud de l'arbre.

2.3.1 Backtrack et maintient d'arc-cohérence

L'algorithme de base pour résoudre les CSPs est l'algorithme du Backtrack chronologique [62]. Le principe consiste à générer des instanciations partielles consistantes, puis étendre ces instanciations par une nouvelle variable. Il s'agit d'une procédure d'énumération récursive qui commence par une affectation vide, et qui, à chaque itération, choisis une valeur d'une variable courante. Des valeurs sont alors associées progressivement aux variables, jusqu'à ce que toutes les contraintes soient satisfaites (i.e., génération d'une solution), ou qu'une contrainte soit violée (i.e., instanciation non consistante) menant vers un retour arrière.

L'algorithme remet en cause la dernière valeur affectée, il revient sur cette variable et essaye une autre valeur de son domaine. Dans le cas où toutes les valeurs d'une variable sont testées, sans succès, l'algorithme revient sur la variable précédente. A chaque retour arrière sur une variable donnée, toutes ces valeurs sont restaurées. La complexité de cette méthode est en $O(ed^n)$, avec $e = |\mathcal{C}|$, $n = |\mathcal{X}|$ et d la cardinalité du plus grand domaine. Sur des problèmes de grandes tailles, cette méthode devient impraticable. L'algorithme de Backtrack peut être amélioré en pratique par l'ajout d'un filtrage des domaines (en pré-traitement ou au cours de la recherche après chaque affectation de variable) [118, 16, 41].

Supposons qu'en cherchant une solution, l'algorithme de Backtrack donne à une variable X_i une valeur a_i qui exclut toutes les valeurs possibles pour une autre variable X_j. L'algorithme de Backtrack ne s'en rendra compte qu'au moment d'instancier X_j. Par ailleurs, étant donné que le retour en arrière est chronologique, la remise en cause de l'affectation de a_i à X_i pourrait se faire tardivement, pénalisant les performances de l'algorithme. Cette situation pourrait être évitée en prenant en compte le fait que $A[X_i] = a_i$ ne pourra jamais faire partie d'une solution. Pour cela, on pourrait utiliser le filtrage pour vérifier qu'une affectation de valeur a_i ne rende pas l'instanciation courante inconsistante. Cette procédure est définie par l'algorithme 3 définie à la section 2.3.3.

Une autre piste d'amélioration concerne le retour-arrière lui-même. Alors que le Backtrack chronologique revient, en cas d'échec, sur le dernier choix effectué, d'autres mécanismes analysent les causes de l'échec :

- soit pour choisir un meilleur point de retour (*Backjumping* [58], *Graph-Based Backjumping* [42], *Conflict-Directed Backjumping* [111]) ;
- soit pour identifier les instanciations partielles qui ne participent à aucune solution et ainsi éviter des explorations inutiles (*Local Change* [119], *Dynamic Backtracking* [61] et `mac-dbt` [85]).

Une description détaillée de ces algorithmes est donnée dans [43].

2.3.2 Ordre de choix des variables et des valeurs

Ordonner les variables et les valeurs avant ou pendant la résolution d'un CSP est connu comme une autre voie importante d'amélioration pour l'algorithme de BT. On distingue généralement deux types d'ordre : statique dont les choix sont effectués avant la recherche et sont figés ; dynamique dont les choix dépendent du sous-arbre à explorer (par exemple, nombre de variables affectées, taille des domaines, etc.) et sont effectués à chaque nœud au cours de la recherche.

Le choix de variable permet de réduire la taille de l'espace de recherche, généralement en provoquant le plus rapidement possible un échec (first-fail) [69]. De nombreuses heuristiques de choix de variable ont été proposées, on distingue notamment :

- l'heuristique domaine minimum (dom) [69] favorisant les variables avec les domaines les plus petits ;
- l'heuristique degré maximum [44] favorise les variables dont les degrés sont les plus élevés ;
- l'heuristique domaine/degré (dom/deg) [16] propose de tirer parti des informations des deux heuristiques précédentes en favorisant les variables dont le ratio de la taille de leur domaine par rapport à leur degré est le plus faible ;

Algorithm 3: RESOLUTION

```
1  Entrées : 𝒳 : ensemble de variables de décision ;
2  Sortie : D : un ensemble de valeurs de domaines ;
3  Début
4      D ← PROPAGER(D, 𝒞)
5      Si ∃X_i ∈ 𝒳 t.q. dom(x_i) est vide
6          retourne Échec
7      Si ∃X_i ∈ 𝒳 t.q. |dom(X_i)| > 1
8          Sélectionner X_i ∈ 𝒳 t.q. |dom(X_i)| > 1 Pour chaque a_{i_j} ∈ dom(X_i) faire
9              RESOLUTION(D ∪ {X_i ← a_{i_j}})
10     Sinon
11         retourne D
```

— l'heuristique domaine/degré futur (dom/fdeg) [128] qui est une variante de dom/deg, où le degré futur d'une variable est utilisé à la place du degré. Le degré futur d'une variable correspond au nombre de contraintes portant sur une variable et dont au moins une variable n'est pas affectée.

Pour chaque variable sélectionnée, le choix de la valeur permet d'aiguiller la recherche vers les branches les plus prometteuses de l'arbre de recherche. Parmi les heuristiques de choix de valeurs, nous retrouvons :

- l'heuristique lexicographique (lex) dont les valeurs sont simplement ordonnées selon l'ordre de leur domaine ;
- l'heuristique *basée « look-ahead »* [54] qui classe les valeurs selon l'ordre croissant du nombre de retraits provoqués par l'affectation de celle-ci ;
- l'heuristique *basée sur les explications* [33] qui utilise les informations liées au retrait d'une valeur (c'est-à-dire aux affectations justifiant ce retrait) ;
- l'heuristique *basée sur les impacts* [113] qui ordonne les valeurs par ordre croissant d'impact. L'impact associé à une valeur mesure l'influence de son affectation sur la taille de l'espace de recherche.

2.3.3 Algorithme de résolution

La résolution d'un CSP s'effectue selon l'algorithme 3. À chaque nœud de l'arbre de recherche, la procédure RESOLUTION sélectionne une variable non encore instanciée (ligne 8) selon l'ordre de choix de variables défini par l'utilisateur. La variable sélectionnée est alors instanciée avec l'une des valeurs a_{i_j} de son domaine (ligne 8). Lorsque l'une des contraintes ne peut être satisfaite, c'est-à-dire lorsque l'un des domaines $D_i = dom(X_i)$ est vide (ligne 5), un BACKTRACK est effectué. Au contraire, l'algorithme retourne une solution (ligne 11) lorsque tous les domaines des variables ont été réduit à des singletons ($|dom(X_i)| = 1$).

2.4 Modélisation des contraintes

Il existe différents types de contraintes dont l'expression dépend de la modélisation et de l'approche adoptée pour résoudre le problème modélisé. En effet, la manière dont les contraintes sont écrites affecte l'efficacité du modèle résultant, car elle affecte la façon dont les contraintes se propageront pendant la recherche. *Harvey et Stuckey* [70] font ainsi remarquer que « Une propriété troublante et peu étudiée des solveurs basés sur la propagation est que la forme d'une contrainte peut modifier la quantité d'informations que la propagation découvre ».

Afin de trouver la meilleure modélisation pour un problème, il est alors nécessaire de connaître les types de contraintes supportées par le solveur de contraintes et le niveau de cohérence appliqué à chaque contrainte. Il faut également connaître la complexité des algorithmes de propagation correspondants [116].

2.4.1 Contraintes Réifiées

Les contraintes réifiées [5] sont un type particulier de contraintes qui associe une variable X de domaine $D = \{0, 1\}$ à une contrainte C de manière à refléter la satisfaction de la contrainte (valeur 1), ou sa non-satisfaction (valeur 0) :

$$X = 1 \iff C \text{ avec } D_X = \{0, 1\} \text{ et } X \notin var(C)$$

Les contraintes réifiées sont utiles pour exprimer des expressions logiques, ou exprimer qu'un certain nombre de contraintes doivent être satisfaites (min, max, exactly). En termes d'expressivité, les contraintes réifiées peuvent former une contrainte sur des contraintes, notamment, pour exprimer une disjonction de contraintes.

Par exemple, pour exprimer la condition qu'au moins une des deux contraintes $\{C_1, C_2\}$ doit être satisfaite, il est possible d'associer ces contraintes à des variables b_1 et b_2 et ajouter la contrainte $b_1 + b_2 \geq 1$ comme suit :

$$\begin{aligned} b_1 &\iff C_1 \\ b_2 &\iff C_2 \\ b_1 + b_2 &\geq 1 \end{aligned}$$

La propagation des contraintes réifiées s'effectue alors comme suit :
— si la contrainte C_i est satisfaite alors on propage $b_i = 1$;
— si la contrainte C_i est insatisfiable alors on propage $b_i = 0$;
— si la variable $b_i = 1$ alors la contrainte C_i doit être satisfaite ;
— si la variable $b_i = 0$ alors la négation de la contrainte C_i doit être satisfaite.

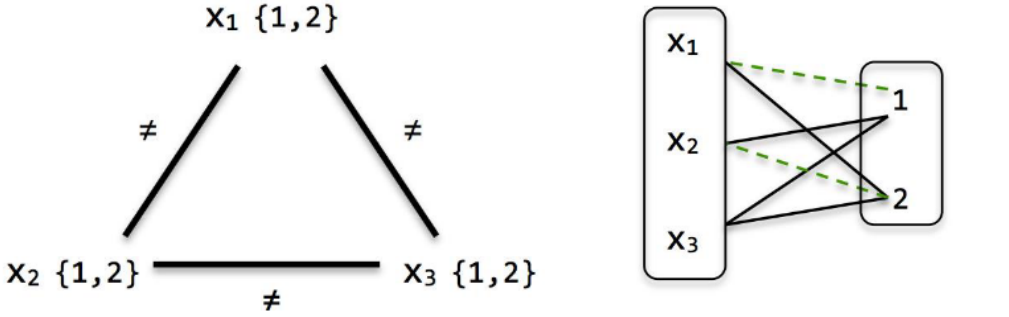
2.4.2 Contraintes Globales

Une contrainte globale est une contrainte qui permet de capturer une relation entre un nombre quelconque et non-borné de variables de décision. L'intérêt est d'avoir un mécanisme de raisonnement sur toute la structure d'un sous-problème, auquel est associé un algorithme de filtrage dédié. En effet, le mécanisme de filtrage présenté précédemment, possède un raisonnement sur chaque contrainte indépendamment des autres, ce qui représente une faiblesse. En d'autres termes, un CSP domaine-consistant ne garantit pas l'existence de solutions.

Exemple 2.4.1.
Soit le CSP suivant avec :
— $\mathcal{X} = \{X_1, X_2, X_3\}$,
— $\mathcal{D} = \{D_1, D_2, D_3\}$, *avec* $D_1 = D_2 = D_3 = \{1, 2\}$
— $\mathcal{C} = \{C_1(X_1, X_2), C_1(X_1, X_3), C_3(X_2, X_3)\}$, *avec*
$C_1(X_1, X_2) : X_1 \neq X_2$
$C_2(X_1, X_3) : X_1 \neq X_3$
$C_3(X_2, X_3) : X_2 \neq X_3$

Ce CSP est domaine-consistant, car aucune valeur ne peut être filtrée. En d'autres termes, dans chaque contrainte $C(X_i, X_j)$, chaque valeur v_i d'une variable X_i possède un support dans le domaine $D(X_j)$ et vice versa. Pourtant, ce CSP n'a pas de solutions. Ceci illustre clairement la faiblesse d'une telle modélisation (voir figure 2.2a).

Un raisonnement global est alors nécessaire pour améliorer le filtrage opéré sur les domaines. Par ailleurs, le filtrage qu'opère une contrainte globale ne peut pas être efficacement propagé par les algorithmes génériques d'arc-consistance tels que $AC-3$ [101], $AC-2001$ [17], sauf le cas où la décomposition de la contrainte globale est Berge-Acyclique [9], on dit alors que la contrainte globale est AC-décomposable, ce qui veut dire que l'on peut assurer le même niveau de consistance sur cette décomposition. De plus, *Cohen et Jeavons* [38] ont montré que lorsqu'une décomposition AC-décomposable existe, celle ci est forcement Berge-acyclique. Pour les contraintes globales, des algorithmes de filtrage (propagateurs) sont construits à partir de la sémantique du problème abordé, dans le but d'établir un filtrage efficace en une complexité polynomiale en temps sur la taille du problème.



(a) Utilisation de contraintes simples (b) Contrainte globale All-Different

Figure 2.2 – Deux modélisation de la contrainte de différence

L'une des contraintes globales les plus connues est la contrainte All-Different [114]. Elle exprime le fait que toutes les valeurs des variables $X_1, X_2, ..., X_n$ de sa portée doivent être différentes. Les contraintes du CSP de l'exemple 2.4.1 pourraient alors être remplacées par la contrainte globale All-Different(X_1, X_2, X_3).

All-Different(X_1, X_2, X_3) associe un algorithme de filtrage dédié, basé sur le couplage maximum (variable - valeur). La contrainte de différence est représentée par un graphe bi-parti (voir figure 2.2b). Le propagateur peut alors détecter que l'on ne peut obtenir un couplage maximum et donc que le CSP ne contienne pas de solution.

Plusieurs contraintes globales ont été proposées, comme la contrainte Regular [110], la contrainte Element [75]. Il existe également un catalogue où les contraintes globales sont regroupées [12]. Dans le chapitre 4, nous présenterons la contrainte globale ClosedDiversity [79] pour l'extraction de motifs diversifiés.

2.5 Modèles PPC pour l'extraction de motifs

Comme nous l'avons vu au chapitre 1, l'extraction de motifs consiste à trouver les régularités dans les jeux de données. Afin de prendre en compte les préférences de l'utilisateur, on utilise des mesures d'intérêts ainsi que des contraintes afin de produire des motifs ou des ensembles de motifs intéressants. L'objectif de la fouille de motifs est donc similaire à celui de la PPC qui consiste à modéliser un problème avec un ensemble de contraintes puis rechercher les solutions satisfaisant ces contraintes. Étant donné la généricité de la PPC, il est alors possible de modéliser le problème d'extraction de motifs sous forme de CSP.

Plusieurs travaux récents ont été menés pour modéliser le problème d'extraction de motifs en utilisant le formalisme CSP [39, 90, 65, 134, 97, 11, 10, 89, 137]. Le premier modèle PPC

utilisé pour la fouille de motifs fréquents et fermés a été proposé par [39]. Ce modèle est basé sur des contraintes réifiées [5] faisant intervenir les items et les transactions d'un jeu de données. Par la suite, [97] ont proposé la première contrainte globale pour produire des motifs fréquents et fermés.

Nous présentons dans cette section des approches exploitant ces deux expressions des contraintes : des contraintes réifiées, des contraintes globales et des `XOR` pour l'échantillonnage de motifs.

2.5.1 Modèle Réifié

Pour l'extraction de motifs, De Raedt *et al.* [39] ont proposé un modèle réifié pour représenter les items et les transactions du jeu de données. Ainsi, ils utilisent deux vecteurs de variables booléennes $\langle I_1, \ldots, I_n \rangle$ pour représenter les items des motifs et $\langle T_1, \ldots, T_m \rangle$ pour représenter les transactions couvertes par les motifs. Chaque variable I_i (respectivement T_i) permet de représenter la présence ou l'absence des items (respectivement des transactions) dans les motifs.

Définition 41 (Contraintes de motifs fréquents)**.**
L'extraction de motifs fréquents peut être représenté par les contraintes réifiées suivantes :

$$\forall t \in \mathcal{T} : T_t = 1 \iff \sum_{i \in \mathcal{I}} I_i(1 - \mathcal{D}_{ti}) = 0 \tag{2.1}$$

$$\forall i \in \mathcal{I} : I_i = 1 \iff \sum_{t \in \mathcal{T}} T_t \mathcal{D}_{ti} \geq \theta \tag{2.2}$$

avec θ le seuil de fréquence minimum et $\mathcal{D}_{ti}$ la présence ($\mathcal{D}_{ti} = 1$) ou l'absence ($\mathcal{D}_{ti} = 0$) de l'item i dans la transaction t.

La contrainte 2.1 est une reformulation de la contrainte de couverture : $T = \varphi(X)$, avec $\varphi(X) = \{t \in \mathcal{T} \mid \forall i \in X : \mathcal{D}_{ti} = 1\}$. Cette contrainte permet de retrouver les différentes transactions couvertes par un motif. De même, la contrainte 2.2 est une reformulation de la contrainte de fréquence $\sum_{t \in \mathcal{T}} T_t \geq \theta$ qui garantie que le support de chaque motif est supérieur à θ. L'extraction des motifs consiste alors à trouver les instanciations I_i et T_t satisfaisant les contraintes 2.1 et 2.2.

Pour l'extraction de motifs fermés, on pourrait ajouter aux contraintes de la définition 41 des contraintes de fermeture.

Définition 42 (Contraintes de motifs fermés)**.**
L'extraction de motifs fréquents fermés est réalisée en combinant les contraintes 2.1 et 2.2 avec la contrainte de fermeture suivante :

$$\forall i \in \mathcal{I} : I_i = 1 \iff \sum_{t \in \mathcal{T}} T_t(1 - \mathcal{D}_{ti}) = 0 \tag{2.3}$$

avec θ le seuil de fréquence minimum et $\mathcal{D}_{ti}$ la présence ($\mathcal{D}_{ti} = 1$) ou l'absence ($\mathcal{D}_{ti} = 0$) de l'item i dans la transaction t.

Le modèle réifié CP4IM présente l'avantage de permettre une simplicité de la modélisation. En effet, il permet une représentation simplifiée des motifs grâce aux vecteurs sur les items et les transactions. Par ailleurs, CP4IM bénéficie de la déclarativité du formalisme CSP qui permet l'ajout simplifié de contraintes. Toutefois, ce modèle est handicapé par le nombre de contraintes qui sont utilisées. En effet, la réification nécessite d'associer chaque variable avec une contrainte (voir section 2.4.1). Cela a pour conséquence l'utilisation d'un nombre de contraintes qui peut devenir très grand pour certains jeu de données. Ainsi, pour représenter la couverture des motifs avec la contrainte 2.1, il est nécessaire d'utiliser $m = |\mathcal{T}|$ contraintes. De même, il est nécessaire d'utiliser $n = |\mathcal{I}|$ contraintes pour garantir le respect de la fréquence minimum par les

motifs. CP4IM utilise ainsi $n + m$ contraintes pour l'extraction de motifs fréquents et $2n + m$ contraintes pour l'extractions de motifs fréquents fermés. De plus, l'arité élevée des contraintes réifiées pose un problème supplémentaire. En effet, pendant la résolution, le domaine D_i de chaque variable X_i est réduit progressivement jusqu'à l'obtention d'une solution ou la détection d'une inconsistance. Lorsqu'une contrainte C_j est d'arité n, il faudra alors propager C_j sur chaque variable $X_{j_k} \in var(C_j) = \{X_{j_1}, \dots X_{j_n}\}$. Ainsi, lorsque l'arité est élevée, les opérations de propagation deviennent complexes, impactant négativement les performance de résolution du CSP.

Le modèle réifié passe alors difficilement à l'échelle lorsqu'on l'utilise sur de grand jeux de données étant donné que le temps nécessaire pour la résolution devient important. Pour résoudre ce problème, l'extraction des motifs peut être modélisée sous forme de contrainte globale.

2.5.2 Modèle par contrainte globale : ClosedPatterns

Lazaar *et al.* [97] ont proposé la première contrainte globale pour produire des motifs fréquents et fermés. Ils utilisent un vecteur X de variables booléennes $\langle X_1, \dots, X_{|\mathcal{I}|}\rangle$ pour représenter les motifs. Chaque variable X_i représente la présence de l'item $i \in \mathcal{I}$ dans le motif. Nous utiliserons les notations suivantes : $X^+ = \{i \in \mathcal{I} \,|\, dom(X_i) = \{1\}\}$ l'ensemble des items présents, $X^- = \{i \in \mathcal{I} \,|\, dom(X_i) = \{0\}\}$ l'ensemble des items absents et $X^* = \{i \in \mathcal{I} \,|\, i \notin X^+ \cup X^-\}$.

Définition 43 (ClosedPatterns). *Soit X un vecteur de variables booléennes, θ un seuil de support minimum et $\mathcal{D}$ un jeu de données. La contrainte globale $\textsc{ClosedPatterns}_{\mathcal{D},\theta}(x)$ est vérifiée si et seulement si X^+ est à la fois fermé et fréquent.*

Définition 44 (Extension propre [140]). *Un motif non nul P est une extension propre de Q si et seulement si $\mathcal{V}_{\mathcal{D}}(P \cup Q) = \mathcal{V}_{\mathcal{D}}(Q)$.*

Règles de filtrage. Lazaar *et al.* ont proposé trois règles de filtrage pour ClosedPatterns. La première règle permet d'étendre un motif X^+ avec un item i lorsque $X^+ \cup \{i\}$ est une extension propre de X^+ (voir Définition 44). Dans ce cas, on supprime la valeur 0 de $dom(X_i)$. La seconde règle permet de vérifier la fréquence du motif $X^+ \cup \{i\}$ et de supprimer la valeur 1 de $dom(X_i)$ si son support est inférieur au seuil θ. La troisième règle supprime la valeur 1 de $dom(X_i)$ lorsque $\mathcal{V}_{\mathcal{D}}(X^+ \cup \{i\}) \subset \mathcal{V}_{\mathcal{D}}(X^+ \cup \{j\})$, avec j un item absent ($j \in X^-$).

Exemple 2.5.1.
Nous reprenons dans la table 2.1 le jeu de données d'illustration utilisé au chapitre 1.

$\mathcal{T}$	Items					
t_1	A					
t_2	A	B		D		
t_3	A			D	E	
t_4	A			D	E	
t_5		B		D	E	
t_6		B		D	E	
t_7			C	D	E	
t_8			C	D	E	
t_9			C	D		F
t_{10}			C	D		F
t_{11}			C		E	F

Table 2.1 – Jeu de données d'illustration de ClosedPatterns.

Pour l'extraction de motifs fermés de fréquence $\theta \geq 3$ avec ClosedPatterns, on utilise le CSP suivant :

- $\mathcal{X} = \{X_A, X_B, X_C, X_D, X_E, X_F\}$
- $D = \{D_A, D_B, D_C, D_D, D_E, D_F\}$, *avec* $D_X = \{0, 1\}\ \forall X \in \mathcal{X}$
- $\mathcal{C} = \textsc{ClosedPatterns}(\mathcal{D}, \mathcal{X}, \theta)$

Nous avons représenté dans la figure 2.3 la recherche des motifs fermés avec ClosedPatterns. *Chaque nœud en blanc représente une instanciation partielle. La valeur du nœud est égale à la prochaine variable X à assigner. Chaque arrête représente l'assignation d'une valeur à la variable X représentée dans le nœud parent. Enfin, les nœuds en vert représentent les instanciations complètes, c'est à dire les motifs fermés.*

On remarque qu'en assignant (avec l'algorithme 3 par exemple) la valeur 1 *à la variable X_A, la 2^{de} règle de filtrage de* ClosedPatterns *filtrera la valeur* 1 *des domaines D_B, D_C, D_E et D_F. On pourra alors assigner X_D successivement avec* 1 *et* 0 *pour obtenir les motifs fermés AD et A. Ces deux motifs correspondent aux instanciations complètes $A(\mathcal{Y}_1) = \langle 1, 0, 0, 1, 0, 0\rangle$ pour AD et $A(\mathcal{Y}_2) = \langle 1, 0, 0, 0, 0, 0\rangle$ pour A.*

De même, en assignant les valeurs 0 *à X_A et* 1 *à X_B, la 1^{re} règle de filtrage de* ClosedPatterns *filtrera les valeurs* 0 *du domaine D_D et* 1 *des domaines D_C, D_E, D_F. On obtient alors l'instanciation complète $A(\mathcal{Y}_3) = \langle 0, 1, 0, 1, 0, 0\rangle$.*

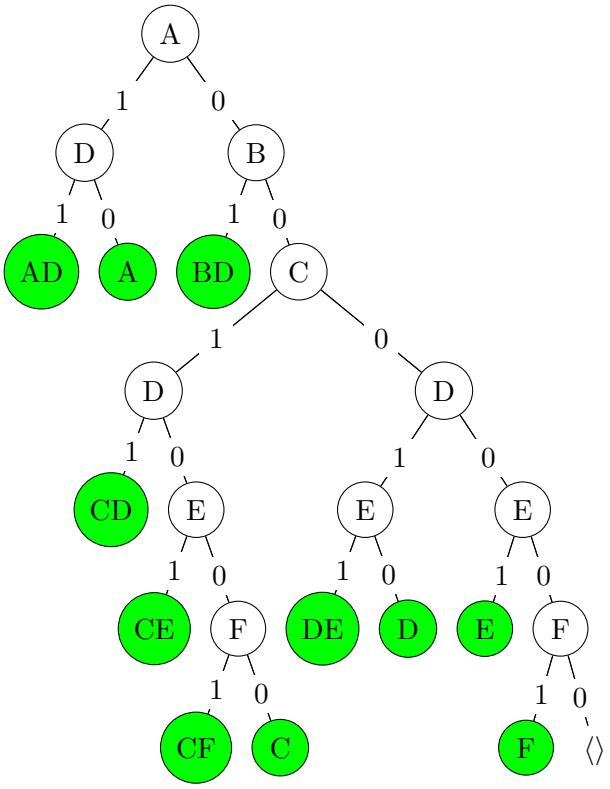


FIGURE 2.3 – Arbre de recherche des motifs fermés du jeu de données de la table 2.1 avec ClosedPatterns

L'utilisation d'une contrainte globale pour l'extraction des motifs présente l'avantage de réduire considérablement le nombre de contraintes utilisées. En effet, les contraintes globales simplifient, grâce aux règles de filtrage, l'expression des différentes contraintes. Par ailleurs, le raisonnement globale rend moins complexe la résolution du CSP. Ainsi, ClosedPatterns permet de passer de $2n+m$ contraintes à une seule contrainte pour l'extraction de motifs fréquents et fermés. Toutefois, il n'est pas toujours aisé de modéliser les contraintes globales. En effet, les règles de filtrage ne sont pas faciles à trouver et peuvent rendre la modélisation complexe. Dans la section suivante, nous présenterons un modèle *PPC* qui s'appuie sur l'échantillonnage de motifs.

2.5.3 Échantillonnage de motifs et contraintes XOR

Le formalisme CSP peut être utilisé pour l'échantillonnage de motifs. Dzyuba *et al.* ont ainsi proposé dans [51] un outils appelé FLEXICS qui exploite le framework WEIGHTGEN [35] afin d'échantillonner des motifs proportionnellement à une mesure φ.

WEIGHTGEN est un outils permettant d'échantillonner les solutions d'un problème SAT, c'est-à-dire les valeurs de littéraux qui permettent à la formule booléenne

$$\phi = Cl_1 \wedge Cl_2 \wedge ... \wedge Cl_m$$

d'être vraie. Ce problème peut être modélisé avec un ensemble de variables X_i booléennes ($D_{X_i} = \{0, 1\}$) représentant les littéraux et des contraintes $C_i : X_j + X_k + ... + X_l \geq 1$ représentant les clauses Cl_i.

Pour échantillonner les solutions, Chakraborty *et al.* [35] proposent de partitionner l'espace de recherche des solutions avec des contraintes `XOR` aléatoires. `XOR` ou *OU Exclusif* noté $\oplus$ est un opérateur utilisé en logique propositionnelle ayant la table de vérité suivante :

$\otimes$	0	1
0	0	1
1	1	0

Définition 45 (Contraintes XOR)**.**
Les contraintes `XOR` s'expriment sous forme d'une formule

$$\phi = \oplus b_i.X_i$$

telle que $\phi = b_0$ où :

- *les b_i sont les coefficients de la contrainte : $b_i \in \{0, 1\}$. Ils déterminent l'apparition d'une variable X_i dans la formule ;*
- *b_0 est le bit de parité de la contrainte.*

La résolution de la contrainte `XOR` consiste alors à trouver les instanciations de variables qui rendent possible l'égalité $\phi = b_0$. Chaque contrainte `XOR` divise l'espace de recherche en deux, les m contraintes `XOR` permettent ainsi de diviser la taille de l'espace de recherche par 2^m. L'échantillonnage d'une solution par WEIGHTGEN consiste alors à partitionner l'espace de recherche en des cellules avec des contraintes `XOR` aléatoires puis à tirer un ensemble de valeurs de littéraux qui rendent vraie la formule booléenne ϕ. Pour cela, il procède en deux étapes :

1. Une étape d'estimation du nombre m de contraintes `XOR` nécessaires pour le partitionnement. En effet, les contraintes `XOR` sont ajoutées au fur et à mesure afin de réduire progressivement les cellules à une taille désirée. Cette taille est mesurée à l'aide d'une fonction de poids qui associe à chaque solution de la cellule une valeur numérique ;
2. La deuxième étape consiste à générer m contraintes `XOR` aléatoires pour partitionner l'espace de recherche puis à tirer les solutions proportionnellement à leur poids.

Dzyuba *et al.* exploitent le cadre de ce travail pour l'échantillonnage de motifs. Ils proposent ainsi d'exploiter WEIGHTGEN pour partitionner l'espace de recherche des motifs en des cellules puis de tirer dans différentes cellules des motifs proportionnellement à leur poids. Leur outils, FLEXICS, comporte deux implémentations en fonction de l'oracle utilisé pour extraire les motifs. En effet, pour évaluer le poids d'une cellule, il est nécessaire d'énumérer les motifs qu'elle contient. Les auteurs de FLEXICS proposent alors la possibilité d'utiliser deux oracles :

- un oracle exploitant l'algorithme ECLAT pour énumérer les motifs. Cette oracle donne l'implémentation EFLEXICS ;
- un oracle utilisant CP4IM, donnant l'implémentation GFLEXICS.

Pour la pondération des motifs, FLEXICS permet un choix entre différentes fonctions comme la fréquence ou la pureté (purity). Par ailleurs, la méthode d'échantillonnage ajoute une diversité (implicite) entre les motifs échantillonnés. En effet, la partition de l'espace de recherche permet d'obtenir des cellules de tailles réduites dans lesquelles il est plus facile de réaliser un tirage des motifs. L'utilisation des contraintes `XOR` permet alors à FLEXICS de tirer des motifs qui décrivent différentes régions du jeu de données, étant données que les différentes cellules sont différentes les unes des autres. Cependant, les cellules exploitées par FLEXICS pour tirer ses motifs ne sont pas toujours mutuellement exclusives. En effet, pour une paire de cellules $(Cell_i, Cell_j)$, il est possible d'avoir $Cell_i \cap Cell_j = \mathcal{P} \neq \emptyset$. De ce fait, il est possible d'avoir des redondances dans l'ensemble $\mathcal{X}$ de motifs final.

2.6 Conclusion

La programmation par contraintes offre un cadre déclaratif permettant de modéliser et de résoudre les problèmes de satisfaction de contraintes. Elle permet une manipulation générique des variables et des contraintes et exploite des méthodes de filtrages et des outils de résolution efficaces. Il est alors possible de l'exploiter pour modéliser et résoudre des problèmes comme l'extraction ou l'échantillonnage de motifs. Ainsi, dans le chapitre 4, nous exploiterons le formalisme CSP pour proposer une modélisation du problème d'extraction de motifs diversifiés.

CHAPITRE 3

Fouille interactive de motifs et apprentissage

Contents

3.1 Introduction . . . 57
3.2 Cadre interactif pour la fouille de motifs . . . 58
3.3 Représentation des motifs . . . 59
3.4 Retours de l'utilisateur . . . 61
3.4.1 Feedback par rangement . . . 62
3.4.2 Feedback quantitatif . . . 62
3.4.3 Feedback implicite . . . 63
3.4.4 Synthèse . . . 64
3.5 Apprentissage des préférences de l'utilisateur . . . 64
3.5.1 Tâches d'apprentissage des préférences . . . 64
3.5.2 Apprentissage d'une fonction de poids . . . 65
3.6 Sélection des motifs . . . 68
3.7 Schéma général de la fouille interactive de motifs et instanciation . . . 69
3.7.1 Schéma général de la fouille interactive de motifs . . . 69
3.7.2 LetSIP . . . 69
3.8 Conclusions . . . 70

3.1 Introduction

Comme nous l'avons présenté dans les chapitres précédents, la fouille de motifs est une tâche qui permet de trouver des régularités dans les jeux de données. Ces régularités, appelées motifs, permettent chacune de décrire de manière concise la structure d'un sous-ensemble du jeu de données. Les motifs extraits par une méthode d'extraction de motifs peuvent alors servir comme ingrédients d'une analyse plus approfondie : en tant que caractéristiques ou règles pour la construction de classificateurs par exemple, ou en tant que descripteurs de clusters ou de sous-groupes, qui sont interprétés par des utilisateurs finaux humains [148].

L'extraction de ces motifs a fait l'objet de nombreux travaux [3, 40, 109, 64] et différentes approches ont été proposées :

— La fouille de motifs sous contraintes locales permet à l'utilisateur d'exprimer ses préférences. Cependant, les motifs extraits peuvent être trop nombreux pour être analysés et peuvent présenter de nombreuses redondance.

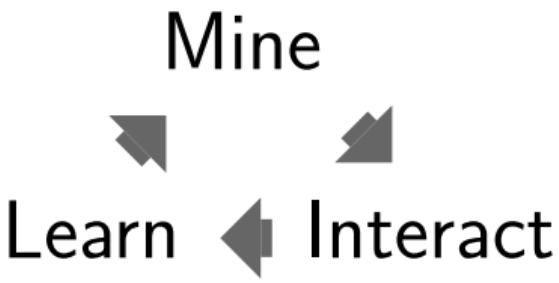


FIGURE 3.1 – Approche générale de fouille et d'apprentissage actif de motifs

— Les représentations condensées, comme les motifs fermés, permettent de réduire la redondance, mais le problème d'explosion des motifs demeure.
— La fouille d'ensemble de motifs permet de prendre en compte les relations entre les motifs et de nombreuses méthodes proposées permettent de réduire la redondance entre les motifs. Cependant, l'extraction d'ensembles de motifs peut être une tâche difficile à cause de la complexité des algorithmes (exemples : PATTERNSTEAM, KRIMP).
— L'échantillonnage permet d'obtenir des ensemble de motifs de façon efficace mais peut nécessiter l'intervention de l'utilisateur car les motifs extraits prennent difficilement en compte ses préférences (exemples : CFTP, GIBBS).

Des travaux récents [20, 47, 48] ont alors entrepris d'introduire l'utilisateur dans le processus de fouille. La prise en compte des préférences de l'utilisateur permet ainsi de guider la recherche vers les motifs qu'il préfère. Dans ce chapitre, nous présenterons dans la section 3.2 le cadre de la fouille interactive de motifs ainsi que ses différents ingrédients nécessaires. Puis, nous discuterons les défis à relever par la fouille interactive de motifs. Nous détaillerons notamment la représentation des motifs à la section 3.3, les retours de l'utilisateur à la section 3.4 et l'apprentissage des préférences à la section 3.5.

3.2 Cadre interactif pour la fouille de motifs

L'intégration de l'utilisateur dans le processus de fouille consiste à prendre en compte les préférences de celui-ci afin de guider la recherche vers des motifs intéressants. Pour cela, les processus de fouille interactive de motifs se déroulent dans un framework à trois composantes dans lesquelles un ensemble d'actions sont effectuées de façon successive. Les composantes du framework sont alors exploitées dans une boucle comme le montre la figure 3.1 :

1. **Extraction** (MINE) : cette étape consiste à extraire un ensemble de motifs de taille suffisamment réduite pour permettre à l'utilisateur de pouvoir l'analyser et exprimer ses préférences.
2. **Interaction** (INTERACT) : à cette étape, les motifs extraits précédemment sont présentés à l'utilisateur afin qu'il exprime ses préférences. Il s'agira alors pour lui d'indiquer les motifs qu'il préfère et ceux qui lui déplaisent.
3. **Apprentissage** (LEARN) : les préférences exprimées par l'utilisateur sont généralisées afin d'apprendre un modèle de préférence.
4. **Fouille de motifs** (encore) : un nouvel ensemble de motifs **bénéficiant du modèle de préférence** extrait est présenté à l'utilisateur. Une boucle s'établit alors entre l'étape 2 et l'étape 4.

La fouille interactive de motifs nécessite de relever différents défis qui se posent au niveau de chaque étape de la boucle. Pour la composante MINE, étant donné qu'il s'agit d'un processus

interactif itératif, il est nécessaire que la méthode d'extraction de motifs soit efficace. En effet, une trop longue attente de l'utilisateur ferait perdre au processus son caractère interactif. Par ailleurs, la méthode d'extraction doit pouvoir intégrer le modèle de préférence appris à l'étape 3 afin de pouvoir extraire au fil des itérations des motifs qui correspondent de plus en plus au préférences exprimées. Enfin, il est nécessaire que la méthode d'extraction puisse présenter des ensembles de motifs présentant peu de redondance entre les motifs. En effet, il est difficile pour un utilisateur d'exprimer des préférences sur des motifs qui sont similaires et qui présentent peu de diversité.

Afin que l'utilisateur puisse exprimer ses préférences, il est également nécessaire de simplifier l'expression de celles-ci dans la composante Interact. Pour un utilisateur donné, il est ainsi plus facile de s'exprimer sur une paire de motifs (feedback binaire) que sur un ensemble de k motifs, avec $k > 2$ (feedback gradué). Cependant, les préférences exprimées doivent pouvoir être précises et s'exprimer sur un ensemble de motifs plus ou moins grand. Dans ce cas, le feedback binaire est moins intéressant que le feedback gradué.

À l'étape 3, la composante Learn généralise les préférences et apprend un modèle de préférences. Pour faciliter cette tâche, il est important que le processus d'apprentissage permette un apprentissage facilité afin de ne pas perturber l'interactivité. Par ailleurs, le modèle de préférence appris doit pouvoir être assez expressif afin de permettre au modèle d'évoluer au fil des itérations.

La fouille interactive de motifs permet alors d'obtenir un framework permettant de mieux représenter les préférences de l'utilisateur grâce à un *apprentissage actif* à partir des retours exprimés. L'intérêt des motifs n'est alors plus déterminés par le jeu de données uniquement mais également par les retours U de l'utilisateur obtenus grâce à des interactions :

$$\Phi : \mathcal{L} \times U \rightarrow \mathbb{R}$$

Dans les sections suivantes, nous présenterons différents ingrédients nécessaires à la fouille interactive comme les descripteurs des motifs. Nous détaillerons également différentes formes de retours de l'utilisateur ainsi que l'apprentissage et l'exploitation des préférences de celui-ci pour extraire des motifs intéressants..

3.3 Représentation des motifs

L'un des ingrédients importants de la fouille interactive de motifs est la description des motifs. En effet, pour extraire des motifs intéressants pour l'utilisateur, il est nécessaire de caractériser ses préférences par des valeurs permettant de sélectionner les plus pertinents. Cette sélection nécessite alors d'évaluer un ensemble de caractéristiques propres à chaque motif.

Exemple 3.3.1.
Un banquier pourrait considérer les conditions suivantes avant d'attribuer un crédit à un client : l'âge, le statut professionnelle, le revenu, l'existence d'un crédit en cours. En représentant chacune de ces conditions par un item i, le banquier pourrait alors privilégier :

1. *les clients ayant un emploi et des revenus supérieurs à 2500 €/mois : ses préférences portent alors sur des motifs contenants ces deux items ;*
2. *les clients ayant moins de 55 ans : dans ce cas, ses préférences excluent ou défavorisent les clients de plus de 55 ans ;*
3. *les clients qui remplissent au moins les deux conditions précédentes : ses préférences portent sur les motifs d'une longueur $l \geq 2$.*

Il est alors important de pouvoir caractériser chaque client par rapport à chacune de ces conditions afin d'évaluer plus facilement leur possibilité d'accès au crédit.

La représentation des motifs permet ainsi de déterminer la manière dont l'utilisateur fait référence aux objets intéressants pour lui. Elle peut alors avoir une influence sur la recherche des motifs à présenter à l'utilisateur [63]. En effet, si les descripteurs utilisés ne sont pas pertinents, l'apprentissage des préférences pourrait ne pas être assez précis et la méthode de fouille échouera à retourner à l'utilisateur les motifs qu'il préfère.

En reprenant l'exemple 3.3.1, il serait impossible de satisfaire la préférence 3 du banquier si la représentation des motifs ne permet de connaître leur longueur.

Pour représenter les motifs, les modèles les plus généralement utilisés exploitent les caractéristiques statiques des motifs. On parle ainsi de **descripteurs statiques**. L'ensemble des descripteurs obtenus forment un vecteur $\mathcal{F} = \langle F_1, \ldots, F_n \rangle$. Pour décrire un motif X, il faut alors déterminer leurs différentes valeurs X_{F_i} avec chaque descripteur F_i. Il en résulte la description suivante $X_{\mathcal{F}}$ du motif X :

$$X_{\mathcal{F}} = \langle X_{F_1}, \ldots, X_{F_n} \rangle.$$

Différents descripteurs statiques sont utilisés dans la littérature :

- Les descripteurs binaires : ils permettent d'attribuer une valeur binaire $X_{F_i} \in \{0, 1\}$ à chaque motif X. On retrouve notamment :
  - les items : pour chaque item i, $X_{\mathcal{I}_i} = 1$ si l'item i est présent dans le motif X et 0 sinon ;
  - les transactions : pour chaque transaction t_j, $X_{\mathcal{T}_j} = 1$ si le motif X couvre la transaction $\mathcal{T}_j$.
- Les descripteurs numériques décrivent les motifs par des valeurs numériques, i.e. $X_{F_i} \in \mathbb{R}$. Ces valeurs sont généralement obtenues à la suite d'une mesure effectuée sur le motif. On retrouve ainsi des descripteurs décrivant les motifs par leur fréquence ou leur longueur, mais également par l'*aire*, le *contraste par rapport à un attribut cible*, le *cosinus*, les mesures de *Yules* et de *Laplace* [13].

Ces différents descripteurs peuvent être combinés afin de décrire de façon plus détaillé les différents motifs.

Exemple 3.3.2.
Reprenons le jeu de données d'illustration utilisé au chapitre 1 et représenté dans la table 3.1.

$\mathcal{T}$	Items					
t_1	A					
t_2	A	B		D		
t_3	A			D	E	
t_4	A			D	E	
t_5		B		D	E	
t_6		B		D	E	
t_7			C	D	E	
t_8			C	D	E	
t_9			C	D		F
t_{10}			C	D		F
t_{11}			C		E	F

TABLE 3.1 – Jeu de données d'illustration.

Considérons les motifs $X_1 = ADE$, $X_2 = CF$ et $X_3 = D$. En utilisant comme descripteurs l'ensemble des items, l'ensemble des transactions, la longueur des motifs et leur fréquence, nous obtenons les représentations suivantes :

- *o* $X_{1_{\mathcal{I}}} = \langle 1,0,0,1,1,0\rangle$, $X_{1_{\mathcal{T}}} = \langle 0,0,1,1,0,0,0,0,0,0,0\rangle$, $X_{1_{length}} = \langle 0.5\rangle$ [2], $X_{1_{freq}} = \langle 0.182\rangle$ [3]. *Le vecteur descripteur de* X_1 *vaut alors :*

$$X_{1_{\mathcal{F}}} = \langle X_{1_{\mathcal{I}}}, X_{1_{\mathcal{T}}}, X_{1_{length}}, X_{1_{freq}}\rangle = \langle 1,0,0,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0.5,0.182\rangle$$

- *o De la même manière, nous obtenons :*

$$X_{2_{\mathcal{F}}} = \langle X_{2_{\mathcal{I}}}, X_{2_{\mathcal{T}}}, X_{2_{length}}, X_{2_{freq}}\rangle = \langle 0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,1,1,0.33,0.273\rangle$$

$$X_{3_{\mathcal{F}}} = \langle X_{3_{\mathcal{I}}}, X_{3_{\mathcal{T}}}, X_{3_{length}}, X_{3_{freq}}\rangle = \langle 0,0,0,1,0,0,0,1,1,1,1,1,1,1,1,1,0,0.17,0.818\rangle$$

Ces différents descripteurs peuvent être exploités par des fonctions afin d'évaluer la qualité des motifs. Ils peuvent alors servir pendant l'étape d'extraction des motifs pour sélectionner les motifs intéressants. Pour cela, on associe à chaque descripteur F_i une valeur réelle w_i représentant le degré d'importance de ce descripteur. Le poids d'un motif X est alors évalué en exploitant les poids associés aux descripteurs et une fonction de préférences Φ :

$$\Phi : \mathcal{L} \times w_{\mathcal{F}} \times X_{\mathcal{F}} \rightarrow \mathbb{R}. \tag{3.1}$$

Ce poids w_i va ainsi évoluer au fil des itérations grâce à l'apprentissage qui va permettre de mettre à jour ses valeurs.

Discussion
Ces représentations de motifs sont faciles à mettre en œuvre et à utiliser. De plus, ils permettent de décrire les motifs à la fois syntaxiquement (avec les items, les transaction ou la fréquence) et sémantiquement (le cosinus pour évaluer la diversité par exemple). Cependant, l'une de leurs limites vient de l'utilisation de caractéristiques statiques pour décrire les motifs. En effet, ces descripteurs sont fixés au début du processus interactif et ne peuvent pas évoluer de façon dynamique au cours des itérations. Or, au cours de l'apprentissage, certains descripteurs peuvent être facultatifs pendant quelques itérations car ils n'influencent pas le processus. Il pourrait alors être intéressant de pouvoir faire varier les descripteurs des motifs à chaque itération de la fouille interactive.

Dans le chapitre 6, nous présenterons une nouvelle classe de descripteurs dynamiques qui permettent de caractériser les motifs à partir des retours de l'utilisateur.

3.4 Retours de l'utilisateur

Comme nous l'avons indiqué à la section 3.3, les motifs peuvent être représentés par un ensemble de descripteurs. Ces descripteurs étant associés à des poids w_i, on obtient la préférence de l'utilisateur pour un motif X en évaluant son poids $\Phi(X)$ [84]. La fonction Φ permet alors d'établir une relation de préférence entre les motifs $X \in \mathcal{L}$.

Définition 46 (Relation de préférence)**.**
Une relation de préférence, notée $\succ$*, est une relation binaire qui établit un ordre de préférence entre les motifs* $X \in \mathcal{L}$*. La relation* $X \succ Y$ *est vérifiée ssi l'utilisateur préfère le motif* X *à* Y*. En l'absence de préférence ou en cas d'égalité des préférences pour* X *et* Y*, on parle de relation d'indifférence [145].*

2. 3 items sur $|\mathcal{I}| = 6$.
3. 2 transactions couvertes sur 11.

Définition 47 (Relation d'indifférence)**.**
La relation d'indifférence, notée $\sim$*, est une relation binaire qui établit une absence de préférence entre deux motifs* X *et* Y *:*

$$X \sim Y \iff \neg(X \succ Y) \wedge \neg(Y \succ X) \tag{3.2}$$

La relation d'indifférence peut se traduire par l'égalité des importances de X et Y pour l'utilisateur ou par son manque d'intérêt pour ces deux motifs [145].

Ainsi, pour apprendre les préférences Φ de l'utilisateur, nous apprenons une fonction φ permettant de l'approximer : $\varphi \sim \Phi$. Pour cela, nous exploitons les retours de l'utilisateur, notés U, sur un ensemble de motifs noté $\mathcal{X}$ (appelé également *requête*). Ces retours qui peuvent prendre différentes formes :

- un retour ou feedback par rangement où les préférences s'expriment sous forme de rangement entre les motifs ;
- un feedback quantitatif dans lequel l'utilisateur exprime ses préférences sous forme d'évaluation numérique ou symbolique sur les motifs ;
- un feedback implicite où les préférences sur les motifs sont extraites d'une forme implicite de retour.

3.4.1 Feedback par rangement

L'utilisateur peut exprimer ses préférences sous forme de rangement des motifs. Ce feedback par rangement consiste en une comparaison des différentes paires (X, Y) de motifs [145]. On obtient alors la relation d'ordre suivante :

$$\forall X, Y,\ X \succ Y \iff \text{l'utilisateur préfère X à Y.} \tag{3.3}$$

Le rangement des motifs par paires s'effectue alors selon un ordre déterminé par l'intérêt de l'utilisateur. Ainsi, si $\Phi(X) > \Phi(Y)$ alors $X \succ Y$.

Exemple 3.4.1.
Soient les motifs A, AB et BC extraits par la méthode d'extraction de l'étape 1 du framework de fouille interactive de motifs. En présentant ces motifs à l'utilisateur, on pourrait obtenir les rangements suivants des motifs :

$$AB \succ A, AD \succ BC, A \succ BC$$

Rüping [117] propose ainsi d'exprimer les préférences de l'utilisateur sous forme de rangement sur les sous-groupes de motifs. De même, dans [142, 49, 50, 48], un ensemble de motifs est extrait à chaque itération et présenté à l'utilisateur qui effectue un rangement total ou partiel des motifs.

3.4.2 Feedback quantitatif

Le feedback quantitatif consiste à attribuer aux motifs des valeurs numériques ou symboliques traduisant les préférences de l'utilisateur [96]. Formellement, étant donné un motif X et une fonction ϕ^{num}, le feedback quantitatif consiste à attribuer une valeur $\phi^{num}(X)$ à X. Ces valeurs $\phi^{num}(X)$ permettent alors d'exprimer le degré d'importance de chaque motif X. Les motifs recevant les poids les plus élevés sont ainsi ceux qui sont intéressants pour l'utilisateur tandis que ceux qui reçoivent des poids moins élevés ont moins d'intérêt.

Ce modèle est mis en œuvre dans [19] où, à chaque itération, lorsque l'utilisateur préfère le motif X au motif Y ($X \succ Y$), on obtient $\phi^{num}(X) > \phi^{num}(X)$. La fonction $\phi^{num}(X)$ induit donc un ordre de préférences des différents motifs traduisant ainsi les préférences de l'utilisateur pour chacun d'eux.

Motif	Feedback
A	6
AB	10
BC	1

TABLE 3.2 – Exemple de feedback quantitatif.

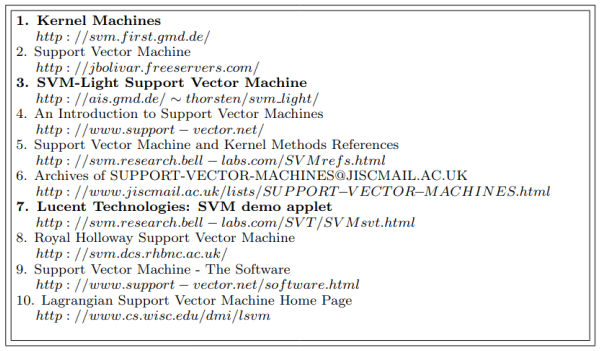


FIGURE 3.2 – Ensemble de liens présentés à l'utilisateur. Les liens en gras sont ceux sur lesquels l'utilisateur a cliqué

Exemple 3.4.2.
En reprenant les motifs de l'exemple 3.4.1, un exemple de feedbacks pourrait être celui de la table 3.2

Les retours de l'utilisateur peuvent également être représentés sous forme binaire sur les motifs. Bhuitan *et al.* l'ont utilisé dans [20]. Il s'agit pour l'utilisateur de donner des avis positifs ou négatifs sur les différents motifs qui lui sont présentés.

Exemple 3.4.3.
Soit l'ensemble de motifs $\mathcal{X}$ suivants : $\mathcal{X} = \{A, AB, BC\}$. Un exemple de feedbacks pourrait être celui de la table 3.3. On obtient alors la fonction ϕ^{bin} suivante :

Motif	Feedbacks	
	Symbolique	Numérique
A	j'aime	0
AB	j'aime	1
BC	je n'aime pas	1

TABLE 3.3 – Exemple de feedback binaire

$$\phi^{bin} : X \mapsto \phi^{bin}(X) \in \{0, 1\}$$

avec $\phi^{bin}(X) = 0$ si l'utilisateur n'aime pas le motif et $\phi^{bin}(X) = 1$ s'il l'aime.

3.4.3 Feedback implicite

Le feedback implicite est une forme de retour de l'utilisateur qui lui permet d'exprimer des relations de préférences entre les motifs sans utiliser de valeurs numériques ou des rangements. La relation de préférences doit alors être extraite à partir du feedback avant d'être exploitée.

Ce modèle est utilisé par *Joachims* qui propose dans [83] un système générique qui extrait les feedbacks de l'utilisateur à partir de ses clics sur un ensemble de liens hypertextes qui lui sont envoyés (voir figure 3.2).

En présentant à l'utilisateur le résultat d'une requête, le système considère en effet que ce dernier cliquera sur les liens qui sont intéressants pour lui. Dans la figure 3.2, les liens (qui peuvent être considérés comme des motifs) intéressants sont X_1, X_3 et X_7. On peut alors considérer les autres motifs $X_{i_{i \notin \{1,3,7\}}}$ comme non intéressants. Les rangements suivants peuvent alors être déduits :

$$X_1 \succ X_4, \qquad X_3 \succ X_2, \qquad X_7 \succ X_5, \qquad X_3 \succ X_9.$$

Cependant, ce système ne permet pas d'obtenir une relation d'ordre sur tous les motifs, ce qui peut limiter la précision de l'apprentissage. En effet, il n'est pas possible de comparer les motifs $X_{i_{i \in \{1,3,7\}}}$ entre eux ni les motifs $X_{i_{i \notin \{1,3,7\}}}$, rendant difficile l'utilisation de ce modèle.

3.4.4 Synthèse

Le modèle de feedback quantitatif présente l'avantage de permettre une représentation simple du degré d'importance de chaque motif pour l'utilisateur. En effet, à partir de l'exemple 3.4.2, on comprend que les motifs préférés par l'utilisateur sont ceux recevant une grande valeur de ϕ^{num}. Cependant, ce modèle est limité par la difficulté de choix des valeurs de ϕ^{num}. En effet, ce modèle nécessite une bonne compréhension de l'échelle des valeurs numériques. La différence entre une échelle de 1 à 5 et une échelle de 1 à 10 peut alors être difficile à établir.

Le modèle de feedback par rangement des motifs présente l'avantage d'affranchir l'utilisateur d'une évaluation numérique des motifs qui lui sont présentés. En effet, il ne nécessite pas une bonne compréhension de l'échelle des valeurs numériques. Cependant, le rangement des motifs nécessite plus d'effort car il passe par la comparaison de plusieurs paires de motifs afin d'établir un rangement.

Le modèle implicite permet de résoudre partiellement cette difficulté en permettant à l'utilisateur d'établir des relations d'ordre sans rangements ou évaluation binaire. Cependant, ce modèle est confronté à la problématique des relations d'indifférences qui peuvent limiter la précision de l'apprentissage à cause de l'absence de préférences.

Ainsi donc, malgré leurs différences, nous pouvons constater que tous ces modèles permettent d'établir une relation de préférence entre les différents motifs X. Ces relations étant déterminées par les préférences de l'utilisateur, alors, il est possible d'établir un rangement des motifs $X \in \mathcal{L}$ avec la fonction Φ (voir l'équation 3.1).

Dans les différentes expérimentations, étant donné les difficultés à disposer d'un utilisateur toujours disponible, il est alors possible d'émuler l'utilisateur avec la fonction Φ. On peut utiliser comme fonction la *surprisingness* [4] [48] ou d'autres méthodes plus complexes pouvant être modélisées sous forme de contraintes [142].

3.5 Apprentissage des préférences de l'utilisateur

Dans la section précédente, nous avons présenté les feedbacks de l'utilisateur. Ces retours permettent à l'utilisateur d'exprimer ses préférences. L'objectif de l'interaction est alors de guider la recherche vers des motifs ou groupes de motifs intéressants pour l'utilisateur. La composante « learn » (ou apprentissage) a ainsi pour objectif d'apprendre les préférences de l'utilisateur à partir des retours exprimés sur les motifs.

3.5.1 Tâches d'apprentissage des préférences

L'apprentissage des préférences s'est principalement développé autour de la problématique du rangement des motifs. En d'autres termes, il s'agit de déterminer comment ordonner les motifs en fonction des préférences de l'utilisateur. Différentes tâches d'apprentissage ont alors été

4. $surp(X) = \max\{sup_{\mathcal{D}}(X) - \prod_{i=1}^{|X|} sup_{\mathcal{D}}(\{i\}), 0\}$.

identifiées :

1. L'apprentissage de label aussi connu sous le terme « label ranking » : la tâche d'apprentissage consiste à attribuer des classes à des motifs selon les préférences de l'utilisateur. Elle s'apparente à la classification et consiste à associer à chaque motif $X \in \mathcal{X}$ une liste ordonnée de labels (ou classes) $c_i \in \mathcal{C}$:

$$X \mapsto c_{\pi(1)} \succ_X c_{\pi(2)} \succ_X \dots \succ_X c_{\pi(k)} \tag{3.4}$$

avec $\mathcal{C} = \{c_1, \dots, c_k\}$ et $\pi(i)$ l'indice du label de rang i dans le rangement associé à X. *Vembu et Gärtnert* ont établi dans [136] un état des lieux de différents algorithmes de label ranking ([68, 57, 82, 37, 98]). Ces algorithmes apprennent notamment des fonctions de poids, des modèles ou des fonctions de rangement.

2. L'apprentissage d'instances aussi appelé « instance ranking » : la tâche consiste ici à apprendre un rangement de motifs labellisés. Il est similaire au *label ranking* mais se différencie par l'apprentissage appris. En effet, l'objectif de l'apprentissage est d'apprendre un rangement des motifs plutôt que celui des labels. Pour cela, l'*instance ranking* nécessite d'utiliser des labels ordonnés $\mathcal{C} = \{c_1, \dots c_k\}$ tels que $c_1 \prec \dots \prec c_k$. Une fonction de rangement est alors apprise permettant de ranger les motifs X en fonction du label qui leur est associé : les motifs ayant les labels les mieux classés dans $\mathcal{C}$ recevront ainsi les meilleurs rangs. Un état des lieux des méthodes d'instance ranking a été réalisé par *Tino Werner* dans [141] où on retrouve des applications dans le traitement d'images et de textes [34, 88, 121].

3. L'apprentissage sur les objets ou « object ranking » est un type d'apprentissage similaire à l'*instance ranking* et qui permet d'apprendre le rangement des motifs. Il se différencie cependant de l'*instance ranking* par l'absence de labels associés aux motifs. L'objectif est ainsi d'apprendre une fonction de rangement φ qui produit un rangement des motifs $X \in \mathcal{X}$ selon l'ordre de préférence de l'utilisateur. Les méthodes d'*object ranking* réalisent l'apprentissage à partir d'un ensemble d'apprentissage de motifs $\mathcal{X}_{train}$ tel que $\mathcal{X}_{train} = \{X_1, \dots X_n\}$ et d'un ensemble de relations de préférences $X_i \succ X_j$ entre des paires de motifs $(X_i, X_j) \in \mathcal{X}_{train} \times \mathcal{X}_{train}$. Kamishima *et al.* ont réalisé dans [87] une étude comparative des méthodes d'object ranking parmi lesquelles on retrouve RankSVM [76, 83].

Les tâches d'apprentissage présentées précédemment peuvent être mis en œuvre à l'aide de techniques plus ou moins similaires [56]. Ces techniques se distinguent alors par leur manière d'aborder l'apprentissage des préférences de l'utilisateur. On retrouve entre autres techniques :

— l'apprentissage d'une fonction de poids ;
— l'apprentissage d'une relation de préférences ;
— l'apprentissage de préférences basé sur les modèles.

Les détails de ces techniques sont donnés dans [56]. Dans la suite de ce chapitre, nous nous intéresserons uniquement aux techniques d'apprentissage appliquées aux motifs et non aux labels. Nous considérerons donc uniquement la tâche *d'object ranking* et nous utiliserons, comme technique, l'apprentissage d'une fonction de poids.

3.5.2 Apprentissage d'une fonction de poids

Il s'agit d'une technique d'apprentissage des préférences de l'utilisateur consistant à apprendre une fonction de poids (ou fonction d'utilité) φ qui approxime la fonction de préférences Φ de l'utilisateur. En effet, la fonction apprise φ doit pouvoir attribuer des poids élevés aux

motifs intéressants pour l'utilisateur et des poids faibles pour les moins intéressants. Autrement dit, la fonction apprise φ doit pouvoir attribuer les meilleurs rangs aux motifs intéressants pour l'utilisateur et les rangs les plus faibles pour les moins intéressants. On pourra ainsi établir un rangement des motifs selon un ordre appris $\hat{\mathcal{R}}$ proche de l'ordre de l'utilisateur $\mathcal{R}^*$.

Définition 48 (Rangements $\mathcal{R}^*$ et $\hat{\mathcal{R}}$)**.**
La fonction de préférences de l'utilisateur Φ permet d'établir un ordre de préférences $\mathcal{R}^*$ des motifs qui s'exprime comme suit :

$$\mathcal{R}^* : X_{\pi(1)} \succ_\Phi \cdots \succ_\Phi X_{\pi(n)} \tag{3.5}$$

avec $\pi(i)$ l'indice du motif associé au rang i.

La fonction de poids φ permet d'établir un ordre de préférence appris $\hat{\mathcal{R}}$ des motifs qui s'exprime comme suit :

$$\hat{\mathcal{R}} : X_{\pi(1)} \succ_\varphi \cdots \succ_\varphi X_{\pi(n)} \tag{3.6}$$

Pour ces deux fonctions, on désignera par $r(X)$ le rang du motif X. On obtient alors :

$$r(X_i) = \pi^{-1}(i) \tag{3.7}$$

Deux ingrédients sont nécessaires pour assurer la qualité de l'apprentissage, la diversité des motifs des différentes requêtes et la précision de la fonction d'apprentissage :

— La diversité peut être obtenue grâce à des mesures d'intérêt (voir la section 1.3.2).
— La précision de l'apprentissage, elle, est assurée grâce à une fonction objectif f^{loss} qui permet de minimiser les discordances entre les deux rangements $\mathcal{R}^*$ et $\hat{\mathcal{R}}$.

On retrouve deux approches d'apprentissage [50] et pour ces deux approches, l'apprentissage se fait sur plusieurs itérations t afin d'améliorer la qualité de φ^t ainsi que le rangement $\hat{\mathcal{R}}^t$ associé. Dans ce qui suit, on notera par $\mathcal{X}^t$, l'ensemble des motifs présentés à l'utilisateur à l'itération t.

A. Apprentissage sur des paires de motifs

Dans cette approche, on commence par transformer les rangements $\mathcal{R}^*$ et $\hat{\mathcal{R}}^t$ en des ensembles de rangements binaires $\mathcal{R}^*_{bin}$ et $\hat{\mathcal{R}}^t_{bin}$ sur les paires de motifs :

$$\mathcal{R}^*_{bin} = \{(X_{\pi(i)} \succ_\Phi X_{\pi(j)}),\ \forall i, j \in [1, n],\ i < j\}$$

et

$$\hat{\mathcal{R}}^t_{bin} = \{(X_{\pi(i)} \succ_{\varphi^t} X_{\pi(j)}),\ \forall i, j \in [1, n],\ i < j\}$$

avec $X_{\pi(i)}, X_{\pi(j)} \in \mathcal{X}^t$.

L'apprentissage permet d'obtenir un rangement $\hat{\mathcal{R}}^t_{bin}$ proche de $\mathcal{R}^*_{bin}$. L'objectif consiste alors à apprendre une fonction φ^t permettant d'obtenir des rangements similaires : $X_i \succ_{\varphi^t} X_j$ ssi $X_i \succ_\Phi X_j$.

La précision de l'apprentissage se mesure en dénombrant les rangements discordants $X_i \succ_{\varphi^t} X_j$. En d'autres termes, la fonction d'apprentissage est d'autant plus précise que le nombre de rangements incorrects $X_i \succ_{\varphi^t} X_j$ est minimal. La tâche d'apprentissage s'apparente ainsi à une classification $(X_i - X_j, +)$ dans laquelle les paires de motifs sont classées selon que le rangement $X_i \succ_{\varphi^t} X_j$ est correct ou pas :

— le rangement $X_i \succ_{\varphi^t} X_j$ est correct lorsque $X_i \succ_\Phi X_j$;
— le rangement $X_i \succ_{\varphi^t} X_j$ est incorrect lorsque $X_j \succ_\Phi X_i$.

Les méthodes adoptant cette approche minimisent une fonction objectif $f^{loss_{pair}}$ qui comptabilisent les paires discordantes au fil des itérations t :

$$\sum_{t=1}^{m} \sum_{X_i, X_j \in \mathcal{X}^t} f^{loss_{pair}}(\mathcal{R}^*, \hat{\mathcal{R}}^t) \tag{3.8}$$

Cette approche a été exploitée par *Joachims* dans [83] où l'auteur a proposé une méthode appelée RankSVM. Cette méthode utilise la régression ordinale pour apprendre la fonction φ^t. D'autres méthodes exploitent également la classification pour l'apprentissage et se distinguent par la fonction objectif utilisée. Ainsi, Burges *et al.* ont développé RankNet [30] qui utilise l'entropie croisée pour mettre à jour la fonction d'apprentissage φ^t. Freund *et al.* propose d'utiliser les réseaux de neurones dans leur méthode RankBoost [53] où la fonction objectif utilise la méthode de descente du gradient.

Dzyuba *et al.* [48] proposent une méthode, dénommée LetSIP, qui exploite la régression logistique pour l'apprentissage. Cette méthode exploite la méthode stochastique de descente des coordonnées (Stochastic Coordinate Descent ou SCD) [122] pour minimiser la fonction objectif et mettre à jour la fonction φ^t.

B. Apprentissage sur des listes de motifs

En exploitant les paires de motifs, l'approche par paire minimise le nombre de discordances, mais n'offre pas de garantie sur le rang de chaque motif pris individuellement.

Exemple 3.5.1.
Soit l'ensemble de motifs $\mathcal{X}^t = \{X_1, X_2, X_3, X_4\}$ *tel que* $X_4 \succ_{\varphi^t} X_1, X_4 \succ_{\varphi^t} X_3, X_1 \succ_{\varphi^t} X_2$. *Ces rangements binaires permettent d'établir que :* $X_4 \succ_{\varphi^t} X_1 \succ_{\varphi^t} X_2$. *Par contre, il n'est pas possible de déterminer le rang de* X_3 *dans* $\mathcal{X}^t$.

Afin de palier à cette limite, il est nécessaire d'exploiter toutes les paires de motifs qu'il est possible de former avec les motifs de $\mathcal{X}^t$. Cela a pour conséquence de rendre l'apprentissage avec RankSVM très coûteux étant donné le nombre potentiellement élevé de paires de motifs.

L'apprentissage sur les listes propose alors d'exploiter l'ensemble des motifs en même temps plutôt que de le subdiviser en plusieurs paires de motifs. Cette approche est utilisée dans ListNet [34] où les auteurs proposent de trouver la fonction φ^t qui minimise la somme suivante :

$$\sum_{t=1}^{m} f^{loss_{list}}(\mathcal{R}^*, \hat{\mathcal{R}}^t) \tag{3.9}$$

avec $\hat{\mathcal{R}}^t$ le rangement des motifs $X_i \in \mathcal{X}^t$ avec la fonction φ^t apprise à l'itération t. L'optimisation de la somme exprimée dans l'équation 3.9 est réalisée avec la méthode de descente du gradient.

C. Synthèse

Ces deux approches d'apprentissage exploitent des poids w_i associés à chaque descripteur F_i des motifs. L'apprentissage permet alors d'améliorer à chaque itération la précision de la fonction d'apprentissage grâce à une minimisation continue de la fonction objectif. Le processus interactif s'arrête après l'atteinte d'un critère portant par exemple sur la valeur de la fonction objectif. L'utilisateur a également la possibilité de mettre fin au processus après un nombre d'itérations T fixé.
Dans la section suivante, nous discuterons des différentes stratégies exploitées pour sélectionner les motifs intéressants.

3.6 Sélection des motifs

La sélection des motifs optimaux est une tâche NP-complète [4]. En effet, la méthode de sélection doit pouvoir suffisamment explorer l'espace de recherche afin de proposer différents motifs à l'utilisateur. Cette démarche permet à l'exploration de couvrir plus de motifs et facilite ainsi l'apprentissage. Par ailleurs, la sélection des motifs doit intégrer les préférences apprises de l'utilisateur afin de sélectionner les motifs intéressants pour lui.

Différentes méthodes existantes exploitent alors des heuristiques combinant une mesure de qualité, de diversité et de densité des motifs :

— la qualité d'un motif traduit à quel point ce motif est proche de ce que l'utilisateur veut ;
— la diversité permet de favoriser les motifs qui n'ont pas été vus par l'utilisateur auparavant ;
— la densité permet d'exploiter la structure du motif.

Qualité. La mesure de qualité peut s'exprimer comme suit :

$$f^{qual}(X) = \mu \times \varphi(X) + (1 - \mu) \times \Phi(X)$$

— $\varphi(X)$ est le score prédit à l'aide des pondérations actuelles apprises ;
— $\Phi(X)$ est le score prédit à l'aide d'une mesure de qualité objective simulant les préférences de l'utilisateur.

L'intuition derrière cette formule est d'approximer une fonction de qualité étant donné deux mesures de qualité φ et Φ en utilisant une interpolation linéaire, où μ est le paramètre d'interpolation qui détermine quel point de données guide la valeur de la qualité.

Diversité. La diversité signifie que l'on présente à l'utilisateur des motifs qu'il n'a pas encore vus auparavant. La diversité est ici mesurée sous forme d'une mesure de distance, par exemple la distance euclidienne :

$$Diversite(X, q) = \min_{Y \in q} dist(X, Y)$$

Densité. La densité signifie que nous présentons à d'utilisateurs des motifs qui sont localisés dans des régions denses de l'ensemble de données :

$$Densite(X, S) = \frac{1}{|S|} \sum_{Y \in S} dist(X, Y)$$

Heuristiques. Plusieurs heuristiques on été proposées permettant d'exploiter la qualité et la diversité d'un motif :

— Maximal Margin Relevance (MMR) [123] : vise à sélectionner un motif de haute qualité et qui est diversifié avec un compromis $\alpha \in [0, 1]$ sur la qualité et la diversité :

$$\text{MMR}(X, S) = \alpha \times f^{qual}(X) + (1 - \alpha) \times Diversite(X, S)$$

— Relevance, Diversity and Density (RDD) [143] : vise à sélectionner un motif dense et de haute qualité qui est également diversifié avec un compromis α sur la qualité et la diversité et un compromis β sur la densité et la diversité :

$$\text{RDD}(X, S) = \alpha \times f^{qual}(X) + \beta \times Density(X, \mathcal{X}) + (1 - \alpha - \beta) \times Diversite(X, S)$$

— Global MMR (GMMR) [46] fonctionne comme MMR mais garantit une diversité entre les motifs X sélectionnés et ceux déjà extraits au cours des itérations précédentes $\mathcal{Q}$:

$$\text{Global-MMR}(X, S) = \alpha \times f^{qual}(X) + (1 - \alpha) \times Diversite\_Globale(X, \mathcal{Q})$$

avec $Diversite\_Globale(X, \mathcal{Q}) = \min_{Y \in \mathcal{Q}} dist(X, Y)$

Algorithm 4: Schéma général de la fouille interactive de motifs

1 **Entrée** : $\mathcal{D}$: jeu de données, $\mathcal{X}$: ensemble de motifs, taille de la requête k, nombre d'itérations T
2 **Sortie** : φ : fonction d'apprentissage ;
3 **begin**
4 ▷ Initialisations
5 $\mathcal{U} \leftarrow \emptyset$; $\varphi^0 \leftarrow$ initial function estimates
6 **for** $t = 1, 2 \dots T$ **do**
7 select a query $\mathcal{X}^t$ based on φ^{t-1} ▷ Mine
8 show query $\mathcal{X}^t$ to the user and get feedback $\mathcal{R}^{*^t}$ ▷ Interact
9 $\mathcal{U} \leftarrow \mathcal{U} \cup \mathcal{R}^{*^t}$
10 compute φ^t based on φ^{t-1} and $\mathcal{U}$ ▷ Learn φ
11 **return** φ;

— Dzyuba *et al.* proposent également dans [48] une nouvelle heuristique permettant d'extraire des motifs intégrant la diversité et les préférences de l'utilisateur via des poids appris lors des précédentes itérations. Pour garantir une forme de diversité, ils proposent d'utiliser Flexics [51] pour extraire les motifs en utilisant des contraintes `XOR` (voir la section 2.5.3). Pour prendre en compte les préférences apprises de l'utilisateur, Flexics utilise la fonction $\varphi_{logistic}$ comme fonction de pondération des motifs.

Dans la section suivante, nous présenterons le schéma général de la fouille interactive de motifs ainsi qu'une instanciation de ce cadre.

3.7 Schéma général de la fouille interactive de motifs et instanciation

3.7.1 Schéma général de la fouille interactive de motifs

Le schéma général de la fouille interactive de motifs est décrit par l'algorithme 4. Après une phase d'initialisation, le processus d'apprentissage se déroule de manière itérative pendant un nombre raisonnable d'itérations T, qui dépend de la tâche à accomplir. Soit Φ la véritable fonction de préférences de l'utilisateur. L'algorithme maintient une approximation φ^t de la véritable fonction Φ, où $t \in [T]$ est l'indice d'itération. À chaque itération, l'algorithme sélectionne une requête $\mathcal{X}^t$ à poser à l'utilisateur (ligne 7), généralement en exploitant l'approximation actuelle φ^t. Un sous-ensemble de k motifs sont présentées à l'utilisateur qui exprime ses préférences sous forme d'un rangement (ligne 8). Le retour de l'utilisateur est ensuite utilisé (éventuellement avec tous les retours reçues jusqu'à présent) pour calculer une nouvelle approximation φ^{t+1} de la fonction Φ (ligne 10). L'algorithme termine après un nombre fini T d'étapes, après quoi, la fonction apprise φ est retournée.

3.7.2 LetSIP

LetSIP est un outil de fouille interactive de motifs qui a été proposé par Dzyuba *et al.* dans [48]. Nous décrivons ci-dessous une instanciation de cet outil selon le schéma général de l'algorithme 4.

Fouille de motifs par échantillonnage. Rappelons que l'objectif principal est de guider la recherche vers des motifs qui sont "subjectivement" intéressants pour l'utilisateur. LetSIP utilise une fonction logistique $\varphi_{logistic}$ paramétrée pour mesurer l'intérêt/la qualité d'un motif

donné :

$$\varphi_{logistic}(X, w, A) = A + \frac{1 - A}{A + e^{-w.X}}$$

où X est un vecteur de descripteurs statiques du motif X, w représente le vecteur des poids des descripteurs et A un paramètre contrôlant la valeur de la fonction logistique $\varphi_{logistic}$: $\varphi_{logistic} \in [A, 1]$. Les motifs sont alors échantillonnés proportionnellement à $\varphi_{logistic}$ en utilisant FLEXICS [51]. Ces pondérations reflètent les contributions des descripteurs à l'intérêt du motif. Notons que ces dernières sont apprises sur la base des retours de l'utilisateur.

Interaction avec l'utilisateur et apprentissage à partir des feedback. LETSIP exploite un modèle de feedback par rangement, où l'utilisateur est invité à fournir un ordre total sur un (petit) nombre de motifs de taille k selon un ordre décroissant de ses préférences. Pour émuler les préférences de l'utilisateur, Dzyuba *et al.* proposent différentes fonctions comme par exemple la *surprinsingness* ou encore la mesure de χ^2. Pour apprendre la fonction logistique, une réduction du classement des motifs à une classification binaire des vecteurs de différence des descripteurs des motifs est appliquée en utilisant la méthode de stochastique de descente des coordonnées (SCD) [122] pour minimiser la perte logistique (voir la section 3.5).

Sélection des motifs à présenter à l'utilisateur. Premièrement, un ensemble de ℓ [5] meilleurs motifs issus de l'itération précédente sont conservés et les $(k - \ell)$ motifs restants sont ensuite échantillonnés avec FLEXICS. Cela permet à l'utilisateur de relier les requêtes les unes aux autres. Deuxièmement, pour biaiser l'échantillonnage vers des motifs de meilleure qualité, Dzyuba *et al.* ont proposé une stratégie, dénotée $Top(m)$, qui consiste à sélectionner les m motifs de meilleure qualité dans une cellule.

3.8 Conclusions

Dans ce chapitre, nous avons présenté la fouille interactive de motifs qui permet d'introduire l'utilisateur dans le processus de fouille en prenant en compte ses préférences pour la recherche de motifs intéressants. Le processus de fouille se déroule de manière itérative et permet la découverte de motifs de plus en plus intéressants. La fouille interactive de motifs s'apparente alors à une tâche d'optimisation au cours de laquelle une fonction d'apprentissage des préférences φ est optimisée afin d'extraire des motifs pertinents.

Différentes approches existent dans la littérature et qui proposent de prendre en compte les préférences de l'utilisateur. Cependant, ces méthodes exploitent des descripteurs statiques qui n'évoluent pas au cours de l'apprentissage et ne permettent pas de décrire de façon dynamique les motifs. Par ailleurs, la diversité, qui est nécessaire pour mieux connaître les préférences de l'utilisateur sur tous les descripteurs, n'est pas toujours assurée de façon efficace.

Dans le chapitre 6, nous proposerons une nouvelle approche de fouille interactive de motifs permettant d'exploiter des descripteurs dynamiques qui évoluent au cours des itérations. Par ailleurs, nous exploiterons notre nouvelle méthode d'échantillonnage, SDIVJAX, qui sera présentée au chapitre 4, pour extraire les motifs diversifiés.

5. Ce paramètre est dénommé "query retention" dans LETSIP.

Troisième partie

Contributions : diversité, échantillonnage et apprentissage

CHAPITRE 4

Extraction de motifs diversifiés

Contents

4.1 Introduction 73
4.2 Modèles FullCP pour la contrainte de diversité 75
4.2.1 Encodage PPC pour la contrainte linéaire de diversité 75
4.2.2 Contraintes linéaires pour la diversification des solutions 76
4.2.3 Procédure de recherche 77
4.3 Relaxations anti-monotones de la contrainte de diversité 77
4.3.1 Motivation 78
4.3.2 Problème relaxé 79
4.3.3 Borne Inférieure de Jaccard 80
4.3.4 Borne supérieure de Jaccard 82
4.4 La contrainte globale ClosedDiversity 84
4.5 Booster la recherche de motifs diversifiés 86
4.5.1 Motif Témoin Positif 86
4.5.2 MINCOV : une heuristique de branchement basée sur les fréquences estimées . 87
4.5.3 WITNESS : une heuristique de branchement basée sur les motifs témoins positifs 88
4.5.4 Stratégies d'exploration des sous-arbres témoins 88
4.5.5 Évitement des motifs diversités faux positifs 88
4.6 Échantillonnage de motifs diversifiés 89
4.6.1 Adaptations de l'algorithme WEIGHTGEN 89
4.6.2 Oracles de sélection des motifs diversifiés 90
4.6.3 Estimation et contrôle du nombre de contraintes `XOR` 91
4.6.4 Résolution et propagation des contraintes XOR 92
4.7 Conclusions 93

4.1 Introduction

Les approches décrites dans le chapitre 1 ont été formulées de manière à ce que les ensembles solutions soient évaluées par rapport à tous les ensembles de motifs de la théorie. Or, l'extraction des motifs peut se faire selon une autre approche. L'utilisateur pourrait en effet être intéressé par des motifs dont l'intérêt est évalué par rapport à un sous-ensemble d'autres motifs préalablement extraits. Pour cela, l'extraction des motifs se déroule comme suit :

1. extraire un motif X de la théorie $\mathcal{T}h(\mathcal{L}_{\mathcal{I}}, \mathcal{D}, C)$;

2. évaluer $\mathbb{X} \cup X$ avec la contrainte $\mathbb{C}$ et ajouter X à l'ensemble, initialement vide, de motifs $\mathbb{X}$ lorsque la contrainte $\mathbb{C}$ est satisfaite ou s'arrêter sinon ;
3. effectuer des traitements sur le jeu de données $\mathcal{D}$;
4. retourner à l'étape 1.

Nous proposons d'exploiter cette approche dans ce chapitre pour trouver des motifs diversifiés : étant donné un historique $\mathcal{H}$ de motifs, nous recherchons les motifs X vérifiant la contrainte $\mathbb{C}(\mathcal{H})$: $\textsc{Jaccard}(X, H) < J_{max} \forall H \in \mathcal{H}$, avec J_{max} un seuil de diversité. Lorsque la contrainte est respectée, le motif X est ajouté à l'historique et l'exploration se poursuit. S'il n'existe pas de motifs respectant la contrainte, l'exploration s'arrête.

Cette approche à l'avantage d'être exhaustive, étant donné que tout l'espace de recherche est exploré et tous les motifs évalués. Par ailleurs, elle est plus performante car l'évaluation des motifs est moins complexe.

Nous avons choisi d'utiliser l'indice de Jaccard comme mesure d'évaluation de la redondance. Un indice de Jaccard faible indique ainsi une faible similarité entre les couvertures des motifs. Cependant, un indice de Jaccard élevé est synonyme de beaucoup de redondances dans leurs couvertures. L'indice de Jaccard peut être utilisé avec un seuil J_{max} afin de limiter la redondance entre les différentes paires de motifs.

Définition 49 (Contrainte de Jaccard maximum)**.**
Soit P et Q deux motifs. Étant donné l'indice de Jaccard Jac ainsi qu'un seuil de redondance maximum J_{max}, on dit que P et Q sont diversifiés entre eux si et seulement si $Jac(P, Q) \leq J_{max}$. Nous noterons cette contrainte c_{Jac}.

Notre objectif est d'exploiter la contrainte de Jaccard maximum c_{Jac} pendant l'exploration de l'espace de recherche afin d'élaguer les motifs non diversifiés. Pour cela, nous proposons de maintenir un *historique* $\mathcal{H}$ des différents motifs extraits afin de garantir que tous les nouveaux motifs soient diversifiés par rapport à ceux déjà extraits. Étant donné que l'indice de Jaccard évalue la similarité entre des paires de motifs, nous proposons d'agréger les valeurs de diversité des différentes paires de motifs. Le problème $\textsc{MaxDiverseKSet}(k)$ (voir définition 27) peut alors être approximé comme suit :

Définition 50 (L'indice de Jaccard pour la diversité)**.**
Étant donné $\mathcal{H} = \{H_1, \ldots, H_k\}$ un historique de motifs diversifiés entre eux, la mesure de la redondance Jac et le seuil de redondance maximum J_{max}, l'extraction de motifs diversifiés par rapport à $\mathcal{H}$ consiste à trouver les motifs P tels que

$$\forall H \in \mathcal{H}, Jac(P, H) \leq J_{max}$$

Ainsi, pour être diversifiés, les motifs P devront avoir des indices de Jaccard par rapport aux motifs H de $\mathcal{H}$ inférieurs à J_{max}.

Dans ce chapitre, nous présenterons deux approches exploitant la PPC pour l'extraction de motifs diversifiés. Nous commençons par présenter notre première contribution, un premier encodage de la contrainte de diversité (i.e. contrainte de Jaccard maximum) qui exploite des contraintes linéaires. Cet encodage permet de garantir que tous les motifs solutions ont un indice de Jaccard inférieur à seuil défini. Ensuite, nous détaillons notre seconde contribution, une nouvelle contrainte globale ClosedDiversity pour l'extraction de motifs fréquents, fermés et diversifiés. ClosedDiversity exploite une relaxation de la contrainte c_{Jac} pour dériver des propriétés de monotonie, permettant d'élaguer efficacement les motifs non diversifiés pendant l'exploration de l'espace de recherche. Enfin, nous proposerons d'exploiter ClosedDiversity dans un cadre d'échantillonnage afin de permettre le tirage d'ensembles de motifs diversifiés.

4.2 Modèles FullCP pour la contrainte de diversité

L'un des principaux avantages des solveurs de programmation par contraintes est qu'ils ne sont pas limités aux principes de « monotonie » et d'« anti-monotonie » exploités dans de la fouille classique de motifs. Pour profiter de cet avantage, nous proposons un premier encodage de la contrainte de diversité (voir la section 4.2.1). Notre idée est de formuler cette contrainte sous la forme d'une contrainte linéaire obtenue en décomposant l'indice de Jaccard en opérations simples sur des ensembles utilisant des variables booléennes intermédiaires représentant les couvertures des motifs. À la section 4.2.2, nous introduisons deux modèles PPC pour l'extraction de motifs fréquents, fermés et diversifiés, exploitant cet encodage.

4.2.1 Encodage PPC pour la contrainte linéaire de diversité

Soit H un motif de l'historique $\mathcal{H}$ et P un motif. Soit la contrainte de diversité $Jac(H,P) \leq J_{max}$ sur les deux motifs H et P. Pour évaluer l'expression $|\mathcal{V}_{\mathcal{D}}(H) \cap \mathcal{V}_{\mathcal{D}}(P)|$, il nécessaire de calculer l'ensemble $\mathcal{V}_{\mathcal{D}}(H) \cap \mathcal{V}_{\mathcal{D}}(P)$, puis de sommer les variables qui le représente. L'expression $|\mathcal{V}_{\mathcal{D}}(H) \cup \mathcal{V}_{\mathcal{D}}(P)|$ peut être réécrite comme suit :

$$|\mathcal{V}_{\mathcal{D}}(H) \cup \mathcal{V}_{\mathcal{D}}(P)| = |\mathcal{V}_{\mathcal{D}}(H)| + |\mathcal{V}_{\mathcal{D}}(P)| - |\mathcal{V}_{\mathcal{D}}(H) \cap \mathcal{V}_{\mathcal{D}}(P)|.$$

Nous modélisons la couverture $\mathcal{V}_{\mathcal{D}}(P)$ (resp. $\mathcal{V}_{\mathcal{D}}(H)$) par un vecteur Y^P (resp. Y^H) de m variables booléennes transactionnelles, avec $(Y_j^P = 1)$ (resp. $(Y_j^H = 1)$) ssi $P \subseteq t_j$ (resp. $H \subseteq t_j$). Maintenant, l'expression $|\mathcal{V}_{\mathcal{D}}(H) \cap \mathcal{V}_{\mathcal{D}}(P)|$ peut être réécrite en comptant le nombre de transactions pour lesquelles les deux variables Y_j^P et Y_j^H sont égales à 1. Cela peut être exprimé de la manière suivante : $\sum_{j \in \mathcal{T}} Y_j^P \times Y_j^H$. Comme $\mathcal{V}_{\mathcal{D}}(H)$ est un ensemble de constantes, alors les Y_j^H le sont aussi. La proposition 2 donne l'encodage de la contrainte de diversité.

Proposition 2 (Encodage PPC de la contrainte de Jaccard maximum).
Soit un motif H de l'historique $\mathcal{H}$, et un motif P. Soit Y^P et Y^H deux vecteurs de variables booléennes associées à $\mathcal{V}_{\mathcal{D}}(P)$ et $\mathcal{V}_{\mathcal{D}}(H)$ respectivement. Nous avons alors :

$$Jac(H,P) \leq J_{max} \quad \Leftrightarrow \sum_{j \in \mathcal{T}} \alpha_j \times Y_j^P \leq \beta \tag{4.1}$$

avec,

$$\alpha_j = Y_j^H - J_{max} \times (1 - Y_j^H)$$
$$\beta = J_{max} \times \sum_{j \in \mathcal{T}} Y_j^H$$

Preuve 1. *$\forall H \in \mathcal{H}$, on a :*

$$\begin{aligned} Jac(H,P) \leq J_{max} &\Leftrightarrow |\mathcal{V}_{\mathcal{D}}(H) \cap \mathcal{V}_{\mathcal{D}}(P)| \leq J_{max} \times |\mathcal{V}_{\mathcal{D}}(H) \cup \mathcal{V}_{\mathcal{D}}(P)| \\ &\Leftrightarrow |\mathcal{V}_{\mathcal{D}}(H) \cap \mathcal{V}_{\mathcal{D}}(P)| - J_{max} \times |\mathcal{V}_{\mathcal{D}}(H) \cup \mathcal{V}_{\mathcal{D}}(P)| \leq 0 \end{aligned}$$

Comme

$$|\mathcal{V}_{\mathcal{D}}(H) \cap \mathcal{V}_{\mathcal{D}}(P)| = \sum_{j \in \mathcal{T}} Y_j^P \times Y_j^H$$
$$|\mathcal{V}_{\mathcal{D}}(P)| = \sum_{j \in \mathcal{T}} Y_j^P$$
$$|\mathcal{V}_{\mathcal{D}}(H)| = \sum_{j \in \mathcal{T}} Y_j^H$$

Nous obtenons

$$
\begin{aligned}
Jac(H,P) \leq J_{max} &\Leftrightarrow \sum_{j\in\mathcal{T}} y_j^P \times Y_j^H - J_{max} \times \left(\sum_{j\in\mathcal{T}} Y_j^P + \sum_{j\in\mathcal{T}} Y_j^H - \sum_{j\in\mathcal{T}} Y_j^P \times Y_j^H \right) \leq 0 \\
&\Leftrightarrow \sum_{j\in\mathcal{T}} Y_j^P \times Y_j^H - J_{max} \times \left(\sum_{j\in\mathcal{T}} (1 - Y_j^H) \times Y_j^P + \sum_{j\in\mathcal{T}} Y_j^H \right) \leq 0 \\
&\Leftrightarrow \sum_{j\in\mathcal{T}} \left(Y_j^H - J_{max} \times \left(1 - Y_j^H\right) \right) \times Y_j^P - J_{max} \times \sum_{j\in\mathcal{T}} Y_j^H \leq 0 \\
&\Leftrightarrow \sum_{j\in\mathcal{T}} \alpha_j \times Y_j^P \leq \beta
\end{aligned}
$$

où

$$
\begin{aligned}
\alpha_j &= Y_j^H - J_{max} \times (1 - Y_j^H) \\
\beta &= J_{max} \times \sum_{j\in\mathcal{T}} Y_j^H
\end{aligned}
$$

4.2.2 Contraintes linéaires pour la diversification des solutions

Dans cette section, nous présenterons les deux modèles PPC exploités pour l'extraction de motifs fermés et diversifiés. Le premier modèle, noté FullCP-1, est une extension du modèle réifié présenté à la section 2.5.1 auquel on ajoute la contrainte de diversité (cf. équation 4.1) à chaque fois qu'un nouveau motif est ajouté à l'historique de solutions $\mathcal{H}$. Le deuxième modèle modèle, noté FullCP-2, combine une extension de la contrainte globale ClosedPatterns (section 2.5.2) avec la contrainte de diversité. Ces deux modèles garantissent un calcul exact de la diversité des motifs et ne nécessitent aucune relaxation.

Modèle FullCP-1

Soit $\mathcal{H} = \{H_1, \ldots, H_k\}$ l'historique des k motifs diversifiés extraits et J_{max} le seuil de redondance maximum entre les motifs. Le modèle FullCP-1 utilise un vecteur X de n variables booléennes, tel que X_i représente la présence de l'item i dans le motif P. Il utilise également un vecteur Y^P de m variables booléennes, où Y_j^P la présence/absence du motif P dans la transaction t_j ($j \in \mathcal{T}$). Enfin, un vecteur de constante Y^H est utilisé pour représenter la couverture $\mathcal{V}_{\mathcal{D}}(H)$ du motif $H \in \mathcal{H}$. Quatre contraintes sont alors exploitées par FullCP-1 :

$$
\text{FullCP-1}_{\mathcal{D},\theta,J_{max}}(X, Y^P, Y^H, \mathcal{H}) \equiv \begin{cases}
\forall j \in \mathcal{T} : (Y_j^P = 1) \Leftrightarrow \sum_{i\in\mathcal{I}} X_i \times (1 - \mathcal{D}_{ti}) = 0 & (1) \\
\forall i \in \mathcal{I} : (X_i = 1) \Leftrightarrow \sum_{j\in\mathcal{T}} Y_j^P \times (1 - \mathcal{D}_{ti}) = 0 & (2) \\
\sum_{j\in\mathcal{T}} Y_j^P \geq \theta & (3) \\
\forall H \in \mathcal{H} : \sum_{j\in\mathcal{T}} \alpha_j \times Y_j^P \leq \beta & (4)
\end{cases}
$$

Le rôle de chaque est contrainte est décrit ci-dessous :

- (1) encode la couverture de X (i.e. du motif P) ;
- (2) assure que X est fermé ;
- (3) assure que X est fréquent par rapport au seuil de fréquence θ ;
- (4) est la contrainte de diversité permettant d'assurer que $Jac(H, X) \leq J_{max}$, pour tout $H \in \mathcal{H}$.

Modèle FullCP-2

Le modèle FullCP-1 présenté précédemment n'est pas efficace et ne passe pas à l'échelle principalement à cause du grand nombres de contraintes réifiées utilisées. Nous proposons alors

un nouveau modèle qui étend la contrainte globale ClosedPatterns [97] en introduisant un nouveau vecteur Y de variables booléennes $(Y_1, \ldots, Y_{|\mathcal{T}|})$ afin de représenter la couverture des différents motifs. Y_j représente alors la présence/absence du motif courant dans la transaction t_j. La nouvelle contrainte, dénommée $\text{ClosedCover}_{\mathcal{D},\theta}(X, Y)$, permet ainsi de remplacer les contraintes (1-3) dans le modèle FullCP-2 comme suit :

$$\text{FullCP-2}_{\mathcal{D},\theta,J_{max}}(X, Y^P, Y^H, \mathcal{H}) \equiv \begin{cases} \text{ClosedCover}_{\mathcal{D},\theta}(X, Y^P) \\ \forall H \in \mathcal{H} : \sum_{j \in \mathcal{T}} \alpha_j \times Y_j^P \leq \beta \end{cases}$$

La contrainte globale ClosedCover est définie comme suit :

Définition 51 (ClosedCover)**.**
Soit X et Y deux vecteur de variables booléennes, θ un seuil de fréquence minimum et $\mathcal{D}$ un jeu de données. Étant donné une instanciation partielle des variables X et Y, avec $X^+ = \{i \in \mathcal{I} \mid X_i = 1\}$, la contrainte $\text{ClosedCover}_{\mathcal{D},\theta}(X, Y)$ est vérifiée ssi $sup_{\mathcal{D}}(X^+) \geq \theta$, X^+ est fermée, et si $\mathcal{V}_{\mathcal{D}}(X^+) = Y^+$, avec $Y^+ = \{j \in \mathcal{T} \mid Y_j = 1\}$.

Filtrage Le propagateur utilisé par ClosedCover exploite les règles de filtrage de ClosedPatterns ainsi que d'autres règles portant sur le vecteur de variables booléennes Y. Ainsi, étant donné une instanciation partielle X, pour chaque $j \in Y^*$, les règles suivantes permettent de filtrer les valeurs inconsistantes de $dom(Y_j)$:

— $1 \notin dom(Y_j)$ ssi : $j \notin \mathcal{V}_{\mathcal{D}}(X^+)$;
— $0 \notin dom(Y_j)$ ssi : $j \in (\mathcal{V}_{\mathcal{D}}(X^+) \cap \mathcal{V}_{\mathcal{D}}(X^*))$

La première règle permet de filtrer les transaction non couvertes par le motif courant X^+, tandis que la deuxième règle évite d'élaguer les transactions couvertes à la fois par le motif courant X^+ et par les items libres (non encore instanciés) X^*. Comme la contrainte de diversité pourrait modifier les domaines des variable Y, pouvant conduire à des inconsistance avec la couverture des motifs dans ClosedCover, nous ajoutons un test de consistance avant chaque propagation et qui retourne un échec lorsque la condition suivante est vérifiée : $\exists j \in \mathcal{T} : dom(Y_j) = 0 \wedge j \in \mathcal{V}_{\mathcal{D}}(X^+ \cup X^*)$.

4.2.3 Procédure de recherche

Les deux modèles FullCP présentés précédemment sont exploités au sein de la procédure de recherche suivante. Au départ, l'historique $\mathcal{H}$ est vide. Il est ensuite mis à jour de façon incrémentale par l'ajout de motifs fréquents, fermés et diversifiés rencontrés pendant la recherche. Premièrement, les contraintes du modèle sont initialisées soit par les contraintes $(1) - (3)$ du modèle FullCP-1, soit par la contrainte globale ClosedCover du modèle FullCP-2. Ensuite, à chaque fois qu'une nouvelle solution H est trouvée, elle est ajoutée à l'historique $\mathcal{H}$ et une contrainte dynamique (voir l'équation 4.1) est postée entre les motifs H et les futurs motifs X^+. Soit H_i la solution courante ajoutée à l'historique $\mathcal{H}$ tel que $\mathcal{H} = \{H_1, \ldots, H_i\}$, nous avons alors $Jac(H_j, X^+) \leq J_{max}, \forall H_j \in \mathcal{H}$ t.q. $j \leq i$. Ce processus s'arrête lorsqu'il n'existe aucune solution satisfaisant les contraintes du modèle.

Les deux modèles FullCP-1 et FullCP-2 nécessitent l'utilisation de $|\mathcal{H}|$ contraintes linéaires dynamiques postées pendant la recherche, en plus de leur contraintes initiales. La gestion de ce grand nombre de contraintes rend alors ces deux modèles très coûteux pour les solveurs PPC. Par ailleurs, ces contraintes qui peuvent impliquer un grand nombre de variables ($|\mathcal{T}|$ variables)) sont caractérisées par une propagation particulièrement faible.

4.3 Relaxations anti-monotones de la contrainte de diversité

Dans cette section, nous présentons notre seconde contribution pour l'encodage de la contrainte de diversité. Cet encodage, qui exploite deux relaxations de la contrainte de diversité, permet de

dériver des propriétés de monotonie à partir de ces relaxations pour réduire efficacement l'espace de recherche.

4.3.1 Motivation

La figure 4.1 présente, sous forme d'un treillis, les motifs extraits à partir d'un jeu de données contenant 5 items et 100 transactions, avec un seuil de fréquence minimum $\theta = 0.07\%$ et un seuil de redondance maximum $J_{max} = 0.19$. Dans cette figure, chaque nœud représente un motif avec les informations suivantes : les items du motif, son support et son indice de Jaccard par rapport à l'historique $\mathcal{H} = \{BE\}$. Les arêtes entre les nœuds représentent les relations de spécialisation ou de généralisation entre les différents nœuds. Ainsi, le motif BCD, qui est une une spécialisation des motifs BC, BD et CD, a un support de 16, un indice de jaccard de 0.25.

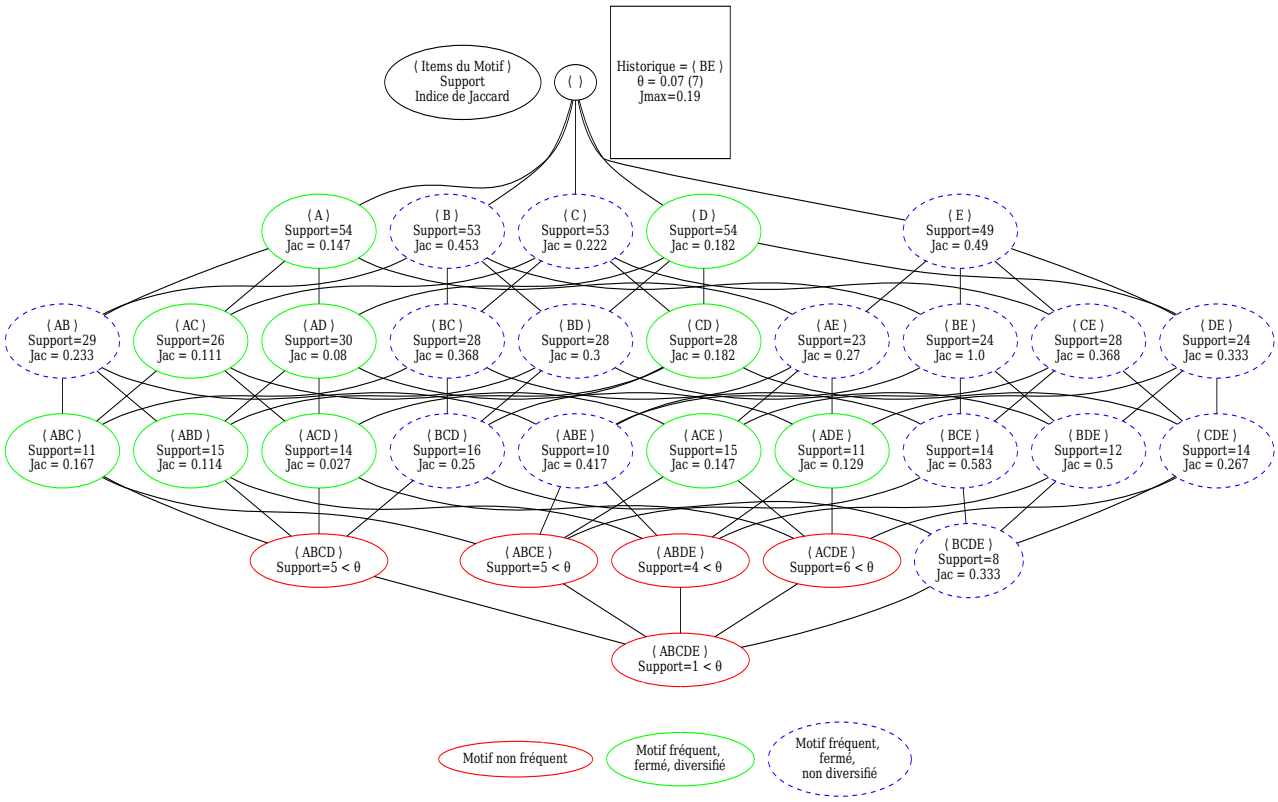


FIGURE 4.1 – Treillis des motifs fermés et fréquents ($\theta = 7$)

Comme le stipule la proposition 3, l'indice de Jaccard n'est ni monotone ni anti-monotone. Par conséquent, il n'est pas possible d'exploiter l'indice de Jaccard dans la contrainte de diversité pour filtrer les motifs non diversifiés.

Proposition 3.
Soient P, Q et P' trois motifs tels que $P \subset P'$. Alors $Jac(P, Q)$ peut être égal, plus grand ou plus petit que $Jac(P', Q)$.

Pour illustrer la proposition 3, considérons les motifs C, CD et BCD de la figure 4.1. Le motif C n'est pas diversifié par rapport à BE car $Jac(C, BE) = 0.222 \geq J_{max}$. Or, le motif CD est diversifié car $Jac(CD, BE) = 0.182 \leq J_{max}$. Par conséquent, la contrainte c_{Jac} n'est donc pas monotone. Elle n'est pas non plus anti-monotone ($Jac(A, BE) < J_{max}$ mais $Jac(AE, BE) > J_{max}$).

Sur la base de la proposition ci-dessus, il n'est donc pas possible d'effectuer un élagage direct des motifs non diversifiés car on pourrait supprimer d'autres motifs qui peuvent être diversifiés.

Pour cela, nous proposons d'approximer la théorie de la contrainte de Jaccard maximum c_{Jac} par une collection plus grande correspondant à l'espace des solutions de sa *relaxation* c^r_{Jac} : $Th(c_{Jac}) \subseteq Th(c^r_{Jac})$.

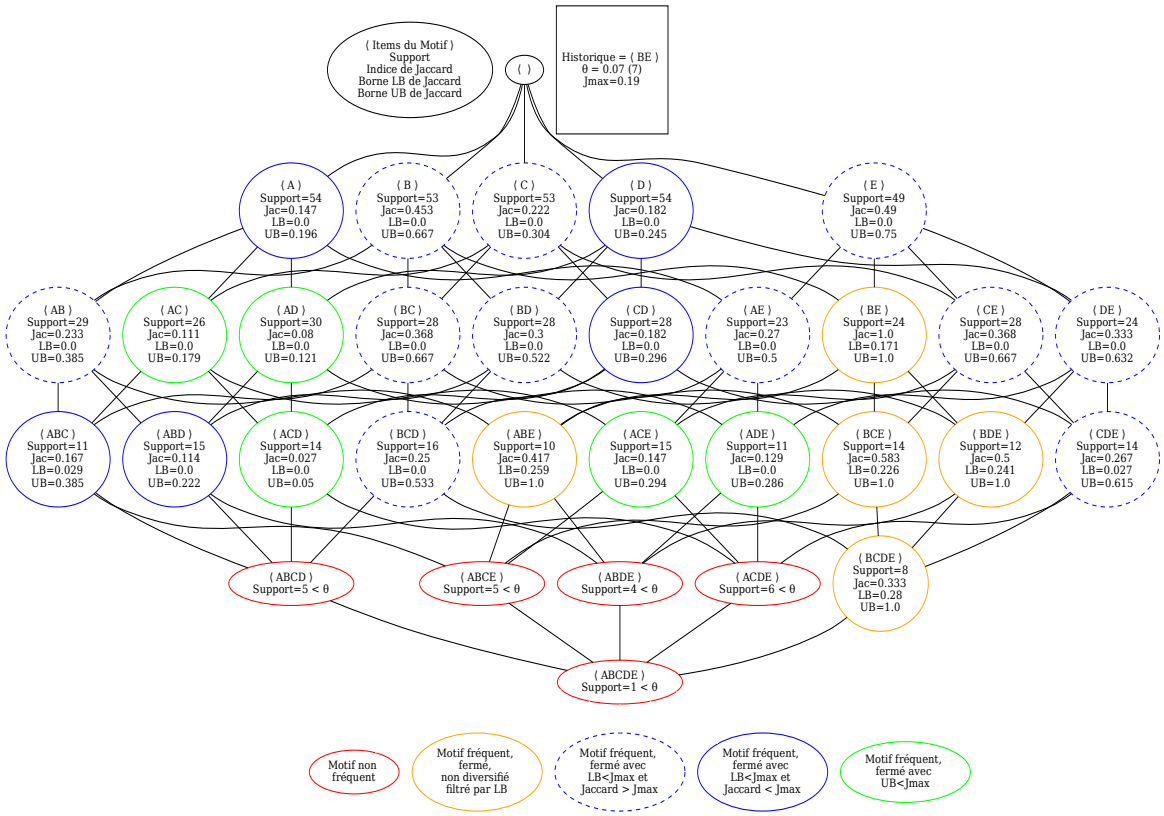


FIGURE 4.2 – Treillis des motifs fermés et fréquents ($\theta = 7$)

Nous proposerons dans cette section deux relaxations anti-monotones de l'indice de Jaccard. Ces relaxations exploitent des bornes de l'indice de Jaccard afin d'approximer la théorie $Th(c_{Jac})$:

— une première relaxation exploitant une borne inférieure de l'indice de Jaccard afin de filtrer les motifs non diversifiés ;
— une deuxième relaxation exploitant une borne supérieure de l'indice de Jaccard utilisée pour certifier la diversité des motifs.

4.3.2 Problème relaxé

Comme indiqué précédemment, nous proposons une approximation de la théorie de la contrainte de Jaccard c_{Jac} par un ensemble plus grand correspondant à l'espace des solutions de sa *relaxation* c^r_{Jac} : $Th(c_{Jac}) \subseteq Th(c^r_{Jac})$. L'attrait de cette approche est que nous pouvons bénéficier de propriétés de monotonie appropriées de la contrainte relaxée qui peuvent être efficacement exploitées pour la réduction de l'espace de recherche. Par ailleurs, une telle approche nous permet de préserver les solutions [8] de c_{Jac} puisque l'élagage induit par la relaxation ne supprime pas les motifs satisfaisant c_{Jac}.

Dans ce qui suit, nous formalisons cette intuition en définissant deux relaxations exploitant une *borne supérieure* $\overline{c}_{Jac}$ et une *borne inférieure* $\underline{c}_{Jac}$ de l'indice de Jaccard. Nous démontrons par la suite que $\underline{c}_{Jac}$ est monotone, ce qui nous permet de définir une règle de filtrage exploitant la monotonie de la borne inférieure. Par ailleurs, nous proposons une nouvelle heuristique de

choix de variables exploitant l'anti-monotonie de la borne supérieure. Cette heuristique permet d'accélérer la recherche de motifs diversifiés (voir la section 4.5.3).

Définition 52 (Problème relaxé)**.**
Étant donné un historique $\mathcal{H}$ de k motifs diversifiés, un seuil de Jaccard maximum J_{max}, une borne inférieure LB_J et une borne supérieure UB_J sur l'indice de Jaccard, le problème relaxé consiste à extraire des motifs candidats P tels que $\forall H \in \mathcal{H}$, $LB_J(P,H) \leq J_{max}$. Lorsque $UB_J(P,H) \leq J_{max}$ pour tout $H \in \mathcal{H}$, alors, la contrainte de Jaccard est satisfaite.

4.3.3 Borne Inférieure de Jaccard

Comme nous l'avons mentionné plus haut, la contrainte de Jaccard maximum ne permet pas d'identifier les motifs témoins de manière simple pendant l'exploration. Nous définissons alors une borne inférieure qui nous permettra d'utiliser un motif comme témoin négatif pour toutes ses spécialisations. Pour cela, nous commençons par définir la notion de *couverture résiduelle* d'un motif.

Définition 53 (Couverture résiduelle)**.**
Soit P et Q deux motifs. La couverture résiduelle de P par rapport à Q est définie comme suit : $\mathcal{V}_Q^{pr}(P) = \mathcal{V}_\mathcal{D}(P) \backslash \{\mathcal{V}_\mathcal{D}(P) \cap \mathcal{V}_\mathcal{D}(Q)\}$.

Nous pouvons dériver une borne inférieure de l'indice de Jaccard entre un motif H et la spécialisation du motif P en considérant le cas où la couverture de l'intersection de ces deux motifs soit la plus petite possible, tandis que la couverture résiduelle de chaque motif reste la plus grande possible. La valeur de Jaccard la plus petite est celle qui réduit le numérateur de l'indice de Jaccard à 0, ce qui n'est pas toujours possible avec une contrainte de fréquence. Le dénominateur, d'autre part, se compose de $|\mathcal{V}_\mathcal{D}(H)|$ (qui ne change pas) et d'une partie de la couverture de P qui ne couvre pas H, i.e. $\mathcal{V}_H^{pr}(P)$.

Le lemme 1 introduit une propriété sur la couverture résiduelle d'un motif qui sera utilisée dans les différentes preuves à venir.

Lemme 1 (Couverture résiduelle)**.**
Considérons un motif H de l'historique $\mathcal{H}$. Soit P et Q deux motifs. Si P est un sur-motif de Q, alors $|\mathcal{V}_H^{pr}(P)| \leq |\mathcal{V}_H^{pr}(Q)|$.

Preuve 2. *L'ingrédient essentiel de cette preuve est donné par*

$$\mathcal{V}_H^{pr}(Q) = \mathcal{V}_\mathcal{D}(Q) \backslash \{\mathcal{V}_\mathcal{D}(Q) \cap \mathcal{V}_\mathcal{D}(H)\} \Rightarrow \mathcal{V}_H^{pr}(Q) \cap \{\mathcal{V}_\mathcal{D}(Q) \cap \mathcal{V}_\mathcal{D}(H)\} = \emptyset.$$

Pour $Q \subseteq P$ on a $\mathcal{V}_\mathcal{D}(P) \subseteq \mathcal{V}_\mathcal{D}(Q)$. Nous pouvons alors distinguer quatre cas :

1. $\{\mathcal{V}_\mathcal{D}(Q) \backslash \mathcal{V}_\mathcal{D}(P)\} \cap \{\mathcal{V}_\mathcal{D}(Q) \cap \mathcal{V}_\mathcal{D}(H)\} = \emptyset, \{\mathcal{V}_\mathcal{D}(Q) \backslash \mathcal{V}_\mathcal{D}(P)\} \cap \mathcal{V}_H^{pr}(Q) = \emptyset \Rightarrow \mathcal{V}_H^{pr}(P) = \mathcal{V}_H^{pr}(Q)$
2. $\{\mathcal{V}_\mathcal{D}(Q) \backslash \mathcal{V}_\mathcal{D}(P)\} \cap \{\mathcal{V}_\mathcal{D}(Q) \cap \mathcal{V}_\mathcal{D}(H)\} \neq \emptyset, \{\mathcal{V}_\mathcal{D}(Q) \backslash \mathcal{V}_\mathcal{D}(P)\} \cap \mathcal{V}_H^{pr}(Q) = \emptyset \Rightarrow \mathcal{V}_H^{pr}(P) = \mathcal{V}_H^{pr}(Q)$
3. $\{\mathcal{V}_\mathcal{D}(Q) \backslash \mathcal{V}_\mathcal{D}(P)\} \cap \{\mathcal{V}_\mathcal{D}(Q) \cap \mathcal{V}_\mathcal{D}(H)\} = \emptyset, \{\mathcal{V}_\mathcal{D}(Q) \backslash \mathcal{V}_\mathcal{D}(P)\} \cap \mathcal{V}_H^{pr}(Q) \neq \emptyset \Rightarrow \mathcal{V}_H^{pr}(P) \subset \mathcal{V}_H^{pr}(Q)$
4. $\{\mathcal{V}_\mathcal{D}(Q) \backslash \mathcal{V}_\mathcal{D}(P)\} \cap \{\mathcal{V}_\mathcal{D}(Q) \cap \mathcal{V}_\mathcal{D}(H)\} \neq \emptyset, \{\mathcal{V}_\mathcal{D}(Q) \backslash \mathcal{V}_\mathcal{D}(P)\} \cap \mathcal{V}_H^{pr}(Q) \neq \emptyset \Rightarrow \mathcal{V}_H^{pr}(P) \subset \mathcal{V}_H^{pr}(Q)$

Ces différents cas nous permettent ainsi de conclure que $\mathcal{V}_H^{pr}(P) \subseteq \mathcal{V}_H^{pr}(Q)$.

Comme indiqué précédemment, notre intuition est d'utiliser le Jaccard d'un motif pour dériver des propriétés à partir de la valeur minimum de Jaccard de toutes ses spécialisations. Dans [79], nous avons présenté une première borne LB_J^{old} de Jaccard. Dans ce qui suit, nous proposons une nouvelle borne LB_J plus resserrée.

Proposition 4 (Nouvelle borne Inférieure)**.**
Considérant un motif H de l'historique $\mathcal{H}$. Soit P un motif rencontré pendant la recherche tel que $sup_{\mathcal{D}}(P) \geq \theta$, et $\mathcal{V}_H^{pr}(P)$ la couverture résiduelle de P par rapport à H.

$$LB_J(H,P) = \begin{cases} \frac{\theta - |\mathcal{V}_H^{pr}(P)|}{|\mathcal{V}_{\mathcal{D}}(H)| + |\mathcal{V}_H^{pr}(P)|} & si\ (\mathcal{V}_H^{pr}(P) < \theta) \\ 0 & si\ non \end{cases}$$

est une borne inférieure de $Jac(H,P)$.

Preuve 3. $\forall H \in \mathcal{H}$ *on a :*

$$\begin{aligned} |\mathcal{V}_{\mathcal{D}}(P)| \geq \theta &\Leftrightarrow |\mathcal{V}_{\mathcal{D}}(H) \cap \mathcal{V}_{\mathcal{D}}(P)| + |\mathcal{V}_H^{pr}(P)| \geq \theta \\ &\Leftrightarrow |\mathcal{V}_{\mathcal{D}}(H) \cap \mathcal{V}_{\mathcal{D}}(P)| \geq \theta - |\mathcal{V}_H^{pr}(P)| \end{aligned}$$

Puisque $|\mathcal{V}_{\mathcal{D}}(H) \cup \mathcal{V}_{\mathcal{D}}(P)| = |\mathcal{V}_{\mathcal{D}}(H)| + |\mathcal{V}_H^{pr}(P)|$, alors :

$$Jac(H,P) = \frac{|\mathcal{V}_{\mathcal{D}}(H) \cap \mathcal{V}_{\mathcal{D}}(P)|}{|\mathcal{V}_{\mathcal{D}}(H) \cup \mathcal{V}_{\mathcal{D}}(P)|} = \frac{|\mathcal{V}_{\mathcal{D}}(H) \cap \mathcal{V}_{\mathcal{D}}(P)|}{|\mathcal{V}_{\mathcal{D}}(H)| + |\mathcal{V}_H^{pr}(P)|} \geq \frac{\theta - |\mathcal{V}_H^{pr}(P)|}{|\mathcal{V}_{\mathcal{D}}(H)| + |\mathcal{V}_H^{pr}(P)|}$$

Comme $Jac(H,P) \geq 0$, si $|\mathcal{V}_H^{pr}(P)| \geq \theta$, le numérateur devient égal à 0,
et $\frac{0}{|\mathcal{V}_{\mathcal{D}}(H)| + |\mathcal{V}_H^{pr}P|} = \frac{0}{\mathcal{V}_{\mathcal{D}}(H)} = 0$.

Proposition 5. *Soit $LB_J^{old}(P,H) = \frac{\theta - |\mathcal{V}_H^{pr}(P)|}{|\mathcal{V}_{\mathcal{D}}(P)| + |\mathcal{V}_{\mathcal{D}}(H)| + |\mathcal{V}_H^{pr}(P)| - \theta}$ la borne inférieure de Jaccard introduite dans [79]. La borne $LB_J(P,H)$ est plus resserrée que $LB_J^{old}(P,H)$.*

Preuve 4. *La preuve de la proposition 5 est relativement triviale*

$$\begin{aligned} |\mathcal{V}_{\mathcal{D}}(P)| \geq \theta &\Leftrightarrow |\mathcal{V}_{\mathcal{D}}(P)| + |\mathcal{V}_{\mathcal{D}}(H)| + |\mathcal{V}_H^{pr}(P)| - \theta \geq |\mathcal{V}_{\mathcal{D}}(H)| + |\mathcal{V}_H^{pr}(P)| \\ &\Leftrightarrow \frac{1}{|\mathcal{V}_{\mathcal{D}}(P)| + |\mathcal{V}_{\mathcal{D}}(H)| + |\mathcal{V}_H^{pr}(P)| - \theta} \leq \frac{1}{|\mathcal{V}_{\mathcal{D}}(H)| + |\mathcal{V}_H^{pr}(P)|} \end{aligned}$$

On obtient alors : $LB_J^{old}(P,H) \leq LB_J(P,H)$.

Proposition 6 (**Monotonie de LB_J**)**.**
Soit $H \in \mathcal{H}$ un motif. Pour tous motifs $Q \subseteq P$, la relation $LB_J(H,P) \geq LB_J(H,Q)$ est vraie.

Preuve 5. *Puisque $|\mathcal{V}_H^{pr}(P)| \leq |\mathcal{V}_H^{pr}(Q)|$ (voir lemme 1), on a*

$$\begin{aligned} LB_J(H,P) = \frac{\theta - |\mathcal{V}_H^{pr}(P)|}{|\mathcal{V}_{\mathcal{D}}(H)| + |\mathcal{V}_H^{pr}(P)|} &\geq \frac{\theta - |\mathcal{V}_H^{pr}(Q)|}{|\mathcal{V}_{\mathcal{D}}(H)| + |\mathcal{V}_H^{pr}(P)|} \\ &\geq \frac{\theta - |\mathcal{V}_H^{pr}(Q)|}{|\mathcal{V}_{\mathcal{D}}(H)| + |\mathcal{V}_H^{pr}(Q)|} = LB_J(H,Q) \end{aligned}$$

Proposition 7 (Filtrage utilisant LB_J).
Soit H un motif de l'historique $\mathcal{H}$. Pour tout motif P, si $LB_J(H,P) > J_{max}$, alors $Jac(H,P) > J_{max}$ et P n'est pas diversifié. Par ailleurs, toute spécialisation $Q \supseteq P$ n'est pas non plus diversifiée.

La proposition 7 établie un résultat important permettant de définir une condition de filtrage exploitant la monotonie de la borne inférieure. Ainsi, en combinant une fonction monotone croissante et un seuil maximum rend la contrainte elle-même anti-monotone. Effet, si $LB_J(H,P) > J_{max}$, alors aucun motif $Q \supseteq P$ ne pourra satisfaire la contrainte de Jaccard maximum. De ce fait, nous pouvons filtrer le motif Q. Notre contrainte globale exploite cette règle de filtrage dans le but de réduire l'espace de recherche (voir la section 4.4).

Exemple 4.3.1. *La figure 4.2 illustre l'exploitation de la borne inférieure sur le même jeu de données que celui de la figure 4.1. Pour chaque motif, nous reportons la valeur de la borne inférieur LB de l'indice de Jaccard. Les motifs ayant un indice de Jaccard inférieur à $J_{max} = 0.19$ sont représentés par des nœuds de couleur bleu. Les motifs avec un indice de Jaccard supérieur à J_{max} et une borne LB inférieure à J_{max} sont représentés par des nœuds en pointillés bleu. Enfin, les motifs ayant une borne LB supérieure à J_{max} sont représentés en orange et correspondent aux motifs filtrés. Par exemple, le motif C n'est pas filtré comme dans la figure 4.1, ce qui permet de générer le motif diversifié CD. Par contre le motif BDE ainsi que toutes ses spécialisations (i.e. le motif $BCDE$ seront filtrés car non diversifiés, i.e. $LB_J(BDE, BE) = 0.241 > 0.19$.*

4.3.4 Borne supérieure de Jaccard

Comme notre relaxation approxime la théorie de la contrainte de Jaccard c_{Jac}, c'est-à-dire $Th(c_{Jac}) \subseteq Th(c^r_{Jac})$, on pourrait alors avoir un motif P (un *faux positif*) avec $LB_J(P,H) < J_{max}$ alors que $Jac(P,H) > J_{max}$ (voir les motifs marqués en bleu avec des traits pointillés dans la figure 4.2). Pour prendre en considération ce cas, nous proposons une borne supérieure sur l'indice de Jaccard pour évaluer la satisfaction de la contrainte de Jaccard maximum. Les motifs P tels que $UB_J(P,H) \leq J_{max}$, $\forall H \in \mathcal{H}$ sont alors appelés des *témoins positifs*. Cette borne supérieure est alors utilisée dans une nouvelle heuristique de choix de variables afin de guider la recherche vers des ensemble de motifs diversifiés (voir la section 4.5).

Pour dériver une borne supérieure de l'indice de Jaccard, nous utilisons le raisonnement inverse de celui de la borne inférieure : le Jaccard le plus élevé possible sera atteint si $\mathcal{V}_\mathcal{D}(H) \cap \mathcal{V}_\mathcal{D}(P)$ reste inchangé et que l'ensemble $\mathcal{V}^{pr}_H(P)$ est réduit au maximum (en dessous du seuil de fréquence minimum). Si l'intersection est supérieure ou égale à θ, dans le pire des cas (conduisant au Jaccard le plus élevé), tout futur motif P' couvrira uniquement les transactions de l'intersection. Dans le cas contraire, le dénominateur devra contenir peu de transactions de $\mathcal{V}^{pr}_H(P)$, c'est-à-dire exactement $\theta - |\mathcal{V}_\mathcal{D}(H) \cap \mathcal{V}_\mathcal{D}(P)|$ transactions.

Proposition 8 (Borne supérieure).
Soit H un motif de l'historique $\mathcal{H}$, et P un motif tel que $sup_\mathcal{D}(P) \geq \theta$.

$$UB_J(H,P) = \frac{|\mathcal{V}_\mathcal{D}(H) \cap \mathcal{V}_\mathcal{D}(P)|}{|\mathcal{V}^{pr}_P(H)| + \max\{\theta, |\mathcal{V}_\mathcal{D}(H)| \cap |\mathcal{V}_\mathcal{D}(P)|\}}$$

est une borne supérieure de $Jac(H,P)$.

Preuve 6. $\forall H \in \mathcal{H}$ *on a :*

$$\Rightarrow Jac(H,P) = \frac{|\mathcal{V}_\mathcal{D}(H) \cap \mathcal{V}_\mathcal{D}(P)|}{|\mathcal{V}_\mathcal{D}(H) \cap \mathcal{V}_\mathcal{D}(P)| + |\mathcal{V}^{pr}_H(P)| + |\mathcal{V}^{pr}_P(H)|}$$

$$\xrightarrow{|\mathcal{V}_H^{pr}(P)|+|\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(P)|\geq\theta} |\mathcal{V}_H^{pr}(P)| \geq \theta - |\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(P)|$$

$$\Rightarrow Jac(H,P) \leq \frac{|\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(P)|}{|\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(P)| + |\mathcal{V}_P^{pr}(H)| + \max\{0, \theta - |\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(P)|\}}$$

$$\Rightarrow Jac(H,P) \leq \frac{|\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(P)|}{|\mathcal{V}_P^{pr}(H)| + \max\{\theta, |\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(P)|\}}$$

Exemple 4.3.2. *Tous les motifs fréquents, fermés et diversifiés avec une valeur de UB_J inférieure à J_{max} sont marqués avec des lignes en vert dans la figure 4.2. Par ailleurs, le motif AC est un témoin positif car $UB_J(AC, BE) = 0.179 < J_{max}$.*

Notre borne supérieure peut être utilisée pour évaluer la contrainte de Jaccard pendant l'extraction des motifs diversifiés. En effet, pendant la recherche, si la valeur de la borne supérieure de Jaccard d'un motif candidat P est inférieure à J_{max}, alors la contrainte c_{Jac} est vérifiée. Par ailleurs, si la borne supérieure est *monotone décroissante* (c'est à dire anti-monotone), alors P pourra être un témoin positif pour toutes ses spécialisations par rapport à la contrainte de Jaccard maximum.

Proposition 9 (Anti-monotonie de UB_J).
Soit H un motif de l'historique $\mathcal{H}$. Pour tous motifs $P \subseteq Q$, l'expression $UB_J(H,P) \geq UB_J(H,Q)$ est vérifiée.

Preuve 7. *$\forall Q \supset P, \{\} \subseteq \mathcal{V}_\mathcal{D}(Q) \subseteq \mathcal{V}_\mathcal{D}(P)$, et un motif de l'historique $H \in \mathcal{H}$:*

$$UB_J(H,P) = \frac{|\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(P)|}{|\mathcal{V}_P^{pr}(H)| + \max\{|\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(P)|, \theta\}}$$

Trois situations doivent alors être analysées :

1. $|\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(P)| > \theta, |\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(Q)| > \theta$:

$$UB_J(H,P) = \frac{|\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(P)|}{|\mathcal{V}_P^{pr}(H)| + |\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(P)|} = \frac{|\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(P)|}{|\mathcal{V}_\mathcal{D}(H)|}$$

$$\geq \frac{|\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(Q)|}{|\mathcal{V}_\mathcal{D}(H)|} = UB_J(H,Q)$$

2. $|\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(P)| > \theta, |\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(Q)| \leq \theta \Rightarrow \mathcal{V}_P^{pr}(H) + \theta \geq \mathcal{V}_\mathcal{D}(H)$:

$$UB_J(H,P) = \frac{|\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(P)|}{|\mathcal{V}_P^{pr}(H)| + |\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(P)|} = \frac{|\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(P)|}{|\mathcal{V}_\mathcal{D}(H)|}$$

$$\geq \frac{|\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(Q)|}{|\mathcal{V}_Q^{pr}(H)| + \theta} = UB_J(H,Q)$$

3. $|\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(P)| \leq \theta, |\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(Q)| \leq \theta \xrightarrow{|\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(P)|\geq|\mathcal{V}_\mathcal{D}(H)\cap\mathcal{V}_\mathcal{D}(Q)|} \mathcal{V}_P^{pr}(H) \leq$

$\mathcal{V}_Q^{pr}(H)$:

$$UB_J(H,P) = \frac{|\mathcal{V}_\mathcal{D}(H) \cap \mathcal{V}_\mathcal{D}(P)|}{|\mathcal{V}_P^{pr}(H)| + \theta} \geq \frac{|\mathcal{V}_\mathcal{D}(H) \cap \mathcal{V}_\mathcal{D}(Q)|}{|\mathcal{V}_P^{pr}(H)| + \theta}$$
$$\geq \frac{|\mathcal{V}_\mathcal{D}(H) \cap \mathcal{V}_\mathcal{D}(Q)|}{|\mathcal{V}_Q^{pr}(H)| + \theta} = UB_J(H,Q)$$

Étant donné que la propriété d'anti-monotonie est vérifiée dans les trois cas, alors elle l'est également pour la borne supérieure.

4.4 La contrainte globale ClosedDiversity

Dans cette section, nous montrons comment exploiter notre relaxation LB au sein de de la contrainte globale ClosedPatterns afin d'extraire des motifs fréquents fermés et diversifiés (FFDiv). Nous proposerons alors une nouvelle contrainte globale notée ClosedDiversity. Contrairement aux deux modèles FullCP présentés à la section 4.2, ClosedDiversity tire partie de la monotonie de la borne inférieure afin de réduire l'espace de recherche.

Définition 54 (Contrainte globale ClosedDiversity).
Soit X un vecteur de variables booléennes representant les items, $\mathcal{H}$ un historique de motifs FFDiv (initialement vide), θ un seuil de fréquence minimum, J_{max} un seuil de Jaccard maximum et $\mathcal{D}$ un jeu de données. La contrainte globale $\text{ClosedDiversity}_{\mathcal{D},\theta}(X, \mathcal{H}, J_{max})$ est vérifiée si et seulement si : (1) *X^+ est fermé,* (2) *X^+ est fréquent, $sup_\mathcal{D}(X^+) \geq \theta$;* (3) *X^+ est diversifié, $\forall H \in \mathcal{H}, LB_J(X^+, H) \leq J_{max}$.*

Initialement vide, l'historique $\mathcal{H}$ est mis à jour de façon incrémentale par notre contrainte globale en y ajoutant les motifs FFDiv extraits pendant la recherche. Cela est rendu possible grâce à la condition (3) qui est une condition nécessaire pour que X^+ soit diversifié. En effet, il possible d'avoir $LB_J(X^+, H) \leq J_{max}$ alors que $Jac(X^+, H) > J_{max}$.

Le propagateur de ClosedDiversity exploite les règles de filtrages de ClosedPatterns (voir la section 2.5.2). Il utilise également notre relaxation LB_J afin de filtrer les items libres i ne conduisant pas à une solution diversifiée contenant X^+.

Nous noterons par X^-_{Freq} l'ensemble des items filtrés par la règle sur la fréquence, et par X^-_{Div} les items filtrés par la règle sur la diversité. X^-_{Freq} et X^-_{Div} forment alors une partition de X^-.

Proposition 10 (Règles de filtrage de ClosedDiversity).
Étant donné un historique $\mathcal{H}$ de motifs deux à deux FFDiv, X un motif partiel, et un item libre $i \in X^*$, $X^+ \cup \{i\}$ ne pourra pas être étendu à un motif diversifié si l'une de deux conditions ci-dessous est vérifiée :
1) si $\exists H \in \mathcal{H}$ s.t. $LB_J(H, X^+ \cup \{i\}) > J_{max}$, alors on filtre 1 de $dom(X_i)$;
2) si $\exists k \in X^-_{Div}$ t.q. $\mathcal{V}_\mathcal{D}(X^+ \cup \{i\}) \subseteq \mathcal{V}_\mathcal{D}(X^+ \cup \{k\})$, alors $LB_J(H, X^+ \cup \{i\}) > LB_J(H, X^+ \cup \{k\}) > J_{max}$, on filtre 1 de $dom(X_i)$.

La validité de la règle de filtrage (2) associée à $k \in X^-_{Div}$ se déduit en raisonnant sur les items absents.

Preuve 8. *La preuve de la règle de filtrage* (1) *est une conséquence de la proposition 6. Nous donnons ci-après la preuve de la règle* (2)*. Soit $P = X^+ \cup \{i\}$, $Q = X^+ \cup \{k\}$ t.q. $i \in X^*$ et $k \in X^-_{Div}$ et $H \in \mathcal{H}$.*

$$LB_J(H,P) = \frac{\theta - |\mathcal{V}_H^{pr}(P)|}{|\mathcal{V}_\mathcal{D}(H)| + |\mathcal{V}_H^{pr}(P)|}$$

Algorithm 5: Filtrage de la contrainte globale ClosedDiversity

1 **In** : θ, J_{max} : seuil de fréquence et de diversité ; $\mathcal{H}$: historique des solutions trouvées pendant la recherche ;
2 **InOut** : $X = \{X_1 \dots X_n\}$: variables booléennes d'items ;
3 **Début**
4 **Si** ($|\mathcal{V}_{\mathcal{D}}(X^+)| < \theta \vee !\mathcal{P}Growth_{LB}(X^+, \mathcal{H}, J_{max})$) **alors retourner faux**;
5 **Pour chaque** $i \in X^*$ **faire**
6 **Si** ($|\mathcal{V}_{\mathcal{D}}(X^+ \cup \{i\})| < \theta$) **alors**
7 $dom(X_i) \leftarrow dom(X_i) - \{1\}$;
8 $X^-_{Freq} \leftarrow X^-_{Freq} \cup \{i\}$; $X^* \leftarrow X^* \setminus \{i\}$;
9 **continuer** ;
10 **Si** ($|\mathcal{V}_{\mathcal{D}}(X^+ \cup \{i\})| = |\mathcal{V}_{\mathcal{D}}(x^+)|$) **alors**
11 $dom(X_i) \leftarrow dom(X_i) - \{0\}$;
12 $X^+ \leftarrow X^+ \cup \{i\}$; $X^* \leftarrow X^* \setminus \{i\}$;
13 **Si** ($!\mathcal{P}Growth_{LB}(X^+ \cup \{i\}, \mathcal{H}, J_{max})$) **alors**
14 $dom(X_i) \leftarrow dom(X_i) - \{1\}$;
15 $X^-_{Div} \leftarrow X^-_{Div} \cup \{i\}$; $X^* \leftarrow X^* \setminus \{i\}$;
16 **continuer** ;
17 **Pour chaque** $k \in (X^-_{Freq} \cup X^-_{Div})$ **faire**
18 **Si** ($\mathcal{V}_{\mathcal{D}}(X^+ \cup \{i\}) \subseteq \mathcal{V}_{\mathcal{D}}(X^+ \cup \{k\})$) **alors**
19 $dom(X_i) \leftarrow dom(X_i) - \{1\}$
20 **si** $k \in X^-_{Freq}$ **alors**
21 $X^-_{Freq} \leftarrow X^-_{Freq} \cup \{i\}$;
22 **sinon**
23 $X^-_{Div} \leftarrow X^-_{Div} \cup \{i\}$;
24 $X^* \leftarrow X^* \setminus \{i\}$;
25 **arrêter** ;
26 **retourner vrai**;
27 **Fonction** $\mathcal{P}Growth_{LB}(x, \mathcal{H}, J_{max})$ *: Booléen*
28 **Pour chaque** $H \in \mathcal{H}$ **faire**
29 **Si** ($LB_J(H, x) > J_{max}$) **alors**
30 **retourner faux**
31 **retourner vrai**

$$|\mathcal{V}_{\mathcal{D}}(P)| \leq |\mathcal{V}_{\mathcal{D}}(Q)| \Rightarrow |\mathcal{V}^{pr}_H(P)| \leq |\mathcal{V}^{pr}_H(Q)|$$

$$\Rightarrow LB_J(H, P) \geq \frac{\theta - |\mathcal{V}^{pr}_H(Q)|}{|\mathcal{V}_{\mathcal{D}}(H)| + |\mathcal{V}^{pr}_H(P)|} \geq \frac{\theta - |\mathcal{V}^{pr}_H(Q)|}{|\mathcal{V}_{\mathcal{D}}(H)| + |\mathcal{V}^{pr}_H(Q)|} = LB_J(H, Q)$$

Algorithme de filtrage. Le propagateur de ClosedDiversity est décrit par l'algorithme 5. Il prend en entrée les variables X, un seuil de fréquence θ, une seuil de Jaccard maximum J_{max} et un historique $\mathcal{H}$ de motifs deux à deux FFDiv. Il commence par calculer la couverture du motif X^+, puis vérifie si ce dernier est fréquent et diversifié (voir la fonction $\mathcal{P}Growth_{LB}$)). Dans le cas où il n'est pas fréquent ou non diversifié, la contrainte n'est plus respectée et un échec est retourné (ligne 4).

L'algorithme 5 étend les règles de filtrage de ClosedPatterns en ajoutant une règle sur la diversité du motif $X^+ \cup \{i\}$ (voir proposition 10). Pour chaque motif $H \in \mathcal{H}$, la fonction

$\mathcal{PG}rowth_{LB}(X^+ \cup \{i\}, \mathcal{H}, J_{max})$ calcule la valeur de $LB_J(X^+ \cup \{i\}, H)$ et vérifie s'il existe un motif H pour lequel $LB_J(X^+ \cup \{i\}, H) > J_{max}$ (lignes 28-29). S'il existe un motif pour lequel cette inégalité est vraie, alors on retourne la valeur faux (ligne 30). Puisque $X^+ \cup \{i\}$ ne peut pas conduire à un motif FFDiv par rapport à l'historique $\mathcal{H}$, on filtre la valeur 1 de $dom(X_i)$ (ligne 16), X^-_{Div} et X^* sont mis à jour et la recherche se poursuit avec les autres items libres. Sinon, on retourne la valeur vrai. Par ailleurs, on retire la valeur 1 du domaine de chaque variable libre $i \in X^*$ dont la couverture est un sur-ensemble de la couverture d'un item absent $k \in (X^-_{Freq} \cup X^-_{Div})$ (lignes 17-25).

Proposition 11 (Consistance et complexité temporelle).
L'algorithme 5 maintient la consistance d'arc (ou consistance de domaine [81]) avec une complexité temporelle en $\mathcal{O}(n^2 \times m)$.

Preuve 9. ***(1) Arc consistance.*** *L'algorithme 5 utilise deux groupes de règles de filtrage : (i) les règles portant sur la fermeture permettant de maintenir l'arc-consistance (voir [97] pour la preuve) et (ii) la règle basée sur la relaxation LB permettant de garantir que chaque item libre $i \in X^*$, ne pouvant étendre un motif courant X^+ vers un motif FFDiv, sera nécessairement filtré (proposition 10). Par construction, toute valeur $X_i \in X$ ne conduisant pas à une solution satisfaisant la contrainte* ClosedDiversity *sera supprimée de $dom(X_i)$. Par conséquent, la contrainte* ClosedDiversity *maintient l'Arc-consistante sur les variables de X.*

(2) Complexité temporelle. *La ligne 4 de l'algorithme s'exécute en $\mathcal{O}(n \times m)$. La fonction $\mathcal{PG}rowth_{LB}$ s'exécute en $\mathcal{O}(m \times |\mathcal{H}|)$ car chaque appel à $LB_J(H, X)$ s'exécute en $\mathcal{O}(m)$, ainsi le filtrage de domaine s'effectue en $\mathcal{O}(n \times m \times |\mathcal{H}|)$. Si l'historique $\mathcal{H}$ contient $|\mathcal{H}| \leq n$ éléments, alors le filtrage de domaine s'effectue en $\mathcal{O}(n^2 \times m)$. De plus, les filtrages portant sur la fréquence et la fermetures s'effectuent en $\mathcal{O}(\frac{n^2}{4} \times m)$: (i) le filtrage des items non fréquents se fait en $\mathcal{O}(n \times m)$ car le test de la ligne 6 se fait en $\mathcal{O}(m)$; (ii) le filtrage des items n'apparaissant dans la clôture d'un motif se fait en en $\mathcal{O}(n \times m)$ car le test de la ligne 12 est réalisé en $\mathcal{O}(m)$; (iii) enfin, le filtrage des lignes 17 à 25 s'effectue en $\mathcal{O}(\frac{n^2}{4} \times m)$ car le test d'inclusion de la ligne 18 se fait en $\mathcal{O}(m)$ et $|X^-| + |X^*|$ vaut au plus n, ainsi cette règle de filtrage est alors vérifiée avec au plus $\frac{n}{2} \times \frac{n}{2}$ opérations (c'est à dire $\mathcal{O}(\frac{n^2}{4})$). Comme cette dernière règle de filtrage est dissociée de la règle de filtrage sur la diversité (c'est pour cette raison que nous continuons à la ligne 16), la complexité temporelle de l'algorithme de filtrage est alors de $\mathcal{O}(n^2 \times m)$.*

4.5 Booster la recherche de motifs diversifiés

La contrainte globale ClosedDiversity est exploitée au niveau de la propagation du solveur de contraintes, c'est-à-dire à la ligne 4 de l'algorithme 3. Dans cette section, nous exploitons les notions de motifs témoins et de *fréquences estimées* pour guider la recherche vers des motifs diversifiés. Pour cela, nous proposons deux nouvelles heuristiques de choix de variables décrites dans les lignes 7 à 9 de l'algorithme 3. Enfin, nous montrons comment exploiter la contrainte de Jaccard maximum ainsi que notre relaxation pour réduire le nombre de faux positifs dans les motifs diversifiés.

4.5.1 Motif Témoin Positif

La borne supérieure de Jaccard est exploitée afin de certifier la satisfaction de la contrainte de Jaccard maximum. Pendant la recherche, à chaque extension d'un motif partiel X^+ avec un item libre i, la valeur de sa borne supérieure est calculée incrémentalement. Ainsi, si pour tout $H \in \mathcal{H}$, $UB(X^+ \cup \{i\}, H) \leq J_{max}$, alors c_{Jac} sera satisfaite. Le motif $X^+ \cup \{i\}$ peut alors être

Algorithm 6: Choix des variables de ClosedDiversity

Entrée : J_{max} : seuil de diversité ; $\mathcal{H}$: historique des solutions ;
Sortie : Indice du premier motif témoin ou X^{es} item ayant la plus petite fréquence estimée
begin
  $X^{es} \leftarrow LazyEstimateCover(X^+, X^*)$;
  Pour chaque $i \in X^*$ **faire**
    Si $(\mathcal{P}Growth_{UB}(X^+ \cup \{i\}, \mathcal{H}, J_{max}))$ **alors**
      retourner $\langle i, \textbf{vrai}\rangle$;
  retourner $\langle X^{es}, \textbf{faux}\rangle$
Fonction $LazyEstimateCover(X^+, X^*)$ *: Entier*
  $X^{es} \leftarrow argmin_{i \in X^*}(eSup_{\mathcal{D}}(i, X^+))$;
  diff $\leftarrow (|\mathcal{V}_{\mathcal{D}}(x^+)| - |\mathcal{V}_{\mathcal{D}}(X^+ \cup \{X^{es}\})|)$;
  Pour chaque $i \in x^* \setminus \{X^{es}\}$ **faire**
    $eSup_{\mathcal{D}}(i, X^+ \cup \{X^{es}\}) \leftarrow eSup_{\mathcal{D}}(i, X^+) -$ diff ;
    Si $(eSup_{\mathcal{D}}(i, X^+ \cup \{X^{es}\}) < \theta)$ **alors**
      $eSup_{\mathcal{D}}(i, X^+ \cup \{X^{es}\}) \leftarrow |\mathcal{V}_{\mathcal{D}}(X^+ \cup \{X^{es}\}) \cap \mathcal{V}_{\mathcal{D}}(i)|$;
  retourner X^{es}
Fonction $\mathcal{P}Growth_{UB}(X^+ \cup \{j\}, \mathcal{H}, J_{max})$ *: Booléen*
  foreach $H \in \mathcal{H}$ **do**
    if $(UB_J(H, x^+ \cup \{j\}) > J_{max})$ **then**
      retourner faux
  retourner vrai

utilisé comme *motif témoin positif*. Par ailleurs, étant donné que UB est anti-monotone (voir la proposition 9), alors toutes les spécialisations de $X^+ \cup \{i\}$ satisferont la contrainte de Jaccard par rapport à $\mathcal{H}$ et au seuil de Jaccard maximum J_{max}.

4.5.2 Mincov : une heuristique de branchement basée sur les fréquences estimées

La fréquence d'un motif peut être calculée en calculant la cardinalité de l'intersection des couvertures des items du motif : $sup_{\mathcal{D}}(X^+) = |\cap_{i \in X^+} \mathcal{V}_{\mathcal{D}}(i)|$, l'intersection entre chaque paire de couvertures se faisant par une opération de ET-logique sur chaque bit de la couverture. Afin de réduire le nombre d'opérations d'intersections qui peuvent être très coûteuses, nous calculons les fréquences estimées $eSup_{\mathcal{D}}(i, X^+)$ de chaque item $i \in \mathcal{I}$ à partir des items de X^+. L'estimation de la fréquence nous permet d'avoir une *borne inférieure* de la fréquence $|\mathcal{V}_{\mathcal{D}}(X^+ \cup \{i\})|$. Ainsi, si $eSup_{\mathcal{D}}(i, X^+) \geq \theta$ alors $|\mathcal{V}_{\mathcal{D}}(X^+ \cup \{i\})| \geq \theta$. De cette manière, nous évitons le calcul coûteux de la fréquence réelle et nous calculons l'intersection des couvertures que lorsque $eSup_{\mathcal{D}}(i, X^+) < \theta$. Par ailleurs, la fréquence estimée peut-être exploitée pour dériver une heuristique de choix de variables. En effet, brancher sur la variable ayant la plus petite fréquence estimée (en utilisant la borne inférieure de la fréquence) permet d'accélérer l'activation d'une de nos règle de filtrage (voir algorithme 5). L'espace de recherche peut alors être réduit plus rapidement, ce qui nous permet d'améliorer les performances de ClosedDiversity. Cette nouvelle heuristique de choix de variable sera notée Mincov.

4.5.3 Witness : une heuristique de branchement basée sur les motifs témoins positifs

Dans l'algorithme 6, nous proposons une nouvelle heuristique des choix de variables. Au début de la recherche, la fréquence estimée de chaque item $i \in X^*$ est initialisée à $eSup_{\mathcal{D}}(i, \emptyset) = |\mathcal{V}_{\mathcal{D}}(i)|$. Puis, lorsqu'un item libre j est ajouté au motif partiel X^+, la fréquence estimée de tous les items libres est mise à jour (la fonction $LazyEstimateCover$). Cette opération permet alors de trouver la variable X^{es} ayant la plus petite fréquence estimée (ligne 10). Par la suite, étant donné que la couverture du motif partiel courant perdra certaines transactions, les items $i \in X^* \setminus \{X^{es}\}$ verront leur support réduit d'une valeur maximum $diff = |\mathcal{V}_{\mathcal{D}}(X^+)| - |\mathcal{V}_{\mathcal{D}}(X^+ \cup \{X^{es}\})|$ (ligne 11). Les fréquences estimées des items i sont alors mises à jour à $eSup_{\mathcal{D}}(i, X^+) - diff$ (lignes 13 à 15). Si $eSup_{\mathcal{D}}(i, X^+) < \theta$, alors la fréquence estimée de i est mise à jour avec la fréquence réelle de l'extension du motif (ligne 15). Dans le cas contraire ($eSup_{\mathcal{D}}(i, X^+) \geq \theta$), nous avons la garantie que l'extension vers l'item i ne conduira pas à un motif non fréquent ($|\mathcal{V}_{\mathcal{D}}(X^+ \cup \{i\})| \geq \theta$).

Enfin, la fonction $\mathcal{P}Growth_{UB}(X^+ \cup \{i\}, \mathcal{H}, J_{max})$ permet d'évaluer la diversité de l'extension de X^+ avec un item libre i. Elle permet de vérifier si cette extension peut conduire à un motif témoin. Pour cela, on évalue la valeur de la borne supérieure de $X^+ \cup \{i\}$. Si $UB_J(H, X^+ \cup \{i\}) \leq J_{max} \forall H \in \mathcal{H}$, alors $X^+ \cup \{i\}$ peut conduire à un motif témoin positif (ligne 18 à 20). Dans ce cas, la contraire de Jaccard est satisfaite et l'item i est retourné. Cet item sera alors exploité pendant la recherche (ligne 7) afin d'accélérer le calcul de la diversité des motifs. Nous noterons pas Witness cette nouvelle heuristique de choix de variable qui permet de brancher sur la première variable d'item libre satisfaisant la propriété des motifs témoins.

4.5.4 Stratégies d'exploration des sous-arbres témoins

Soit N le nœud associé au motif courant X^+ étendu avec un item libre $\{i\}$ (c'est à dire $X^+ \cup \{i\}$). Lorsque le nœud N est détecté comme motif témoin positif pendant l'étape de branchement, alors tous les sur-motifs dérivés de N satisferont également la contrainte de Jaccard maximum. Comme ces motifs sont plus susceptibles d'avoir des couvertures similaires, donc un Jaccard assez élevé entre eux, nous proposons une stratégie simple, notée WIT-FirstSOL, qui génère le premier motif FFDiv du sous-arbre enraciné dn N et d'ignorer les autres motifs. Ainsi, nous générons le premier motif diversifié fermé à partir de N , l'ajoutons à l'historique courant et continuons l'exploration de l'espace de recherche restant en utilisant le nouvel historique. Cette stratégie n'effectue donc pas une exploration complète des sous-arbres témoins. Même si elle ne garantit pas d'ajouter à l'historique le motif FFDiv ayant le meilleur Jaccard, elle a l'avantage d'être rapide.

4.5.5 Évitement des motifs diversités faux positifs

Comme nous l'avons vu dans la section 4.3.4, notre relaxation peut générer des motifs diversifiés mais qui sont des faux positifs, c'est-à-dire des motifs P tels que $LB_J(H, P) < J_{max}$ mais avec $Jac(H, P) > J_{max}$. Pour remédier à ce problème, nous avons proposé à la section 4.5.3 une nouvelle heuristiques Witness pour sélectionner les items satisfaisant la contrainte de diversité. Pour aller plus loin, nous proposons d'évaluer au niveau de chaque nœud feuille de l'arbre de recherche (c'est-à-dire à chaque fois que ClosedDiversity nous permet d'obtenir une nouvelle solution) si la nouvelle solution satisfait la contrainte de Jaccard maximum avant la mise à jour de l'historique $\mathcal{H}$. Soit X^+ une instanciation complète des variables X générée par ClosedDiversity, si pour tout $H \in \mathcal{H}$ on a $Jac(H, X^+) \leq J_{max}$, alors X^+ est ajouté $\mathcal{H}$. Dans le cas où nous utilisons l'heuristique Witness, ce test n'est effectué que si aucun motif témoin n'est trouvé (voir l'algorithme 6). Par contre, pour l'heuristique Mincov, ce test doit être effectué à chaque nouvelle solution, étant donné que l'approximation de la théorie de la contrainte de Jaccard maximum est basée sur notre relaxation par la borne inférieure. Nous noterons toutes les méthodes exploitant ce test par Méthode+Jaccard

4.6 Échantillonnage de motifs diversifiés

Les techniques d'échantillonnage de motifs constituent un axe de recherche prometteur pour contrôler le nombre de motifs produits et garantir des temps de réponses raisonnables. En effet, un des points forts de l'échantillonnage de motifs est d'offrir un accès direct à l'ensemble des motifs à faible coût tout en garantissant une forme de diversité entre les motifs échantillonnés, grâce notamment à son caractère non-exhaustif et à sa nature aléatoire. Toutefois, les principales méthodes qui ont été proposées pour l'échantillonnage de motifs ne permettent pas de contrôler de manière exacte cette diversité.

Dans cette section, nous montrons comment exploiter la contrainte globale ClosedDiversity au sein d'un outil d'échantillonnage afin de contrôler de manière explicite la diversité des motifs échantillonnés. Pour cela, nous proposons d'utiliser Weightgen [35] pour échantillonner des motifs en utilisant comme oracle pour l'énumération des motifs, la contrainte globale ClosedDiversity. Le cadre obtenu est alors similaire à celui de Flexics [51]. Notre nouvel outils, noté SDivJaX pour « **S***ample* **Div***erse pattern with* **Ja***ccard and* **X***or constraints* » c'est-à-dire « *Échantillonnage de motifs diversifiés avec l'indice de Jaccard et des contraintes* `XOR` », tire partie de la diversité implicite apportée par les contraintes `XOR` de Weightgen et la diversité explicite de la contrainte globale ClosedDiversity.

Nous commencerons par présenter dans la section 4.6.1 l'algorithme Weightgen et les principales adaptations que nous avons apporté à cet algorithme. Ensuite, dans la section 4.6.2, nous détaillerons ces adaptations. En particulier, nous présenterons deux nouveaux oracles qui exploitent ClosedDiversity pour l'énumération de motifs au niveau de chaque cellule et qui vont servir à l'étape d'échantillonage. Enfin, nous décrirons dans la section 4.6.4 l'algorithme de filtrage des contraintes `XOR` que nous avons implanté.

4.6.1 Adaptations de l'algorithme Weightgen

Basiquement, l'algorithme Weightgen réduit l'espace des solutions aléatoirement à l'aide de contraintes `XOR`, puis fait un tirage pondéré dans cet espace. Ces contraintes portent sur la présence des items dans le motif. Plus précisément, il partitionne l'espace de recherche des motifs en des cellules puis tire dans différentes cellules des motifs proportionnellement à leur poids.

Pour obtenir la "bonne" taille de cellule [6] désirée, lors de l'étape d'initialisation, l'algorithme commence par estimer le nombre de contraintes `XOR` qu'il faut ajouter en moyenne durant la phase d'échantillonage (voir la ligne 4, algorithme 7). Cette taille est calculée en additionnant les poids de toutes les solutions du problème (voir la ligne 15, algorithme 7). Finalement, vient la phase d'échantillonnage (voir la ligne 10, algorithme 7) où, pour chaque motif échantillonné, les opérations ci-dessous sont répétées :

1. générer les contraintes `XOR` estimées lors de la phase d'initialisation (ligne 8) ;
2. énumérer les solutions dans le sous-espace défini par les contraintes `XOR` (ligne 14) ;
3. calculer le poids de ces solutions (ligne 15) ;
4. Si le poids total ne convient pas, cellule trop grande ou trop petite, ajouter ou enlever une contrainte puis revenir à l'étape 2 ;
5. tirer un motif aléatoirement suivant la pondération choisie.

Modifications de Weightgen. Les modifications que nous avons apporté concernent principalement la prise en compte explicite de la diversité dans l'étape d'énumération des solutions dans chaque cellule (cf. la fonction ClosedXorSolve) et le contrôle du nombre de contraintes `XOR` dans le cas où la cellule devient trop petite (cf. les lignes 20-21, algorithme 7).

6. L'algorithme contient un paramètre κ, nommé "sampling error tolerance", qui correspond à la tolérence sur la taille de la cellule et qui permet d'avoir une solution plus rapidement au détriment du respect de la bonne distribution.

Algorithm 7: Updated WEIGHTGEN

Entrée : $\mathcal{D}, \theta, J_{max}, w$: fonction de poids, κ : tolérance d'erreur, k : nombre de tirages
Entrée/Sortie : $X = \{X_1 \dots X_n\}$: variables booléennes ;
Début
  $N_{XOR} \leftarrow EstimationXOR()$;
  $loThresh \cong (1+\kappa)/\kappa^2$, $hiThresh \cong (1+\kappa)^3/\kappa^2$;
  $i \leftarrow 1$, $\mathcal{H} \leftarrow \emptyset$;
  Pour $(i \leq k)$ **faire**
    $InitXORs \leftarrow \{RandomXOR() \times N_{XOR}\}$;
    ▷ Retourner une solution tirée aléatoirement
    $P \leftarrow$ GENERER $(\kappa, [loThresh, hiThresh], InitXORs)$;
    si $(P \neq$ NULL$)$ **alors** $\mathcal{H} \leftarrow \mathcal{H} \cup P$;
  retourner $\mathcal{H}$;
Fonction GENERER$(\kappa, [lT, hT], XORs)$ *: Motif*
  $Solutions \leftarrow$ **ClosedXorSolve**$(X, \theta, J_{max}, XORs)$
  $PoidsCellule \leftarrow \sum_{s \in Solutions} w(s)$
  si $(PoidsCellule \in [lT, hT])$ **alors**
    retourner ECHANTILLONNER$(Solutions, w)$
  si $(PoidsCellule > hT)$ **alors**
    retourner GENERER$(\kappa, [loThresh, hiThresh], XORs \cup RandomXOR()$
  Si $(|XORs| > 0)$
    retourner GENERER$(\kappa, [loThresh, hiThresh], XORs - RandomXOR())$
  Sinon
    retourner NULL

4.6.2 Oracles de sélection des motifs diversifiés

L'exploitation de WEIGHTGEN pour l'échantillonnage des motifs permet à FLEXICS de tirer des motifs proportionnellement à leur poids (évalué grâce à une mesure de qualité). Par ailleurs, avec les contraintes `XOR`, il est possible d'obtenir une diversité implicite entre les motifs échantillonnés car ceux-ci sont extraits dans différentes zone de l'espace de recherche. Toutefois, cette diversité n'est pas garantie. En effet, étant donné le caractère aléatoire des contraintes `XOR` générées et le fait que chaque tirage soit indépendant, les cellules obtenues par l'application de ces contraintes peuvent se chevaucher. Il en résulte alors la possibilité de tirer deux motifs identiques ou très similaires dans différentes cellules. Pour ce faire, nous proposons deux oracles d'énumération des motifs permettant d'ajouter explicitement une contrainte de diversité : SDIVJAX-1 et SDIVJAX-2. Par ailleurs, en contraignant l'espace de recherche de chaque cellule par l'ajout de la contrainte de diversité, nous espérons réduire le temps d'exploration au niveau de chaque cellule.

SDivJaX-1. L'objectif de cet oracle est d'extraire un ensemble de motifs diversités localement à chaque cellule, puis échantillonner un motif parmi cet ensemble. Pour cela, dans chaque cellule, la contrainte globale CLOSEDDIVERSITY est utilisée pour énumérer les motifs fréquents fermés et diversifiés qui satisfont les contraintes `XOR` générées par WEIGHTGEN. Il en résulte alors un historique $\mathcal{H}_{local}$ de motifs diversifiés entre eux. Un motif est par la suite tiré de cet historique local et l'opération se répète jusqu'à l'obtention du nombre k de motifs demandés.

Étant donné le caractère local de la diversité des motifs échantillonnés, l'historique $\mathcal{H}_{local}$ exploité par CLOSEDDIVERSITY est réinitialisé à l'ensemble vide, à chaque nouveau tirage de WEIGHTGEN. Dans chaque cellule i, CLOSEDDIVERSITY énumère donc un ensemble de motifs $\mathcal{H}^i_{local} = \{H^1_i, H^2_i, \dots, H^n_i\}$ tel que :

$$\forall j, \ell \in [1, n] \wedge (j > \ell) : LB_J(H^j_i, H^\ell_i) \leq J_{max}$$

Un motif est alors tiré de cet ensemble proportionnellement à son poids et le processus se

poursuit jusqu'à l'extraction complète du nombre k d'échantillons demandés. Cet ensemble $\mathcal{H}$ de k motifs se présente alors comme suit :

$$\mathcal{H} = \{H_1, H_2, \ldots, H_k\}$$

avec $H_j \in \mathcal{H}^j_{local}$ où $\mathcal{H}^j_{local}$ est l'historique local de motifs extraits par ClosedDiversity à la cellule j.

Cette approche nous permet de bénéficier des performances de ClosedDiversity et ainsi d'accélérer l'échantillonnage de chaque motif au sein de chaque cellule. Par ailleurs, le filtrage effectué avec la borne LB_J nous permet d'obtenir des cellules de tailles plus réduites par rapport à celles de EFlexics et GFlexics. De ce fait, moins de contraintes XOR sont nécessaires pour échantillonner chaque motif, ce qui permet un gain supplémentaire en termes de performance. Toutefois, en réinitialisant l'historique des motifs $\mathcal{H}_{local}$ à chaque nouvelle cellule, la méthode ne permet pas de garantir une diversité globale des motifs échantillonnés, c'est à dire ceux de $\mathcal{H}$. Nous proposons de remédier à cette situation avec l'approche SDivJaX-2.

SDivJaX-2. Avec SDivJaX-2, nous maintenons un historique $\mathcal{H}_{global}$ des différents motifs échantillonnés à travers les différentes cellules. Cet historique est mis à jour après chaque tirage de chaque motif et est utilisé pour assurer la diversité entre motifs déjà échantillonnés et les motifs solutions des prochaines cellules.

Pour garantir la diversité entre les différents motifs échantillonnés, nous utilisons la contrainte globale ClosedDiversity qui prend en paramètre un historique $\mathcal{H}_{global}$ initialement vide. Contrairement à l'approche SDivJaX-1, $\mathcal{H}_{global}$ n'est pas mis à jour avec tous les motifs découverts dans une même cellule. Notons par $\mathcal{H}^{i-1}_{global}$ l'historique obtenu après l'échantillonnage des $(i-1)$ premières cellules. Au départ, $\mathcal{H}^1_{global} = \emptyset$. Pour échantillonner un motif dans la $i$ème cellule, nous procédons comme suit :

— extraction de l'ensemble $S = \{s_1, ..., s_n\}$ de motifs fréquents, fermés et diversifiés par rapport à $\mathcal{H}^{i-1}_{global}$, tel que

$$\forall\, s_j \in S, LB_J(s_j, \mathcal{H}^{i-1}_{global}) \leq J_{max}$$

— tirage d'un motif $s_j \in S$ proportionnellement à son poids et mise à jour de $\mathcal{H}^i_{global}$ comme suit :

$$\mathcal{H}^i_{global} \leftarrow \mathcal{H}^{i-1}_{global} \cup \{s_j\}$$

Soit k le nombre de tirages à réaliser, après l'échantillonnage des k motifs, nous avons $\mathcal{H}_{global} = \mathcal{H}^k_{global} = \{s_1, s_2, \ldots, s_k\}$, avec

$$\forall i, j \in [1, k] \wedge (i > j) : LB_J(s_i, s_j) \leq J_{max}$$

4.6.3 Estimation et contrôle du nombre de contraintes `XOR`

Les deux approches que nous avons détaillé précédemment ont nécessité quelques adaptations du processus d'échantillonnage de Weightgen et plus particulièrement dans le cas de SDivJaX-2. En effet, étant donné qu'un historique global des solutions est maintenu, l'espace de recherche se réduit à chaque mise à jour de $\mathcal{H}_{global}$. De ce fait, l'estimation faite par Weightgen du nombre de contraintes `XOR` nécessaires à l'échantillonnage devient de plus en plus inexact. En effet, en maintenant le même nombre de contraintes XOR tout au long de l'échantillonnage des k motifs ainsi qu'un historique de plus en plus grand, la phase d'énumération de motifs dans une cellule peut échouer car l'espace de recherche devient trop contraint.

Nous avons proposé de modifier l'algorithme de Weightgen afin d'adapter le nombre de contraintes XOR lorsque la taille de la cellule devient trop petite en raison de l'exploitation d'un

historique global $\mathcal{H}_{global}$. Ainsi, lorsque le nombre de contraintes `XOR` ne permet pas d'obtenir suffisamment de motifs (ligne 21), nous autorisons le retrait d'une contrainte afin de moins contraindre l'espace de recherche. Le même raisonnement est également appliqué lors du maintien d'un historique local $\mathcal{H}_{local}$ (cas de l'oracle SDivJaX-1). Notons, toutefois, que dans ce cas, la taille de cellule est moins contrainte. Dans le cas où il n'y a pas de solutions et que le nombre de contraintes `XOR` est nulle (ligne 22), alors nous pouvons arrêter la recherche. En effet, ce cas correspond à une situation où il n'y a plus aucune solution diversifiée. Il est alors inutile de poursuivre la résolution.

La phase d'estimation du nombre de contraintes `XOR` nécessaires pour obtenir une "bonne" taille de cellule peut également impacter le processus d'échantillonnage. Pour nos deux oracles, nous avons utilisé la contrainte globale ClosedDiversity pour déterminer la taille de la cellule. Nous avons également étudié l'impact d'utiliser la contrainte globale ClosedPatterns comme moyen d'estimer la taille d'une cellule.

4.6.4 Résolution et propagation des contraintes XOR

Dans la pratique, l'algorithme Weightgen va ajouter plusieurs contraintes `XOR`. On se retrouve alors avec un système d'équations linéaires que l'on va pouvoir résoudre avec la méthode du pivot de Gauss en remplaçant les contraintes `XOR` par une addition dans le corps $\mathbf{F}_2$, qui a la même table de vérité que le `XOR`. En nous appuyons sur la méthode du pivot de Gauss, nous avons implanté un propagateur de contraintes `XOR`. Ce propagateur utilise une matrice de taille $(m, n+1)$, où m est le nombre de contraintes et n le nombre d'items du jeu de données, la dernière colonne représentant le bit de parité. Ces matrices sont obtenues en traduisant les différentes contraintes XORs en des sommes de coefficients binaires :

$$\oplus a_i x_i = b \Rightarrow \sum a_i = b$$

Ainsi, dans chaque contrainte k, si une variable x_i participe à la contrainte, alors on aura $\mathcal{M}[k][i] = 1$, sinon $\mathcal{M}[k][i] = 0$.

Exemple 4.6.1. *Par exemple, pour un jeu de données avec 5 items, les contraintes `XOR` suivantes :*

$$\begin{cases} x_1 \oplus x_5 = False \\ x_1 \oplus x_3 \oplus x_5 = False \\ x_4 \oplus x_5 = True \end{cases} \quad donneront \Rightarrow \quad \left(\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right)$$

L'algorithme 8 détaille le principe du filtrage des contraintes `XOR`. À chaque étape de propagation, la matrice $\mathcal{M}$ est mise à jour (ligne 5) à partir du motif partiel courant X^+. Pour toute variable X_i instanciée à 1 qui apparaît dans une contrainte `XOR`, le bit de parité de la contrainte est inversé et le coefficient de cette variable dans les lignes correspondantes de la matrice est mis à 0. La figure 4.3 illustre cette opération sur les lignes 1 et 2 de l'étape 5 et à la ligne 2 de l'étape 3.

Ensuite, nous vérifions si la mise à jour de la matrice ne conduit pas à un cas d'inconsistance, c'est à dire une ligne vide avec un bit de parité égal à 1. Dans ce cas le système de contraintes `XOR` est insatisfiable (étape 5, figure 4.3). Si ce test ne conduit pas à un échec, alors une opération d'échelonnement (ligne 12) est réalisée avec la méthode de Gauss-Jordan pour obtenir une matrice centrée-réduite (étape 3, figure 4.3). Lors de l'échelonnement, deux situations peuvent se présenter :

— Si une ligne devient vide alors que son membre de droite est égal à 1, le système est insatisfiable et la branche de recherche en cours se termine ;
— Si une ligne ne contient qu'une seule variable libre, elle est alors affectée à son membre droit dans la ligne. Par exemple, à partir de la ligne 2 de l'étape 3, nous pouvons filtrer la valeur 1 du domaine de X_3.

Algorithm 8: Propagateur de contraintes `XOR`

```
1  Entrée : I, T, M : matrice des coefficients
2  Entrée/Sortie : X = {X_1 ... X_n} : variables booléennes ;
3  Début
4      X^+ ← {i | X_i = 1}, X^- ← {i | X_i = 0}, X^* = {i ∈ I | i ∉ X^+ ∪ X^-}
5      Pour (i ∈ X^+ ∧ ligne r ∈ M) faire
6          Si (M[r][i] = 1) alors parity_r ← 1 − parity_r
7          M[r][i] ← 0
8      Si checkInconsistency () alors fails();
9      Si (X^* = ∅)
10         retourner vrai;
11     Sinon
12         Echelonner (M) ;
13         Si checkInconsistency ()
14             fails();
15         Sinon
16             fixerVariables () ;
17     retourner vrai;
18 Fonction checkInconsistency () : Booléen
19     Pour chaque ligne r ∈ M faire
20         Si ((parity_r = 1) ∧ (Σ_{i=1}^{n} M[r][i] = 0)) alors
21             retourner vrai
22     retourner faux
23 Procédure fixerVariables ()
24     Pour chaque ligne r ∈ M faire
25         Si (Σ_{i=1}^{n} M[r][i] = 1) alors
26             dom(X_i) ← dom(X_i) − {1 − parity_r};
27             (X^+ ← X^+ ∪ {i} ∨ X^- ← X^- ∪ {i} ;)
```

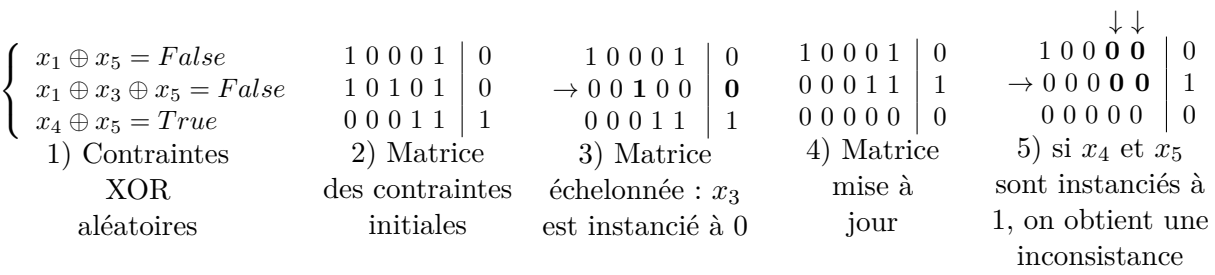


FIGURE 4.3 – Exemple de propagation de contraintes `XOR` utilisant l'élimination de Gauss.

4.7 Conclusions

Dans ce chapitre, nous avons proposé un premier encodage de la contrainte de diversité (i.e. contrainte de Jaccard maximum) qui exploite des contraintes linéaires. Nous avons montré comment exploiter cet encodage pour extraire des motifs fréquents, fermés et diversifiés. Ensuite, nous avons introduit un second encodage permettant d'utiliser des relaxations anti-monotones de l'indices de Jaccard pour l'extraction de motifs diversifiés. Ces relaxations exploitent des bornes inférieures et supérieures de l'indice de Jaccard afin d'élaguer de façon efficace les motifs non diversifiés. Notre approche est formalisée par la contrainte globale ClosedDiversity qui permet de contrôler la diversité des motifs avec un seuil de similarité maximum J_{max}. Enfin, nous avons montré comment exploiter la contrainte globale ClosedDiversity au sein d'un outil d'échantillonnage afin de contrôler de manière explicite la diversité des motifs échantillonnés.

Ces différentes méthodes feront l'objet d'évaluations dans le chapitre 5.

CHAPITRE 5

Évaluations des méthodes de réductions de la redondance

Contents

5.1 Étude expérimentale de ClosedDiversity 96
5.1.1 Jeux de données de FIMI et paramétrages 96
5.1.2 Choix des méthodes de l'état de l'art 96
5.1.3 Implantation et évaluation 98
5.2 Résultats et Discussions 99
5.2.1 Apports d'une borne plus resserrée 99
5.2.2 Évaluations de la qualité de la diversification 99
5.2.3 Caractéristiques des motifs extraits 104
5.2.4 Taille des ensembles de motifs 105
5.2.5 Temps d'exécution 106
5.2.6 Évaluation des heuristiques de choix de variables 110
5.2.7 Analyse qualitative des bornes 112
5.3 Étude expérimentale de SDivJaX 113
5.3.1 Protocole expérimental 113
5.3.2 Temps d'exécution de SDivJaX 113
5.3.3 Impact de l'étape d'estimation des `XOR` sur SDivJaX 115
5.3.4 Évaluation de la qualité de diversification 116
5.4 Conclusions 117

Dans ce chapitre, nous présentons les différentes expérimentations que nous avons réalisé pour comparer et évaluer les apports pratiques de notre contrainte globale ClosedDiversity, par rapport aux méthodes complètes de réduction de la redondance comme PatternsTeam et Picker et aux méthodes d'échantillonnage de motifs comme Flexics, CFTP et Gibbs. Nous commençons par décrire les jeux de données utilisés, le protocole expérimental, ensuite nous discutons les résultats obtenus en termes de diversité des solutions, de temps de calcul et de quelques caractéristiques sur les motifs extraits. Enfin, nous évaluons les apports de SDivJaX et ses différentes variantes pour l'échantillonnage de motifs diversifiés en termes de temps de calcul.

TABLE 5.1 – Descriptif des bases de transactions retenues pour nos expérimentations

Base	$\|\mathcal{T}\|$	$\|\mathcal{I}\|$	ρ	Type de données	Taille
CHESS	3 196	75	49%	game steps	**239 700**
HEPATITIS	137	68	50%	medical data	**9 316**
KR-VS-KP	3 196	73	49.32%	game (King-Rook vs. King-Pawn)	**233 308**
HEART-CLEVELAND	296	95	47.37%	heart disease data	**28 120**
SPLICE1	3 190	287	21%	genetic sequences	**915 530**
MUSHROOM	8 124	119	19%	species of mushrooms	**966 756**
CONNECT	67 557	129	33%	game steps	**8 714 853**
BMS1	59 602	497	0.5%	web click stream	**29 622 194**
T10I4D100K	100 000	10	1%	synthetic dataset	**100 000 000**
T40I10D100K	100 000	40	4%	synthetic dataset	**100 000 000**
PUMSB	49 046	7 117	1%	census data	**349 060 382**
RETAIL	88 162	16 470	0.06%	retail market basket data	**1 452 028 140**

5.1 Étude expérimentale de ClosedDiversity

5.1.1 Jeux de données de FIMI et paramétrages

Nous avons sélectionné plusieurs bases réelles et synthétiques de grande taille du dépôt FIMI [7]. Ces bases possèdent différentes caractéristiques selon les domaines d'application. La table 5.1 résume pour chaque base, le nombre de transactions $|\mathcal{T}|$, le nombre d'items $|\mathcal{I}|$, la densité de la base ρ et la taille de la base (i.e., $|\mathcal{T}| \times |\mathcal{I}|$). Le choix des bases est motivé par la variété de leur nombre de transactions, nombre d'items et la densité. Certaines bases, comme HEPATITIS et CHESS sont très denses (resp. 50% et 49%). D'autres au contraire sont très creuses, comme T10I4D100K et RETAIL (resp. 1% et 0.06%). La différence entre ces deux types de bases est que les bases denses ont tendance à produire un nombre important de motifs fermés, comparées aux bases dites creuses. À noter que la taille des bases de notre benchmark varie de l'ordre $\approx$ de 10^5 à 10^9. Pour nos expérimentations, nous avons sélectionné les paramétrages ci-dessous :

- **Seuil de support minimum** θ : nous avons sélectionné pour chaque base trois seuils de fréquence afin d'obtenir différents nombres de motifs fréquents et fermés tels que $|Th(c)| \leq 15 \times 10^3$, $30 \times 10^3 \leq |Th(c)| \leq 10^6$, et $|Th(c)| > 10^6$. La seule exception concerne les bases très creuses comme RETAIL et PUMSB pour lesquelles nous n'arrivons pas à obtenir un grand nombre de motifs. Nous avons choisi d'utiliser CLOSEDPATTERNS comme modèle de base afin de déterminer des seuils de fréquences intéressants. Par la suite, nous désignons par MUSHROOM-5, l'instance correspondant à MUSHROOM avec un seuil $\theta = 5\%$.
- **Seuil de diversité** J_{max} : le choix de J_{max} est déterminant dans notre méthode car il permet de définir le niveau de redondance entre les motifs et permet ainsi de filtrer les motifs présentant beaucoup de similarité. Pour étudier l'impact de la variation de ce seuil sur les solutions obtenus, nous avons fait varier J_{max} entre 0.05 et 0.7. Pour les autres expérimentations, nous utiliserons 0.05 comme valeur par défaut.
- **Heuristiques de choix de variables** : nous avons évalué les deux heuristiques de choix de variables : MINCOV et WITNESS. CLOSEDDIV-MINCOV (resp. CLOSEDDIV-WITNESS) désignera alors l'heuristique exploitant MINCOV (resp. WITNESS).

5.1.2 Choix des méthodes de l'état de l'art

Afin de montrer l'intérêt de notre contrainte globale CLOSEDDIVERSITY, nous avons réalisé différentes comparaisons avec plusieurs méthodes de l'état de l'art. Ces comparaisons portent sur les temps d'exécution, la qualité des ensembles de motifs extraits en terme de diversité et certaines caractéristiques des motifs extraits. Ces comparaisons incluent aussi bien des méthodes

7. fimi.ua.ac.be/data

de réduction de la redondance (PatternsTeam, Picker et Krimp) que des méthodes d'échantillonnage (Flexics, CFTP et Gibbs) qui ne sont pas explicitement des méthodes de réduction de la redondance mais qui permettent d'introduire de la diversité dans les motifs extraits.

A. Méthodes de réduction de la redondance

Comme nous l'avons vu à la section 1.7, différentes approches de réduction de la redondance ont été proposées. Nous nous sommes particulièrement intéressés aux méthodes non supervisées comme PatternsTeam [94] et Picker [29]. Rappelons que l'ensemble des motifs extraits par ces méthodes n'est pas toujours diversifié au sens de notre contrainte de Jaccard maximum. Par exemple, lorsque la couverture d'un motif i_1 recouvre la moitié de la couverture d'un autre motif i_2, ces deux motifs seront considérés comme non-redondants et intéressants, mais avec ClosedDiversity ils ne seront pas retenus dans le résultat final car trop similaires (i.e. indice de Jaccard élevé).

PatternsTeam et Picker nécessitent en entrée un ensemble de motifs de taille k. Comme valeur de départ, nous avons fixé k au nombre de motifs obtenus avec ClosedDiversity. Cependant, lorsque les performances des deux méthodes ne permettent pas d'extraire les k motifs dans une limite de temps de 24 heures, notamment pour PatternsTeam, alors, nous fixons la valeur de k à 50, puis réduisons progressivement le nombre de motifs de 1 jusqu'à l'obtention de la bonne valeur de k. Les différentes valeurs de k de chaque méthode sont reportées dans la table 5.3.

Nous nous sommes également comparés à Krimp [138] qui permet de réduire la redondance de façon implicite en sélectionnant les motifs qui permettent de réaliser la meilleure compression du jeu de données. Dans nos expériences, Krimp sélectionne à partir de l'ensemble de tous les motifs fermés extraits à partir d'un jeu de données particulier.

B. Méthodes d'échantillonnage

Nous avons sélectionné trois méthodes d'échantillonnage de motifs :

- Flexics [51] qui exploite des contraintes `XOR` afin de partitionner l'espace de recherche en des cellules à partir desquelles les différents motifs sont extraits. Nous avons considéré les deux variantes de Flexics qui diffèrent par leurs oracles. La première, GFlexics, s'appuie sur le modèle réifié CP4IM. La seconde, EFlexics, utilise une extension de l'algorithme Eclat [146]. Pour nos expérimentations, nous avons lancé Weightgen avec des valeurs de $\kappa \in \{0,1,0,5,0,9\}$ [51]. Nous rapportons les résultats correspondant au meilleur réglage du paramètre κ.
- Gibbs [14] qui exploite une mesure d'intérêt subjective. Pour nos expérimentations, nous avons utilisé le même protocole expérimentale que [14] avec un nombre d'itérations p fixé à 1000 ($1k$) et à 10000 ($10k$).
- CFTP [22] qui exploite une approche en deux étapes pour échantillonner des motifs en utilisant une mesure de qualité φ. Dans nos expérimentations, nous avons utilisé une mesure de qualité $\varphi = freq$ correspondant à la fréquence ainsi que des facteurs de fréquence $c \in \{2,3,\ldots,9\}$ noté $freq^c$.

C. Approches PPC exploitant l'indice de Jaccard

Pour évaluer l'intérêt de notre relaxation, nous avons comparé ClosedDiversity avec des approches PPC qui utilisent une évaluation exacte de l'indice de Jaccard :

- ClosedP+Jaccard qui applique un test de diversité à chaque fois qu'une nouvelle solution est trouvée avant de l'ajouter à l'historique $\mathcal{H}$. Cette méthode garantie une diversité parfaite de l'ensemble final de motifs, conformément à la contrainte de diversité.
- ClosedDiv+Jaccard qui applique le même test que précédemment avec ClosedDiversity afin d'éliminer les faux positifs.

— Le modèle FullCP-2 présenté à la section 4.2.2.

5.1.3 Implantation et évaluation

La mise en oeuvre des différentes contraintes globales ont été réalisées en Java, sous `Choco` [112], une bibliothèque Java dédiée au développement des programmes à contraintes. Nous avons implémenté nous même l'outil Picker et avons récupéré l'implantation de PatternsTeam auprès de ses auteurs. Toutes les expérimentations ont été menées sous Linux sur un **AMD Opteron** 6174**,** 2.2 **GHz** disposant d'une mémoire **RAM de** 256 **Go**. Nous avons utilisé une limite de temps de 24 heures et un espace mémoire alloué à la JVM de 30 Go. Comme seuil de diversité, nous avons fixé J_{max} à 5%. Nous avons également choisi d'utiliser comme heuristique par défaut de choix de variables Mincov. Pour le choix des valeurs, le branchement se fait toujours d'abord sur la plus grande valeur du domaine de chaque variable. Tous les codes sources de nos méthodes sont disponibles à l'adresse https://github.com/lobnury/ClosedDiversity.

Nos expérimentations ont pour objectif d'adresser les questions de recherche suivantes :

- **Q**$_1$ Comment ClosedDiversity se compare aux autres méthodes de réduction de la redondance et aux approches par échantillonnage (CFTP et Gibbs) en termes de **diversité de l'ensemble de motifs** ?
- **Q**$_2$ Comment les motifs extraits par ClosedDiversity se comparent en termes de **longueur** et de **couverture** autres méthodes spécialisées de l'état de l'art ?
- **Q**$_3$ Comment ClosedDiversity se compare, en termes de **temps de calcul** et de **nombre de motifs** extraits, aux approches PPC (ClosedPatterns, FullCP-2 et ClosedDiv+Jaccard) et aux autres méthodes spécialisées (Flexics, CFTP et Gibbs) ?
- **Q**$_4$ Comment les deux heuristiques de choix de variables Witness et Mincov se comparent en termes de temps de calcul ?
- **Q**$_5$ Quelle est la qualité des deux bornes en termes de distance les séparant de l'indice de Jaccard ?

L'évaluation de la qualité d'un ensemble peut être une tâche difficile étant donné que la perception de la qualité dépend des besoins de l'utilisateur.

Pour évaluer la diversité d'un ensemble de motifs, nous continuons d'utiliser l'indice de Jaccard de chaque paire de motifs. Cependant, les méthodes d'agrégation statistiques de ces valeurs de Jaccard, comme le maximum , la moyenne ou le minimum, ne permettent pas d'avoir une vision d'ensemble sur la diversité de l'ensemble des motifs. Par exemple, un ensemble de motifs, ayant un Jaccard moyen de 0.5 sur les différentes paires pourrait être constitué de motifs qui se chevauchent tous à peu près à moitié mais également de motifs X qui, étant donné un autre motif Y, ont exactement la même couverture que Y ou ont une couverture totalement disjointe. Pour mieux rendre compte de la diversité d'un ensemble de motifs, nous avons choisi de représenter de façon visuelle la *distribution* des indices de Jaccard des différentes paires. Comme les fonctions de densité de probabilité peuvent être sujettes à des variation dans la distribution conduisant des formes visuellement plutôt distinctes, nous avons décidé d'utiliser à la place les *fonctions de répartition cumulative* (ou Cumulative Distribution Function (CDF)) sur les indices de Jaccard de chaque paire de motifs. Soit un ensemble de motifs $\mathcal{H} = \{P_1, \ldots, P_k\}$, la distribution est représentée comme suit :

$$CDF_{\mathcal{H}}(\tau) = \#\{(i,j) | Jac(P_i, P_j) \leq \tau, 1 \leq i < j \leq k\} \cdot \frac{2}{k(k-1)}$$

$\frac{2}{k(k-1)}$ est le facteur de normalisation permettant d'avoir une distribution comprise entre 0 et 1. Ainsi, pour tout indice de Jaccard τ, $CDF_{\mathcal{H}}(\tau)$ indique la proportion de paires de motifs ayant un indice de Jaccard *inférieur* à τ. Ainsi, les courbes les plus à gauche sur les graphiques représentent les ensembles les plus diversifiés car indiquant une valeur de Jaccard plus faible. De même pour les courbes les plus hautes.

5.2 Résultats et Discussions

Dataset $\lvert\mathcal{I}\rvert \times \lvert\mathcal{T}\rvert$ $\rho(\%)$	$\theta(\%)$	#Motifs		Temps d'exécutions (s)		#Nœuds	
		LB_J	LB_J^{old}	LB_J	LB_J^{old}	LB_J	LB_J^{old}
Chess	20	**65**	96	**3.40**	5.87	**57**	436
75 × 3196	15	**238**	393	**26.18**	**75.40**	**1,154**	1,855
49.33%	10	**1,622**	4,204	**728.13**	**3825.29**	**7,774**	18,270
Heart-cleveland	10	**1,470**	3,496	**73.39**	257.39	**3,735**	**7,977**
95 × 296	8	**4,761**	12,842	**542.75**	2527.38	**11,441**	28,221
47.37%	6	**20,490**	58,240	**7668.52**	46163.06	**46,506**	124,705
Splice1	10	**413**	422	**27.75**	25.25	**825**	**843**
287 × 3190	5	**7,920**	8,781	**4214.48**	5616.47	**15,886**	**17,594**
20.91%	2	-	-	-	-	-	-
Mushroom	5	**548**	727	**52.21**	60.70	**1,357**	1,704
112 × 8124	1	**9,935**	12,139	**8976.82**	12532.95	**20,924**	25,154
18.75%	0.5	**23,931**	27,768	**50646.09**	64829.06	**49,406**	56,873
BMS1	0.15	**592**	609	**61531.32**	68312.38	**1,118**	1,220
497 × 59602	0.14	**647**	f668	**66287.70**	68049.00	**1,298**	1,339
0.51%	0.12	**778**	823	**74801.13**	79704.88	**1,560**	**1,651**

Table 5.2 – Analyse comparative des deux bornes inférieures

5.2.1 Apports d'une borne plus resserrée

Nous avons comparé expérimentalement notre nouvelle borne LB_J à l'ancienne borne proposée dans LB_J^{old} [79]. Les résultats de la table 5.2 montrent clairement l'intérêt de calculer des bornes plus serrées, plus particulièrement sur les instances denses. En effet, LB_J permet d'obtenir un meilleur filtrage des motifs non diversifiés et améliore les performances de LB_J^{old}. Sur les jeux de données denses, l'amélioration est plus importante. La borne LB_J permet pour certaines instances, comme Heart-Cleveland avec $\theta = 6\%$, de réduire le nombre de motifs de plus de moitié. Pour cette instance, le temps de calcul est également amélioré et passe de 46163.06 secondes pour LB_J^{old} à 7668.52 secondes pour LB_J. Le nombre de nœuds lui passe de 124705 pour LB_J^{old} à 46506 pour LB_J.

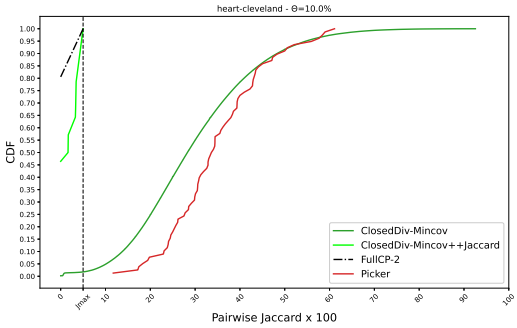
Pour les jeux de donnes modérément denses, les réductions en termes de nombres de motifs et de nœuds et en temps de calcul sont moins importantes. Pour autant, les résultats de LB_J demeurent meilleurs que ceux de LB_J^{old}.

5.2.2 Évaluations de la qualité de la diversification

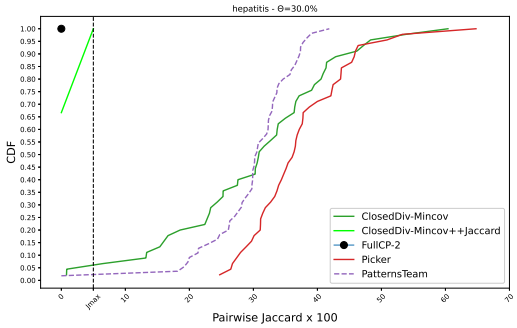
Les figures 5.1, 5.2, et 5.3 comparent les CDFs de ClosedDiversity et des autres méthodes. Des résultats complémentaires sur d'autres jeux de données de l'UCI sont disponibles en Annexe (voir figures A.3 et A.2).

a)- Comparaison des CDFs : ClosedDiv-Mincov vs Picker vs PatternsTeam

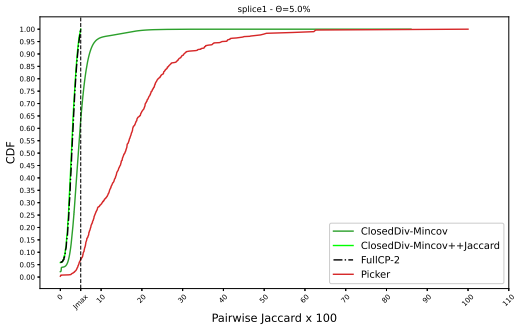
La figure 5.1 montre les résultats de comparaison avec Picker et PatternsTeam. Pour les jeux de données denses, nous remarquons ainsi que les différentes variantes de ClosedDiv extraient toujours des paires de motifs ayant un indice de Jaccard très faible. La fonction de répartition croît ensuite graduellement, dû au fait que nous utilisons une relaxation de l'indice de Jaccard pour extraire les motifs. Cependant, pour les approches qui utilisent une évaluation exacte de l'indice de Jaccard pour filtrer les motifs non diversifiés (FullCP-2 et ClosedDiv-Mincov+Jaccard), les graphiques des CDFs se positionnent toujours en haut à droite de la figure, ce qui traduit une meilleure diversité de l'ensemble de motifs produit par ces méthodes. Sur le jeu de données dense pour lequel Picker et PatternsTeam ont réussi à produire un résultat, nous pouvons voir que les valeurs Jaccard par paire ne sont jamais très faibles, puisque la



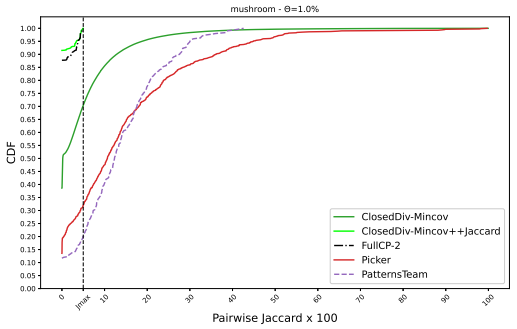
(a) HEART-CLEAVELAND ($\theta = 10\%$)



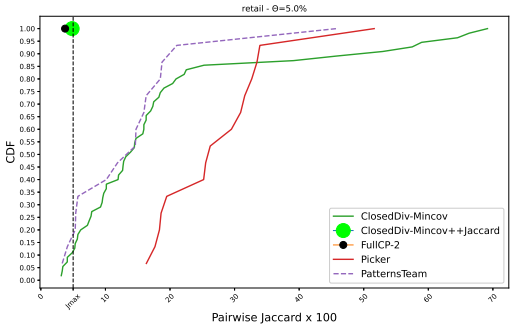
(b) HEPATITIS ($\theta = 30\%$)



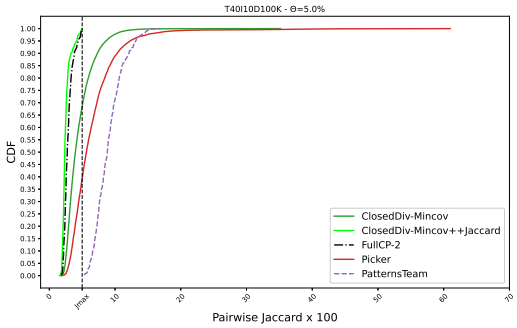
(c) SPLICE1 ($\theta = 5\%$)



(d) MUSHROOM ($\theta = 1\%$)



(e) RETAIL ($\theta = 5\%$)

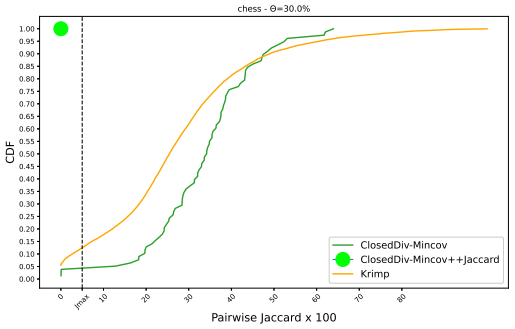


(f) T40I10D100K ($\theta = 5\%$)

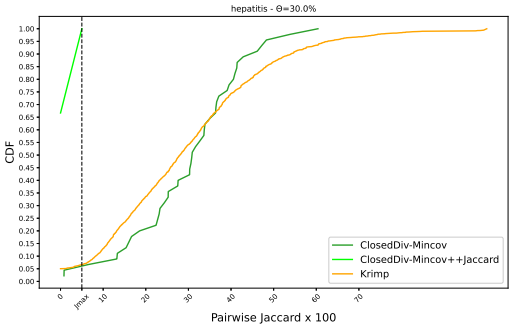
FIGURE 5.1 – Évaluation de la redondance globale des paires de motifs de CLOSEDDIV, PICKER, PATTERNSTEAM et FULLCP-2. Les abscisses sont étiquetés avec les valeurs de Jaccard des paires de motifs (multipliées par 100 pour une meilleure lisibilité) et les ordonnées avec des valeurs de CDFs associées comprises entre 0 et 1. Pour toute valeur Jaccard donnée, les courbes indiquent quel pourcentage de paires de motifs a un indice de Jaccard inférieur à cette valeur.

courbe des CDFs est inférieur à celles de CLOSEDDIV et FULLCP-2 au début. Ceci est un effet du critère de sélection des deux méthodes, qui préfèrent les motifs qui se chevauchent partiellement avec d'autres pour produire une partition de données bien équilibrée, mais rejettent également ceux qui se chevauchent trop fortement.

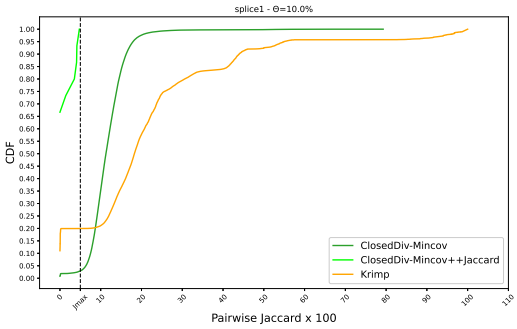
Pour les jeux de données modérément denses et creux, les courbes des CDFs sont beaucoup plus raides, ce qui indique qu'il est plus facile de trouver des paires de motifs avec un faible Jaccard. Là encore, les graphiques de PICKER et PATTERNSTEAM restent en dessous de ceux des autres approches, à l'exception de RETAIL avec $\theta = 5\%$, qui ne produit que très peu de motifs. Comme le nombre de motifs extraits sur ces jeux de données est plus faible, les motifs sont moins susceptibles de se chevaucher. Ainsi, la sélection de motifs présentant quelques chevauchements, comme le font PICKER et PATTERNSTEAM, entraînera donc des valeurs de Jaccard pour chaque paire plus mauvaises que lors de l'utilisation d'une méthode permettant de réduire redondance.



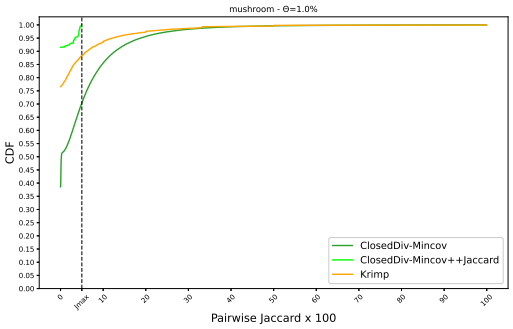
(a) Chess ($\theta = 30\%$)



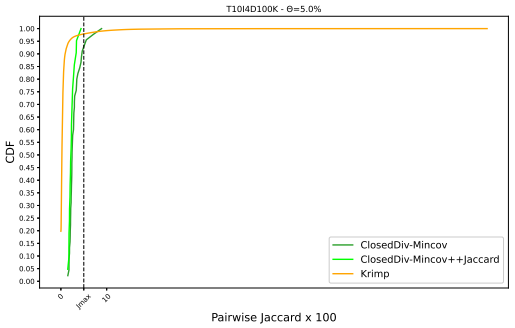
(b) Hepatitis ($\theta = 30\%$)



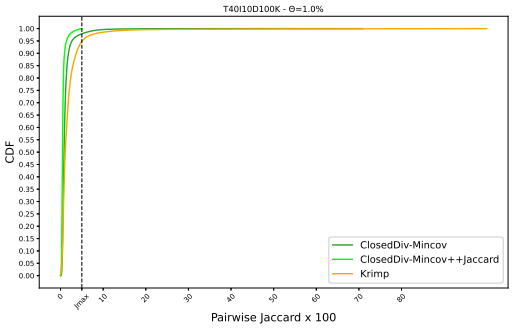
(c) Splice1 ($\theta = 10\%$)



(d) Mushroom ($\theta = 1\%$)



(e) T10I4D100K ($\theta = 5\%$)



(f) T40I10D100K ($\theta = 1\%$)

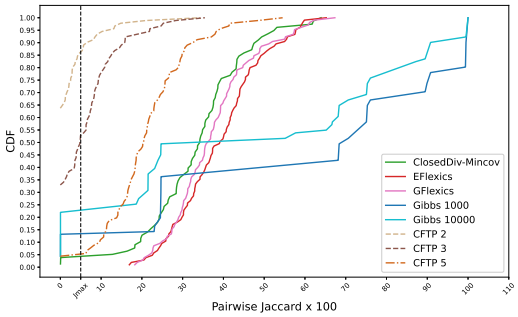
Figure 5.2 – Évaluation de la redondance globale des paires de motifs de ClosedDiv-Mincov et Krimp

b)- Comparaison des CDFs : ClosedDiv-Mincov vs Krimp

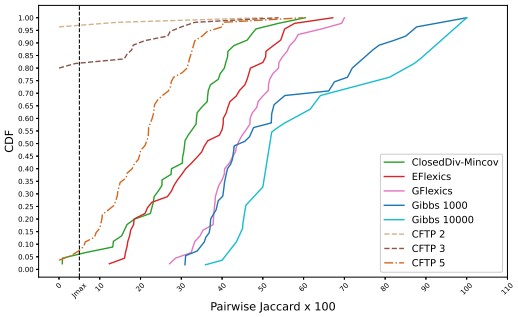
Les résultats de la comparaison avec Krimp, illustrés à la figure 5.2, semblent surprenants : alors que ClosedDiv-Mincov+Jaccard, à l'exception de T10I4D100K, conduit toujours à une meilleure distribution de l'indice de Jaccard par paires, Krimp reste colmpétitif et permet d'obtenir des résultats plus ou moins meilleurs que ceux de ClosedDiv-Mincov. Sur les jeux de données denses comme Chess et Hepatitis, les graphiques indiquent que les ensembles de motifs de Krimp donnent, en proportion, les meilleures valeurs de Jaccard, avant de tomber finalement en dessous de ClosedDiv-Mincov. Sur Mushroom (modérément dense) et T10I4D100K (creux), les graphiques montrent que Krimp reste meilleur à $\theta = 5\%$. Par contre, sur Splice, ClosedDiv-Mincov a systématiquement de meilleures valeurs de paires de Jaccard. Une autre observation intéressante à relever et que Krimp produit plus de paires de motifs ayant des indice de Jaccard élevés. Cela peut s'expliquer par la nature de la méthode, qui nécessite de rajouter de nouveaux motifs à l'ensemble qui ne sont pas diversifiés et qui seraient rejetés par ClosedDiv-Mincov.

c)- Comparaison des CDFs : ClosedDiv-Mincov vs Flexics vs CFTP vs Gibbs

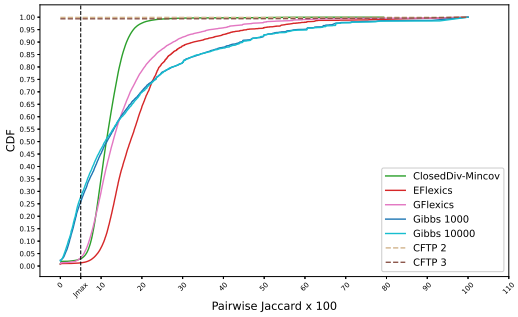
La figure 5.3 montre les résultats des méthodes d'échantillonnage. Premièrement, nous consta-



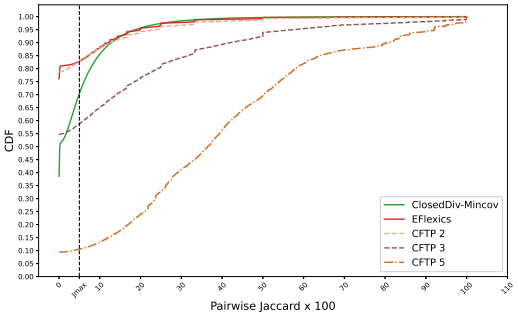
(a) Chess ($\theta = 30\%$)



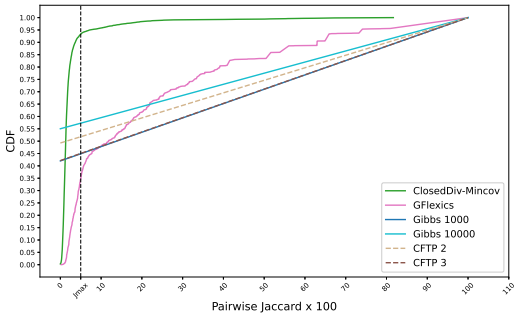
(b) Hepatitis ($\theta = 30\%$)



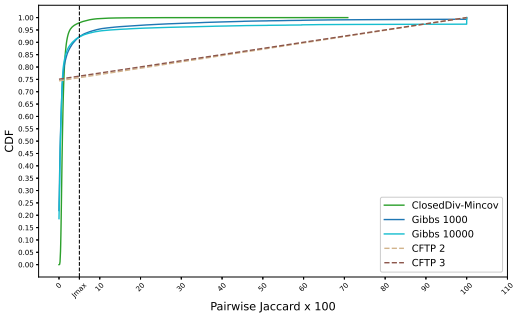
(c) Splice1 ($\theta = 10\%$)



(d) Mushroom ($\theta = 1\%$)



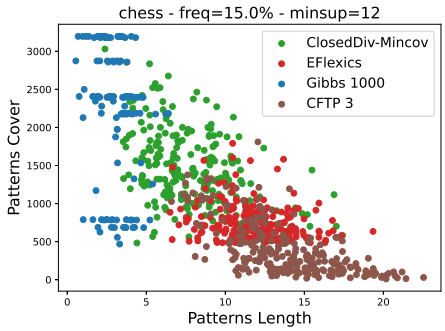
(e) Retail ($\theta = 1\%$)



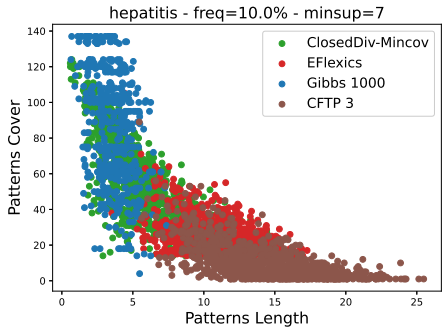
(f) T40I10D100K ($\theta = 1\%$)

Figure 5.3 – Évaluation de la redondance globale des paires de motifs de ClosedDiv-Mincov, Flexics, Gibbs et CFTP. Les abscisses sont étiquetés avec les valeurs de Jaccard des paires de motifs (multipliées par 100 pour une meilleure lisibilité) et les ordonnées avec des valeurs de CDFs associées comprises entre 0 et 1. Pour toute valeur Jaccard donnée, les courbes indiquent quel pourcentage de paires de motifs a un indice de Jaccard inférieur à cette valeur

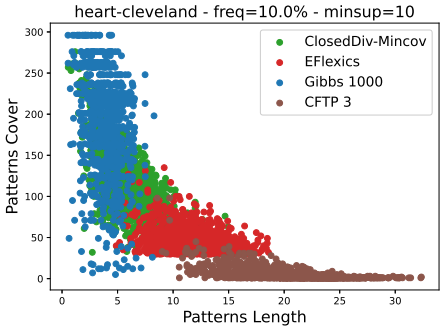
tons que les motifs obtenus par Gibbs conduisent à des indices de Jaccard plus élevés sur l'ensemble des paires de motifs. Au contraire, CFTP assemble des ensembles à faible redondance sur les jeux de données denses. Toutefois, cette faible redondance devient plus importante à mesure que le facteur de fréquence c augmente. Cela s'explique, comme nous le verrons plus tard, par le fait que les motifs échantillonnés avec un facteur de fréquence faible ont un support très bas, ce qui réduit les possibilités de redondance dans l'ensemble solution. Ainsi, pour $c = 5$, les motifs sont beaucoup plus redondants, et sur Mushroom conduisent de loin à des valeurs Jaccard les plus élevées. Enfin, EFlexics et GFlexics conduisent à une distribution plus ou moins bonne des valeurs de Jaccard, bien que les résultats de ces deux méthodes soient moins bons que ceux de ClosedDiv-Mincov.



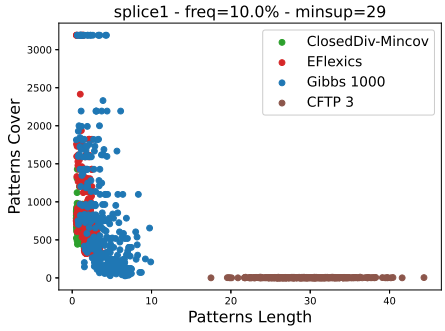
(a) Chess ($\theta = 15\%$)



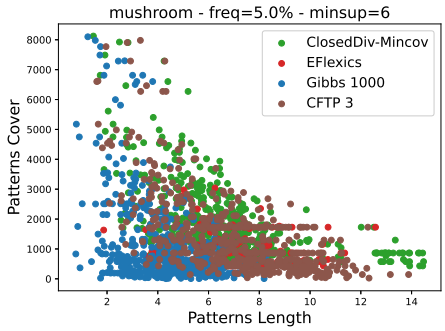
(b) Hepatitis ($\theta = 10\%$)



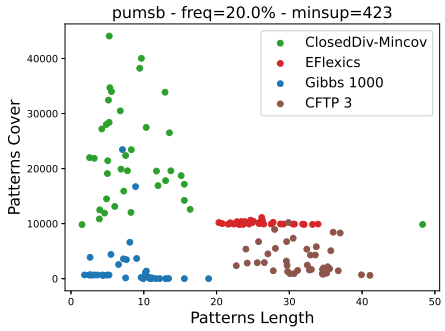
(c) Heart-cleveland ($\theta = 10\%$)



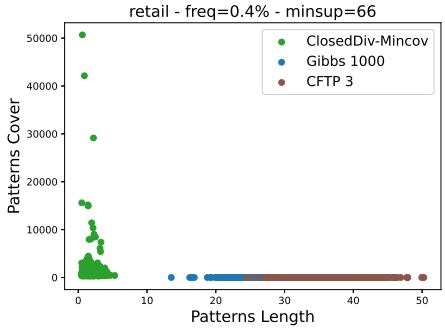
(d) Splice1 ($\theta = 10\%$)



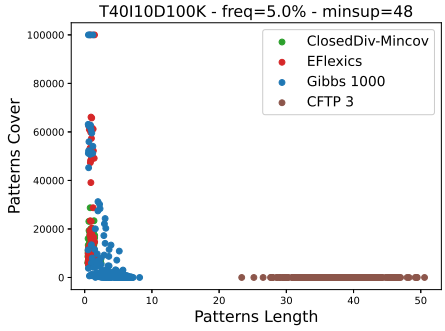
(e) Mushroom ($\theta = 5\%$)



(f) Pumsb ($\theta = 20\%$)



(g) Retail ($\theta = 0.4\%$)



(h) T40I10D100K ($\theta = 5\%$)

Figure 5.4 – Nuages de points obtenus à partir des descriptions des motifs, c'est à dire la longueur et la taille de la couverture, sur différents jeux de données.

5.2.3 Caractéristiques des motifs extraits

Dans la section 5.2.2, nous avons démontré la capacité de ClosedDiv à réduire la redondance des ensemble de motifs. Nous examinons dans cette section les caractéristiques des motifs extraits (taille et couverture) par les différentes approches. La figure 5.4 montre les nuages de points obtenus à partir des descriptions des motifs, c'est à dire la longueur et la taille de la couverture. Ce graphique est très informatif car il nous renseigne sur la forme des motifs découverts par ClosedDiv et par les méthodes d'échantillonnage. Gibbs favorise l'échantillonnage de motifs courts avec une grande couverture, tandis que CFTP (avec $c = 3$ comme facteur de fréquence) privilégie l'échantillonnage de motifs longs ne couvrant que peu de transactions. C'est la raison pour laquelle les indices Jaccard entre toutes les paires de motifs renvoyées par CFTP sont très faibles. En effet, avec de petites couvertures (moins de 10 transactions), il est très probable que deux couvertures ne se croisent pas, ce qui conduit à un indice Jaccard de 0. ClosedDiv-Mincov se situe au milieu, et renvoie des motifs bien distribués en termes de longueur et de taille de couverture. Flexics est également capable de renvoyer des motifs de diverses longueurs mais moins distribués en termes de taille de couverture. Sur des jeux de données très creux, bien que les motifs retournés par CFTP et Gibbs sont de taille variable, ils ne couvrent malheureusement qu'une ou deux transactions.

Jeux de données		#Motifs							#Nœuds				
$\lvert\mathcal{I}\rvert \times \lvert\mathcal{T}\rvert$ $\rho(\%)$	$\theta(\%)$	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(1)	(2)	(3)	(4)	(5)
Chess	30	5,316,468	**2**	14	**2** (433,165)	**2**	OOM	OOM	10,632,935	10,632,935	**57**	866,335	5,316,469
75 × 3,196	20	22,808,625	**3**	65	**4** (194,270)	-	OOM	OOM	45,617,249	45,617,249	**318**	388,545	-
49.33%	15	50,723,131	**4**	238	**5** (73,152)	-	OOM	OOM	101,446,261	101,446,261	**1,154**	146,313	-
	10	OOM	**8**	1,622	**8** (36,732)	-	OOM	OOM	246,486,145	**7,774**	73,483	-	
Hepatitis	30	83,048	**2**	11	**3** (3,485)	**2**	10	11	166,095	166,095	**28**	6,972	83,049
68 × 137	20	410,318	**4**	45	**4** (3,913)	**4**	10	11	820,635	820,635	**129**	7,829	41,0321
50.00%	10	1,827,264	**9**	1,018	**9** (12,408)	**9**	10	11	3,654,527	3,654,527	**2,545**	24,829	1,826,753
Kr-vs-kp	30	5,219,727	**2**	14	**2** (432,426)	**2**	OOM	OOM	10,439,453	10,439,453	**57**	864,857	5,219,728
73 × 3,196	20	21,676,719	**3**	64	**4** (138,029)	**3**	OOM	OOM	43,353,437	43,353,437	**307**	276,063	21,676,721
49.32%	10	OOM	**7**	1,609	**7** (27,601)	-	OOM	OOM	OOM	218,876,381	**7,703**	55,221	-
Connect	30	460,357	**1**	19	3 (106,705)	-	OOM	OOM	920,713	920,713	**89**	213,415	-
129 × 67,557	18	2,005,476	**3**	141	4 (542,858)	-	OOM	OOM	4,010,951	4,010,951	**699**	1,085,725	-
33.33%	15	3,254,780	**4**	297	5 (519,639)	-	OOM	OOM	6,509,559	6,509,559	**1,389**	1,039,290	-
	10	8,035,412	**7**	1,907	9 (73,634)	-	OOM	OOM	16,070,823	16,070,823	**7,815**	147,302	-
Heart-cleveland	10	12,774,456	9	1,470	**8** (37,115)	9	13	OOM	25,548,911	25,548,911	**3,735**	74,233	12,774,464
95 × 296	8	23,278,687	13	4,761	**10** (4,817)	13	14	OOM	46,557,373	46,557,373	11,441	**9,648**	23,278,699
47.37%	6	43,588,346	19	20,490	**12** (10,718)	19	13	OOM	87,176,691	87,176,691	46,506	**21,456**	43,588,355
Splice1	10	1,606	**5**	413	6 (1,042)	5	50	OOM	3,211	3,211	**825**	2,083	1,610
287 × 3,190	5	31,441	99	7,920	**89** (19,449)	99	43	OOM	62,881	62,881	**15,886**	38,898	31,539
20.91%	2	589,588	448	-	**416** (355,612)	448	??	OOM	1,179,175	1,179,175	-	**711,229**	590,035
Mushroom	5	8,977	13	548	**11** (881)	13	37	25	17,953	17,953	**1,357**	1,768	8,989
112 × 8,124	1	40,368	110	9,935	**72** (2,119)	110	41	25	80,735	80,735	20,924	**4,249**	40,477
18.75%	0.8	47,765	199	12,743	**90** (2,554)	199	41	25	95,529	95,529	26,660	**5,128**	47,963
	0.5	62,334	278	23,931	**121** (3,056)	278	42	25	124,667	12,4667	49,406	**6,120**	62,611
T40I10D100K	8	138	**7**	125	10 (129)	**7**	121	30	275	275	249	257	**144**
942 × 100,000	5	317	**22**	284	33 (289)	**22**	123	30	633	633	567	577	**338**
4.20%	1	65,237	**373**	7,217	395 (9,774)	-	135	OOM	130,473	130,473	**14,517**	19,631	-
Pumsb	40	-	-	**4**	-	-	OOM	OOM	-	-	**15**	-	-
2,113 × 49,046	30	-	-	**14**	-	-	OOM	OOM	-	-	**59**	-	-
3.50%	20	-	-	**39**	-	-	OOM	OOM	-	-	**206**	-	-
T10I4D100K	5	11	**6**	11	7 (77)	**6**	10	10	21	21	21	21	**16**
870 × 100,000	1	386	**83**	360	93 (363)	**83**	375	30	771	771	720	726	**468**
1.16%	0.5	1,074	**185**	607	208 (650)	**185**	555	30	2,147	2,147	**1,238**	1,323	1,259
BMS1	0.15	1,426	**69**	592	73 (926)	**69**	303	30	2,851	2,851	**1,186**	1,854	1,495
497 × 59,602	0.14	1,683	**67**	647	75 (980)	**67**	311	30	3,365	3,365	**1,298**	1,961	1,750
0.51%	0.12	2,374	**76**	778	79 (909)	**76**	326	30	4,747	4,747	**1,560**	1,819	2,450
retail	5	17	**2**	12	**2** (13)	**2**	6	6	33	33	23	25	**18**
16470 × 88,162	1	160	**60**	105	63 (114)	**60**	70	30	319	319	**218**	234	219
0.06%	0.4	832	275	515	**272** (550)	275	317	30	1663	1663	**1,071**	1,143	1106

Table 5.3 – Analyse comparative du nombres de motifs et du nombre de nœuds explorés par les différentes approches PPC. (1) : ClosedP (2) : ClosedP+Jaccard (3) ClosedDiv-Mincov (4) : ClosedDiv-Mincov+Jaccard (5) : FullCP-2 (6) : Picker (7) : PatternsTeam. " – " dépassement de la limite de temps. OOM : out of memory.

5.2.4 Taille des ensembles de motifs

La table 5.3 indique, pour chaque jeux de données et seuil de fréquence, le nombre de motifs extraits par ClosedP, ClosedDiv-Mincov, Picker, PatternsTeam ainsi que par les approches PPC qui effectuent une évaluation exacte de l'indice de Jaccard. Nous rapportons également le nombre de noeuds explorés. Les valeurs entre parenthèses désignent le nombre de motifs intermédiaires non diversifiés générés par ClosedDiv-Mincov+Jaccard avant l'étape de filtrage des faux positifs.

a)- ClosedDiv-Mincov vs ClosedPatterns :

Les résultats mettent en évidence une grande différence entre les deux approches avec un nombre nettement inférieur de motifs générés par ClosedDiv-Mincov (en milliers) par rapport à ClosedPatterns (en millions). Sur les jeux de données denses et moyennement denses (de Chess à Mushroom), l'écart est largement amplifié, en particulier pour les petites valeurs de θ. Par exemple, sur Chess, le nombre de motifs extraits par ClosedDiv-Mincov est réduit de 99.9% (de $\sim 50 \cdot 10^6$ motifs à 238) pour θ égal à 15%. La densité des ces bases explique le bon comportement de ClosedDiv-Mincov. En effet, comme le nombre de motifs fermés augmente avec la densité, la redondance entre ces motifs augmente également. Sur les jeux de données très creux, ClosedDiv-Mincov produit toujours moins de motifs que ClosedPatterns mais la différence est moins prononcée. Cela s'explique par le fait que sur ces jeux de données, nous avons peu de motifs et que presque tous les motifs sont diversifiés.

b)- ClosedDiv-Mincov vs ClosedP+Jaccard vs ClosedDiv-Mincov+Jaccard vs FullCP-2 :

Comme le montre la table 5.3, le nombre de motifs obtenus avec ClosedP+Jaccard est inférieur à ceux de ClosedPatterns et ClosedDiv-Mincov, en particulier sur les jeux de données denses où la réduction est très impressionnante par rapport à ClosedPatterns (de $\sim 10^6$ à 20 motifs). Plus important encore, ClosedP+Jaccard permet de terminer l'extraction sur les instances où ClosedPatterns se heurte à un *out of memory* (voir Chess et Kr-vs-kp avec un seuil θ de 10%). Sur les jeux de données creux, puisque le nombre de motifs est très faible, la réduction n'est pas énorme. Il est intéressant de remarquer que sur les jeux de données denses, ClosedDiv-Mincov+Jaccard permet de filtrer un grand nombre de faux positifs conduisant à une grande diversité du résultat final. Enfin, sur les instances où FullCP-2 termine l'extraction, ClosedP+Jaccard et FullCP-2 obtiennent le même ensemble de motifs diversifié puisqu'ils explorent l'espace de recherche de la même manière ; la seule différence réside dans la manière dont la diversité de l'ensemble de résultats est assurée. Ainsi, comme le montrent les résultats, ClosedP+Jaccard et FullCP-2 extraient généralement moins de motifs que ClosedDiv et ClosedDiv-Mincov+Jaccard. Les seules exceptions sont les trois jeux de données Heart-cleveland, Splice1 et Mushroom, où ClosedDiv-Mincov+Jaccard extrait moins de motifs.

c)- ClosedDiv-Mincov vs Picker vs PatternsTeam :

Comme le montre les résultats, Picker peut extraire plus de motifs que ClosedDiv-Mincov+Jaccard, notre méthode de réduction de la redondance la plus restrictive. Nous pouvons donc nous attendre à ce que les ensembles de motifs extraits par Picker montrent plus de redondance, car minimiser la redondance n'est pas vraiment l'objectif de Picker, comme mentionné ci-dessus. PatternsTeam extrait moins de motifs que Picker en raison de l'inefficacité de l'implantation de la méthode. Nous avons donc choisi des valeurs de k plus petites, pour limiter la taille des ensembles dans le but d'obtenir des résultats illustratifs. Ces résultats ne peuvent donc pas être utilisés directement pour commenter la redondance des ensembles de motifs dérivés à partir de PatternsTeam.

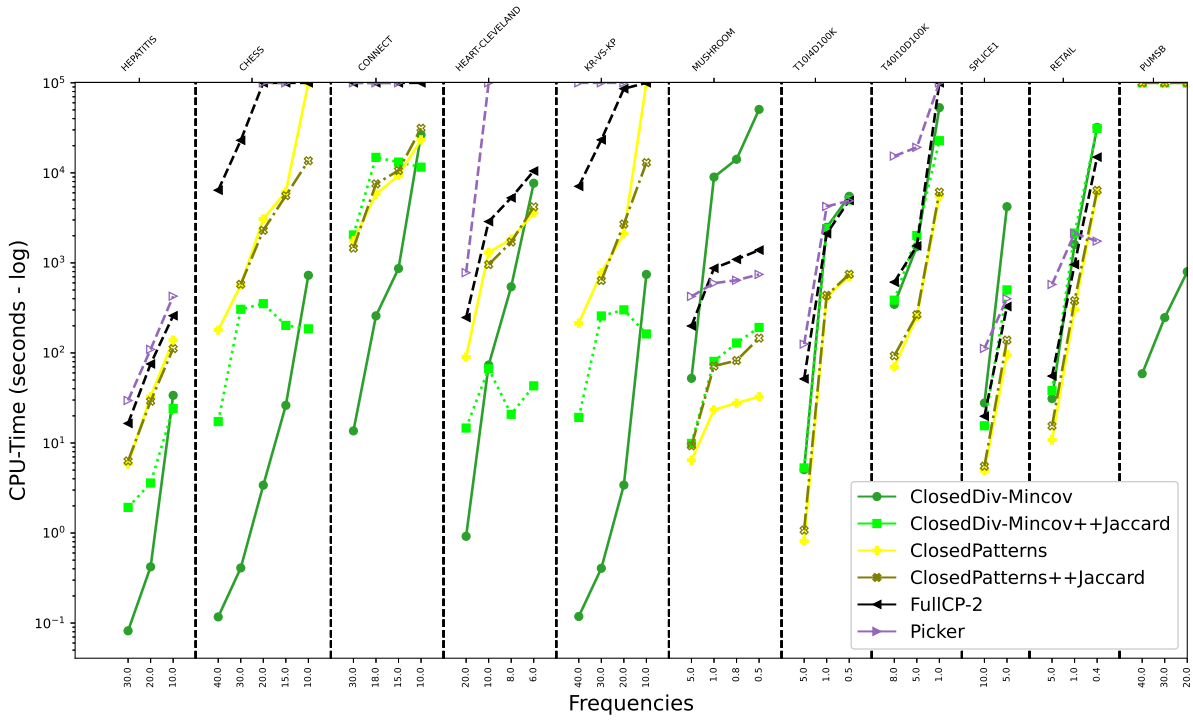


FIGURE 5.5 – Temps d'exécution de CLOSEDP, CLOSEDP+JACCARD, CLOSEDDIV-MINCOV, CLOSEDDIV-MINCOV+JACCARD et FULLCP-2 sur les jeux de données UCI pour différents seuils θ

5.2.5 Temps d'exécution

Les temps d'exécution (en secondes) de toutes les approches sur tous les jeux de données et les seuils associés sont présentés dans les figures 5.5 et 5.6. Nous avons utilisé une échelle logarithmique et avons utilisé un seuil $J_{max} = 0.05$ et $\kappa = 0.5$. Pour les méthodes d'échantillonnage, nous avons fixé le nombre de motifs à échantillonner au nombre de motifs extraits par CLOSEDDIV.

a)- CLOSEDDIV-MINCOV vs CLOSEDP :

CLOSEDDIV-MINCOV présente différents comportements en fonction de la nature du jeu de données. Sur les jeux de données denses ($\rho \geq 30\%$), CLOSEDDIV-MINCOV est plus efficace que CLOSEDPATTERNS et jusqu'à trois ordres de grandeur plus rapide. Sur CHESS (resp. CONNECT), l'accélération est de 1382 (resp. 129) pour $\theta = 30\%$. Pour les instances résultant entre 300 et 5000 motifs fréquents, fermés et diversifiés, CLOSEDDIV-MINCOV est souvent 17 fois plus rapide. Une autre observation importante qui peut être faite est que sur certaines instances (CHESS et KR-VS-KP) avec un seuil de fréquence minimum de %10, CLOSEDPATTERNS ne parvient pas à terminer l'extraction dans la limite de temps. Les bonnes performances de CLOSEDDIV-MINCOV sont principalement dues aux règles de filtrage de LB qui permettent de réduire l'espace de recherche en filtrant plus de valeurs inconsistantes. En effet, CLOSEDDIV-MINCOV produit un nombre de nœuds toujours plus réduit que CLOSEDP. Sur les jeux de données denses, les gains sont très importants avec une moyenne de gains d'environ 99 %. La seule exception est HEART-CLEVELAND pour lequel CLOSEDDIV-MINCOV est plus lent pour les valeurs de $\theta \leq 6\%$. Cela est principalement dû au nombre relativement important de motifs diversifiés (≥ 20000), qui induit un sur-coût élevé pour le calcul de la borne inférieure. Nous observons le même comportement sur les deux jeux de données modérément denses SPLICE1 et MUSHROOM. Sur des jeux de données creux, CLOSEDDIV-MINCOV peut prendre beaucoup plus de temps pour extraire tous les motifs fréquents, fermés et diversifiés. Cela peut s'expliquer par le fait que sur ces instances presque tous les motifs fermés sont diversifiés par rapport à notre borne inférieure LB (en moyenne environ 65 % pour RETAIL et 37 % pour BMS1, (voir Table 5.3)). De ce fait, les motifs non

diversifiés sont rarement filtrées, tandis que le sur-coût du calcul de la borne inférieure pénalise fortement la recherche des solutions. Cela explique également la légère différence dans le nombre de nœuds exploré par les deux méthodes. Enfin, sur jeu de données très creuxPumsb, notre approche est très efficace alors que ClosedPatterns ne parvient pas à terminer l'extraction.

b)- ClosedDiv-Mincov vs ClosedP+Jaccard vs ClosedDiv-Mincov+Jaccard :

Comme le montre la figure 5.5, ClosedP+Jaccard n'est pas efficace et reste très lent par rapport à ClosedDiv-Mincov. Cela s'explique par le grand nombre de motifs fermés que ClosedP+Jaccard doit tester pour assurer la diversité de l'ensemble final. Les seules exceptions concernent les instances avec un historique de très grande taille $\mathcal{H}$ ($\geq$ 20000), où ClosedP+Jaccard est efficace. Sur des jeux de données modérément denses et creux, même si ClosedP+Jaccard nécessite plus de temps que ClosedP, elle reste toujours meilleur que ClosedDiv-Mincov. En effet, comme montré précédemment, ClosedPatterns présente des temps d'exécution inférieurs à ClosedDiv-Mincov et le sur-coût engendré par le test de la contrainte de Jaccard reste négligeable par rapport à celui induit par notre calcul de la borne inférieure. En comparant les comportements de ClosedDiv-Mincov et ClosedDiv-Mincov+Jaccard, nous observons une corrélation entre la taille de l'historique généré et le temps d'exécution. Ainsi, sur les jeux de données denses, ClosedDiv-Mincov surpasse clairement ClosedDiv-Mincov+Jaccard, notamment sur les instances où la taille de l'historique de ClosedDiv-Mincov reste raisonnablement petite et le nombre de solutions intermédiaires générées par ClosedDiv-Mincov+Jaccard est très grand. Cependant, lorsque l'historique devient suffisamment grand ($\geq$ 1000), ClosedDiv-Mincov+Jaccard devient plus efficace comparé à ClosedDiv-Mincov. Ce résultat surprenant peut s'expliquer par la petite taille de l'historique de ClosedDiv-Mincov+Jaccard, qui réduit considérablement le sur-coût lié au calcul de la borne inférieure. Sur des jeux de données modérément denses, on observe le même comportement : ClosedDiv-Mincov+Jaccard est toujours meilleur que ClosedDiv-Mincov et est très comparable à ClosedP+Jaccard. Enfin, sur des jeux de données très creux, aucune approche ne domine clairement l'autre.

Le modèle FullCP-1 se heurte au problème *d'out of memory* sur toutes les instances considérées, dû au nombre très important de contraintes réifiées qui pénalisent fortement le modèle. Ainsi, nous rapportons et discutons uniquement les résultats du modèle FullCP-2 (voir Figure 5.5). Sur les jeux de données denses, FullCP-2 présente un mauvais comportement : il obtient un *Time Out* sur 10 instances (sur 21) et est toujours classé en dernier. Comparé à ClosedP+Jaccard, FullCP-2 reste très lent. Ces résultats montrent clairement l'intérêt de notre cadre de relaxation par rapport à FullCP-2. Sur des jeux de données modérément denses, FullCP-2 bat clairement ClosedDiv-Mincov mais reste moins efficace que ClosedDiv-Mincov+Jaccard : ClosedDiv-Mincov+Jaccard obtient les meilleurs temps d'exécution sur 5 instances, tandis que FullCP-2 est plus rapide sur 2 instances. Notons que ClosedP+Jaccard est classé deuxième sur ces instances. Sur des jeux de données très creux, excepté Pumsb et T40 où ClosedDiv-Mincov est très efficace, FullCP-2 se classe troisième. Encore une fois, ClosedP+Jaccard reste la deuxième meilleure méthode. Ces résultats sont cohérents avec nos observations précédentes puisque ClosedP+Jaccard et FullCP-2 ont tendance à explorer l'espace de recherche de la même manière car les deux exploitent la contrainte globale ClosedPatterns pour extraire des motifs fermés fréquents, mais le sur-coût lié à la gestion d'un grand nombre de contraintes linéaires supplémentaires (au plus k contraintes linéaires de diversité sont ajoutées où $k = |\mathcal{H}|$) pénalise fortement le modèle FullCP-2. Ces résultats démontrent clairement l'avantage de notre cadre de relaxation par rapport au modèle FullCP-2.

c)- ClosedDiv-Mincov vs Flexics vs CFTP vs Gibbs :

La figure 5.6 compare les temps d'exécution de ClosedDiv-Mincov avec ceux des méthodes d'échantillonnage de motifs sur différents jeux de données et différentes valeurs de seuil θ. Nous

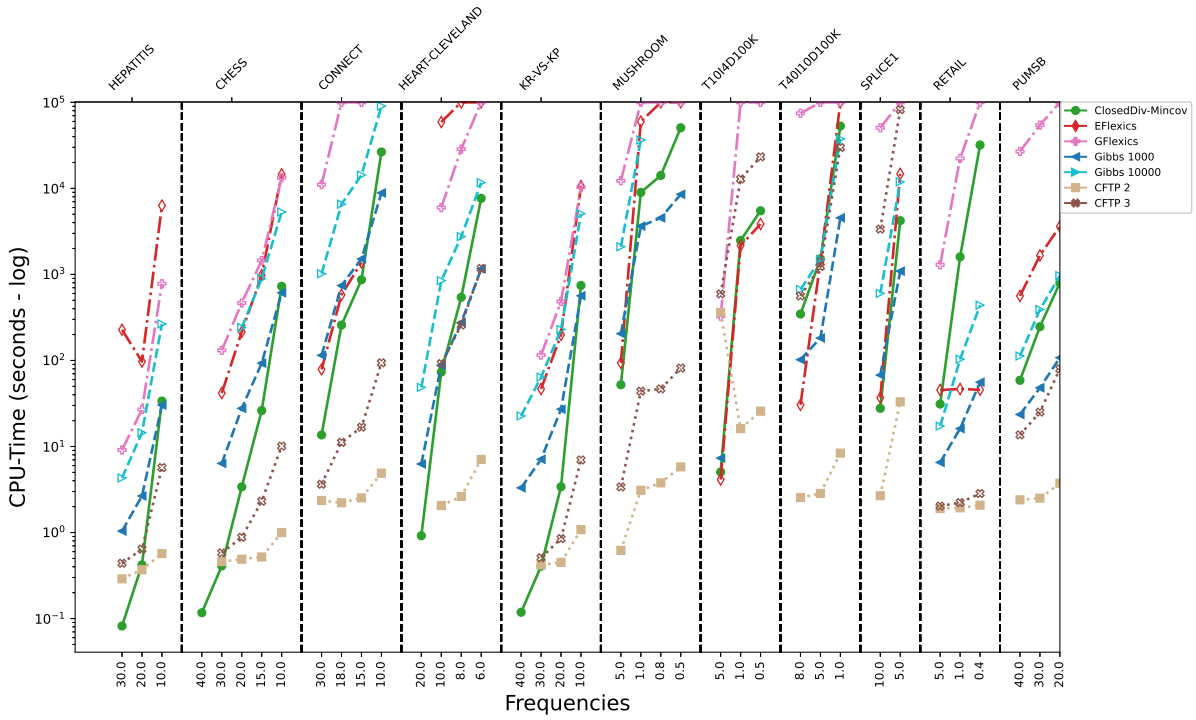
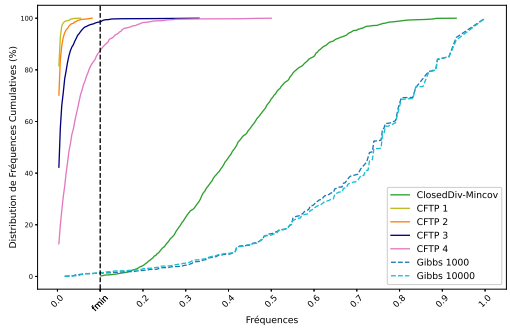


FIGURE 5.6 – Comparaison des temps d'exécution de CLOSEDDIV-MINCOV, FLEXICS, GIBBS et CFTP pour différents seuils θ

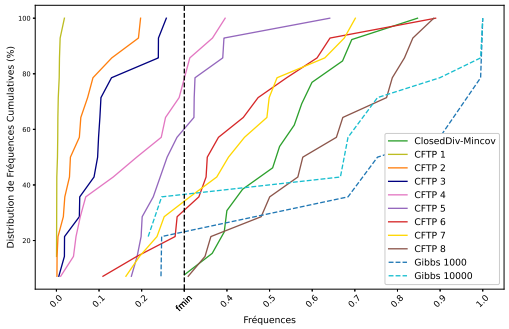
rapportons uniquement les résultats de CFTP pour $c = 2$ et $c = 3$. Les résultats complets sont donnés en annexe A (voir la figure A.1).

Premièrement, CLOSEDDIV-MINCOV domine largement GFLEXICS. Sur plusieurs instances, GFLEXICS ne parvient pas à générer le nombre d'échantillons demandés pour des seuils de fréquence très bas dans la limite de 24 heures. Alors que EFLEXICS est presque toujours plus rapide que GFLEXICS, notre approche est toujours mieux classée que les deux variantes de FLEXICS. La seule exception est le jeu de données RETAIL), démontrant ainsi son utilité pour extraire des motifs diversifiés dans un contexte anytime.

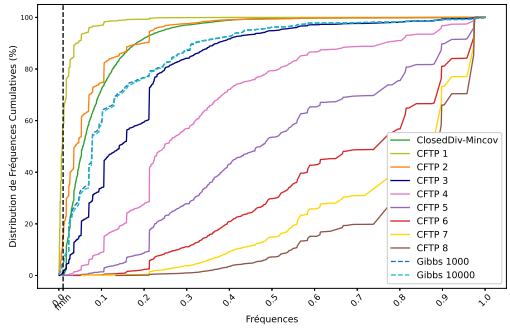
Deuxièmement, CFTP est presque toujours plus rapide que CLOSEDDIV-MINCOV pour des facteurs de fréquence compris entre 1 et 3. Pour ces instances, l'échantillonnage peut être effectué en moins de 10 secondes dans la plupart des cas. En revanche, pour $c = 4$ (voir la figure A.1), CFTP devient sensiblement plus lent et est même surpassé par CLOSEDDIV-MINCOV sur quatre jeux de données (CHESS, HEPATITIS, HEART-CLEVELAND et PUMSB). De plus, CFTP ne parvient pas à terminer l'extraction dans la limite de 24 heures sur les trois jeux de données SPLICE1, T40I10D100K et T10I4D100K. Ce comportement est grandement amplifié pour $c \geq 5$ où CFTP devient moins efficace, voire largement dépassé par CLOSEDDIV-MINCOV. Cependant, malgré les bonnes performances de CFTP, et comme le montrent les résultats des figures 5.7 et A.4, CFTP ne parvient pas à générer des motifs fréquents pour des valeurs de c comprises entre 1 et 3. Pour $c = 4$, plus de 80% des motifs échantillonnés ne sont pas fréquents par rapport au seuil minimal θ. Ce problème peut être partiellement résolu en biaisant la recherche vers des motifs plus fréquents grâce à un échantillonnage proportionnel à $freq^c$ avec $c \geq 5$. Cependant, sur des jeux de données creux, CFTP ne parvient pas à générer des motifs fréquents même pour des valeurs plus élevées de c. La seule exception est le jeu de données MUSHROOM où les motifs échantillonnés par CFTP sont fréquents. En résumé, la méthode CFTP offre des avantages substantiels en temps de calcul par rapport à CLOSEDDIV pour des valeurs de $c \in \{1, 2, 3\}$, mais ne fournit pas de garanties pour échantillionner des motifs fréquents. CLOSEDDIV-MINCOV offre donc clairement un bon compromis entre temps d'exécution et flexibilité.



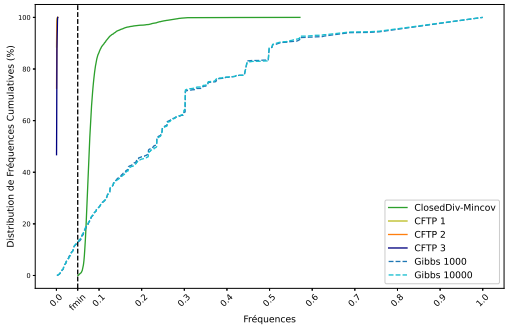
(a) Heart-cleveland ($\theta = 10\%$)



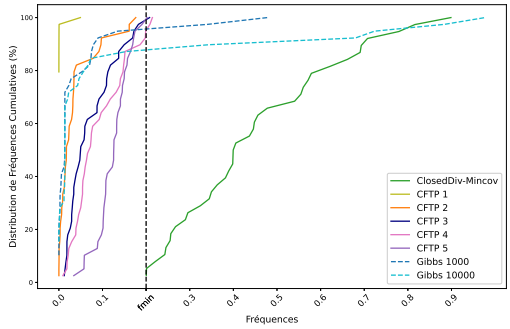
(b) Chess ($\theta = 30\%$)



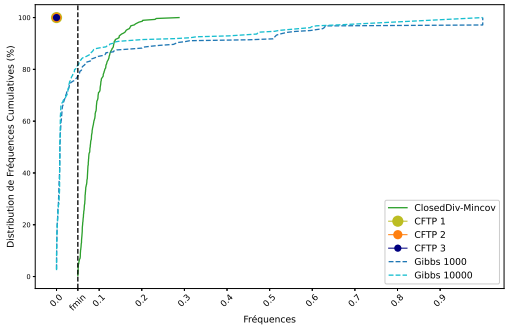
(c) Mushroom ($\theta = 5\%$)



(d) splice1 ($\theta = 5\%$)



(e) Pumsb ($\theta = 20\%$)



(f) T40I10D100K ($\theta = 5\%$)

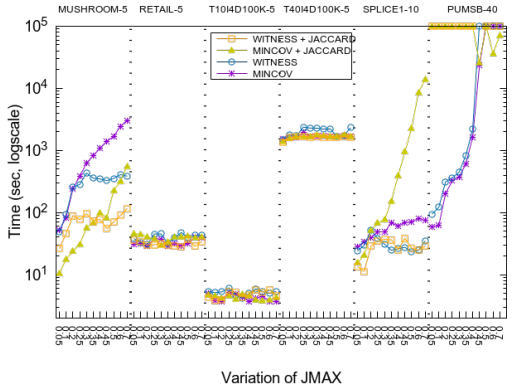
Figure 5.7 – Distribution de fréquences cumulatives des motifs

Troisièmement, la comparaison avec Gibbs montre que, sur des jeux de données denses et pour des valeurs de seuil minimum θ élevées, ClosedDiv-Mincov est considérablement plus rapide, en particulier lorsque le nombre d'itérations p augmente. En effet, la complexité dans le pire des cas de la procédure d'échantillonnage d'une tuile par Gibbs est $\mathcal{O}(p \times n \times m)$. Ainsi, sur Chess avec $\theta = 30\%$, ClosedDiv est 15 fois plus rapide que Gibbs avec $p = 1k$ itérations. Pour $p = 10k$ itérations, l'accélération atteint 130. Cependant, nous pouvons observer que Gibbs offre des avantages substantiels par rapport à ClosedDiv-Mincov pour $p = 1k$ itérations et pour des valeurs très faibles de θ. Fait intéressant, sur les jeux de données creux et pour $p = 1k$ itérations, Gibbs est en moyenne 11 fois plus rapide que ClosedDiv-Mincov, à l'exception du jeu de données retail, où l'accélération atteint 572 pour $\theta = 0,4\%$. Enfin, pour un nombre d'itération $p = 10k$, aucune des deux approches ne domine clairement l'autre. Par ailleurs, l'analyse des distributions de fréquence des motifs échantillonnés (voir les figures 5.7 et A.4) montre que, contrairement à CFTP, Gibbs offre beaucoup plus de garanties pour l'extraction de motifs fréquents. Sur les jeux de données denses, on constate ainsi que les fréquences des motifs sont toutes supérieures au seuil minimum $fmin$. Cependant, ces motifs ont des valeurs

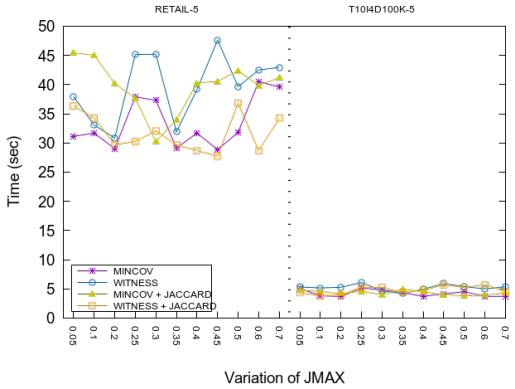
de fréquence inférieures à celles de ClosedDiv-Mincov, ce qui se traduit par des courbes en dessous de celles de ClosedDiv-Mincov. Ainsi, pour une valeur de fréquence donnée f, le nombre de motifs avec une fréquence entre $fmin$ et f est significativement plus élevé pour ClosedDiv-Mincov que pour Gibbs. Toutefois, il est important de noter que sur les jeux de données creux, à l'exception de T40I10D100K, les fréquences des tuiles échantillonnées par Gibbs sont nettement inférieures à la valeur $fmin$, en particulier pour Retail et Pumsb.

5.2.6 Évaluation des heuristiques de choix de variables

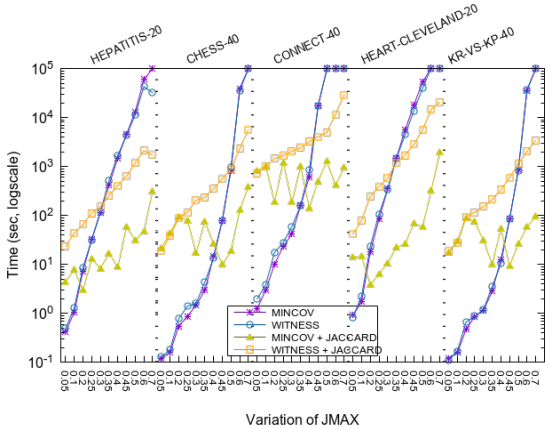
Dans cette expérimentation, nous comparons les deux heuristiques de choix de variables Mincov et Witness et nous étudions l'effet de la variation du paramètre J_{max}. La figure 5.8 montre l'évolution du nombre de motifs extraits pour différentes valeurs de J_{max} comprises entre 0.05 et 0.7. Ces résultats montrent que la taille de l'historique a un impact important sur les temps de calcul. En effet, la taille de l'historique $\mathcal{H}$ croît rapidement avec l'augmentation de J_{max}. Cela induit des sur-coûts importants dans les calculs des bornes inférieures et supérieures. Notons qu'en pratique, les utilisateurs ne sont intéressés que par des petites valeurs de J_{max} car la diversité des motifs obtenus est alors maximale et le nombre de motifs renvoyés devient raisonnable. Cela explique pourquoi nous avons fixé la valeur de J_{max} à 0.05 dans nos expérimentations.



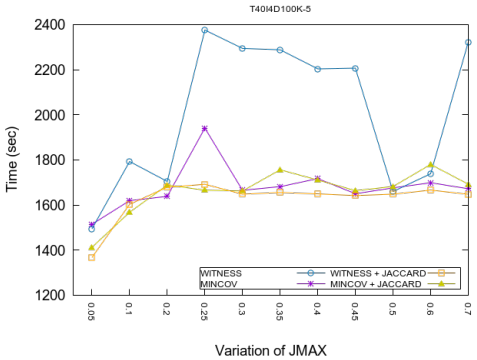
(a) Moderately dense and sparse datasets.



(b) Retail-5 and T10-5 instances : zoom view.



(c) Dense datasets.



(d) T40-5 instance : zoom view.

Figure 5.8 – Comparaison des temps d'exécution de ClosedDiv-Mincov et ClosedDiv-Witness avec et sans test de la valeur de Jaccard

Les figures 5.8a et 5.8c comparent les deux heuristiques Mincov et Witness. Sur les jeux de données denses, les deux heuristiques se comportent de manière très similaire, avec un léger avantage pour Witness pour $J_{max} \geq 0,45$. Par ailleurs, le nombre de motifs témoins extraits est très faible (moins de 1 % pour la plupart des instances) par rapport au nombre de motifs diversifiés extraits par les deux heuristiques (voir la figure 5.9). Cela explique pourquoi Witness

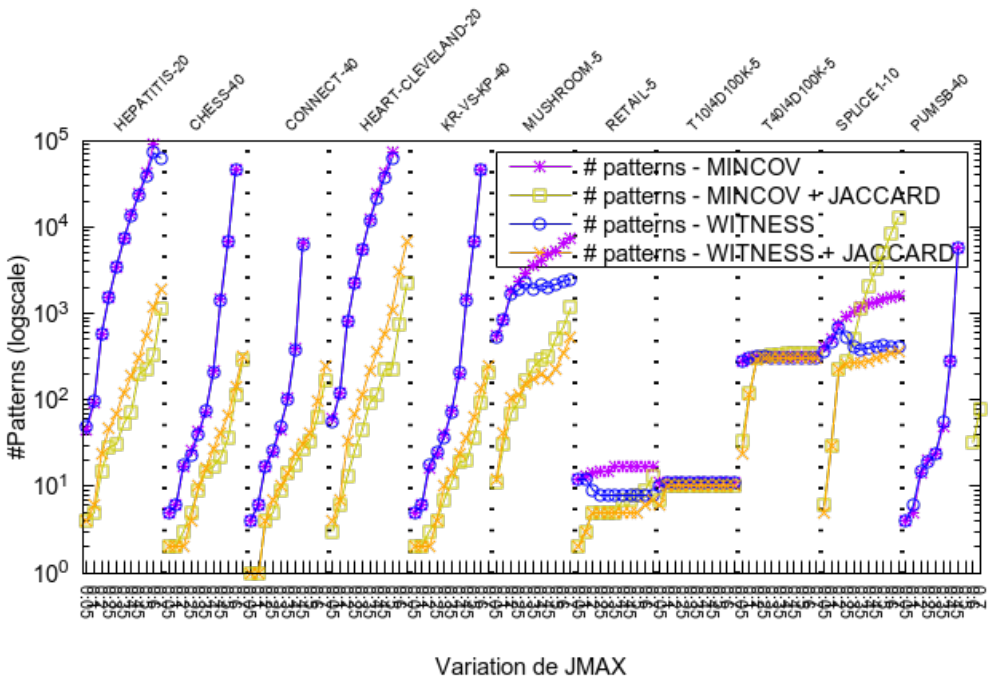
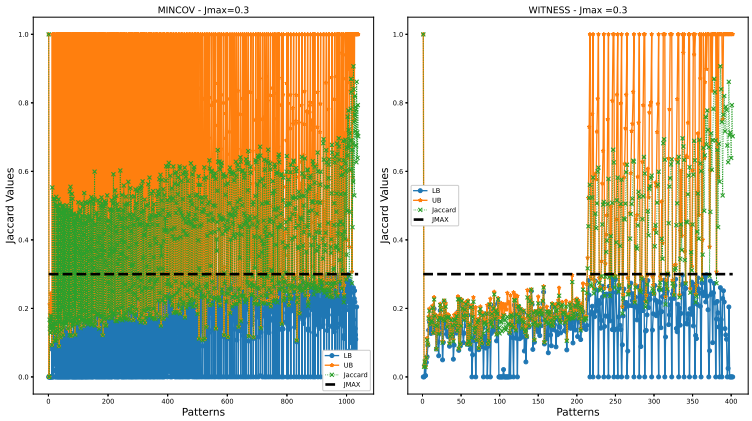


FIGURE 5.9 – Comparaison des nombres de motifs générés par CLOSEDDIV-MINCOV et CLOSEDDIV-WITNESS avec et sans test de la valeur de Jaccard

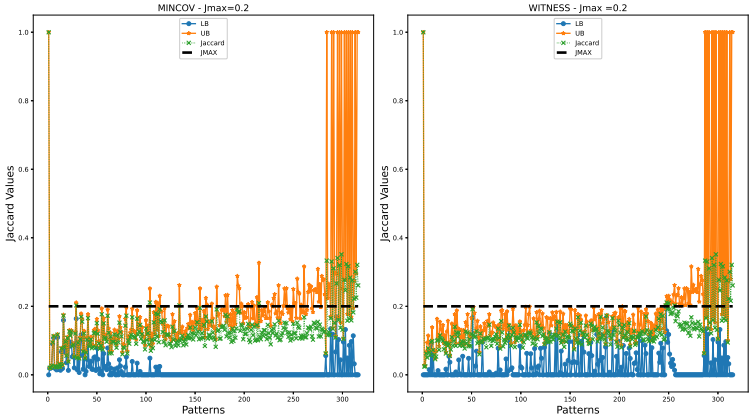
et MINCOV ont le même comportement. Ceci est dû à la stratégie WIT-FIRSTSOL qui évite l'exploration complète des différents sous-arbres témoins rencontrés lors de la recherche.

En comparant MINCOV et WITNESS aux deux approches qui effectuent une évaluation exact de la contrainte de Jaccard maximum, au début, nous observons que CLOSEDDIV-MINCOV et CLOSEDDIV-WITNESS dominent nettement CLOSEDDIV-MINCOV+JACCARD ainsi que CLOSEDDIV-WITNESS+JACCARD. Cependant, lorsque J_{max} augmente, la taille de l'historique augmente également (jusqu'à $\sim 9 \cdot 10^4$ pour MINCOV). Dans ce cas (taille ≥ 1000 pour MINCOV et ≥ 6500 pour WITNESS), les deux approches CLOSEDDIV-MINCOV+JACCARD et CLOSEDDIV-WITNESS+JACCARD deviennent très efficaces. Toutefois, pour CLOSEDDIV-MINCOV+JACCARD et CLOSEDDIV-WITNESS+JACCARD, la taille de $\mathcal{H}$ reste raisonnablement petite. Enfin, CLOSEDDIV-MINCOV+JACCARD domine largement CLOSEDDIV-WITNESS+JACCARD. Ceci est dû au grand nombre de solutions intermédiaires générées par WITNESS, alors que pour MINCOV ce nombre est faible, réduisant ainsi le sur-coût induit par le test exact de la contrainte de diversité.

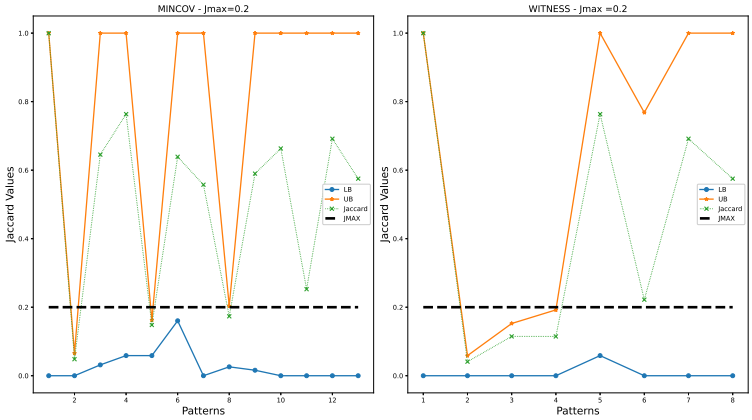
Pour les jeux de données modérément denses, CLOSEDDIV-WITNESS est largement plus efficace que CLOSEDDIV-MINCOV, notamment pour $J_{max} \geq 0.25$. Par ailleurs, CLOSEDDIV-WITNESS permet d'obtenir des ensembles de motifs de taille plus réduite dont une part importante est constituée de motifs témoins : environ 5% pour MUSHROOM et 60% pour SPLICE1. Le comportement CLOSEDDIV-MINCOV+JACCARD est un peu plus complexe, particulièrement pour SPLICE1, en raison de la taille de l'historique qui augmente de manière exponentielle avec l'augmentation de J_{max} (voir la figure 5.9). Cela explique probablement pourquoi cette variante est moins efficace. Enfin, les temps d'exécution de CLOSEDDIV-WITNESS et CLOSEDDIV-WITNESS+JACCARD sont comparables. Pour les jeux de données creux, toutes les heuristiques sont comparables en temps de calcul et en nombre de motifs diversifiés, avec un léger avantage pour les méthodes utilisant une évaluation exacte de la contrainte de Jaccard maximum (voir la figure 5.8).



(a) Splice1 ($\theta = 10\%, J_{max} = 0.3$)



(b) T40 ($\theta = 5\%, J_{max} = 0.2$)



(c) Retail ($\theta = 5\%, J_{max} = 0.2$)

Figure 5.10 – Analyse qualitative des relaxations LB et UB

5.2.7 Analyse qualitative des bornes

Dans cette section, nous analysons la qualité de la relaxation de la contrainte de Jaccard maximum. La figure 5.10 montre l'évolution des valeurs de LB_J, de l'indice de Jaccard et de UB_J des différents motifs pendant l'extraction, pour les deux heuristiques Mincov et Witness sur trois jeux de données (Splice1, T40I10D100K et Retail) avec un seuil de fréquence $\theta = 5\%$ et un seuil de diversité $J_{max} \in \{0.2, 0.3\}$. Pour le premier motif extrait, l'indice de Jaccard et l'UB_J vaut 1, et la valeur de LB_J vaut 0. Les solutions sont ordonnées en fonction des valeurs de UB_J.

Pour la borne inférieure, nous pouvons constater que toutes les valeurs de LB_J sont en-dessous du seuil J_{max} (représentées en rouge). Cela montre à quelle fréquence la règle de filtrage

LB de ClosedDiv est utilisée. Cela confirme également la pertinence de la règle pour l'élagage des motifs fréquents, fermés non diversifiés.

En ce qui concerne la borne supérieure, il est intéressant de voir que les valeurs de UB_J sont très proches de celle du Jaccard, ce qui signifie que notre borne supérieure du Jaccard fournit une relaxation serrée. De plus, un grand nombre de solutions ont des valeurs de UB_J inférieures ou très proches de J_{max}. Ceci est un indicateur de la qualité des motifs trouvés en termes de diversité. Nous rappelons que lorsque $UB_J < J_{max}$, toutes les affectations partielles peuvent être immédiatement étendues à des motifs diversifiés grâce à la propriété d'anti-monotonie de la borne UB donné à la proposition 9. Par ailleurs, nous remarquons que les meilleures solutions (c'est-à-dire celles avec les plus petites valeurs de UB_J et de Jaccard) sont générées au début de la recherche. Ainsi, plus l'historique est grand, moins la qualité des solutions en termes de valeurs de Jaccard et de UB_J est bonne. Enfin, nous pouvons voir que Witness permet de découvrir rapidement un ensemble de motifs de meilleure qualité en termes de valeurs de UB_J et de Jaccard comparativement à Mincov. Cela démontre l'attrait et la force de notre heuristique de branchement UB_J pour favoriser les ensembles de motifs plus diversifiés.

5.3 Étude expérimentale de SDivJaX

Dans cette section, nous présentons l'étude expérimentale menée sur plusieurs bases de transactions de FIMI, pour comparer et évaluer les apports pratiques de SDivJaX et ses différentes variantes par rapport à l'approche Flexics. Nous commençons par décrire le protocole expérimental, ensuite nous discutons les résultats obtenus en termes de temps de calcul et de qualité de diversité des solutions.

5.3.1 Protocole expérimental

La mise en œuvre de notre approche a été réalisée en Java pour la partie ClosedDiversity et en Scala [8] pour la partie Weightgen. Toutes les expérimentations ont été menées sous Linux sur un **AMD Opteron** 6174**,** 2.2 **GHz** disposant d'une mémoire **RAM de** 256 **Go**. Nous avons utilisé une limite de temps de 24 heures et un espace mémoire alloué à la JVM de 30 Go. Comme seuil de diversité, nous avons fixé J_{max} à 5%. Nous avons également choisi d'utiliser comme heuristique par défaut de choix de variables Mincov. Pour mesurer l'impact de l'étape d'estimation du nombre de contraintes `XOR` sur les deux variantes SDivJaX, nous avons implanté deux approches :

— la première, dénommée SDivJaX-CDiv, exploite la contrainte globale ClosedDiversity comme oracle de nos deux méthodes SDivJaX-1 et SDivJaX-2 pendant l'étape d'estimation du nombre de contraintes XOR,

— la seconde, dénommée SDivJaX-CP, exploite la contrainte globale ClosedPatterns comme oracle pour déterminer la "bonne" taille de cellule.

Nous comparons nos deux oracles SDivJaX-1 et SDivJaX-2 aux deux variantes de Flexics (EFlexics et GFlexics). Pour cette dernière, nous avons utilisé les mêmes paramètres que précédemment. Pour toutes les méthodes, le nombres d'échantillons a été fixé au nombre de motifs extraits par ClosedDiversity.

5.3.2 Temps d'exécution de SDivJaX

Le tableau 5.4 compare les temps d'exécution de SDivJaX-1-CDiv et SDivJaX-2-CDiv avec Flexics sur différents jeux de données et pour différentes valeurs de θ. Une première remarque est que pour des valeurs de θ élevées (exceptée Heart-cleveland avec $\theta = 10\%$ et T40I10D100K avec $\theta = 8\%$), SDivJaX-1-CDiv surpasse largement les autres approches, alors que SDivJaX-2-CDiv est très souvent classée en second position. Toutefois, cette tendance

8. https://www.scala-lang.org/

Dataset $\lvert\mathcal{I}\rvert \times \lvert\mathcal{T}\rvert$ $\rho(\%)$	$\theta(\%)$	CDiv		Flexics	
		(1)	(2)	EFlexics	GFlexics
Hepatitis	30	**2.19**	*6.84*	227.34	9.11
68 × 137	20	*88.73*	110.84	97.16	**26.89**
50.00%	10	7119	7509	*6316*	**779.24**
Chess	30	**4.08**	*15.80*	41.83	131.07
75 × 3196	20	*336.86*	639.94	**215.07**	465.12
49.33%	15	5457	2678.18	**981.49**	*1452.23*
	10	35342	61991	*14573*	**13305**
Connect	30	**43.26**	201.68	*78.69*	11073
129 × 67,557	18	-	*	**579.38**	-
33.33%	15	-	*20239*	**1377.09**	-
Heart-cleveland	10	*12711*	40435	59096	**5984**
95 × 296	8	-	**8922**	-	*28629*
47.37%	6	-	*	-	-
Kr-vs-kp	30	**3.71**	*15.32*	46.63	115.42
73 × 3196	20	*263.41*	619.75	**198.84**	484.07
49.32%	10	29426	40933	*10631*	**10341**
Mushroom	5	20479	16737	**93.79**	*12277*
112 × 8124	1	-	-	**59691**	-
18.75%	0.8	-	-	-	-
	0.5	-	-	-	-
T10I4D100K	5	**19.66**	*82.92*	Uns	326.67
	1	*4227*	-	**2163.16**	-
T40I10D100K	8	*555.35*	44683	**30.53**	74677
	5	*2080.23*	-	**1404.82**	-
Splice1	10	*4939*	4473	**36.95**	50622
	5	-	-	**14747**	-
Retail	5	**91.33**	*824.79*	Uns	1294.68
16470 × 88,162	1	**3184.75**	-	Uns	22279
0.06%	0.4	**50267**	-	Uns	-
Pumsb	40	**197.02**	*1818.32*	562.23	26880
2,113 × 49,046	30	*23144*	30267	**1662.70**	54418
3.50%	20	-	*62088*	**3619**	-

Table 5.4 – Analyse comparative des temps d'exécution de SDivJaX avec Flexics. (1) : SDivJaX-1-CDiv, (2) : SDivJaX-2-CDiv. En gras, les résultats de la meilleure méthode. En italique, les résultats de la seconde meilleure méthode.

s'inverse avec la diminution de la valeur de θ, où EFlexics obtient les meilleurs résultats sur la plupart des instances considérées. Notons, toutefois, les bonnes performances de SDivJaX-1-CDiv, en particulier sur le jeu de données Retail, qui est très souvent classée en seconde position.

Les résultats de SDivJaX-2-CDiv ne sont pas compétitifs par rapport aux trois autres approches. Ce comportement peut s'expliquer par la stratégie de gestion de l'historique au niveau de SDivJaX-CDiv. En effet, avec SDivJaX-1-CDiv, l'historique évolue localement et plus rapidement lors de l'exploration d'une cellule, ce qui a pour conséquence de réduire l'espace de recherche et donc d'accélérer le temps d'exploration dans chaque cellule. Au contraire, l'historique maintenu par SDivJaX-2-CDiv est plus globale et augmente que d'un seul motif d'une étape d'échantillonnage à une autre, ce qui pénalise fortement le temps global de résolution de SDivJaX-2. Ce comportement est confirmé par les graphiques de la figure 5.11. En effet, après l'étape d'estimation (cf. traits en pointillés rouge), un sous-ensemble de contraintes XOR sont ajoutées (une contrainte `XOR` dans le cas de Hepatitis). Ensuite, pour échantillonner un motif, SDivJaX-2-CDiv nécessite de rajouter plus de contraintes `XOR` comparé à SDivJaX-1-CDiv, jusqu'à 10 contraintes `XOR` dans le cas de Hepatitis. Rappelons que l'ajout de contraintes `XOR` se fait lorsque la taille de la cellule, donnée par la somme des poids des motifs de la cellule, est

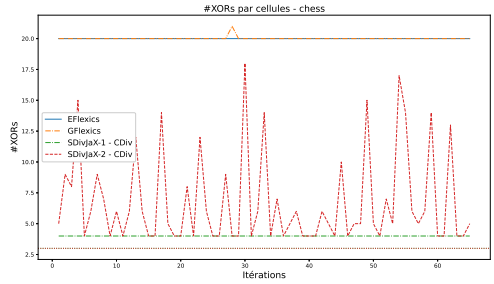
Dataset $\lvert\mathcal{I}\rvert \times \lvert\mathcal{T}\rvert$ $\rho(\%)$	$\theta(\%)$	SDivJaX-1		SDivJaX-2		EFlexics
		Cdiv	ClosedP	Cdiv	ClosedIP	
Hepatitis	30	**2.19**	18.84	6.84	17.87	227.34
68 × 137	20	88.73	92.55	110.84	**21.58**	97.16
50.00%	10	7119	13004	7509	**1511.55**	6316
Chess	30	**4.08**	92.37	15.80	65.56	41.83
75 × 3196	20	336.86	**156.04**	639.94	200.47	215.07
49.33%	15	5457	8075	2678.18	2069.41	**981.49**
	10	35342	-	61991	**11720**	14573
Connect	30	**43.26**	8177	201.68	1006.31	78.69
129 × 67,557	18	-	-	11983	8108	**579.38**
33.33%	15	-	-	20239	27625	**1377.09**
Heart-cleveland	10	12711	20289	40435	**11969**	59096
95 × 296	8	-	-	8922	-	-
47.37%	6	-	-	54821	-	-
Kr-vs-kp	30	**3.71**	99.63	15.32	74.37	46.63
73 × 3196	20	263.41	232.11	619.75	**112.28**	198.84
49.32%	10	29426	89095	40933	44258	**10631**
Mushroom	5	20479	19058	16737	482.80	**93.79**
112 × 8124	1	-	-	-	-	59691
18.75%	0.8	-	-	-	-	-
	0.5	-	-	-	-	-
T10I4D100K	5	19.66	**17.82**	82.92	43.92	Uns
	1	4227	4733	-	-	**2163.16**
T40I10D100K	8	555.35	*	44683	8106	**30.53**
	5	2080.23	2941.09	-	-	**1404.82**
Splice1	10	4939	4244	4473	1382.31	**36.95**
	5	-	-	-	-	14747
Retail	5	**91.33**	*	824.79	158.05	Uns
16470 × 88,162	1	**3184.75**	*	-	-	Uns
0.06%	0.4	**50267**	*	-	-	Uns
Pumsb	40	**197.02**	10065	1818.32	19637	562.23
2,113 × 49,046	30	23144	-	30267	40815	**1662.70**
3.50%	20	-	-	62088	-	**3619**

Table 5.5 – Analyse comparative des temps d'exécution de SDivJaX-CDiv vs SDivJaX-CP pour chacun des deux oracles de SDivJaX. En gras les résultats de la meilleure méthode. La cellule en gris indique, pour chaque oracle, la meilleure alternative

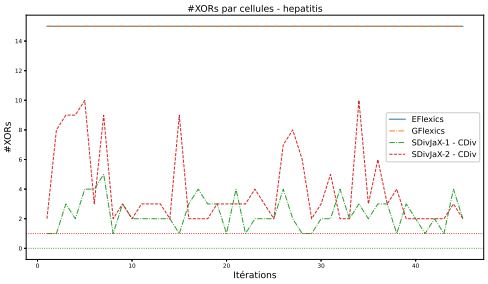
beaucoup plus grande. Cette augmentation significative du nombre de contraintes `XOR` pénalise les performances de la méthode. Notons que sur le jeu de données Retail, aucune contrainte `XOR` n'a été rajoutée car ce dernier contient déjà peu de motifs par rapport aux seuils considérés.

5.3.3 Impact de l'étape d'estimation des `XOR` sur SDivJaX

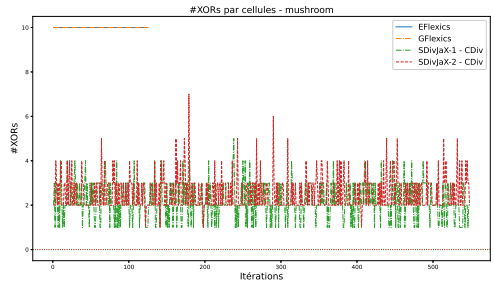
Le tableau 5.5 compare les temps d'exécution de SDivJaX-CDiv avec SDivJaX-CP. Nous pouvons remarquer que sur les instances denses, l'utilisation de ClosedPatterns pour estimer le nombre de contraintes `XOR` permet d'améliorer de manière significative les temps de calcul de SDivJaX-2-CP sur 8 instantes parmi 16 (cellules coloriées en gris) comparé à SDivJaX-2-CDiv. Par ailleurs, il obtient les meilleurs de temps de calcul sur 5 instances (cellules en gras), surpassant même EFlexics : SDivJaX-2-CP est meilleur sur 7 instances parmi 16 contre 4 instances pour EFlexics. Sur les instantes moins denses, à l'exception de Pumsb, SDivJaX-2-CP domine largement SDivJaX-2-CDiv mais reste néanmoins moins performant comparé à SDivJaX-1-CDiv. Les performances de SDivJaX-2-CP peuvent s'expliquer par l'analyse des graphiques de la figure 5.12 qui montrent l'indice de Jaccard moyen entre les premières solutions, en augmentant de manière itérative le nombre de solutions. Comme nous pouvons le constater, les premières solutions de SDivJaX-2-CP sont encore plus diversifiées



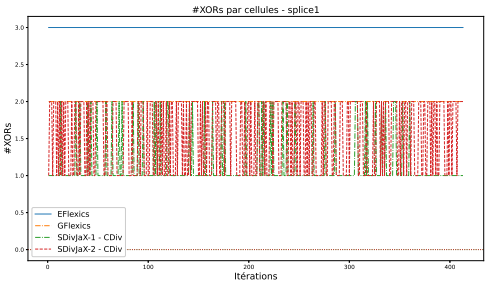
(a) Chess ($\theta = 20\%$)



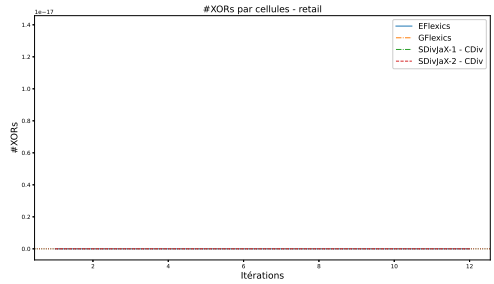
(b) Hepatitis ($\theta = 20\%$)



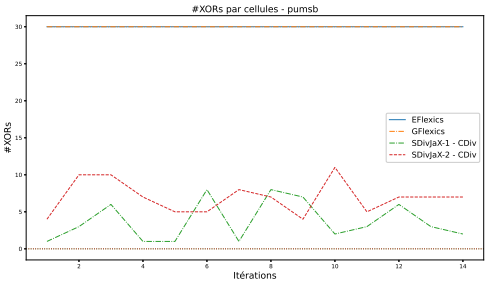
(c) Mushroom ($\theta = 5\%$)



(d) Splice1 ($\theta = 10\%$)



(e) Retail ($\theta = 5\%$)



(f) Pumsb ($\theta = 30\%$)

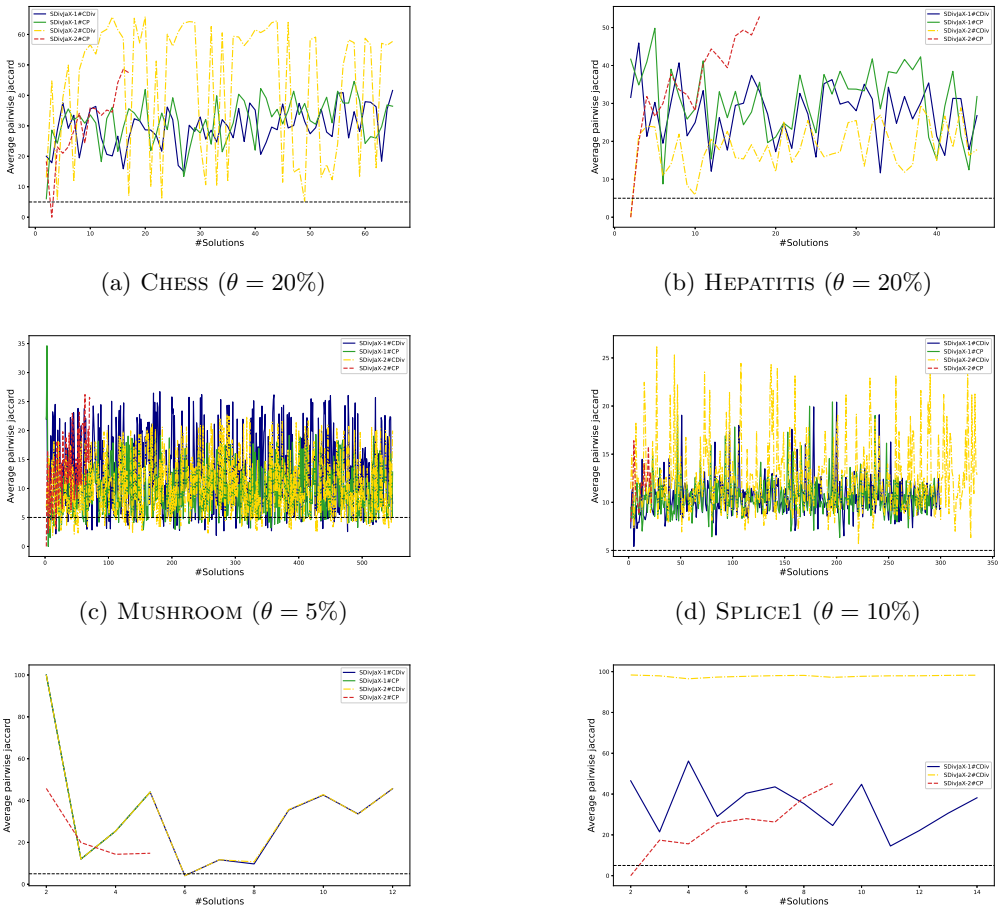
Figure 5.11 – Évolution du nombre de contraintes XOR utilisées par itération par SDivJaX et Flexics.

et la diversité de l'ensemble de motifs extraits au bout d'un certain nombre de tirages n'arrive plus à être améliorée car l'ordre de recherche a un fort impact sur la diversité. Ainsi, la requête SDivJaX-2-CP$(\mathcal{D}, J_{max}, k)$ peut donner un ensemble S de motifs diversifiés tels que $|S| < k$, alors que SDivJaX-2-CDiv continuera à extraire des motifs jusqu'à atteindre le nombre k de motifs demandés.

5.3.4 Évaluation de la qualité de diversification

Dans la figure 5.13, nous représentons les distributions cumulatives d'indice de Jaccard de toutes les paires de motifs de ClosedDiv-Mincov, Flexics et SDivJaX-CDiv sur six jeux de données. Pour les jeux de données denses, les courbes des CDFs de SDivJaX-CDiv se situent au-dessus de celles de ClosedDiv-Mincov, indiquant que SDivJaX-CDiv extrait des ensembles de motifs pour lesquels l'indice de Jaccard des paires est relativement faible. Pour les instances creuses, les deux approches restent comparables.

Comparé à Flexics, les CDFs de SDivJaX-2 indiquent qu'il est plus facile de trouver des paires de motifs avec un faible Jaccard. Notons enfin que, sur les plupart des instances considérées, SDivJaX-1-CDiv extrait des ensembles de motifs moins diversifiés que SDivJaX-2-CDiv. En effet, l'idée de maintenir un historique $\mathcal{H}_{global}$ permet à SDivJaX-2-CDiv d'extraire des ensembles de motifs de meilleure diversité.



(a) Chess ($\theta = 20\%$)

(b) Hepatitis ($\theta = 20\%$)

(c) Mushroom ($\theta = 5\%$)

(d) Splice1 ($\theta = 10\%$)

(e) Retail ($\theta = 5\%$)

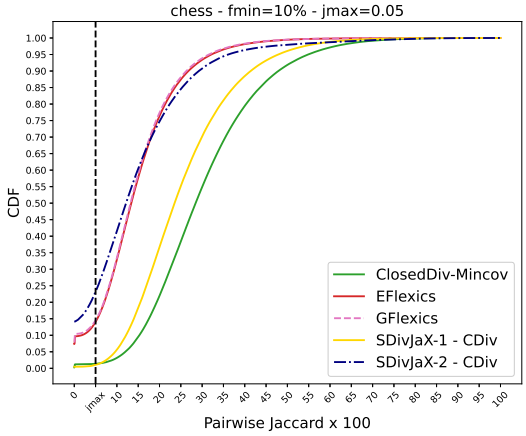
(f) Pumsb ($\theta = 30\%$)

Figure 5.12 – Indice de Jaccard moyen par itérations pour les approches SDivJaX-CDiv.

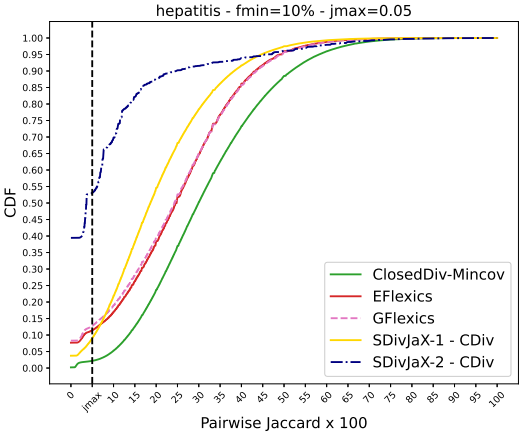
5.4 Conclusions

Dans ce chapitre, nous avons présenté différentes évaluations de notre contrainte globale ClosedDiversity et de notre méthode d'échantillonnage associée SDivJaX. Ainsi, en contrôlant la diversité avec un seuil J_{max}, nous avons montré la capacité de ClosedDiversity à pouvoir d'une part extraire des ensembles de motifs significativement plus petits que l'ensemble des motifs fermés, et d'autre part réduire la redondance entre les motifs extraits. Par ailleurs, nos expérimentations ont montré que les performances de ClosedDiversity étaient meilleures que celles de Flexics, ce qui motive son exploitation pour faire de l'échantillonnage. Toutefois, ces performances restent limitées par rapport à celles de Gibbs et CFTP en utilisant des facteurs de fréquence bas (≤ 5). Cependant, nous avons montré que ClosedDiversity offre plus de garantie sur la qualité des motifs en terme de diversité et présente un meilleur compromis entre diversité et performances comparativement à ces trois méthodes d'échantillonnage.

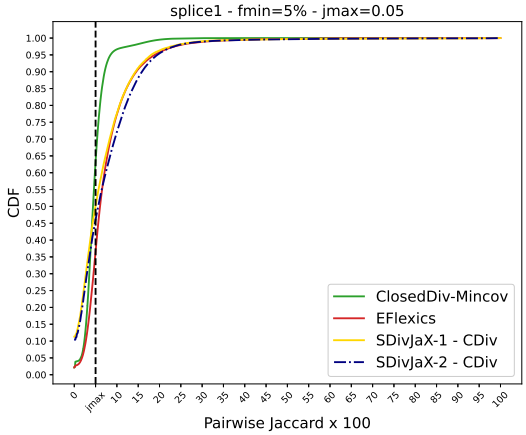
De plus, même si ces performances sont inférieures à celles de ClosedDiversity, nous avons montré que SDivJaX permet d'extraire des échantillons de motifs diversifiés. Les différents résultats montrent que SDivJaX-1 est l'approche la plus performante et que SDivJaX-2 permet d'obtenir des ensembles de motifs relativement plus diversifiés grâce à l'historique global utilisé. Néanmoins, les résultats montrent que la redondance moyenne dans les ensembles de motifs obtenus avec SDivJaX-1 reste peu élevé malgré l'absence d'un historique global des solutions. SDivJaX-1 peut ainsi être utilisé pour extraire des motifs dans un contexte anytime étant donné sa capacité à extraire des motifs relativement rapidement et sa capacité à introduire de la



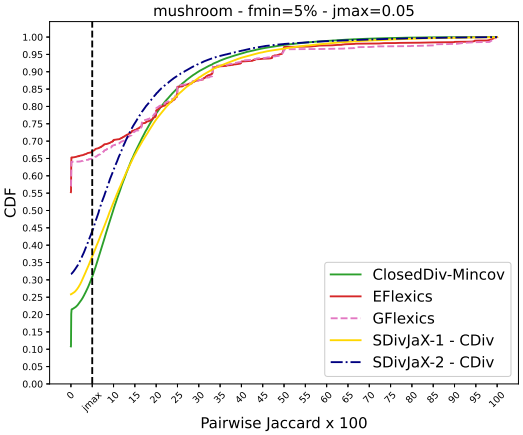
(a) Chess ($\theta = 10\%$)



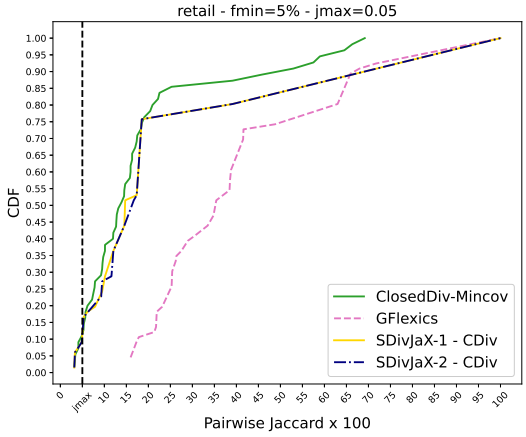
(b) Hepatitis ($\theta = 10\%$)



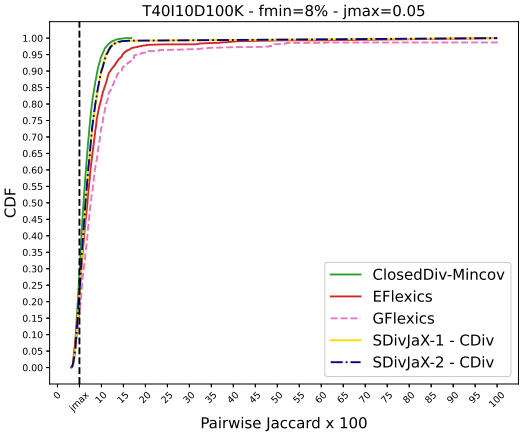
(c) Splice1 ($\theta = 5\%$)



(d) Mushroom ($\theta = 5\%$)



(e) Retail ($\theta = 5\%$)



(f) T10I4D100K ($\theta = 1\%$)

Figure 5.13 – Évaluation de la redondance globale des paires de motifs de ClosedDiv-Mincov, Flexics, et SDivJaX-CDiv

diversité dans les ensembles de motifs. Ainsi, nous exploiterons SDivJaX-1 pour l'apprentissage actif des préférences de l'utilisateur (chapitre 6) en l'utilisant pour échantillonner les motifs diversifiés à présenter à l'utilisateur.

CHAPITRE 6

Motifs discriminants et Apprentissage

Contents

6.1 Introduction 119
6.2 Nouvelle représentation des motifs 120
6.2.1 Motifs discriminants 120
6.3 Motifs discriminants comme descripteurs 122
6.4 Apprentissage à partir des descripteurs discriminants 123
6.4.1 Fonction d'apprentissage des préférences de l'utilisateur 123
6.4.2 Exploitation des descripteurs discriminants 124
6.5 DiSPaLe : Un cadre interactif de fouille de motifs exploitant les discriminants et la diversité 127
6.6 Évaluation expérimentale 127
6.6.1 Protocole expérimental 128
6.6.2 Évaluation de différents paramètres de DiSPaLe 131
6.6.3 Comparaisons entre DiSPaLe et LetSIP 133
6.6.4 Apports de la diversité dans LetSIP et DiSPaLe 134
6.7 Conclusions 136

6.1 Introduction

Dans ce chapitre nous proposons une approche de fouille interactive de motifs basée sur le schéma général proposé dans l'algorithme 4 de la section 3.7. Premièrement, dans cette approche, nous exploitons une méthode d'échantillonnage pour extraire les motifs. Comme nous l'avons montré au chapitre 1, ces méthodes ont l'avantage d'être rapides et d'assurer une forme de diversité dans les ensembles $\mathcal{X}$ de motifs extraits. Plus particulièrement, nous proposons d'exploiter SDivJaX-1 comme méthode d'extraction des motifs. Deuxièmement, pour prendre en compte les intérêts de l'utilisateur et s'assurer que les résultats sont pertinents, nous proposons d'exploiter de nouveaux descripteurs dynamiques pour apprendre une fonction logistique φ. Ces nouveaux descripteurs permettent de mieux comprendre le rangement effectué par l'utilisateur et d'extraire des ensembles de motifs plus concis et diversifiés. Pour cela, nous introduisons la notion de **motifs discriminants**, qui sépare les motifs auxquels l'utilisateur attribue un rang bas de ceux qui ont un rang élevé. En ajoutant ces descripteurs à la liste des descripteurs initiaux, nous pouvons apprendre une fonction de poids φ.

Dans la suite de ce chapitre, nous présenterons les motifs discriminants et leur exploita-

tion pour décrire les motifs de façon dynamique. Puis, nous montrerons comment ces nouveaux descripteurs sont intégrés et exploités pour apprendre les préférences de l'utilisateur. Nous présenterons enfin une nouvelle méthode de fouille interactive de motifs exploitant les motifs discriminants.

6.2 Nouvelle représentation des motifs

Dans la section 3.3, nous avons présenté un ensemble de descripteurs utilisés dans la littérature. Ces descripteurs sont statiques, c'est-à-dire fixés au début du processus interactif et ne peuvent pas varier au fil des itérations. Ils exploitent alors certaines caractéristiques syntaxiques des motifs pour les représenter (comme les items ou les transactions couvertes) [48, 18] mais également des mesures (comme la longueur et la fréquence des motifs) [48].

Une des limites de ces descripteurs vient de leur utilisation comme des éléments indépendants les uns des autres. En effet, étant donné que chaque descripteur décrit une caractéristique particulière du motif, il est naturel de les considérer comme étant indépendants pour sélectionner des motifs *globalement* intéressants pour l'utilisateur. Cependant, il serait intéressant de considérer d'éventuelles interdépendances qui peuvent exister entre ces différents descripteurs. Typiquement, un utilisateur pourrait être intéressé par des « combinaisons particulières d'items ou être désintéressé par des transactions particulières ». Comment exprimer ce type de préférences ? et comment les prendre en compte sous forme de descripteurs ?

Dans cette section, nous introduisons les *motifs discriminants* et nous montrons comment les extraire à partir des motifs présentés à l'utilisateur tout en exploitant ses retours. Nous définissons une nouvelle classe de descripteurs exploitant les discriminants pour améliorer la précision de l'apprentissage. Ces nouveaux descripteurs se présentent alors comme des motifs corrélés au rangement effectué par l'utilisateur, c'est-à-dire des motifs qui ont fortement influencé un bon/mauvais rangement d'un sous-ensemble d'autres motifs.

6.2.1 Motifs discriminants

Les motifs discriminants peuvent être considérés comme étant les sous-motifs qui ont été déterminants dans le rangement effectué par l'utilisateur :

— soit ce sous-motif a permis à un/plusieurs autres motifs d'être bien classés ;
— soit il a provoqué leur rangement dans les rangs les plus bas.

Une manière de modéliser la recherche de motifs discriminants consiste à considérer les rangs numériques donnés aux motifs individuels comme des étiquettes numériques et de considérer la recherche des discriminants comme un problème de régression. Dans notre cas, l'objectif n'est pas de construire un modèle de régression complet, mais seulement d'**extraire un modèle individuel qui est en corrélation avec l'étiquette numérique**. Pour cela, nous exploitons la notion de variance inter-classe.

A. Variance inter-classe des motifs

Pour trouver les motifs corrélés au rangement de l'utilisateur, nous exploitons la mesure de la variance interclasse proposée par Morishita *et al.* dans [104].

Définition 55 (Variance inter-classe)**.**
Soit $\mathcal{X}$ un ensemble de k motifs ordonnés selon l'ordre $\mathcal{R}^*$ défini par l'utilisateur. On note par $\mathcal{X}_y = \{X \in \mathcal{X} | X \supseteq y\}$ l'ensemble des motifs X de $\mathcal{X}$ qui contiennent le sous-motif y, et $\overline{\mathcal{X}}_y = \mathcal{X} - \mathcal{X}_y$. La variance interclasse du sous-motif y est définie comme suit :

$$ICV(y, \mathcal{R}^*) = |\mathcal{X}_y| \cdot (\mu(\mathcal{X}) - \mu(\mathcal{X}_y))^2 + |\overline{\mathcal{X}}_y| \cdot (\mu(\mathcal{X}) - \mu(\overline{\mathcal{X}}_y))^2 \tag{6.1}$$

avec $\mu(\mathcal{X}) = \frac{1}{|\mathcal{X}|} \cdot \sum_{X \in \mathcal{X}} r(X)$, et $r(X)$ le rang du motif X dans $\mathcal{X}$ (voir équation 3.7).

Algorithm 9: Extraction de motifs discriminants

```
1  Fonction MineDiscriminating(X, R*)
2      ICV_max ← 0, X_disc ← ∅
3      I_X ← {i ∈ X | X ∈ X}
4      Y ← I_X
5      Pour chaque i ∈ I_X faire
6          if ICV(i, R*) ≥ ICV_max then
7              ICV_max ← ICV(i, R*)
8              X_disc ← {i}
9      Pour chaque Y ∈ Y faire
10         Tant que ∀i ∈ I_X, ∃X ∈ X tq. i ∉ Y ∧ {Y ∪ i} ⊆ X faire
11             si ICV(Y ∪ i, R*) ≥ ICV_max alors
12                 ICV_max ← ICV(Y ∪ {i}, R*);
13                 X_disc ← Y ∪ {i};
14                 Y ← Y ∪ X_disc;
15     retourner X_disc
```

L'ICV est donc déterminé par l'ensemble de motifs $\mathcal{X}_y \subseteq \mathcal{X}$ et par les rangs $r(X)$ associés aux motifs $X \in \mathcal{X}_y$.

Exemple 6.2.1.
Soit un jeu de donné contenant les items suivants : $\mathcal{I} = \{1, \ldots, 7\}$. *Supposons une requête utilisateur contenant les motifs suivants :* $\mathcal{X} = \{X_1, X_2, X_3, X_4\}$, *avec* $X_1 = \{5, 7\}, X_2 = \{2, 7\}, X_3 = \{1\}, X_4 = \{4\}$ *et soit le rangement* $\mathcal{R}^*$ *de l'utilisateur sur les motifs de* $\mathcal{X}$, *avec* $\mathcal{R}^* = \{X_2 \succ X_1 \succ X_3 \succ X_4\}$. *Pour* $y = \{2\}$, *nous avons* $\mathcal{X}_y = \{X_2\}$ *car* $\{2, 7\} \supseteq \{2\}$, $\overline{\mathcal{X}}_y = \{X_1, X_3, X_4\}$, $\mu(\mathcal{X}_y) = 1$, $\mu(\overline{\mathcal{X}}_y) = \frac{r(X_1)+r(X_3)+r(X_4)}{3} = \frac{(2+3+4)}{3} = 3$ *et* $\mu(\mathcal{X}) = 2.5$. *En appliquant l'équation (6.1), on obtient* $ICV(\{2\}) = (2.5-1)^2 + 3 \cdot (2.5-3)^2 = 3$. *De la même manière,* $ICV(\{0\}) = 0.33$.

B. Extraction de motifs discriminants

La variance inter-classe présentée précédemment permet d'évaluer la corrélation des motifs avec le rangement de l'utilisateur ; les motifs ou sous-motifs les plus corrélés sont ceux ayant les plus grande valeur d'ICV. Notre objectif est alors de trouver les sous-motifs $y \subseteq X \in \mathcal{X}$ ayant la plus grande variance inter-classe ICV. Sémantiquement, il s'agit de trouver les sous-motifs dont la présence dans un ou plusieurs motifs X a influencé leur rangement. Ainsi, si $y \subseteq X$, on peut dire que le rangement de X à la $r(X)^e$ position dans $\mathcal{X}$ est plus susceptible d'être expliqué par la présence du sous-motif y.

L'algorithme 9 décrit la procédure de recherche d'un motif discriminant. Cette recherche est effectuée par la fonction **MineDiscriminating** qui permet d'extraire le motif qui maximise la variance interclasse des motifs. Il prend en entrée un ensemble de motifs ordonnés $\mathcal{X}$ ainsi que le rangement $\mathcal{R}^*$ et retourne le sous-motif ayant la plus grande variance interclasse ICV. L'algorithme commence par évaluer l'ICVde tous les items des motifs $X \in \mathcal{X}$ (boucle 5−8), puis combine progressivement les items entre eux afin d'obtenir des discriminants plus grand et plus précis (boucle 9-14). Avant chaque combinaison, un test est effectué pour s'assurer que le sous-motif résultant est inclus dans l'un des motifs $X \in \mathcal{X}$ (ligne 10). Dans l'affirmative (ligne 11), la valeur maximal d'ICV(c'est-à-dire ICV_{max}) est mise à jour ; le meilleur motif discriminant X_{disc} est alors enregistré (ligne 13) et l'ensemble des sous-motifs pouvant être étendus est mis à jour en y ajoutant X_{disc} (ligne 14). À la fin, le meilleur motif discriminant X_{disc} est retourné

(ligne 15) et pourra être utilisé comme descripteur dans le processus de fouille interactive de motifs.

Proposition 12 (Complexité temporelle)**.**
L'algorithme 9 a une complexité qui est quadratique dans le pire des cas : $\mathcal{O}(n^2)$, avec $n = |I_{\mathcal{X}}|$.

Preuve 10.
La preuve de la proposition 12 se présente comme suit.
La première boucle de l'algorithme 9 s'exécute en $\mathcal{O}(n)$. Pour chaque itération i de la deuxième boucle (lignes 9 à 14), nous avons au plus $(n - i)$ itérations pour la troisième boucle (lignes 10 à 14). Ainsi, ces deux boucles s'effectuent en $\mathcal{O}(\frac{n^2}{2})$.
Les sous-motifs $\mathcal{Y}$ évoluent avec au moins $|X|$ éléments, étant donné que $|X_{disc}| \leq |X|$. Cela ajoute une complexité supplémentaire de $\mathcal{O}(n \times |X|)$. Puisque $|X| \leq n$, alors, la complexité dans le pire des cas est en $\mathcal{O}(n^2)$.

Exemple 6.2.2.
*Reprenons le jeu de données de l'exemple 6.2.1 ainsi que l'ensemble ordonné $\mathcal{X} = \{X_2, X_1, X_3, X_4\}$ tel que $\mathcal{R}^* = \{X_2 \succ X_1 \succ X_3 \succ X_4\}$. Après la première boucle de l'algorithme 9, la variable $\mathcal{Y}$ contient $\{1, 2, 4, 5, 7\}$, $ICV(2, \mathcal{R}^*) = ICV(4, \mathcal{R}^*) = 3$, $ICV(1, \mathcal{R}^*) = ICV(5, \mathcal{R}^*) = 0.33$ et $ICV(7, \mathcal{R}^*) = 4$. Ainsi, $ICV_{max} = 4$ et $X_{disc} = \{7\}$. La deuxième boucle (lignes 9 à 14) cherchera alors la meilleure combinaison de ces items qui maximise la variance inter-classe ICV_{max}. Pour illustrer cette boucle, supposons que $Y = \{2\}$ et $I_{\mathcal{X}} = \{1, 2, 4, 5, 7\}$. La boucle **while** va alors considérer uniquement les items $i \in I_{\mathcal{X}} - \{2\}$ tels que $Y \cup \{i\}$ est un sous-motif des motifs $X \in \mathcal{X}$. Ainsi, on obtient que pour $i = \{7\}, Y \cup \{i\} \subseteq X_2$. Puisque $ICV(\{2, 7\}, \mathcal{R}^*) = 3 < ICV_{max}$, X_{disc} demeure inchangé. Notons que pour $i = \{4\}$ ou $i = \{5\}$, $\nexists X \in \mathcal{X}$ tq. $(\{2\} \cup \{i\}) \subseteq X$. Pour cet exemple, l'algorithme 9 retournera $X_{disc} = \{7\}$.*

6.3 Motifs discriminants comme descripteurs

L'utilisation des motifs discriminants X_{disc} comme descripteurs apporte une sémantique nouvelle dans la description des motifs. En effet, ces nouveaux descripteurs permettent d'expliquer le rang obtenu par un motif X pendant l'itération courante. Plus important encore, ils permettent d'identifier une combinaisons d'items corrélés positivement ou négativement avec le rangement de l'utilisateur. Nous utiliserons donc les motifs discriminants comme des descripteurs numériques binaires F_{disc} tel que pour tout motif X, on a :

$$X_{F_{disc}} = \begin{cases} 1 \text{ si } X_{disc} \subseteq X \\ 0 \text{ si non} \end{cases} \tag{1.2}$$

Étant donné que ces descripteurs dépendent de la requête utilisée, à chaque itération de nouveaux motifs discriminants peuvent être extraits. Cela explique le caractère dynamique dans la description des motifs.

Une première utilisation consisterait à étendre le vecteur des descripteurs des motifs avec les nouveaux discriminants trouvés au fil des itérations. Cette stratégie d'exploitation consiste ainsi à prendre en compte la sémantique de tous les rangements de l'utilisateur dans la description des futurs motifs. On obtient alors le vecteur de descripteurs suivant :

$$\mathcal{F}^* = \langle \underbrace{F_1, \ldots, F_n}_{\text{statiques}}, \underbrace{F_{disc_1}, \ldots, F_{disc_m}}_{\text{discriminants}} \rangle \tag{6.3}$$

Ces nouveaux descripteurs ont l'avantage de permettre une description de plus en plus précise des différents motifs aussi bien d'un point de vue syntaxique (avec les descripteurs statiques) que

sémantique (avec les descripteurs discriminants). Cependant, en allongeant le vecteur $\mathcal{F}$ à chaque itération, on complexifie l'apprentissage qui doit prendre en compte plus d'éléments. Par ailleurs, on pourrait faire face à un problème de sur-apprentissage à cause de l'utilisation de descripteurs trop nombreux. Pour pallier à cela, nous proposons d'utiliser les motifs discriminants comme des descripteurs temporaires. Il s'agit d'ajouter temporairement ces descripteurs F_{disc} au vecteur $\mathcal{F}$ afin d'apprendre un poids $w_{F_{disc}}$. Le poids appris sera alors exploité pour mettre à jour la fonction d'apprentissage φ. Cette procédure sera décrite dans la section 6.4.

Trois types de descripteurs discriminants peuvent être ajoutés à $\mathcal{F}$:

— F_{disc_X} : il s'agit du descripteur décrit à l'équation 6.2. Il s'agit d'un descripteur binaire utilisé pour marquer la présence/absence du discriminant dans un motif $X \in \mathcal{X}$:

$$X_{F_{disc_X}} = \begin{cases} 1 \ si \ X_{disc} \subseteq X \\ 0 \ sinon \end{cases}$$

— $F_{disc_\mathcal{T}}$: il s'agit d'un descripteur numérique qui représente la fréquence du motif discriminant. $F_{disc_\mathcal{T}}$ peut alors prendre les valeurs suivantes :

$$X_{F_{disc_\mathcal{T}}} = \begin{cases} \frac{|\mathcal{V}_\mathcal{D}(X_{disc})|}{|\mathcal{T}|} \ si \ X_{disc} \subseteq X \\ 0 \ sinon \end{cases}$$

— $F_{disc_\mathcal{I}}$ est un descripteur numérique qui représente la longueur du motif discriminant. $F_{disc_\mathcal{I}}$ peut alors prendre les valeurs suivantes :

$$X_{F_{disc_\mathcal{I}}} = \begin{cases} \frac{|X_{disc}|}{|\mathcal{I}|} \ si \ X_{disc} \subseteq X \\ 0 \ sinon \end{cases}$$

En notant $\mathcal{F}_{disc}$ l'ensemble des descripteurs discriminants ajoutés à $\mathcal{F}$, on obtient le descripteur temporaire suivant :

$$\mathcal{F}^* = \langle F_1, \ldots, F_n, F_{disc_X}, F_{disc_\mathcal{T}}, F_{disc_\mathcal{I}} \rangle \tag{6.4}$$

Dans la prochaine section, nous montrerons comment ces nouveaux descripteurs seront exploités pour améliorer l'apprentissage du modèle de préférences.

6.4 Apprentissage à partir des descripteurs discriminants

Dans la section précédente, nous avons présenté une nouvelle classe de descripteurs. Ces descripteurs exploitent les retours de l'utilisateur pour représenter les motifs corrélés au rangement de celui-ci. Dans cette section, nous montrons comment les motifs discriminants sont exploités durant la phase d'apprentissage.

6.4.1 Fonction d'apprentissage des préférences de l'utilisateur

Pour apprendre les préférences de l'utilisateur, nous proposons d'exploiter le modèle proposé par Dzyuba *et al.* dans [48]. Notre démarche consiste ainsi à apprendre une fonction logistique $\varphi_{logistic}$ en exploitant la méthode stochastique de descente des coordonnées (SCD) en mettant à jour les poids w_{F_i} des descripteurs. Ce processus d'apprentissage peut être synthétisé par le schéma de la figure 6.1.

Afin d'exploiter les descripteurs discriminants dans l'apprentissage, nous proposons un nouveau processus qui étend celui de la figure 6.1. Ce processus exploite le nouveau vecteur des descripteurs présenté dans l'équation 6.4. Pour apprendre la fonction de poids $\varphi_{logistic}$, on associe alors de nouveaux poids $w_{F_{disc}}$ aux descripteurs discriminants. L'apprentissage permettra ainsi de mettre à jour à chaque itération t les poids $w^t_{F_i}$ des descripteurs $F_i \in \mathcal{F}^*$ et plus spécialement les poids $w^t_{F_{disc}}$ des descripteurs discriminants F_{disc}. Comme ces descripteurs sont

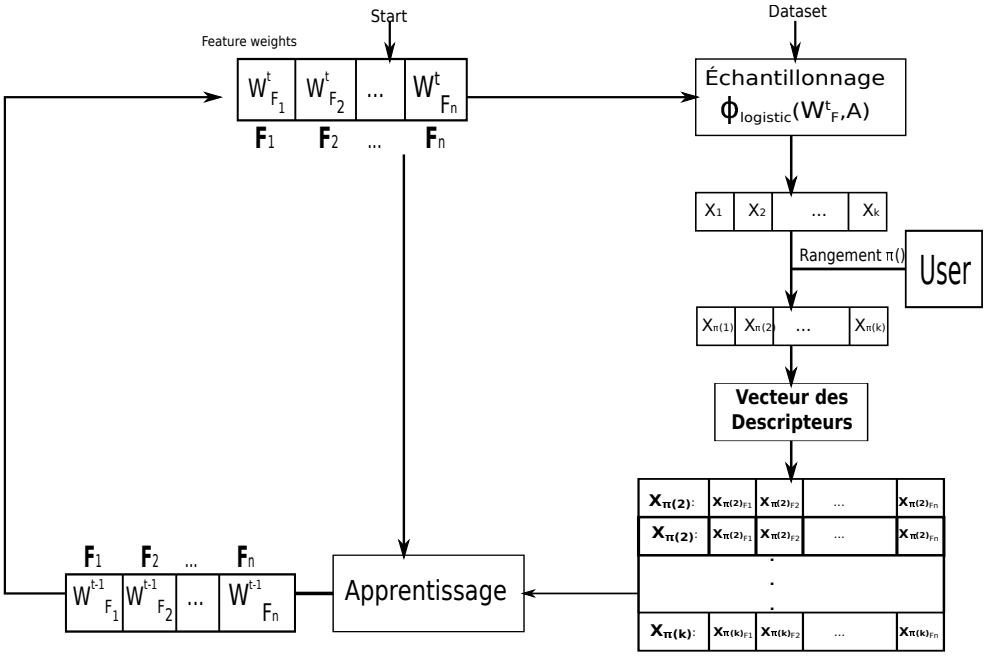


FIGURE 6.1 – Schéma général d'apprentissage des poids des descripteurs statiques.

temporaires, les poids $w^t_{F_{disc}}$ appris sont ensuite agrégés aux poids des descripteurs non discriminants afin d'être exploités à l'itération suivante. Ce nouveau processus, synthétisé par la figure 6.2, se présente comme suit :

(i) Pour chaque motif $X_i \in \mathcal{X}^t$, une re-description est effectuée :

$$\mathcal{X}^t_{\mathcal{F}^*} = \{\langle X_{i_{F_1}}, \ldots, X_{i_{F_n}}, X_{i_{F_{disc_X}}}, X_{i_{F_{disc_\mathcal{T}}}}, X_{i_{F_{disc_\mathcal{I}}}}\rangle,\ \forall i \in [1,k]\}.$$

(ii) De nouveaux poids $w^t_{F_i}$ sont appris pour chaque descripteurs $F_i \in \mathcal{F}^*$. Les poids appris $w^t_{F_{disc}}$ associés aux descripteurs discriminants sont alors agrégés aux poids des descripteurs statiques $F_{i_{i\in[1,n]}}$ en exploitant une fonction d'agrégation multiplicative qui sera présentée à la section suivante.

(iii) Pour terminer, le vecteur des descripteurs est mis à jour en retirant les descripteurs temporaires F_{disc}. On obtient alors l'ensemble des descripteurs $\mathcal{F} = \langle F_1 \ldots F_n\rangle$ auxquels sont associés les poids $w^t_\mathcal{F} = \langle w^t_{F_1} \ldots w^t_{F_n}\rangle$.

Dans la prochaine section, nous détaillerons comment le vecteur de poids $w^t_\mathcal{F}$ est mis à jour en utilisant les poids des descripteurs discriminants $w^t_{F_{disc}}$ via une fonction d'agrégation f^{ag}.

Exemple 6.4.1.
Reprenons l'exemple 6.2.2 et considérons le sous-motif discriminant $X_{disc} = \{7\}$ obtenu avec l'algorithme 9. Supposons que nous utilisons comme descripteurs l'ensemble des items et la fréquence des motifs : $\mathcal{F} = \langle F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_{freq}\rangle$. Le motif $X_1 = \{5,7\}$ est alors représenté par le vecteur descripteur $X_{1_\mathcal{F}} = \langle 0,0,0,0,1,0,1,0.54\rangle$ et le motif $X_3 = \{1\}$ par le vecteur $X_{3_\mathcal{F}} = \langle 1,0,0,0,0,0,0,0.36\rangle$. En utilisant le motif discriminant X_{disc}, on obtient le nouveau vecteur de descripteurs ci-dessous : $\mathcal{F}^* = \langle F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_{freq}, F_{disc_X}, F_{disc_\mathcal{T}}\rangle$. La re-description des motifs donne ainsi :
- *$X_{1_{\mathcal{F}^*}} = \langle 0,0,0,0,1,0,1,0.54, \boldsymbol{1}, \boldsymbol{0.63}\rangle$ car $X_{disc} \subset X_1$;*
- *$X_{3_{\mathcal{F}^*}} = \langle 0,0,0,0,1,0,1,0.36, \boldsymbol{0}, \boldsymbol{0}\rangle$ car $X_{disc} \not\subset X_3$.*

6.4.2 Exploitation des descripteurs discriminants

Comme nous l'avons présenté à la section 6.2.1, l'extraction des motifs discriminants permet d'identifier un sous-ensemble d'items corrélés au rangement de l'utilisateur. Afin de mettre en

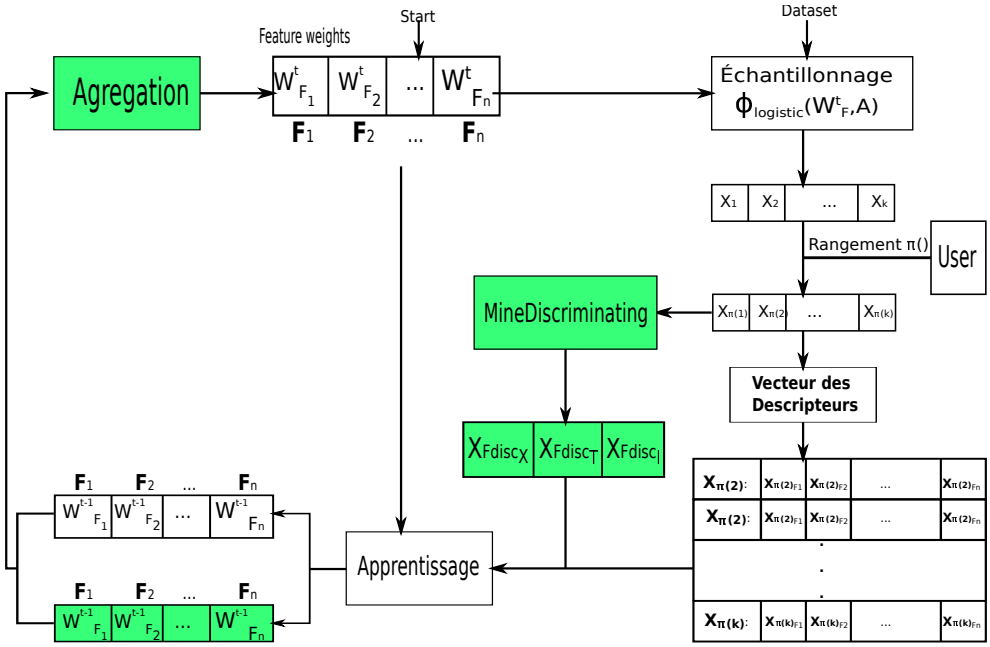


FIGURE 6.2 – Exploitation des descripteurs discriminants dans l'apprentissage des poids.

exergue l'intérêt de ces items pour l'utilisateur, nous proposons alors une double mise à jour des poids qui leur sont associées :

— une première mise à jour effectuée avec la méthode SCD ;
— une deuxième mise à jour effectuée en exploitant les poids des discriminants.

Nous proposons deux types d'agrégation des poids des descripteurs statiques à partir des poids des descripteurs discriminants appris :

— Pour les descripteurs binaires (items et transactions), l'agrégation est effectuée comme suit :
  o $w^t_{F_i} = f^{ag}(w^t_{F_i}, w^t_{F_{disc_X}})\ \forall F_i \in \mathcal{F}_\mathcal{I} \wedge i \in \mathcal{I}[X_{disc}]$, où $\mathcal{F}_\mathcal{I}$ représente l'ensemble des descripteurs des items et $\mathcal{I}[X_{disc}]$ représente les items de X_{disc}.

  o $w^t_{F_i} = f^{ag}(w^t_{F_i}, w^t_{F_{disc_X}})\ \forall F_i \in \mathcal{F}_\mathcal{T} \wedge i \in \mathcal{V}_\mathcal{D}(X_{disc})$, où $\mathcal{F}_\mathcal{T}$ représente l'ensemble des descripteurs des transactions.

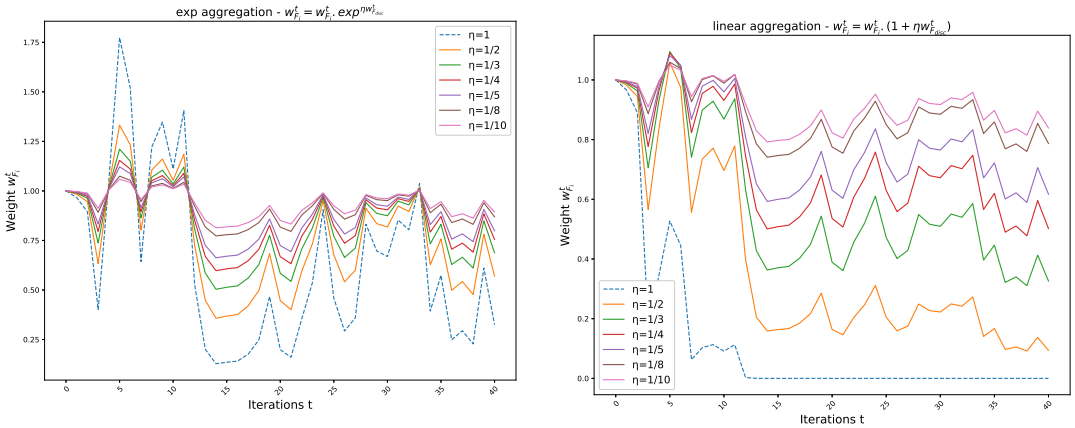
— Pour les descripteurs numériques (fréquence, longueur, ...), l'agrégation est réalisée comme suit :
  o $w^t_{F_{freq}} = f^{ag}(w^t_{F_{freq}}, w^t_{F_{disc_\mathcal{T}}})$, où F_{freq} représente le descripteur sur la fréquence.

  o $w^t_{F_{length}} = f^{ag}(w^t_{F_{length}}, w^t_{F_{disc_\mathcal{I}}})$, où F_{length} représente le descripteur sur la longueur des motifs.

Cette procédure de mise à jour permet ainsi de favoriser d'avantage les descripteurs qui ont permis un rangement des motifs qui les contiennent à des rangs élevés. Cela augmente leur probabilité d'être présents dans les motifs des prochaines itérations. De la même manière, la mise à jour peut être utilisée pour pénaliser d'avantage les descripteurs qui ont influencé un rangement des motifs qui les contiennent à des rangs faibles. Pour cela, nous proposons d'utiliser deux fonctions d'agrégation.

A. Fonctions d'agrégation

Notre approche est inspirée de [6], qui utilise une méthode multiplicative pour la mise à jour



(a) Fonctions d'agrégation exponentielle

(b) Fonctions d'agrégation inéaire

Figure 6.3 – Courbes évolutives des poids pour les deux facteurs multiplicatifs.

de poids w_i associés à des objets i. A chaque itération t, cette méthode met à jour les poids w_i^t des objets i de la manière suivante : $w_i^t = Agreg(w_i^t, c_i^t) = w_i^t \cdot \bigotimes(w_i^t, c_i^t)$, où $\bigotimes(w_i^t, c_i^t)$ est un facteur d'agrégation et c_i^t un gain (resp. pénalité) si $c_i^t \geq 1$ (resp. $c_i^t < 1$).

Notre objectif est d'agréger les poids appris des descripteurs discriminants $w^t_{F_{disc}}$ aux poids des descripteurs statiques $w^t_{F_i}$ des motifs qu'ils représentent. Nous proposons d'utiliser les deux facteurs d'agrégation suivants :

— un facteur d'agrégation linéaire : $\bigotimes(w^t_{F_i}, w^t_{F_{disc}}) = (1 + \eta w^t_{F_{disc}})$;

— un facteur d'agrégation exponentiel : $\bigotimes(w^t_{F_i}, w^t_{F_{disc}}) = exp^{\eta w^t_{F_{disc}}}$;

À chaque itération t, les poids des descripteurs F_i seront donc mis à jour comme suit :

$$w^t_{F_i} = \begin{cases} w^t_{F_i} \times (1 + \eta w^t_{F_{disc}}) \\ ou \\ w^t_{F_i} \times \exp^{\eta w^t_{F_{disc}}} \end{cases} \tag{6.5}$$

Le paramètre $\eta \in]0, \frac{1}{2}]$ est un paramètre de **régularisation** (inspiré de l'apprentissage automatique [120, 105, 86]) pour limiter les fortes variations des poids résultant de cette deuxième mise à jour. En effet, une trop forte variation des poids pourrait conduire à l'extraction de motifs de mauvaise qualité et nuire au processus d'apprentissage.

Les graphiques de la figure 6.3 permettent d'illustrer ce phénomène. Dans ces graphiques, nous représentons l'évolution des poids appris avec les deux facteurs multiplicatifs pour différentes valeurs de η. Ainsi, à chaque itération $t \in [1, 40]$, nous générons aléatoirement un poids $w^t_{F_{disc}} \in [-1, 1]$. Puis, nous mettons à jour $w^t_{F_i}$ qui sera alors incrémenté ($w^t_{F_{disc}} > 0$) ou décrémenté ($w^t_{F_{disc}} \leq 0$).

Comme nous pouvons le constater, pour de grandes valeurs de η, il y a de fortes variations du poids $w^t_{F_i}$ qui atteint des valeurs qui peuvent paraître aberrantes. Le graphique 6.3a montre qu'entre les itérations $t = 4$ et $t = 6$, on obtient des valeurs $w^t_{F_i}$ très grandes pour $\eta = 1$ par rapport aux autres. Dans le cas où ces poids $w^t_{F_i}$ sont utilisés pour pondérer les motifs, de grandes valeurs de poids auront pour conséquence de favoriser très fortement les motifs X tel que $X_{F_i} \neq 0$ au détriment des motifs décrits par d'autres descripteurs. Cette situation pourrait alors conduire à une réduction de la diversité dans les motifs extraits. Nous remarquons également que les poids $w^t_{F_i}$ diminuent avec la valeur de η et restent faibles par rapport aux valeurs obtenues pour $\eta = 1$. Un comportement similaire est observable avec les graphiques de la figure 6.3b.

Exemple 6.4.2.
Considérons l'exemple 6.4.1 vu précédemment. Supposons que nous sommes à l'itération $t = 1$ et que le motif discriminant vaut $X_{disc} = \{7\}$. Le vecteur de poids associé à $\mathcal{F}^*$ est défini par

$$w^1_{\mathcal{F}^*} = \langle w^1_1, w^1_2, w^1_3, w^1_4, w^1_5, w^1_6, w^1_7, w^1_{freq}, w^1_{disc_X}, w^1_{disc_\mathcal{T}} \rangle.$$

Après l'étape d'apprentissage, ce vecteur prend les valeurs suivantes :

$$w^1_{\mathcal{F}^*} = \langle -0.33, 0.99, 0, -0.99, 0.33, 0, 1.33, 0.15, 1.33, 0.84 \rangle.$$

En utilisant la fonction d'agrégation linéaire avec $\eta = 0.2$, nous mettons à jour les poids w^1_7 (car l'item $7 \in X_{disc}$) et w^1_{freq} comme suit :

- $w^1_7 = w^1_7 \cdot (1 + \eta w^1_{disc_X}) = 1.33 \times (1 + (0.2 \times 1.33)) = 1.68$;
- $w^1_{freq} = w^1_{freq} \cdot (1 + \eta w^1_{disc_\mathcal{T}}) = 0.15 \times (1 + (0.2 \times 0.84)) = 0.175$.

Le nouveau vecteur $w^1_\mathcal{F}$ associé à $\mathcal{F}$ vaudra alors :

$$w^1_\mathcal{F} = \langle -0.33, 0.99, 0, -0.99, 0.33, 0, \mathbf{1.68}, \mathbf{0.15} \rangle.$$

Ce nouveau vecteur sera utilisé par $\phi_{logistic}$ afin d'extraire de nouveaux motifs à la prochaine itération $t = 2$.

6.5 DiSPaLe : Un cadre interactif de fouille de motifs exploitant les discriminants et la diversité

Dans cette section, nous présentons une nouvelle approche de fouille interactive de motifs, dénommée DiSPaLe, qui exploite différents briques issues des travaux que nous avons présentés tout au long de ce document. Nous détaillons ci-dessous les trois composants de base de notre approche (voir l'algorithme 10) :

Fouille de motifs par échantillonnage. Comme pour LetSIP, nous utilisons une fonction logistique $\varphi_{logistic}$ comme une fonction de pondération des motifs. Contrairement à LetSIP, à chaque itération, les motifs sont échantillonnés proportionnellement à $\varphi_{logistic}$ en utilisant SDivJaX. De cette manière nous tirons parti à la fois des contraintes XOR et de la contrainte globale ClosedDiversity pour présenter à l'utilisateur des motifs diversifiés. Par ailleurs, la diversité est prise en compte de façon explicite pendant la phase d'extraction des motifs et non pas comme une heuristique post-traitement.

Interaction avec l'utilisateur et apprentissage à partir des feedback. DiSPaLe utilise un modèle de feedback basé sur un classement total des motifs. L'apprentissage de la fonction logistique se réduit à une classification binaire des vecteurs de différence des descripteurs des motifs. DiSPaLe exploite les motifs discriminants comme nouveaux descripteurs dans la phase d'apprentissage.

Sélection des motifs à présenter à l'utilisateur. Comme dans LetSIP, nous conservons les ℓ meilleurs motifs issus de l'itération précédente pour construire notre requête. Toutefois, les $(k - \ell)$ motifs restants sont échantillonnés avec SDivJaX.

6.6 Évaluation expérimentale

Dans cette section, nous évaluons expérimentalement notre approche DiSPaLe pour l'apprentissage des préférences de l'utilisateur. Nous étudierons les apports des discriminants dans la description des motifs ainsi que l'utilisation de la méthode d'extraction SDivJaX dans l'échantillonnage de motifs. Pour cela, nous adoptons les notations suivantes :

Algorithm 10: DiSPaLe (**Di**scriminating **S**ub-**Pa**ttern feature **Le**arning)

Entrée : Jeu de données $\mathcal{D}$, seuil de fréquence minimum θ, taille de la requête k, nombre d'itérations T, ℓ, range A, Descripteurs des motifs $\mathcal{F}$

begin

- $w_{\mathcal{F}}^0 \leftarrow \mathbf{0}\,;\, \mathcal{U} \leftarrow \emptyset\,;\, \mathcal{X}^0 \leftarrow \emptyset$ ▷ Initialisations
- $h_0 = \varphi_{logistic}(w_{\mathcal{F}}^0, A)$ ▷ Fonction de classement
- ▷ Mine, Interact, Learn, Repeat loop
- **for** $t = 1, 2 \ldots T$ **do**
  - $R = \textsc{TakeFirst}(\mathcal{R}^{*^{t-1}}, \ell)$ ▷ Retenir les top-ℓ motifs de l'itération précédente
  - $\mathcal{X}^t \leftarrow R \cup \textsc{SDivJaX}(\mathcal{D}, \theta, h_{t-1}) \times (k - \ell)$ ▷ Exécuter $(k-\ell)$ fois SDivJaX
  - $\mathcal{R}^{*^t} = \textsc{Ranger}(\mathcal{X}^t)\,;\, \mathcal{U} \leftarrow \mathcal{U} \cup \mathcal{R}^{*^t}$ ▷ Rangement de l'utilisateur
  - $X_{disc} \leftarrow$ **MineDiscriminating**$(\mathcal{X}^t, \mathcal{R}^{*^t})$ ▷ Extraction du discriminant et re-description
  - $\mathcal{F}^* \leftarrow \mathcal{F} \cup \mathcal{F}_{disc}$
  - $w_{\mathcal{F}}^{*^t} \leftarrow \textsc{LearnWeights}(\mathcal{U}, \mathcal{F}^*)$, $w_{\mathcal{F}}^t \leftarrow f^{ag}(w_{\mathcal{F}}^{*^t}, w_{F_{disc}})$ ▷ Apprentissage et agrégation
  - $\mathcal{F} \leftarrow \mathcal{F}^* - \mathcal{F}_{disc}$ ▷ Nouvelle re-description
  - $h_t \leftarrow \varphi_{logistic}(w_{\mathcal{F}}^t, A)$ ▷ Nouvelle fonction d'apprentissage
- **return** $w_{\mathcal{F}}^T$

— DiSPaLe-Exp désigne la méthode DiSPaLe utilisant une fonction d'agrégation exponentielle ;

— DiSPaLe-Lin désigne la méthode DiSPaLe utilisant une fonction d'agrégation linéaire ;

— DiSPaLe +Div désigne la méthode DiSPaLe utilisant SDivJaX-1 comme oracle d'extraction des motifs (voir chapitre 4) ;

— Par extension, DiSPaLe-E+Div (resp. DiSPaLe-L+Div) désignera la méthode DiSPaLe-Exp (resp. DiSPaLe-Lin) avec SDivJaX-1.

Nos expérimentations ont pour objectif d'adresser les questions de recherche suivantes :

Q$_1$ Quel effet les paramètres de DiSPaLe (choix de la fonction d'agrégation, choix de la valeur de η, taille de la requête) ont-ils sur la qualité des patterns appris ?

Q$_2$ Quel effet les paramètres de DiSPaLe ont-ils sur la qualité des motifs échantillonnés à l'aide de SDivJaX-1 ?

Nous noterons par LetSIP+Div pour désigner l'utilisation de SDivJaX-1 comme oracle d'extraction des motifs dans LetSIP à la place de EFlexics.

6.6.1 Protocole expérimental

L'évaluation des algorithmes de fouille interactive de motifs peut être difficile. En effet, les experts du domaine sont rares et il est difficile de collecter suffisamment de données pour tirer des conclusions fiables. Pour effectuer une évaluation approfondie, nous émulons les préférences de l'utilisateur à l'aide d'une mesure de qualité Φ cachée. Nous avons suivi le même protocole que Dzyuba *et al.* dans [48] : pour chaque jeu de données, un ensemble $\mathcal{S}$ de motifs fréquents est extrait sans connaissances préalables des préférences de l'utilisateur. Nous supposons l'existence d'un ordre R^* de l'utilisateur dérivé de Φ permettant de classer les motifs de $\mathcal{S}$. Plus précisément, si $X \succ Y$ alors $\Phi(X) > \Phi(Y)$. Ainsi, l'objectif est d'apprendre une fonction logistique $\varphi_{logistic}$ pour l'échantillonnage des motifs fréquents qui approxime Φ. Pour nos expérimentations, nous avons utilisé comme Φ la fonction *surprisingness* (*surp*) qui est définie comme suit :

$$surp(X) = \max\{sup_{\mathcal{D}}(X) - \prod_{i=1}^{|X|} sup_{\mathcal{D}}(\{i\}), 0\}$$

Jeu de données	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{I}\rvert$	θ	$\#\mathcal{S}$
Anneal	812	89	660	149 331
Chess	3 196	75	2 014	155118
German	1 000	110	350	161 858
Heart-cleveland	296	95	115	153 214
Hepatitis	137	68	35	148 289
Kr-vs-kp	3 196	73	2 014	155118
Lymph	148	68	48	146 969
Mushroom	8 124	112	813	155657
Soybean	630	50	28	143 519
Vote	435	48	25	142 095
Zoo-1	101	36	10	151 806

TABLE 6.1 – Caractéristiques des jeux de données utilisés par DISPALE. θ représente la fréquence absolue.

Pour comparer les performances de nos approches (DISPALE, LETSIP+DIV et DISPALE +DIV) avec celles de LETSIP, nous avons utilisé la mesure du *regret*. À chaque itération t, nous évaluons le regret de classer le motif X par Φ. Pour cela, nous calculons le rang centile $pct.rank(X_i)$ par Φ de chaque motif X_i $(1 \leq i \leq k)$ de la requête $\mathcal{X}^t$ comme suit :

$$pct.rank(v_i) = \frac{DI + \frac{DE}{2}}{|\mathcal{S}|}$$

avec $DI = |Y \in \mathcal{S},\ \Phi(Y) < \Phi(X_i)|$ et $DE = |Y \in \mathcal{S}, \Phi(Y) = \Phi(X_i)|$.

En calculant le rang centile des motifs $X_i \in \mathcal{X}^t$ avec la mesure Φ, on obtient ainsi le pourcentage de motifs $Y \in \mathcal{S}$ tels que $\Phi(Y) \leq \Phi(X_i)$. En considérant l'ensemble des motifs de la requête $\mathcal{X}^t$ obtenus avec la fonction apprise $\varphi_{logistic}$ à l'itération t, le rang centile permet de mesurer la capacité de la fonction logistique à extraire des motifs intéressants, c'est-à-dire des motifs X pour lesquels $\Phi(X)$ est élevé. Ainsi, la valeur idéale est 1 (la valeur la plus élevée possible de Φ sur tous les motifs fréquents a un rang centile de 1). Le regret est alors défini comme suit :

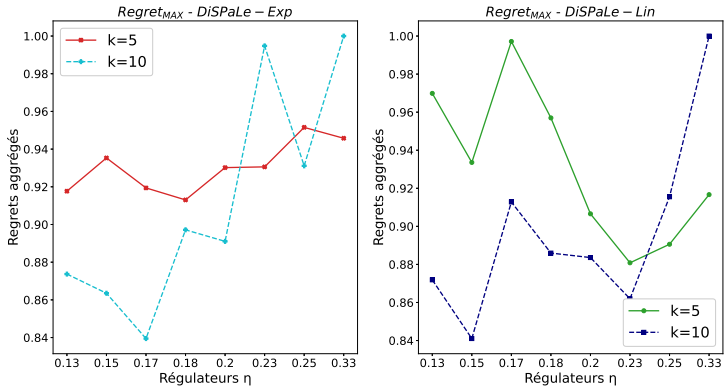
$$Regret_M(\mathcal{X}^t) = 1 - M_{(1 \leq i \leq k)}(pct.rank(X_i))$$

avec $k = |\mathcal{X}^t|$ et $M \in \{\max, Avg\}$. Il peut être évalué en utilisant la moyenne des rangs centiles $M = Avg$, on obtient alors un regret moyen $Regret_{Avg}$. Il peut aussi être évalué en calculant la valeur maximum de ces rangs centiles, dans ce cas, on obtient alors un regret maximum $Regret_{\max}$. Pour chaque jeu de données, nous calculons le *regret cumulé*, additionnée sur les différentes itérations. Étant donné que EFLEXICS et SDIVJAX-1 utilisent des contraintes `XOR` aléatoires, nous lançons chaque expérience 10 fois avec différentes graines aléatoires ; le regret cumulé moyen obtenu est rapporté. Pour s'assurer que toutes les méthodes sont échantillonnées sur les mêmes bases de motifs, à chaque itération, les mêmes ensembles de contraintes XOR sont générés dans les deux approches.

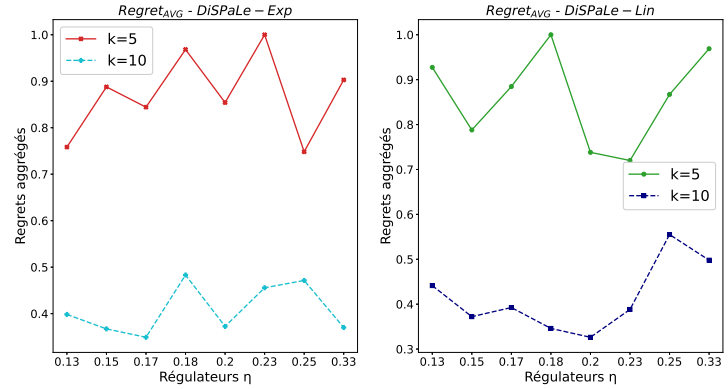
Pour nos évaluation, nous avons utilisé les jeux de données de l'UCI, disponibles dans le dépôt CP4IM [9]. Pour chaque jeu de données, nous avons utilisé un seuil de fréquence minimum tel que la taille de l'ensemble de motifs $\mathcal{S}$ soit approximativement égale à 140 000 motifs fréquents. La table 6.1 résume les différents seuils de fréquence et taille de $\mathcal{S}$. Par ailleurs, chaque expérimentation s'effectue en $T = 100$ itérations (requêtes) et nous utilisons les valeurs par défaut proposées dans [48] pour les paramètres de l'échantillonnage dans EFLEXICS et SDIVJAX : $\lambda = 0.001$, $\kappa = 0.9$, $A = 0.1$. Pour les autres paramètres communs aux approches DISPALE et LETSIP, nous considérons les valeurs suivantes :

— taille de requête $k \in \{5, 10\}$;

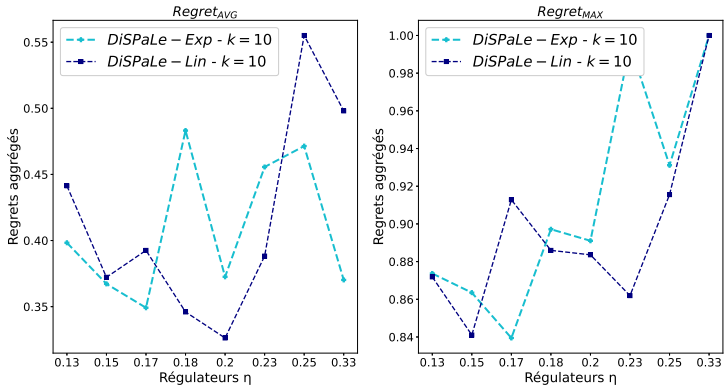
9. https://dtai.cs.kuleuven.be/CP4IM/datasets/



(a) $Regret_{\max}$ de DiSPaLe-Lin et DiSPaLe-Exp pour $k \in \{5, 10\}$.



(b) $Regret_{Avg}$ de DiSPaLe-Lin et DiSPaLe-Exp pour $k \in \{5, 10\}$.

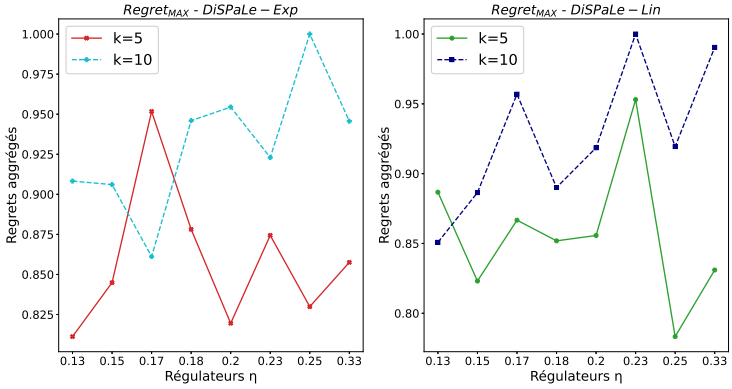


(c) $Regret_{\max}$ et $Regret_{Avg}$ de DiSPaLe-Lin et DiSPaLe-Exp pour $k = 10$.

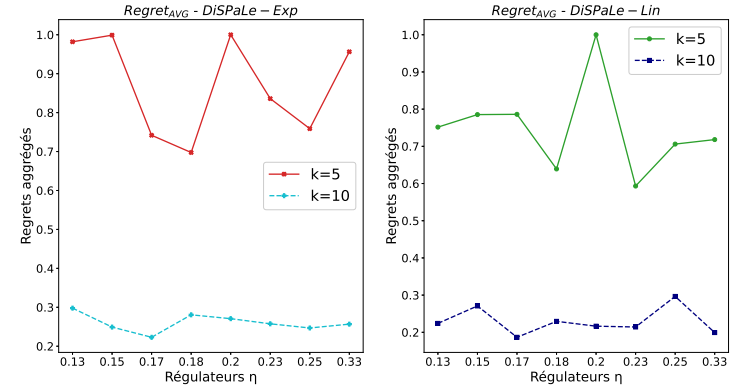
FIGURE 6.4 – Effet du paramètre de régularisation η de DiSPaLe sur les mesures du regret ($Regret_{best}$ et $Regret_{Avg}$) et pour $\ell = 0$. Les résultats sont agrégés sur les jeux de données de la table 6.1 et les trois combinaisons de descripteurs I, IT et ILFT. Les valeurs de regret sont normalisées sur [0,1].

— combinaisons suivantes de descripteurs : *Items* (I), *Items+Transactions* (IT), et *Items+ Longueur + Frequence + Transactions* (ILFT) ;
— paramètre ℓ définissant les ℓ meilleurs motifs issus de l'itération précédente $(t-1)$ et qui seront utilisés pour construire la requête à l'itération t : $\ell \in \{0, 1\}$.

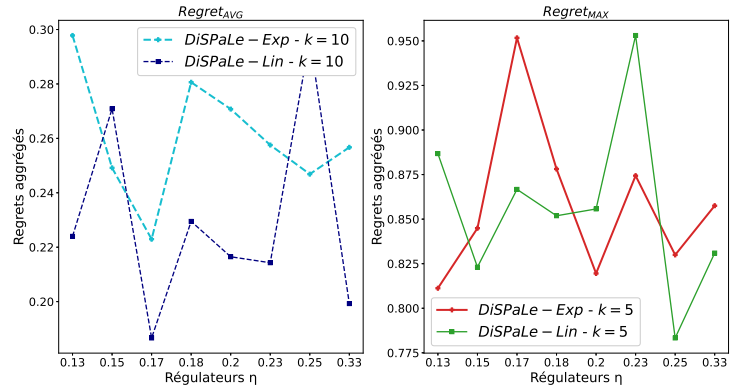
Pour le paramètre de régularisation η de DiSPaLe, nous utilisons les valeurs suivantes : $\eta \in \{0.13, 0.15, 0.17, 0.18, 0.2, 0.23, 0.25, 0.33\}$. Enfin, toutes les expérimentations ont été réalisées sur des processeurs AMD Opteron 6174 (2.2GHz) avec une mémoire RAM de 256 Go et un temps d'exécution maximum de 24 heures.



(a) $Regret_{\max}$ de DiSPaLe-Lin et DiSPaLe-Exp pour $k \in \{5, 10\}$.



(b) $Regret_{Avg}$ de DiSPaLe-Lin et DiSPaLe-Exp pour $k \in \{5, 10\}$.



(c) $Regret_{\max}$ et $Regret_{Avg}$ de DiSPaLe-Lin et DiSPaLe-Exp pour $k = 10$.

FIGURE 6.5 – Effet du paramètre de régularisation η de DiSPaLe sur les mesures du regret ($Regret_{\max}$ et $Regret_{Avg}$) et pour $\ell = 1$. Les résultats sont agrégés sur les jeux de données de la table 6.1 et les trois combinaisons de descripteurs I, IT et ILFT. Les valeurs de regret sont normalisées sur [0,1].

6.6.2 Évaluation de différents paramètres de DiSPaLe

Dans la première partie de l'évaluation expérimentale, nous examinons l'effet de différents réglages de paramètres sur DiSPaLe, en particulier l'influence des différentes combinaisons de descripteurs et comment DiSPaLe se compare à LetSIP.

a) Impact du choix de la fonction d'agrégation et des valeurs de η et k :

Pour évaluer les effets de ces trois paramètres sur le regret, nous avons considéré les 11 jeux de données de la table 6.1. La figure 6.4 montre l'évolution des valeurs de $Regret_{\max}$ et

TABLE 6.2 – Évaluation des descripteurs et comparaison entre (1) (DISPALE-EXP) et (2) (DISPALE-LIN) pour $\eta = 0.17$ et pour les deux mesures $Regret_{\max}$ et $Regret_{Avg}$. Les résultats correspondent aux valeurs agrégées pour $k = 10$ sur les jeux de données de la table 6.1.

	$\ell = 0$				$\ell = 1$			
	$Regret_{\max}$		$Regret_{Avg}$		$Regret_{\max}$		$Regret_{Avg}$	
Descripteurs	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)
I	**114.763**	115.282	554.136	**553.774**	11.450	**11.483**	499.730	499.946
IT	**93.026**	93.218	**492.738**	493.987	11.143	**11.133**	453.744	**453.265**
ITLF	**89.614**	90.027	**487.985**	491.932	11.8	**11.750**	**448.374**	451.422

$Regret_{Avg}$ pour différentes valeurs de η et pour $\ell = 0$. Nous remarquons alors dans la figure 6.4a que les meilleures valeurs de regrets sont obtenues en utilisant une taille de requête $k = 10$. Ces regrets indiquent que notre approche est capable de réaliser un bon apprentissage même lorsque la taille de la requête augmente. De plus, nous relevons que les meilleurs valeurs de regret sont obtenues pour un régulateur $\eta \in \{0.13, 0.15, 0.17\}$. Ces constats restent les mêmes pour la mesure $Regret_{Avg}$ (voir la figure 6.4b) où on obtient les meilleures valeurs de regret pour $k = 10$ et pour $\eta \in \{0.13, 0.15, 0.17\}$.

Pour déterminer la méthode permettant d'obtenir les meilleures valeurs de regret, nous avons représenté dans la figure 6.4c les regrets de DISPALE-EXP et DISPALE-LIN pour $k = 10$. L'analyse de ces graphiques nous permet ainsi de déduire que DISPALE-LIN est la méthode permettant d'obtenir les meilleures valeurs de regret.

Nous avons réalisé les mêmes évaluations pour $\ell = 1$ et reporté les résultats dans la figure 6.5. Nous constatons alors dans la figure 6.5b qu'avec la mesure $Regret_{Avg}$, les meilleurs résultats sont également obtenus avec une taille de requête $k = 10$. Cependant, avec la mesure $Regret_{\max}$ (voir la figure 6.5a), les meilleures valeurs de regret sont obtenues avec $k = 5$ quelle que soit la fonction d'agrégation $\bigotimes$ (linéaire ou exponentielle) utilisée. La figure 6.5c montre enfin que les plus petites valeurs de regret sont obtenus avec DISPALE-LIN.

Ces deux figures confirment ainsi que l'utilisation de petites valeurs de η permet d'éviter des variations trop importantes des poids w_{F_i}. La tableau 6.2 donne une vision détaillée des résultats de DISPALE-EXP et DISPALE-LIN pour chaque combinaison de descripteurs, pour $\eta = 0.17$ et $k = 10$. Nous pouvons remarquer que les deux approches se comportent de manière relativement similaire pour les deux mesures $Regret_{\max}$ et $Regret_{Avg}$. Pour la suite de nos expérimentations, nous choisissons ainsi d'utiliser les paramétrages suivants : $\eta = 0.17$ et $k = 10$.

b) Évaluation de l'impact des descripteurs de motifs :

Afin d'évaluer l'impact des descripteurs des motifs sur l'apprentissage, nous construisons les descripteurs des motifs de manière incrémentale. Nous commençons par évaluer DISPALE avec un liste réduite au descripteur d'items I, nous étendons ensuite cette liste avec de nouveaux descripteurs et analysons les améliorations éventuelles apportées. Les résultats de ces évaluations sont données dans la table 6.3 où, pour chaque combinaison de descripteurs, nous avons évalué DISPALE-LIN pour $k = 10$ et $\ell \in \{0, 1\}$.

Lorsque l'on considère les items comme descripteurs de motifs, LETSIP permet d'obtenir de meilleurs regrets et cela quelque soit la mesure considérée et la valeur de η. Toutefois, l'augmentation du nombre de descripteurs permet d'améliorer le regret de DISPALE-LIN. En effet, l'ajout de nouveaux descripteurs apporte des informations supplémentaires permettant de mieux décrire les motifs et d'améliorer l'apprentissage. Ainsi, pour DISPALE-LIN, nous constatons une diminution importante des valeurs de regret, notamment avec l'ajout du descripteur de transactions T. Pour le descripteur IT et pour $\ell = 0$, le $Regret_{\max}$ de DISPALE-LIN passe de 115.282 à 93.218, soit une amélioration de 19%. Pour $Regret_{Avg}$, cette amélioration est d'environ 11%. De même, pour $l = 1$ le $Regret_{Avg}$ de DISPALE-LIN passe de 499.946 pour le descripteur I à 453.265 pour

TABLE 6.3 – Évaluation des descripteurs et comparaison entre DISPALE-LIN ($\eta = 0.17$) et LETSIP pour les deux mesures $Regret_{\max}$ et $Regret_{Avg}$. Les résultats correspondent aux valeurs agrégées pour $k = 10$ sur les 11 jeux de données de la table 6.1.

	$\ell = 0$				$\ell = 1$			
	$Regret_{\max}$		$Regret_{Avg}$		$Regret_{max}$		$Regret_{Avg}$	
Descripteurs	LETSIP	DISPALE-LIN	LETSIP	DISPALE-LIN	LETSIP	DISPALE-LIN	LETSIP	DISPALE-LIN
I	**112.137**	115.282	554.816	**553.774**	**10.438**	11.483	**496.918**	499.946
IT	108.446	**93.218**	543.556	**493.987**	**10.761**	11.133	483.689	**453.265**
ITLF	106.006	**90.027**	538.848	**491.932**	**11.275**	11.750	490.818	**451.422**

IT. Ces résultats sont donc cohérents avec ceux rapportés par Dzyuba *et al.* dans [48]. Toutefois, nous devons noter que l'impact de chaque descripteur (ou groupe de descripteurs) dépend des caractéristiques des motifs extraits et de la fonction de préférences Φ [50]. Par exemple, pour l'extraction de motifs surprenants, le descripteur de la longueur L est le plus susceptible d'être inclus dans l'ensemble des meilleurs descripteurs car les motifs longs ont tendance à avoir des valeurs plus élevées de *surprisingness*. Dans ce cas, le descripteur des items I peut également être intéressant car les fréquences des items individuels sont directement incluses dans la formule du *surprisingness*. Le descripteur T des transactions a quand à lui une importance qui vient de sa capacité à représenter (indirectement) les interactions entre différents descripteurs.

6.6.3 Comparaisons entre DiSPaLe et LetSIP

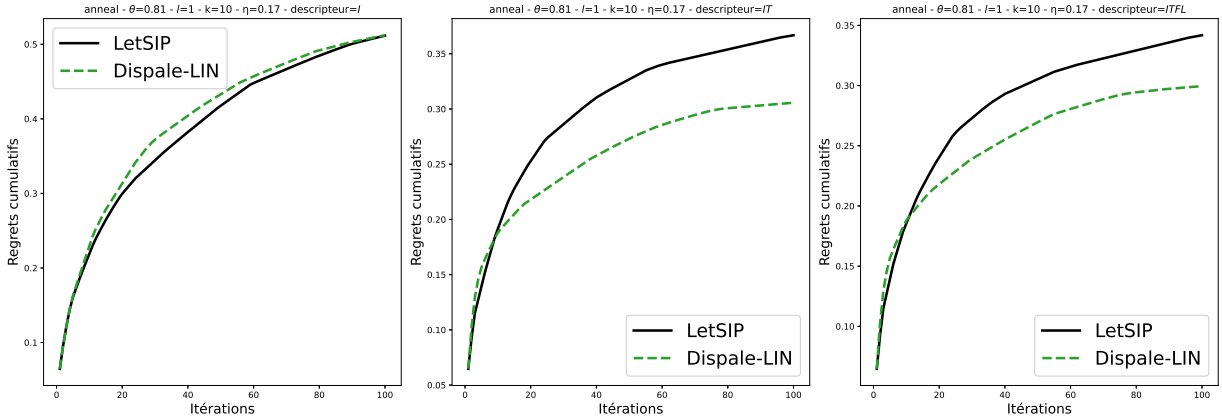
a) Analyse des regrets :

La table 6.3 montre les résultats expérimentaux comparant DISPALE-LIN et LETSIP sur différents paramètres. Nous constatons alors que pour le descripteur I, LETSIP obtient les meilleures valeurs de regret ($Regret_{Avg}$ et $Regret_{\max}$). Cependant, pour $\ell = 0$, le $Regret_{Avg}$ est sensiblement meilleur pour DISPALE-LIN. Par contre, en utilisant les descripteurs IT et $ITLF$, DISPALE-LIN permet d'obtenir des regrets plus réduits quelle que soit la valeur de ℓ (excepté pour le $Regret_{max}$ où LETSIP a de meilleurs résultats lorsque $\ell = 1$ et pour les descripteurs IT et $ITLF$). Ces résultats montrent ainsi que DISPALE-LIN permet de mieux apprendre la fonction de rangement Φ de l'utilisateur. En effet, l'analyse des $Regret_{Avg}$ indique que les motifs sélectionnés par DISPALE-LIN sont de meilleures qualité que ceux sélectionnés par LETSIP. Cela révèle alors que les rangements appris par DISPALE-LIN sur les k motifs sélectionnés par chaque requête sont plus précis que ceux appris par LETSIP. Enfin, nous remarquons que les meilleures valeurs de regrets sont obtenus en fixant ℓ à 1 quelle que soit la méthode évaluée. Ces résultats sont en concordance avec ceux de [48].

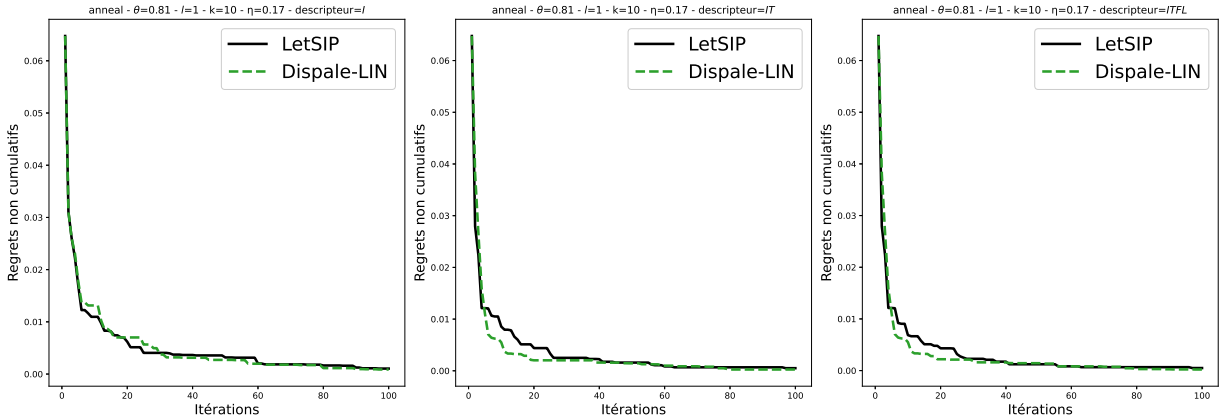
La figure 6.6 présente une vue détaillée des regrets cumulatifs et non cumulatifs de DISPALE-LIN et LETSIP sur le jeu de données ANNEAL (les autres résultats sont donnés en annexe B.1). Ces graphiques montrent l'évolution du regret (cumulatif et non cumulatif) sur 100 itérations d'apprentissage. Le regret cumulatif est obtenu en cumulant le regret calculé à chaque itération du processus d'apprentissage, ce qui donne une courbe croissante. Lorsque le regret s'améliore au fil des itérations, la pente de la courbe devient moins raide, ce qui indique ainsi une meilleure précision de la fonction d'apprentissage $\varphi_{logistic}$. Les graphiques de la figure 6.6 montrent la capacité de DISPALE-LIN à pouvoir identifier les motifs fréquents ayant la plus grande valeur de *surprisingness*, d'où un regret cumulatif ou non cumulatif plus réduit que celui de LETSIP.

b) Analyse des temps d'exécution :

Pour évaluer les performances en temps d'exécutions des deux méthodes DISPALE-EXP et LETSIP, nous avons mesuré pour chaque méthode le temps nécessaire pour effectuer une itération complète. La figure 6.10 montre l'évolution des temps de calcul de DISPALE-EXP et



(a) Comparaison des regrets cumulatifs de DiSPaLe-Lin et LetSIP.



(b) Comparaison des regrets non cumulatifs de DiSPaLe-Lin et LetSIP.

Figure 6.6 – Jeu de données Anneal : vue détaillée des comparaisons entre DiSPaLe-Lin et LetSIP (regret cumulatif et non cumulatif) pour différents descripteurs. $Regret_{\max}$, $k = 10$ et $\ell = 1$.

de LetSIP sur les jeux de données German-credit, Chess, Heart-Cleveland et Zoo-1. Dans l'ensemble, l'ajout des descripteurs discriminants ne pénalise pas significativement les performances de notre approche DiSPaLe-Lin. Ainsi, sur les jeux de données Chess, Heart-Cleveland, Kr-vs-kp et Soybean, DiSPaLe-Lin a de meilleurs temps d'exécution que LetSIP. Sur les trois autres jeux de données, les deux approches ont les mêmes performances, avec néanmoins de meilleures performance pour LetSIP sur les jeux de données German-credit, Hepatitis, Zoo-1 (voir les résultats complémentaires en annexe B.2)

Une itération de DiSPaLe-Lin s'effectue alors en moyenne en $8.84s$ contre $8.81s$ pour LetSIP sur AMD Opteron 6174 à 2.2 GHz. Notons que ce temps d'exécution moyen est fortement lié à la composante d'échantillonnage, alors que le temps d'apprentissage des poids est relativement faible. Comme cela sera montré dans la section suivante, l'utilisation de SDivJaX-1 comme oracle permet d'améliorer les temps d'exécutions.

6.6.4 Apports de la diversité dans LetSIP et DiSPaLe

Dans cette section, nous évaluons l'impact de la diversité sur la qualité de l'apprentissage des deux méthodes LetSIP et DiSPaLe. Pour cela, nous comparerons les regrets obtenus par LetSIP+Div et DiSPaLe +Div avec ceux obtenus avec LetSIP et DiSPaLe.

a) Analyse des regrets :

La table 6.4 présente les résultats de LetSIP et LetSIP+Div pour des seuils de diversité

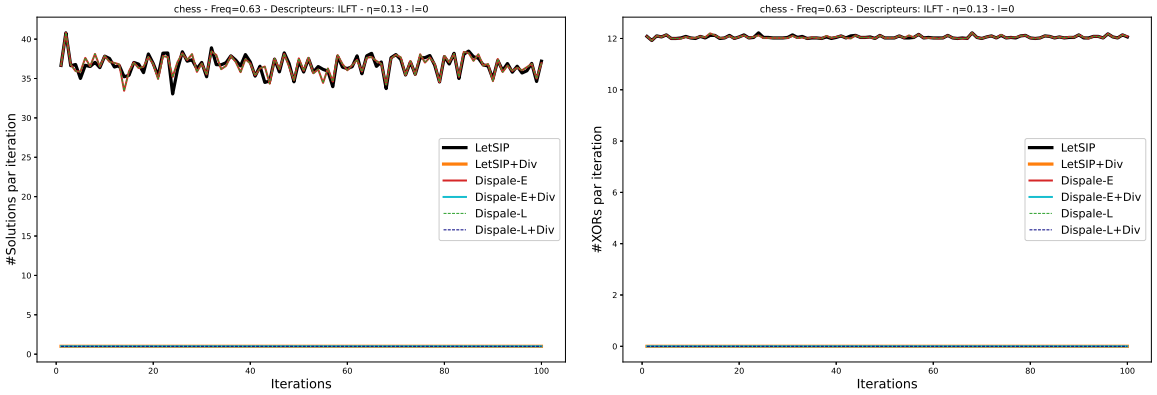
$J_{max} \in \{0.05, 0.1\}$. Ces résultats montrent que les regrets obtenus par LetSIP sont meilleurs que ceux de LetSIP+Div, quelque soit la configuration de paramètres utilisée.

Table 6.4 – Évaluation des descripteurs et comparaison entre LetSIP+Div et LetSIP pour différentes valeurs de J_{max}. Les résultats sont agrégés sur les jeux de données de la table 6.1.

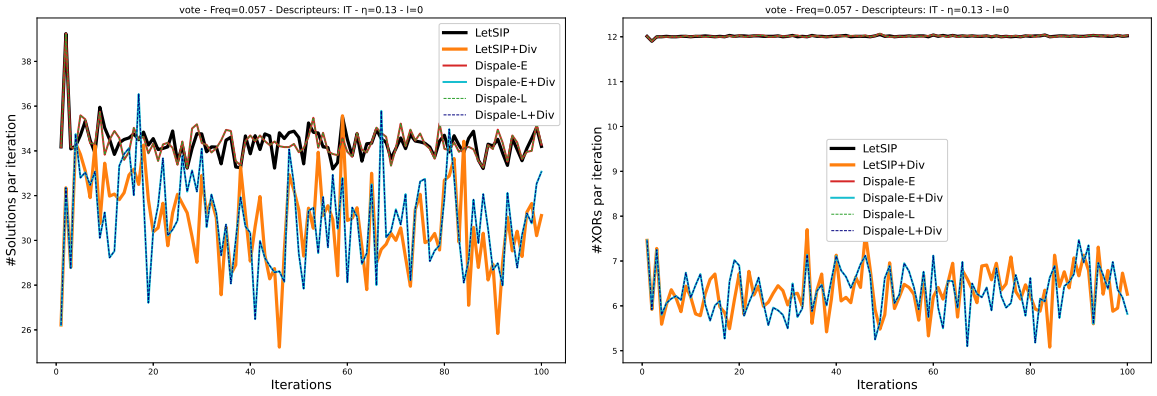
	$J_{max} = 0.05 \quad \wedge \quad \ell = 1$							
	$k = 5$				$k = 10$			
	$Regret_{\max}$		$Regret_{Avg}$		$Regret_{\max}$		$Regret_{Avg}$	
Descripteurs	LetSIP	LetSIP+Div	LetSIP	LetSIP+Div	LetSIP	LetSIP+Div	LetSIP	LetSIP+Div
I	**23.930**	303.124	**493.488**	696.161	**12.280**	299.596	**530.041**	744.815
IT	**23.055**	304.353	**429.837**	674.710	**13.667**	299.779	**474.707**	719.523
ITLF	**23.027**	304.531	**430.148**	676.519	**14.302**	299.896	**481.855**	718.811
	$J_{max} = 0.05 \quad \wedge \quad \ell = 0$							
	$k = 5$				$k = 10$			
	$Regret_{\max}$		$Regret_{Avg}$		$Regret_{\max}$		$Regret_{Avg}$	
Descripteurs	LetSIP	LetSIP+Div	LetSIP	LetSIP+Div	LetSIP	LetSIP+Div	LetSIP	LetSIP+Div
I	**210.562**	428.302	**606.310**	794.980	**113.256**	355.997	**581.242**	795.397
IT	**185.920**	423.707	**539.962**	764.410	**106.145**	357.471	**532.782**	766.422
ITLF	**183.906**	422.050	**536.476**	761.769	**103.849**	356.785	**528.131**	764.837
	$J_{max} = 0.1 \quad \wedge \quad \ell = 1$							
	$k = 5$				$k = 10$			
	$Regret_{\max}$		$Regret_{Avg}$		$Regret_{\max}$		$Regret_{Avg}$	
Descripteurs	LetSIP	LetSIP+Div	LetSIP	LetSIP+Div	LetSIP	LetSIP+Div	LetSIP	LetSIP+Div
I	**23.930**	298.573	**493.488**	709.064	**12.280**	292.853	**530.041**	756.061
IT	**23.055**	297.968	**429.837**	682.219	**13.667**	294.098	**474.707**	728.996
ITLF	**23.027**	297.904	**430.148**	682.073	**14.302**	293.999	**481.855**	729.397
	$J_{max} = 0.1 \quad \wedge \quad \ell = 0$							
	$k = 5$				$k = 10$			
	$Regret_{\max}$		$Regret_{Avg}$		$Regret_{\max}$		$Regret_{Avg}$	
Descripteurs	LetSIP	LetSIP+Div	LetSIP	LetSIP+Div	LetSIP	LetSIP+Div	LetSIP	LetSIP+Div
I	**210.562**	436.399	**606.310**	809.965	**113.256**	358.009	**581.242**	808.377
IT	**185.920**	413.840	**539.962**	778.964	**106.145**	345.637	**532.782**	780.764
ITLF	**183.906**	414.755	**536.476**	779.480	**103.849**	346.290	**528.131**	781.442

Pour mieux comprendre ce comportement, nous avons analysé, pour chaque méthode, le nombre de motifs extraits à chaque itération par EFlexics (utilisé dans DiSPaLe et LetSIP) et SDivJaX-1 (utilisé dans DiSPaLe +Div et LetSIP+Div). La figure 6.7 montre que pour les jeux de données denses comme Chess (figure 6.7a), le nombre de motifs extraits à chaque itération par LetSIP+Div et DiSPaLe +Div est faible (1 motif par itération) comparé à LetSIP et DiSPaLe. En effet, sur les jeux de données denses, en raison d'une très forte redondance entre les couvertures des motifs, SDivJaX-1 ne retourne que des ensembles de motifs très réduits ce qui est insuffisant pour permettre un apprentissage plus précis des préférences de l'utilisateur (voir la vue détaillée des comparaisons entre LetSIP et LetSIP+Div des regrets cumulatifs et non cumulatifs à la figure 6.8). Par ailleurs, en analysant le nombre de contraintes `XOR` utilisées par LetSIP+Div, nous constatons que celui-ci est nulle, ce qui confirme l'hypothèse d'un filtrage important des motifs par ClosedDiversity.

Cependant, pour des jeux de données moins denses, comme Vote (voir la figure 6.7b), nous remarquons que le nombre de motifs extraits par LetSIP+Div est plus importants (en moyenne 27 motifs par itération). En effet, la redondance au niveau des couvertures des motifs étant moins importante et le filtrage également (voir le chapitre 5). Plus important, les résultats des comparaisons des regrets cumulatifs et non cumulatifs (voir la figure 6.9) montrent que LetSIP+Div est bien meilleur comparé à LetSIP. Les résultats des autres jeux de données sont présentés dans l'annexe B.1. Ainsi, les résultats présentés à la table 6.4 incluent des instances où LetSIP+Div est meilleur et d'autres instances qui ne lui sont pas favorables. Par ailleurs, nous



(a) Chess



(b) Vote

FIGURE 6.7 – Nombre de XORs et de Motifs par itérations de LETSIP, LETSIP+DIV, DISPALE et DISPALE +DIV. $k = 10$, $\ell = 0$ et $J_{max} = 0.05$.

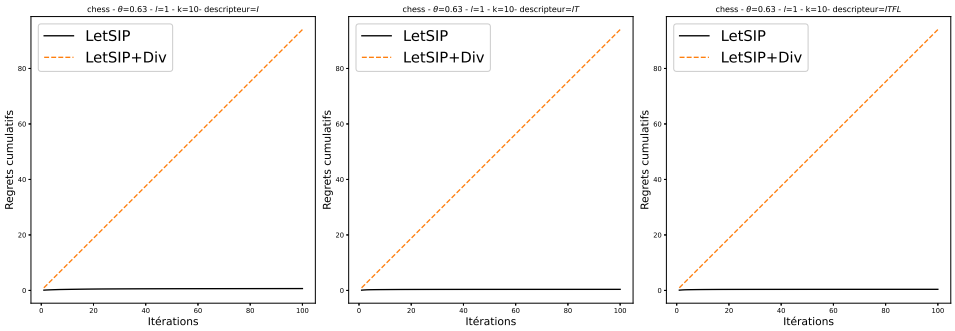
avons dressé le même constat pour DISPALE +DIV, ce dernier utilise également SDIVJAX-1 pour extraire les motifs. Les résultats de DISPALE ne sont dons pas reportés dans ce manuscrit.

b) Analyse des temps d'exécution :

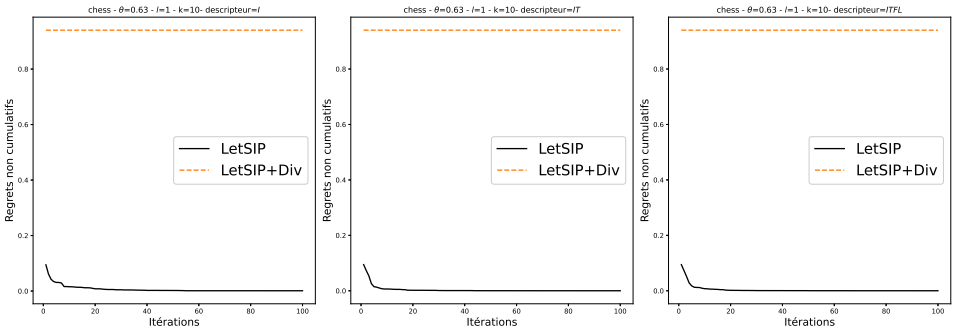
La figure 6.10 compare les performances de LETSIP+DIV à celles de LETSIP sur les jeux de données GERMAN-CREDIT, CHESS, HEART-CLEVELAND et ZOO-1. Ces graphiques montrent ainsi que LETSIP+DIV est plus rapide que LETSIP. Cela peut s'expliquer notamment par l'utilisation de SDIVJAX-1 comme méthode d'extraction des motifs qui permet une extraction plus rapide. En effet, comme nous l'avons montré au chapitre 5, SDIVJAX-1 permet de réduire la taille des cellules obtenues avec les contraintes `XOR` et CLOSEDDIVERSITY en éliminant les motifs non diversifiés, ce qui permet d'accélérer l'exploration. Les seules exceptions à cette observation viennent des jeux de données ZOO-1 et VOTE pour lesquels LETSIP est plus rapide. Par ailleurs, nous remarquons que LETSIP+DIV et DISPALE +DIV (DISPALE-EXP) ont des performances similaires sur la plus part des instances.

6.7 Conclusions

Dans ce chapitre, nous avons proposé une nouvelle approche d'apprentissage des préférences de l'utilisateur exploitant les motifs discriminants. Ces motifs sont corrélés au rangement de l'utilisateur et leur exploitation comme descripteur nous permet d'améliorer la précision de l'apprentissage. Les expérimentations montrent ainsi que notre méthode DISPALE permet d'obtenir de meilleurs résultats que LETSIP.



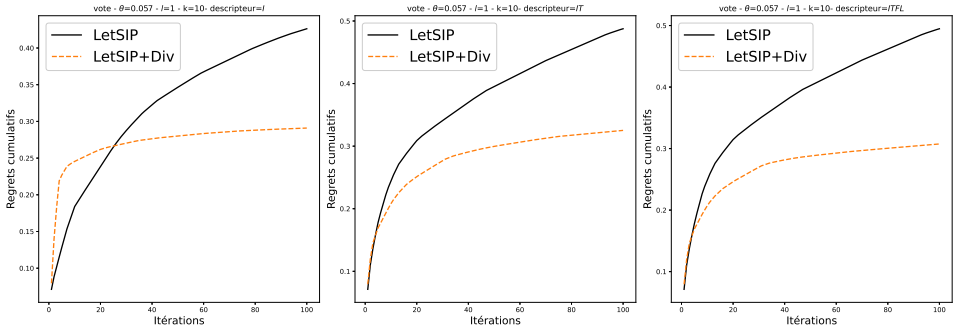
(a) Comparaison des regrets cumulatifs de LetSIP et LetSIP+Div.



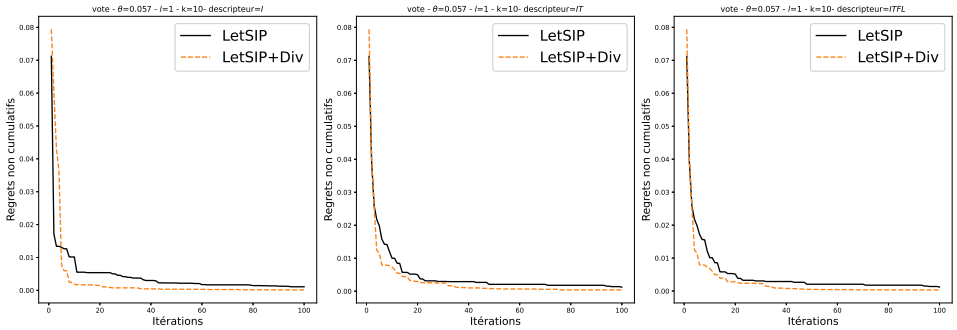
(b) Comparaison des regrets non cumulatifs de LetSIP et LetSIP+Div.

Figure 6.8 – Jeu de données Chess : vue détaillée des comparaisons entre LetSIP et LetSIP+Div (regrets cumulatifs et non cumulatifs) pour différents descripteurs. $Regret_{\max}$, $k = 10$ et $\ell = 1$.

Nous avons également évalué l'apport de la diversité dans LetSIP et dans DiSPaLe. Pour cela, nous avons comparé l'utilisation de SDivJaX comme oracle d'extraction des motifs à celle de EFlexics. Nous montrons ainsi que SDivJaX améliore la qualité de φ lorsque le nombre de motifs diversifiés est élevé. Toutefois, les résultats sont plus mitigés lorsque ce nombre est faible.

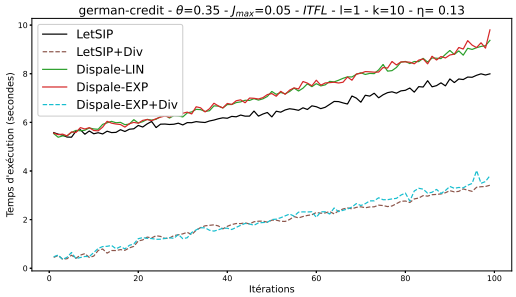


(a) Comparaison des regrets cumulatifs de LetSIP et LetSIP+Div.

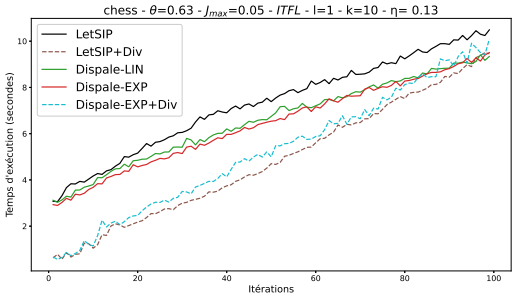


(b) Comparaison des regrets non cumulatifs de LetSIP et LetSIP+Div.

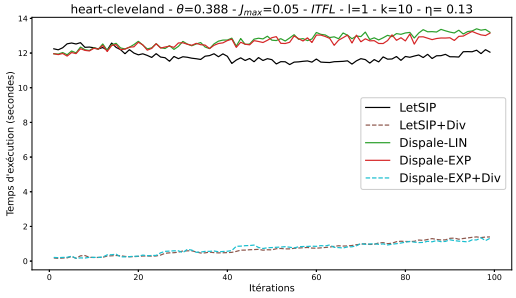
Figure 6.9 – Jeu de données Vote : vue détaillée des comparaisons entre LetSIP et LetSIP+Div (regrets cumulatifs et non cumulatifs) pour différents descripteurs. $Regret_{\max}$, $k = 10$ et $\ell = 1$.



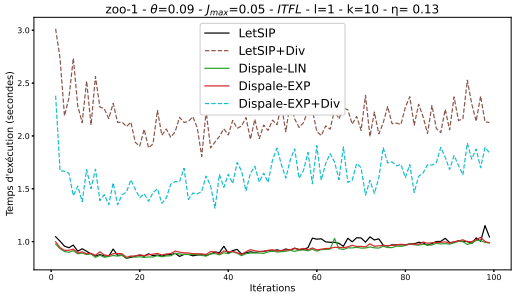
(a) German-credit



(b) Chess



(c) Heart-cleveland



(d) Zoo-1

Figure 6.10 – Temps d'exécution en *sec.* de LetSIP, LetSIP+Div, DiSPaLe-Exp et DiSPaLe-E+Div pour la combinaison de descripteurs ILFT. $k = 10$ et $\ell = 1$.

Quatrième partie

Conclusions et perspectives

CHAPITRE 7

Bilan et perspectives

Contents

7.1 Bilan 141
7.1.1 Méthodes d'extraction de motifs diversifiés 141
7.1.2 Échantillonnage de motifs diversifiés 142
7.1.3 Nouveaux descripteurs dynamique décrivant le rangement de l'utilisateur . . . 142
7.1.4 Apprentissage des préférences avec les descripteurs discriminants 142
7.2 Perspectives 142
7.2.1 Amélioration de la diversité des ensembles de motifs 142
7.2.2 Nouvelles évaluations de la diversité des ensembles de motifs 143
7.2.3 Amélioration des performances de SDivJaX 144
7.2.4 De nouvelles formes retours utilisateur et de représentation des motifs 144

Dans cette partie, nous faisons un bilan des différents travaux menés au cours de cette thèse sur la fouille interactive de motifs et présentons des ouvertures possibles vers d'autres travaux.

7.1 Bilan

L'objectif de cette thèse était de développer de nouvelles méthodes permettant de prendre en compte les préférences utilisateur dans le processus de fouille. Pour y arriver, notre travail s'est principalement focalisé autour des deux axes suivants :

(1) l'extraction rapide de motifs diversifiés ;

(2) la représentation et l'apprentissage des préférences de l'utilisateur.

Nous résumons ci-dessous les principales contributions de cette thèse.

7.1.1 Méthodes d'extraction de motifs diversifiés

Dans le chapitre 4, nous avons proposé deux méthodes de fouille de motifs diversifiés. Ces deux méthodes utilisent l'indice de Jaccard comme mesure de la similarité des motifs ainsi que les règles de filtrage de ClosedPatterns pour assurer la fréquence et la fermeture des motifs. Elles se distinguent néanmoins par leur manière de modéliser la diversité entre les motifs. La première méthode, FullCP, garantit la diversité en la formulant sous forme de contraintes linéaires. Ces contraintes sont obtenues en décomposant l'indice de Jaccard en opérations simples sur des

ensembles utilisant des variables booléennes intermédiaires pour représenter les couvertures des motifs. La deuxième méthode, ClosedDiversity, exploite deux relaxations anti-monotones de l'indice de Jaccard pour assurer la diversité dans les ensembles de motifs produits. De façon générale, nos expérimentations montrent que ClosedDiversity a de meilleures performances que FullCP sur les jeux de données denses qui intègrent beaucoup de redondances. Cependant, sur les jeux de données modérément denses et clairsemés, FullCP permet d'obtenir des performances comparables ou meilleures que ceux de ClosedDiversity.

7.1.2 Échantillonnage de motifs diversifiés

Dans le but d'obtenir une méthode de production rapide de motifs diversifiés, nous avons proposé dans le chapitre 4 SDivJaX qui permet d'échantillonner des motifs diversifiés. Cette méthode exploite ainsi Weightgen et la contrainte globale ClosedDiversity. Elle procède par des réductions successive de la taille de l'espace de recherche avec des contraintes XOR, puis utilise ClosedDiversity comme oracle pour extraire les échantillons de motifs. Cette méthode permet ainsi de bénéficier d'une diversité assurée de manière implicite avec les contraintes XOR et de manière explicite avec ClosedDiversity.

7.1.3 Nouveaux descripteurs dynamique décrivant le rangement de l'utilisateur

Les différents outils de représentations des motifs utilisés dans la littérature ont la particularité d'être statiques et de ne pas évoluer au fil de l'apprentissage. Pour palier à cette rigidité, nous avons proposé une nouvelle classe de descripteurs basés sur les motifs discriminants. Il s'agit de (sous-)motifs permettant de représenter la corrélation entre le rangement de l'utilisateur et la présence d'un sous-ensemble d'items dans les motifs qui lui sont présentés. L'utilisation de ces nouveaux descripteurs en plus des descripteurs statiques (items, transactions, longueur, fréquence) a ainsi permis d'apporter plus de sémantique à la description des motifs.

7.1.4 Apprentissage des préférences avec les descripteurs discriminants

Nous avons proposé une méthode permettant d'agréger les poids appris des descripteurs discriminants sur les poids des autres descripteurs. Par cette méthode, la sémantique des descripteurs discriminants est mise à contribution pour impacter d'avantage les poids des différents descripteurs statiques. Les évaluations effectuées au chapitre 6 montrent alors que notre méthode DiSPaLe permet d'améliorer la qualité de l'apprentissage comparativement à LetSIP et permet de sélectionner des motifs de plus en plus intéressants pour l'utilisateur. Cependant, la qualité de l'apprentissage se dégrade lors de l'utilisation de la méthode d'échantillonnage SDivJaX, principalement sur les jeux de données denses. Cette situation s'explique par l'insuffisance de motifs pendant l'apprentissage à cause de l'utilisation d'un seuil de diversité J_{max} très bas et à cause de la présence de nombreuse redondances dans les jeux de données denses. La précision de l'apprentissage devrait donc s'améliorer après une mise à jour des paramètre de SDivJaX pour permettre l'extraction de motifs en nombre plus importants.

7.2 Perspectives

Les perspectives de recherche s'inscrivent dans la continuité de ces travaux de recherche avec pour objectif d'une part d'améliorer les méthodes proposées dans ce manuscrit et d'autre part de proposer de nouvelles approches inspirées de celles qu nous avons proposées.

7.2.1 Amélioration de la diversité des ensembles de motifs

Dans ce manuscrit, nous avons proposé différentes méthodes permettant de réduire la redondance das les ensembles de motifs. L'une des perspectives de ce travail serait d'explorer les

possibilités d'amélioration de la diversité. Pour cela, plusieurs pistes peuvent être envisagées.

Combinaison de ClosedDiversity avec FullCP. L'une des premières pistes d'amélioration de la diversité dans les ensembles de motifs serait de combiner le modèle FullCP-2 (vu comme des contraintes redondantes) avec la contrainte globale ClosedDiversity. Cette approche a pour objectif de tirer profit des avantages des deux méthodes. La méthode obtenue pourrait ainsi exploiter les règles de filtrage de ClosedDiversity pour filtrer les motifs non diversifiés et garantir la diversité des motifs avec les contraintes linaires de FullCP-2.

Top-k motifs diversifiés. Une autre amélioration de nos méthodes de diversité consiste à extraire les Top-k motifs diversifiés. Il s'agit d'une tâche similaire à celle de MaxDiverseKSet(k) (voir la définition 27) et qui consiste à trouver le meilleur sous-ensemble de k motifs diversifiés. La procédure d'extraction consiste alors à trouver les motifs de la théorie suivante :

$$\mathcal{T}h(2^{\mathcal{L}_\mathcal{I}}, \mathcal{D}, \mathbb{C}) = \{\mathbb{X} \in 2^{\mathcal{L}_\mathcal{I}} \mid \nexists \mathbb{Y} \in 2^{\mathcal{L}_\mathcal{I}} \max_{Y,Y' \in \mathbb{Y}} \textsc{Jaccard}(Y, Y') \leq \max_{X,X' \in \mathbb{X}} \textsc{Jaccard}(X, X')\}$$

avec $taille(\mathbb{X}) = taille(\mathbb{Y}) = k$

L'un des avantages de cette approche est de pouvoir affranchir l'utilisateur de la définition du seuil de diversité J_{max}. En effet, étant donné qu'on cherche le meilleur sous-ensemble de motifs diversifiés, le seuil de diversité sera initialisé et mis à jour par la méthode elle-même. Pour cela, on l'initialise avec un sous-ensemble de k motifs dont on calcule l'indice de Jaccard maximum J_{max}. Cet indice de Jaccard sera alors exploité pour trouver de nouveaux motifs ayant un indice de Jaccard par rapport aux motifs de $\mathbb{X}$ inférieur à J_{max}, tout en mettant à jour au fur et à mesure le seuil J_{max}. Toutefois, les performances d'une telle approche dépendent par le premier sous-ensemble de k motifs extraits permettant d'initialiser le seuil de diversité J_{max} ; cette valeur doit être la plus petite possible pour garantir une exploration rapide de l'espace de recherche.

De nouvelles heuristiques d'exploration pour ClosedDiversity. Une des pistes d'amélioration de notre contrainte globale ClosedDiversity réside dans l'amélioration de l'exploration de l'espace de recherche. En effet, les résultats de ClosedDiversity sont fortement liés à l'ordre de découverte des motifs qui peuvent impacter la qualité de l'ensemble final produit. Une première contribution à l'amélioration de l'exploration de l'espace de recherche a été faite avec l'heuristique Witness qui permet de certifier la diversité des motifs d'un sous-arbre témoin. Nous proposons de définir de nouvelles heuristiques d'exploration dans lesquelles les branchements sur les items e seraient déterminés par l'indice de Jaccard du motif partiel X^+ par rapport à l'historique Jaccard$(X \cup e, \mathcal{H})$. Cette approche orthogonale à nos travaux de thèse, a pour avantage de permettre l'extraction de motifs dont la diversité est garantie à chaque nœud puisque la contrainte de Jaccard est vérifiée. Nous proposons de combiner cette heuristique de branchement avec un tirage aléatoire pour sélectionner des items sur lesquels brancher. Cette approche, malgré sa simplicité, permet en effet d'effectuer une recherche des motifs dans tout l'espace de recherche. Ce qui permet d'améliorer la diversité.

7.2.2 Nouvelles évaluations de la diversité des ensembles de motifs

Dans ce manuscrit, nos méthodes de réduction de la redondance procèdent par paires de motifs pour évaluer la diversité. Par ailleurs, elles sont toutes basées sur l'indice de Jaccard qui leur permet d'évaluer la similarité dans les ensembles de motifs. Cependant, d'autres mesures de similarité des motifs peuvent être exploitées et l'évaluation de la similarité peut se faire de façon globale sur l'ensemble des motifs.

Mesures de la diversité sur les ensembles de motifs. Comme nous l'avons présenté dans le chapitre 1, il existe des mesures d'évaluation de la diversité sur les ensembles de motifs. Ces

mesures calculent la redondance entre les motifs en effectuant une évaluation de tout l'ensemble de motifs en même temps plutôt que de procéder par paires. Une approche similaire a été proposée dans ce sens par Ouali dans [108]. Il s'agit de l'évaluation des transactions redondantes présentées à la section 1.4.2.

Utilisation de nouvelles mesures d'évaluation de la diversité. Tout au long de cette thèse, nous avons utilisé l'indice de Jaccard comme mesure d'évaluation de la redondance entre les paires de motifs. Or, il existe dans la littérature d'autres mesures permettant d'assurer la diversité. Étant donné que nous proposons avec ClosedDiversity un cadre générique d'extraction de motifs, nous pouvons alors d'exploiter l'entropie comme mesure d'évaluation de la diversité à la place de l'indice de Jaccard. Nous proposons alors d'utiliser un modèle proche de celui de PatternsTeam [93]. Il s'agit alors de trouver le sous-ensemble de motifs permettant d'optimiser l'entropie conjointe $H(\mathcal{T}, \mathbb{X})$. On pourra alors proposer une contrainte de diversité telle que le meilleur sous-ensemble de motifs diversifiés ait une entropie supérieure à un seuil H_{max} :

$$H(\mathcal{T}, \mathbb{X}) \geq H_{max}$$

Si la mesure de l'entropie conjointe n'est pas anti-monotone, on pourra proposer une relaxation de la contrainte et exploiter le cadre générique de ClosedDiversity.

7.2.3 Amélioration des performances de SDivJaX

Durant l'évaluation de SDivJaX au chapitre 5, nous avons constaté que SDivJaX-2 était pénalisé par l'utilisation d'un historique globale $\mathcal{H}_{global}$ de motifs. En effet, cet historique étant vide au départ, ClosedDiversity filtre peu de motifs pendant l'échantillonnage. SDivJaX-2 est alors impacté fortement par le surcoût des calculs de LB_J qui dégrade ses performances.
Pour remédier à cette situation, nous proposons d'utiliser un historique hybride $\mathcal{H}_{hybrid}$ qui fonctionnerait comme suit :

— Supposons qu'à l'itération t-1, l'historique $\mathcal{H}_{hybrid}$ contient $t-1$ motifs :

$$\mathcal{H}^{t-1}_{hybrid} = \{s_1, \ldots, s_{t-1}\}$$

— Pendant l'énumération de l'itération t, nous exploitons $\mathcal{H}_{hybrid}$ comme un historique local. De ce fait, nous utilisons ClosedDiversity pour extraire les motifs X_i tels que $LB_J(X_i, \mathcal{H}_{hybrid}) \leq J_{max}$. À la fin de l'énumération, $\mathcal{H}_{hybrid}$ contiendra m motifs supplémentaires :

$$\mathcal{H}^{t}_{hybrid} = \{s_1, \ldots, s_{t-1}, X^t_1, \ldots, X^t_m\}$$

— Au moment du tirage du nouveau motif s_t de l'itération t, on sélectionne alors un motif parmi les m motifs trouvés à l'itération t. On obtiens ainsi un nouvel historique global $\mathcal{H}_{hybrid}$ tel que :

$$\mathcal{H}_{hybrid}\{s_1, \ldots, s_{t-1}, s_t\}$$

avec $s_t \in \{X^t_1, \ldots, X^t_m\}$.

7.2.4 De nouvelles formes retours utilisateur et de représentation des motifs

De nouvelles formes de feedback de l'utilisateur Cette perspective pourrait permettre d'apporter une réponse au défi méthodologique **Q2**. En effet, pour l'instant, les retours de l'utilisateur servent principalement à indiquer les motifs qu'ils préfèrent (ou pas). Cependant, on ne donne pas la possibilité à l'utilisateur d'expliquer pourquoi il effectue certain retour. Ainsi, au lieu de rejeter un motif entier, un utilisateur pourrait indiquer quelques items qu'il ne veut pas voir. L'apprentissage pourrait alors être plus précis. Cette nouvelle forme de feedback nécessitera la définition de primitives de retour pour indiquer comment l'interaction s'effectuera. Il faudra également déterminer le traitement effectué sur le feedback de l'utilisateur. Par exemple, lorsque l'utilisateur indique trois items qu'il ne veut pas, il faudra déterminer si on exclut le 3-itemset ou les items pris individuellement.

L'exploitation des motifs discriminants comme descripteurs permanents. Dans le chapitre 6, nous avons exploité les descripteurs discriminants comme descripteurs temporaires des motifs extraits. Les évaluations effectuées ont alors montré la pertinence de ces nouveaux descripteurs pour représenter les motifs.
Afin d'exploiter pleinement les descripteurs discriminants, une des pistes seraient de les utiliser comme descripteurs permanents. Cette démarche a pour objectif de maintenir chaque descripteur discriminant dans l'ensemble des descripteurs tout au long du processus d'apprentissage. Il serait alors possible de décrire les motifs avec uniquement les descripteurs discriminants. Pour cela, on ajoutera à $\mathcal{F}$ un nouveau descripteur discriminant à chaque itération. Après T itérations, le vecteurs des descripteurs se présentera comme suit :

$$\mathcal{F} = \langle F^1_{disc}, F^2_{disc}, \ldots, F^l_{disc} \rangle$$

où F^t_{disc} représente le descripteur discriminant ajouté à $\mathcal{F}$ à l'itération t.

Cinquième partie

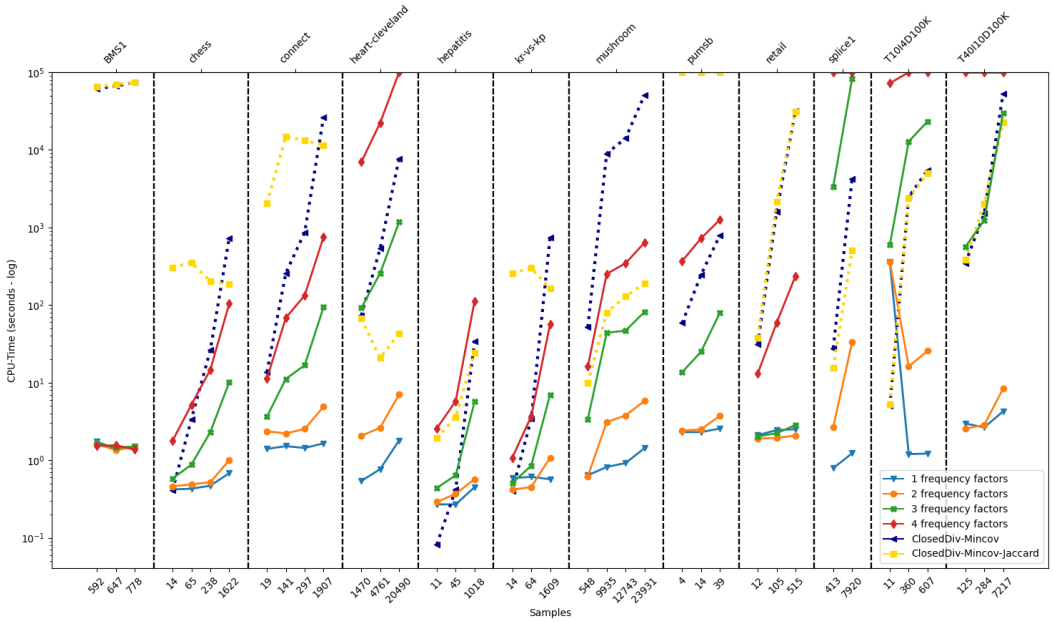
Annexes

ANNEXE A

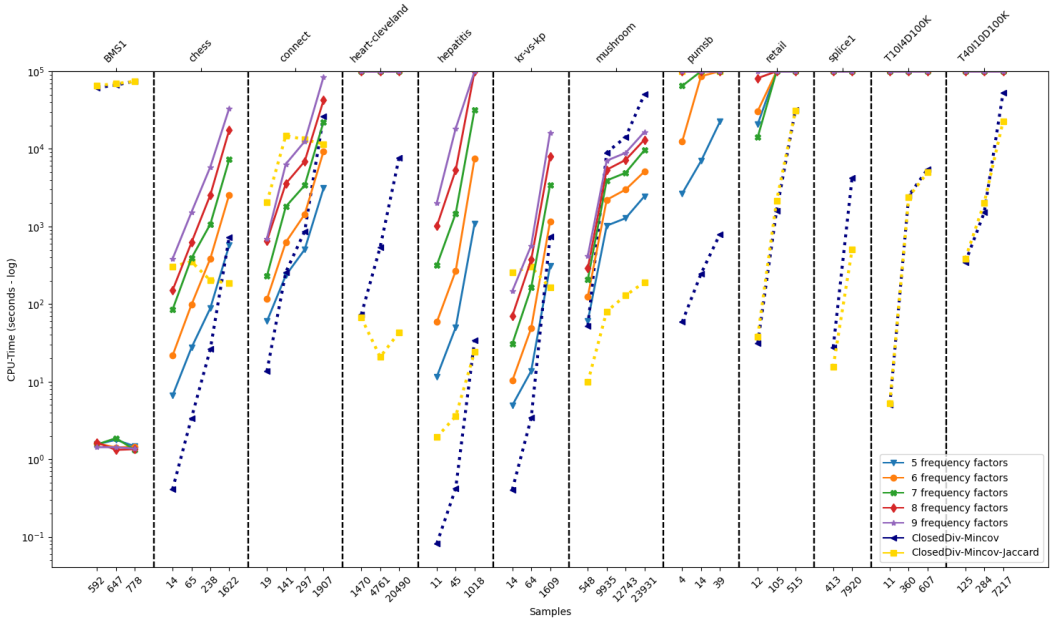
Réduction de la redondance

Dans cette annexe, nous présentons quelques évaluations des méthodes de réduction de la redondance. Ces évaluations se présentent sous forme de graphiques dont une partie a déjà été présentée au chapitre 5.

A.1 Temps d'exécution de ClosedDiv-Mincov et de CFTP



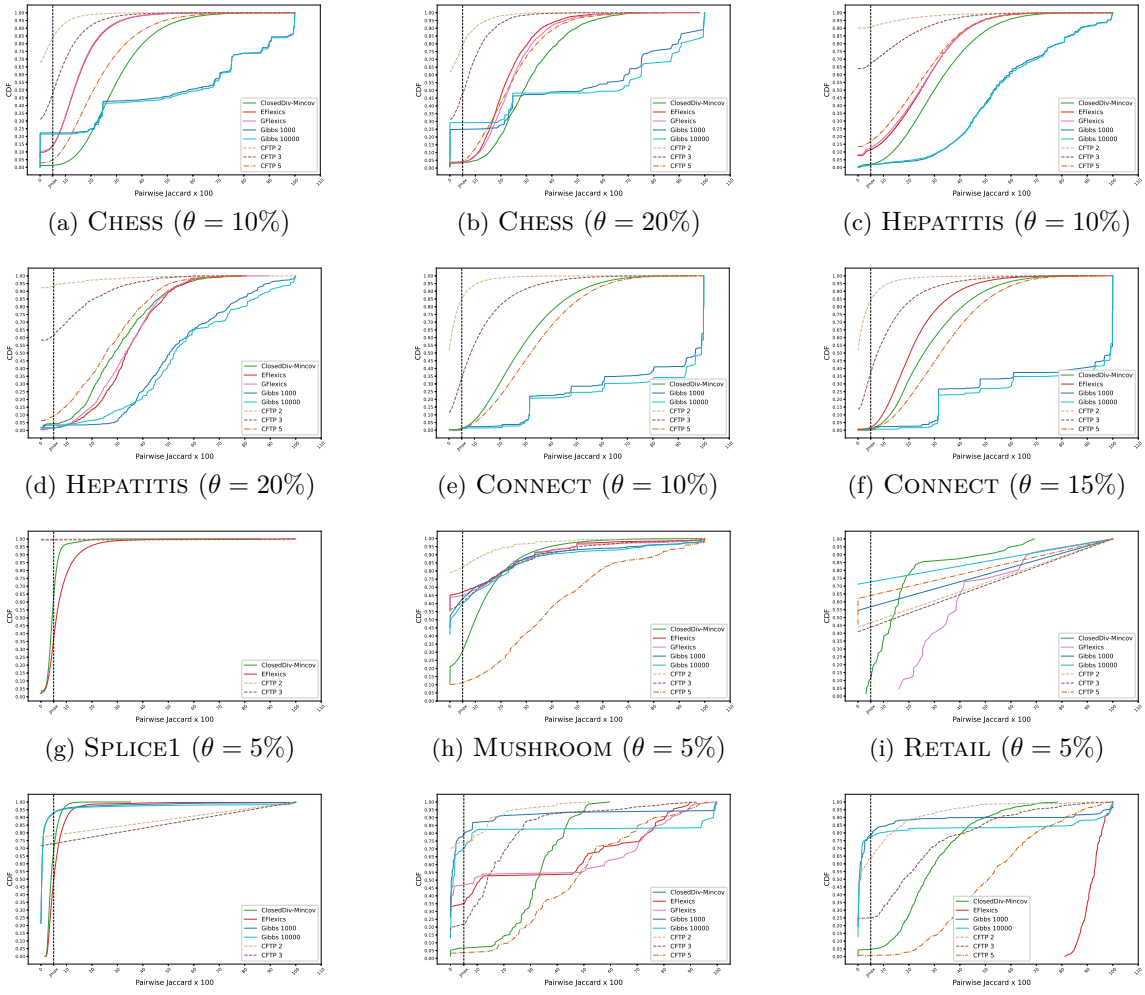
(a) frequency factor $c \in \{1, 2, 3, 4\}$



(b) frequency factor $c \in \{5, 6, 7, 8, 9\}$

FIGURE A.1 – Comparaison des temps d'exécution de CLOSEDDIV ($J_{max} = 0.5$) et CFTP.

A.2 Comparaison de la diversité des paires de motifs de ClosedDiv, Flexics, CFTP et Gibbs



(a) Chess ($\theta = 10\%$) (b) Chess ($\theta = 20\%$) (c) Hepatitis ($\theta = 10\%$)

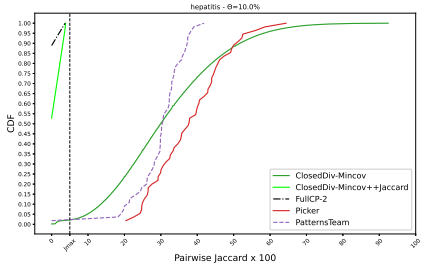
(d) Hepatitis ($\theta = 20\%$) (e) Connect ($\theta = 10\%$) (f) Connect ($\theta = 15\%$)

(g) Splice1 ($\theta = 5\%$) (h) Mushroom ($\theta = 5\%$) (i) Retail ($\theta = 5\%$)

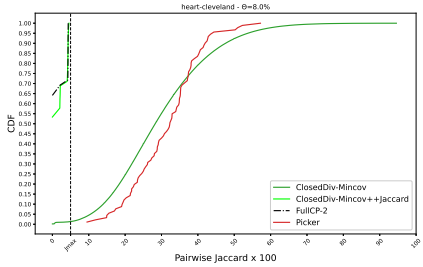
(j) T40 ($\theta = 5\%$) (k) Pumsb ($\theta = 30\%$) (l) Pumsb ($\theta = 20\%$)

Figure A.2 – Comparaison de la diversité des paires de motifs de ClosedDiv-Mincov et des méthodes d'échantillonnages.

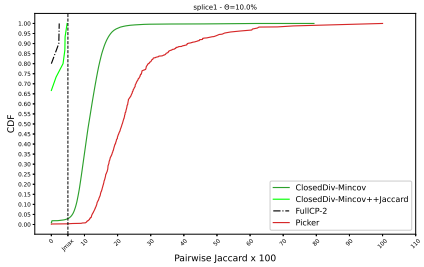
A.3 Comparaison de la diversité des paires de motifs de ClosedDiv, Picker, PatternsTeam et FullCP-2



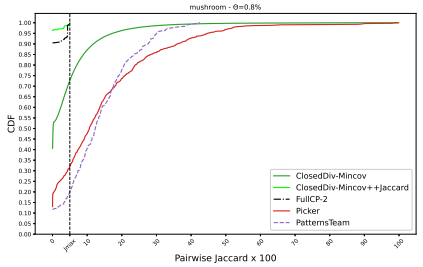
(a) Hepatitis ($\theta = 10\%$)



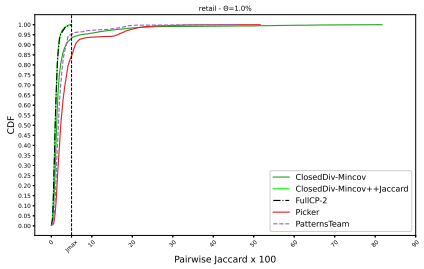
(b) Heart-cleaveland ($\theta = 8\%$)



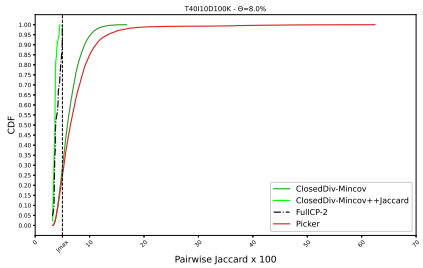
(c) Splice1 ($\theta = 10\%$)



(d) Mushroom ($\theta = 8\%$)



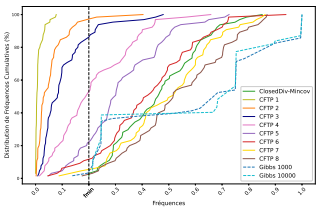
(e) Retail ($\theta = 1\%$)



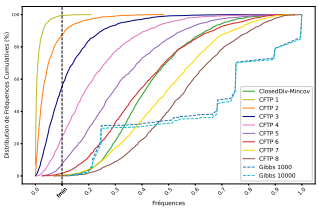
(f) T40 ($\theta = 8\%$)

Figure A.3 – Comparaison de la diversité des paires de motifs de ClosedDiv, Picker, PatternsTeam et FullCP-2.

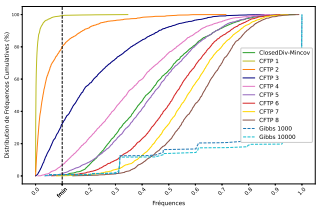
A.4 Analyse de la répartition des fréquences des motifs



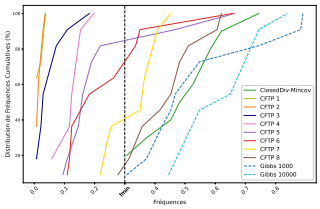
(a) Chess ($\theta = 20\%$)



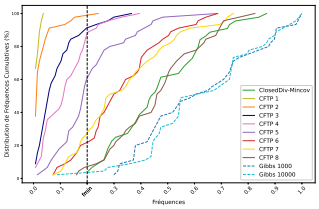
(b) Chess ($\theta = 10\%$)



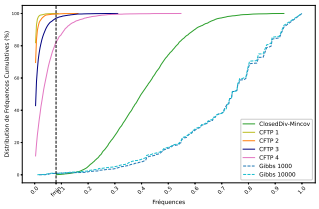
(c) Connect ($\theta = 10\%$)



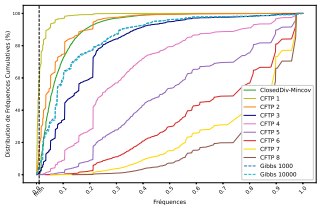
(d) Hepatitis ($\theta = 30\%$)



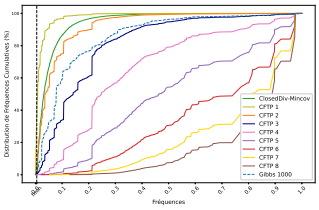
(e) Hepatitis ($\theta = 20\%$)



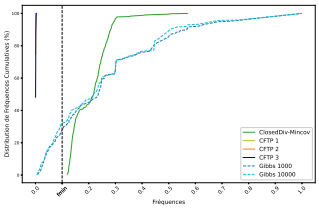
(f) heart-clevelan ($\theta = 8\%$)



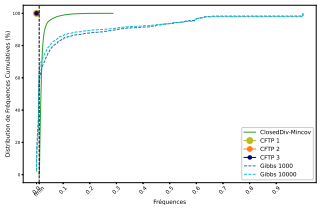
(g) Mushroom ($\theta = 1\%$)



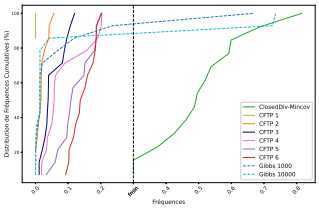
(h) Mushroom ($\theta = 0.5\%$)



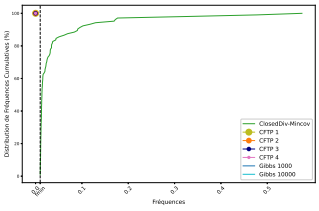
(i) Splice1 ($\theta = 1\%$)



(j) T40 ($\theta = 1\%$)



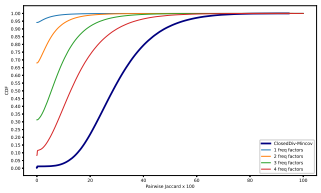
(k) Pumsb ($\theta = 30\%$)



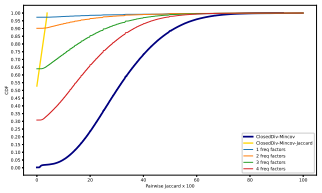
(l) Retail ($\theta = 10\%$)

Figure A.4 – Distribution cumulative des fréquences de ClosedDiv et des méthodes d'échantillonnage

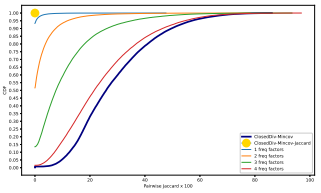
A.5 Analyse de la redondance dans les ensembles de motifs de ClosedDiv et CFTP



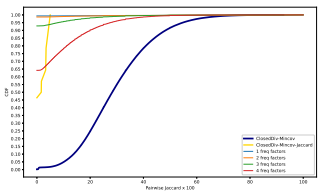
(a) Chess ($\theta = 10\%$)



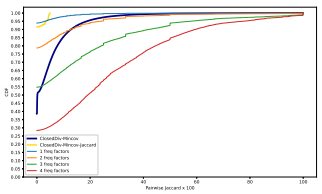
(b) Hepatitis ($\theta = 10\%$)



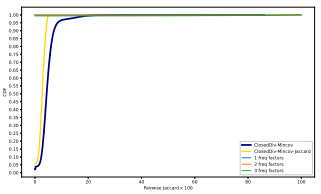
(c) Connect ($\theta = 15\%$)



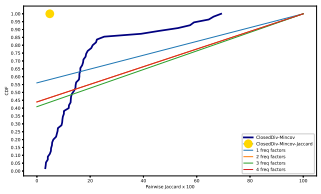
(d) Heart-clevland ($\theta = 10\%$)



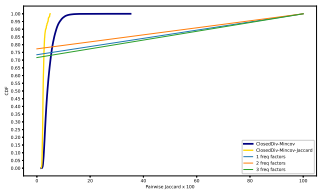
(e) Mushroom ($\theta = 1\%$)



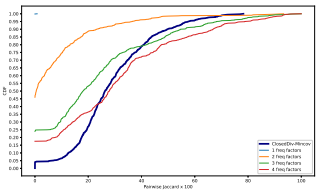
(f) Splice1 ($\theta = 5\%$)



(g) Retail ($\theta = 5\%$)

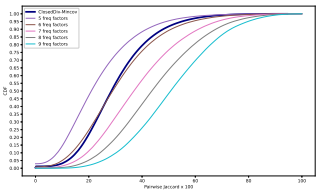


(h) T40I10D100K ($\theta = 5\%$)

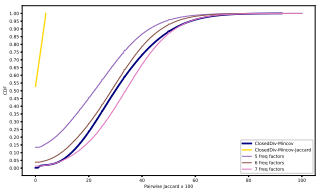


(i) Pumsb ($\theta = 20\%$)

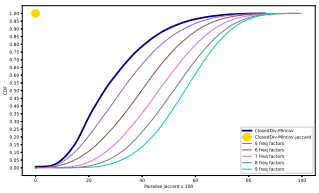
Figure A.5 – Analyse de la redondance dans les ensembles de motifs de ClosedDiv et CFTP. Nous utilisons ici des facteurs de fréquence $c \in \{1, 2, 3, 4\}$.



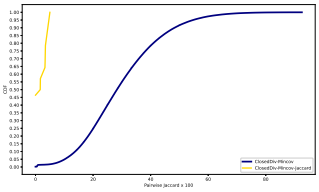
(a) Chess ($\theta = 10\%$)



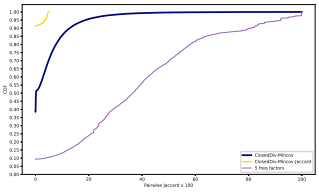
(b) Hepatitis ($\theta = 10\%$)



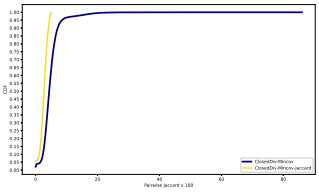
(c) Connect ($\theta = 15\%$)



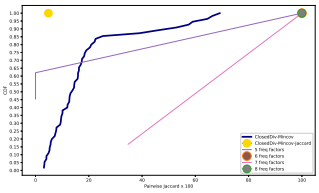
(d) Heart-clevland ($\theta = 10\%$)



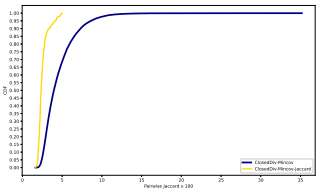
(e) Mushroom ($\theta = 1\%$)



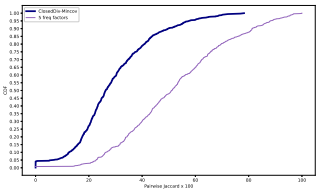
(f) Splice1 ($\theta = 5\%$)



(g) Retail ($\theta = 5\%$)



(h) T40I10D100K ($\theta = 5\%$)



(i) Pumsb ($\theta = 20\%$)

Figure A.6 – Analyse de la redondance dans les ensembles de motifs de ClosedDiv et CFTP. Nous utilisons ici des facteurs de fréquence $c \in \{5, 6, 7, 8, 9\}$.

ANNEXE B

Apprentissage

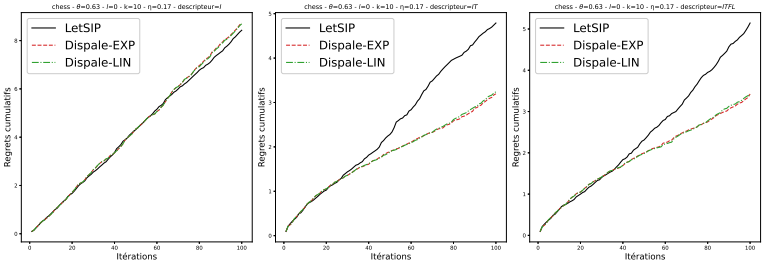
Dans cette annexe, nous présentons quelques évaluations des méthodes d'apprentissage présentées au chapitre 6. Les évaluations se présentent sous forme de graphiques.

B.1 Résultats complémentaires sur le regret

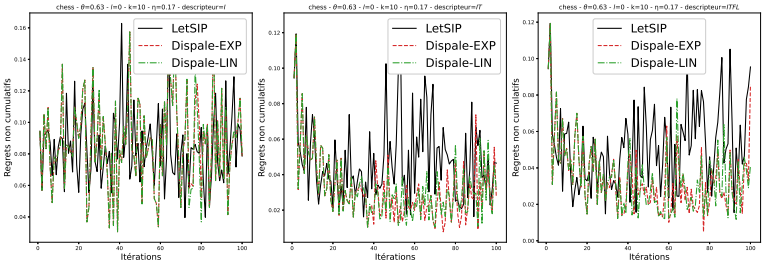
Les figures B.1 à B.4 montre une comparaison détaillée de DiSPaLe avec LetSIP pour différents jeux de données. Il s'agit des regrets cumulatifs $Regret_{\max}$ pour $k = 10$ et $\ell = 0$.

B.2 Résultats complémentaires sur les temps d'execution

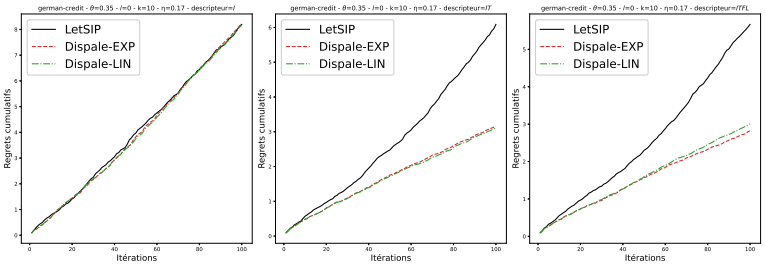
La figure B.5 compare les temps d'exécution de chaque itération de LetSIP et DiSPaLe-Exp. Ces graphiques concernent les exécutions avec comme descripteurs tous les descripteurs (ILFT), une taille de requête $k = 10$ et $\ell = 1$.



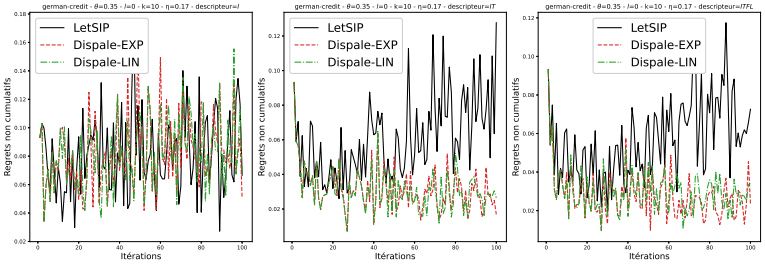
(a) Chess : regrets cumulatifs.



(b) Chess : regret non cumulatif.

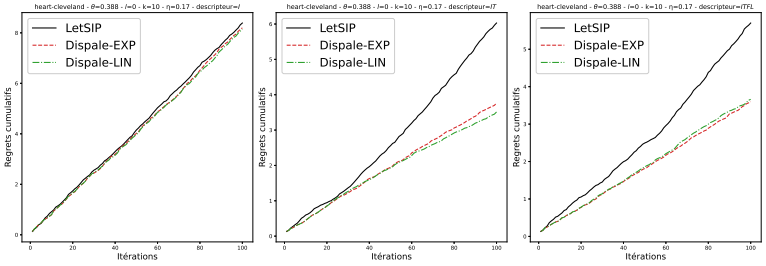


(c) German-credit : regrets cumulatifs.

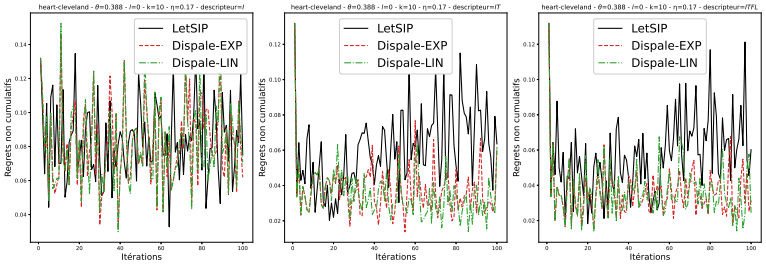


(d) German-credit : regret non cumulatif.

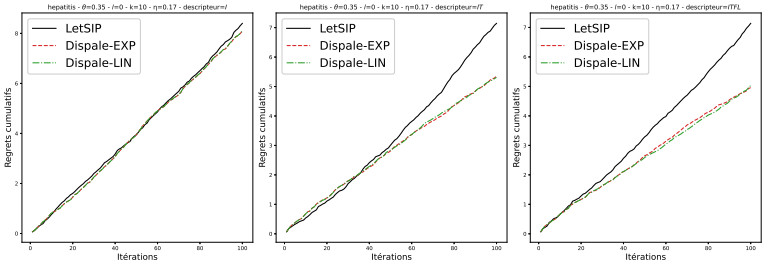
Figure B.1 – Regrets cumulatifs et non cumulatifs $Regret_{\max}$ pour $k = 10$ et $\ell = 0$ (Chess et German-credit).



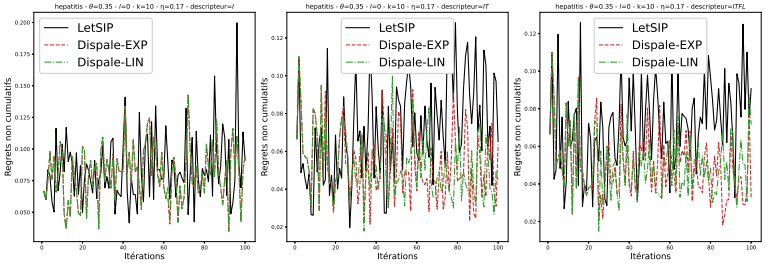
(a) HEART-CLEVELAND : regrets cumulatifs.



(b) HEART-CLEVELAND : regrets non cumulatifs.

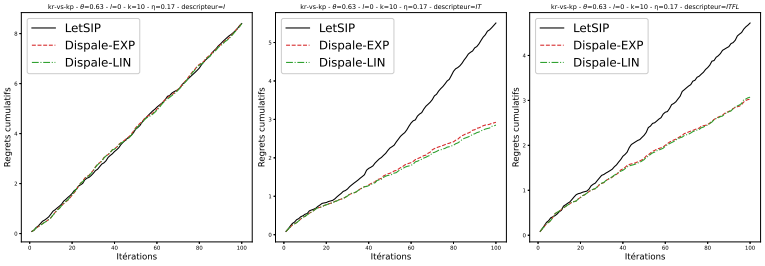


(c) HEPATITIS : regrets cumulatifs.

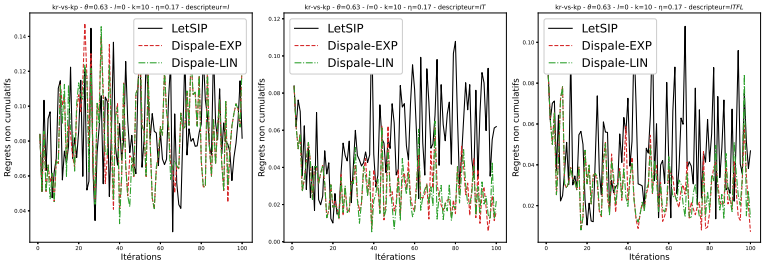


(d) HEPATITIS : regret non cumulatif.

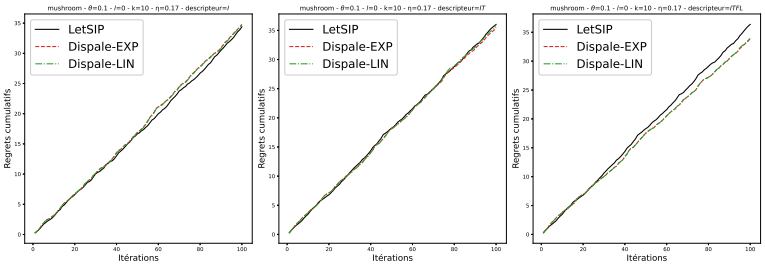
FIGURE B.2 – Regrets cumulatifs et non cumulatifs $Regret_{\max}$ pour $k = 10$ et $\ell = 0$ (HEART-CLEVELAND et HEPATITIS).



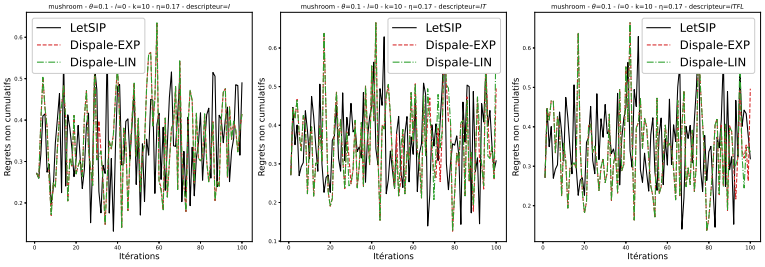
(a) Kr-vs-kp : regrets cumulatifs.



(b) Kr-vs-kp : regret non cumulatif.

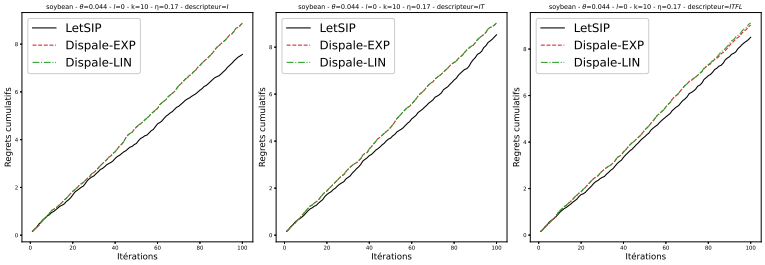


(c) Mushroom : regrets cumulatifs.

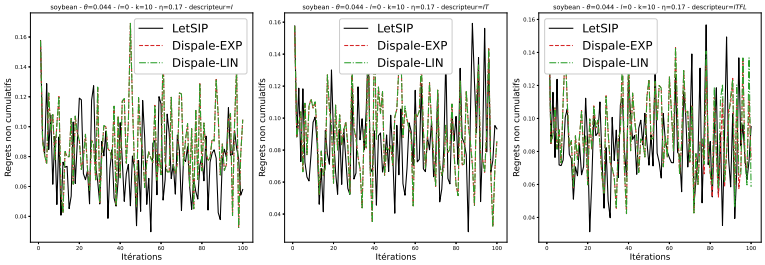


(d) Mushroom : regret non cumulatif.

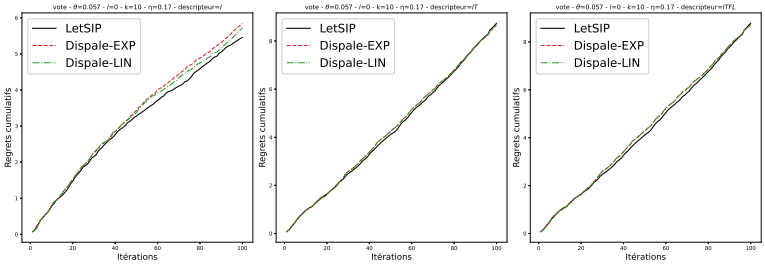
FIGURE B.3 – Regrets cumulatifs et non cumulatifs $Regret_{\max}$ pour $k = 10$ et $\ell = 0$ (KR-VS-KP et MUSHROOM).



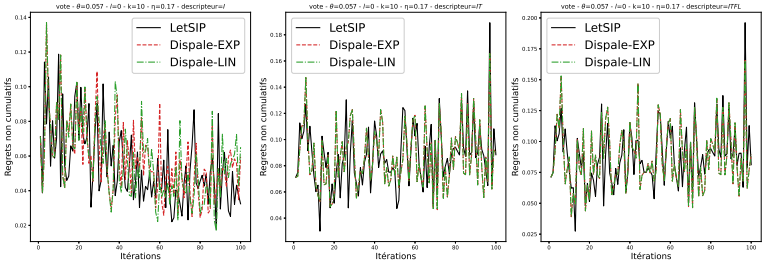
(a) Soybean : regrets cumulatifs.



(b) Soybean : regrets non cumulatifs.

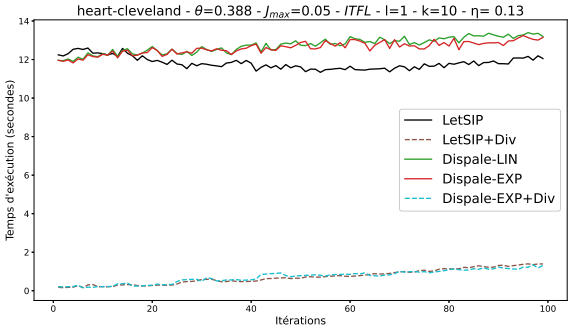


(c) Vote : regrets cumulatifs.

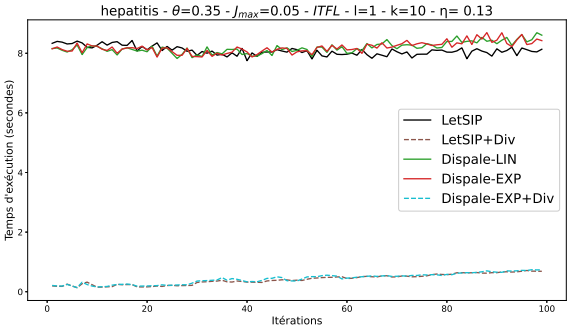


(d) Vote : regrets non cumulatifs.

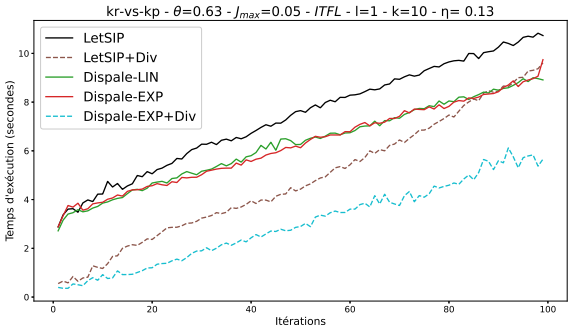
FIGURE B.4 – Regrets cumulatifs et non cumulatifs $Regret_{\max}$ pour $k = 10$ et $\ell = 0$ (SOYBEAN et VOTE).



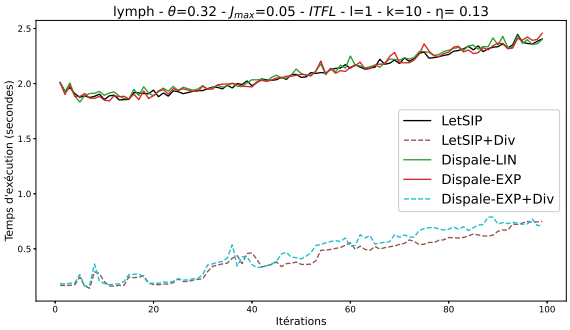
(a) Heart-cleveland



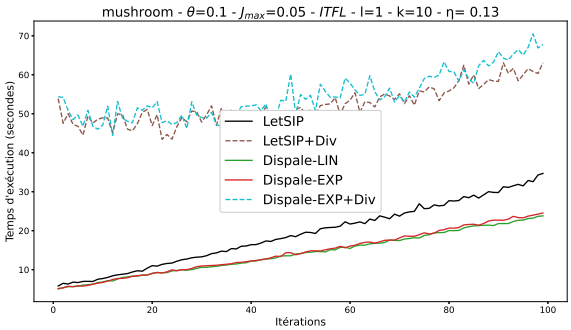
(b) Hepatitis



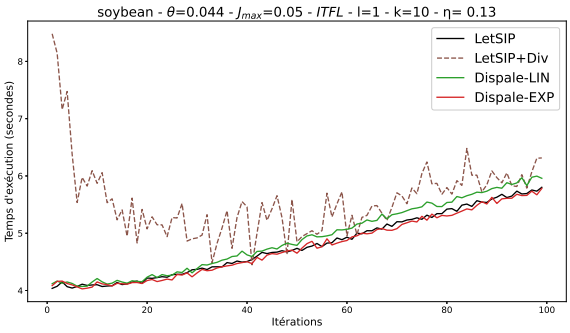
(c) Kr-vs-kp



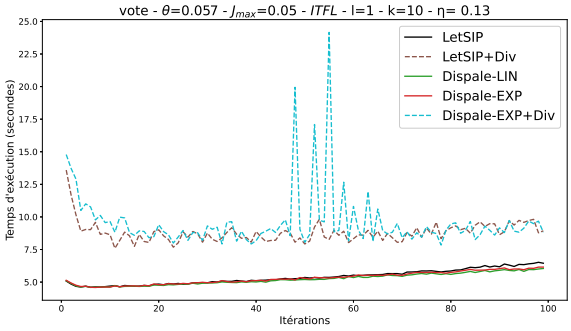
(d) Lymph



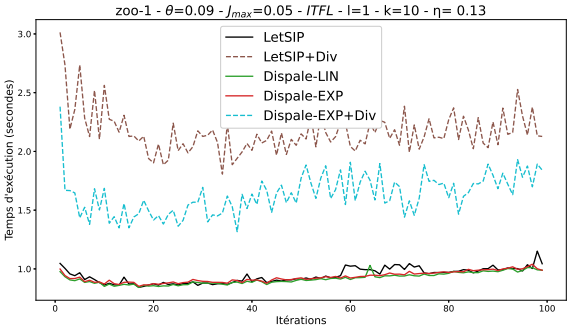
(e) Mushroom



(f) Soybean



(g) Vote



(h) Zoo-1

FIGURE B.5 – Comparaison des durées de chaque itération de LETSIP, LETSIP+DIV, DISPALE et DISPALE +DIV.

Bibliographie

[1] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, May 26-28, 1993*, pages 207–216. ACM Press, 1993.

[2] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI/MIT Press, 1996.

[3] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499. Morgan Kaufmann, 1994.

[4] Nir Ailon. An active learning algorithm for ranking from pairwise preferences with an almost optimal query complexity. *J. Mach. Learn. Res.*, 13 :137–164, 2012.

[5] Krzysztof R. Apt. *Principles of constraint programming.* Cambridge University Press, 2003.

[6] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method : a meta-algorithm and applications. *Theory Comput.*, 8(1) :121–164, 2012.

[7] Stephen D. Bay and Michael J. Pazzani. Detecting change in categorical data : Mining contrast sets. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD 99, page 302–306, New York, NY, USA, 1999. Association for Computing Machinery.

[8] Roberto J. Bayardo. The hows, whys, and whens of constraints in itemset and rule discovery. In *European Workshop on Inductive Databases and Constraint Based Mining, Hinterzarten, Germany, March 11-13, 2004, Revised Selected Papers*, pages 1–13, 2004.

[9] Catriel Beeri, Ronald Fagin, David Maier, and Mihalis Yannakakis. On the desirability of acyclic database schemes. *J. ACM*, 30(3) :479–513, 1983.

[10] M-B. Belaid, C. Bessiere, and N. Lazaar. Constraint programming for association rules. In Tanya Y. Berger-Wolf and Nitesh V. Chawla, editors, *Proceedings of the 2019 SIAM International Conference on Data Mining, SDM 2019, Calgary, Alberta, Canada, May 2-4, 2019*, pages 127–135. SIAM, 2019.

[11] M-B. Belaid, C. Bessiere, and N. Lazaar. Constraint programming for mining borders of frequent itemsets. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 1064–1070. ijcai.org, 2019.

[12] Nicolas Beldiceanu, Mats Carlsson, Sophie Demassey, and Thierry Petit. Global constraint catalogue : Past, present and future. *Constraints An Int. J.*, 12(1) :21–62, 2007.

[13] Nassim Belmecheri, Noureddine Aribi, Nadjib Lazaar, Yahia Lebbah, and Samir Loudni. Boosting the learning for ranking patterns. *CoRR*, abs/2203.02696, 2022.

[14] A. Bendimerad, J. Lijffijt, M. Plantevit, C. Robardet, and T. De Bie. Gibbs sampling subjectively interesting tiles. In *Advances in Intelligent Data Analysis XVIII - 18th International Symposium on Intelligent Data Analysis, IDA 2020, Konstanz, Germany, April 27-29, 2020, Proceedings*, volume 12080 of *Lecture Notes in Computer Science*, pages 80–92. Springer, 2020.

[15] Christian Bessière and Marie-Odile Cordier. Arc-consistency and arc-consistency again. In Richard Fikes and Wendy G. Lehnert, editors, *Proceedings of the 11th National Conference on Artificial Intelligence. Washington, DC, USA, July 11-15, 1993*, pages 108–113. AAAI Press / The MIT Press, 1993.

[16] Christian Bessière and Jean-Charles Régin. MAC and combined heuristics : Two reasons to forsake FC (and cbj ?) on hard problems. In Eugene C. Freuder, editor, *Proceedings of the Second International Conference on Principles and Practice of Constraint Programming, Cambridge, Massachusetts, USA, August 19-22, 1996*, volume 1118 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 1996.

[17] Christian Bessière and Jean-Charles Régin. Refining the basic constraint propagation algorithm. In Bernhard Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pages 309–315. Morgan Kaufmann, 2001.

[18] Mansurul Bhuiyan and Mohammad Al Hasan. Interactive knowledge discovery from hidden data through sampling of frequent patterns. *Stat. Anal. Data Min.*, 9(4) :205–229, 2016.

[19] Mansurul Bhuiyan and Mohammad Al Hasan. PRIIME : A generic framework for interactive personalized interesting pattern discovery. *CoRR*, abs/1607.05749, 2016.

[20] Mansurul Bhuiyan, Snehasis Mukhopadhyay, and Mohammad Al Hasan. Interactive pattern mining on hidden data : a sampling-based solution. In Xue-wen Chen, Guy Lebanon, Haixun Wang, and Mohammed J. Zaki, editors, *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, pages 95–104. ACM, 2012.

[21] Tijl De Bie. Maximum entropy models and subjective interestingness : an application to tiles in binary databases. *Data Min. Knowl. Discov.*, 23(3) :407–446, 2011.

[22] M. Boley, S. Moens, and T. Gärtner. Linear space direct pattern sampling using coupling from the past. In *Proceedings of KDD 12*, pages 69–77. ACM, 2012.

[23] Mario Boley, Thomas Gärtner, and Henrik Grosskreutz. Formal concept sampling for counting and threshold-free local pattern mining. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2010, April 29 - May 1, 2010, Columbus, Ohio, USA*, pages 177–188. SIAM, 2010.

[24] Mario Boley, Claudio Lucchese, Daniel Paurat, and Thomas Gärtner. Direct local pattern sampling by efficient two-step random procedures. In Chid Apté, Joydeep Ghosh, and Padhraic Smyth, editors, *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 582–590. ACM, 2011.

[25] Francesco Bonchi and Claudio Lucchese. Extending the state-of-the-art of constraint-based pattern discovery. *Data and Knowledge Engineering*, 60(2) :377–399, feb 2007.

[26] Jean-François Boulicaut and Artur Bykowski. Frequent closures as a concise representation for binary data mining. In Takao Terano, Huan Liu, and Arbee L. P. Chen, editors,

Knowledge Discovery and Data Mining, Current Issues and New Applications, 4th Pacific-Asia Conference, PADKK 2000, Kyoto, Japan, April 18-20, 2000, Proceedings, volume 1805 of *Lecture Notes in Computer Science*, pages 62–73. Springer, 2000.

[27] Jean-François Boulicaut, Artur Bykowski, and Christophe Rigotti. Free-sets : A condensed representation of boolean data for the approximation of frequency queries. *Data Min. Knowl. Discov.*, 7(1) :5–22, 2003.

[28] Jean-François Boulicaut, Artur Bykowski, editor="Zighed Djamel A. Rigotti, Christophe", Jan Komorowski, and Jan Żytkow. Approximation of frequency queries by means of free-sets. In *Principles of Data Mining and Knowledge Discovery*, pages 75–85, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[29] Björn Bringmann and Albrecht Zimmermann. One in a million : picking the right patterns. *Knowledge and Information Systems*, 18(1) :61–81, 2009.

[30] Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. Learning to rank using gradient descent. In Luc De Raedt and Stefan Wrobel, editors, *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*, volume 119 of *ACM International Conference Proceeding Series*, pages 89–96. ACM, 2005.

[31] Toon Calders and Bart Goethals. Minimal k-free representations of frequent sets. In Nada Lavrac, Dragan Gamberger, Hendrik Blockeel, and Ljupco Todorovski, editors, *Knowledge Discovery in Databases : PKDD 2003, 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Cavtat-Dubrovnik, Croatia, September 22-26, 2003, Proceedings*, volume 2838 of *Lecture Notes in Computer Science*, pages 71–82. Springer, 2003.

[32] Toon Calders and Bart Goethals. Non-derivable itemset mining. *Data Min. Knowl. Discov.*, 14(1) :171–206, 2007.

[33] Hadrien Cambazard. *Résolution de problèmes combinatoires par des approches fondées sur la notion d'explication. (Explanation-based algorithms in Constraint Programming)*. PhD thesis, University of Nantes, France, 2006.

[34] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank : from pairwise approach to listwise approach. In Zoubin Ghahramani, editor, *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, volume 227 of *ACM International Conference Proceeding Series*, pages 129–136. ACM, 2007.

[35] Supratik Chakraborty, Daniel J. Fremont, Kuldeep S. Meel, Sanjit A. Seshia, and Moshe Y. Vardi. Distribution-aware sampling and weighted model counting for sat, 2014.

[36] Raymond Chan, Qiang Yang, and Yi-Dong Shen. Mining high utility itemsets. In *Third IEEE International Conference on Data Mining*, pages 19–26, 2003.

[37] Weiwei Cheng, Eyke Hüllermeier, Willem Waegeman, and Volkmar Welker. Label ranking with partial abstention based on thresholded probabilistic models. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25 : 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 2510–2518, 2012.

[38] David A. Cohen and Peter G. Jeavons. The power of propagation : when GAC is enough. *Constraints An Int. J.*, 22(1) :3–23, 2017.

[39] L. De Raedt, T. Guns, and S. Nijssen. Constraint programming for itemset mining. In *14th ACM SIGKDD*, pages 204–212, 2008.

[40] L. De Raedt and A. Zimmermann. Constraint-based pattern set mining. In *7th SIAM SDM*, pages 237–248. SIAM, 2007.

[41] Romuald Debruyne and Christian Bessière. Domain filtering consistencies. *J. Artif. Intell. Res.*, 14 :205–230, 2001.

[42] Rina Dechter. Enhancement schemes for constraint processing : Backjumping, learning, and cutset decomposition. *Artif. Intell.*, 41(3) :273–312, 1990.

[43] Rina Dechter. *Constraint processing.* Elsevier Morgan Kaufmann, 2003.

[44] Rina Dechter and Itay Meiri. Experimental evaluation of preprocessing techniques in constraint satisfaction problems. In N. S. Sridharan, editor, *Proceedings of the 11th International Joint Conference on Artificial Intelligence. Detroit, MI, USA, August 1989*, pages 271–277. Morgan Kaufmann, 1989.

[45] Lamine Diop. *Echantillonnage sous contraintes de motifs Structures. (Constrained Sampling of Structured Patterns).* PhD thesis, Gaston Berger University, Saint-Louis, Senegal, 2020.

[46] Vladimir Dzyuba. *Mine, Interact, Learn, Repeat : Interactive Pattern-based Data Exploration ; Zoek, Interacteer, Leer, Herhaal : interactieve data-exploratie met patronen.* PhD thesis, Katholieke Universiteit Leuven, Belgium, 2017.

[47] Vladimir Dzyuba and Matthijs van Leeuwen. Interactive discovery of interesting subgroup sets. In Allan Tucker, Frank Höppner, Arno Siebes, and Stephen Swift, editors, *Advances in Intelligent Data Analysis XII - 12th International Symposium, IDA 2013, London, UK, October 17-19, 2013. Proceedings*, volume 8207 of *Lecture Notes in Computer Science*, pages 150–161. Springer, 2013.

[48] Vladimir Dzyuba and Matthijs van Leeuwen. Learning what matters - sampling interesting patterns. In Jinho Kim, Kyuseok Shim, Longbing Cao, Jae-Gil Lee, Xuemin Lin, and Yang-Sae Moon, editors, *Advances in Knowledge Discovery and Data Mining - 21st Pacific-Asia Conference, PAKDD 2017, Jeju, South Korea, May 23-26, 2017, Proceedings, Part I*, volume 10234 of *Lecture Notes in Computer Science*, pages 534–546, 2017.

[49] Vladimir Dzyuba, Matthijs van Leeuwen, Siegfried Nijssen, and Luc De Raedt. Active preference learning for ranking patterns. In *25th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2013, Herndon, VA, USA, November 4-6, 2013*, pages 532–539. IEEE Computer Society, 2013.

[50] Vladimir Dzyuba, Matthijs van Leeuwen, Siegfried Nijssen, and Luc De Raedt. Interactive learning of pattern rankings. *Int. J. Artif. Intell. Tools*, 23(6), 2014.

[51] Vladimir Dzyuba, Matthijs van Leeuwen, and Luc De Raedt. Flexible constrained sampling with guarantees for pattern mining. *Data Mining and Knowledge Discovery*, 31(5) :1266–1293, 2017.

[52] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. Knowledge discovery and data mining : Towards a unifying framework. In Evangelos Simoudis, Jiawei Han, and Usama M. Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, pages 82–88. AAAI Press, 1996.

[53] Yoav Freund, Raj D. Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. In Jude W. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24-27, 1998*, pages 170–178. Morgan Kaufmann, 1998.

[54] Daniel Frost and Rina Dechter. Look-ahead value ordering for constraint satisfaction problems. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes*, pages 572–578. Morgan Kaufmann, 1995.

[55] Ada Wai-Chee Fu, Renfrew W.-w. Kwong, and Jian Tang. Mining N-most interesting itemsets. In Zbigniew W. Ras and Setsuo Ohsuga, editors, *Foundations of Intelligent*

Systems, 12th International Symposium, ISMIS 2000, Charlotte, NC, USA, October 11-14, 2000, Proceedings, volume 1932 of *Lecture Notes in Computer Science*, pages 59–67. Springer, 2000.

[56] Johannes Fürnkranz and Eyke Hüllermeier. Preference learning : An introduction. In Johannes Fürnkranz and Eyke Hüllermeier, editors, *Preference Learning*, pages 1–17. Springer, 2010.

[57] Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. Multilabel classification via calibrated label ranking. *Mach. Learn.*, 73(2) :133–153, 2008.

[58] John Gary Gaschnig. *Performance Measurement and Analysis of Certain Search Algorithms.* PhD thesis, USA, 1979. AAI7925014.

[59] Floris Geerts, Bart Goethals, and Taneli Mielikäinen. Tiling databases. In Einoshin Suzuki and Setsuo Arikawa, editors, *Discovery Science*, pages 278–289, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[60] Liqiang Geng and Howard J. Hamilton. Interestingness measures for data mining : A survey. *ACM Comput. Surv.*, 38(3) :9–es, sep 2006.

[61] Matthew L. Ginsberg. Dynamic backtracking. *J. Artif. Intell. Res.*, 1 :25–46, 1993.

[62] Solomon W. Golomb and Leonard D. Baumert. Backtrack programming. *J. ACM*, 12 :516–524, 1965.

[63] David Graus, Manos Tsagkias, Wouter Weerkamp, Edgar Meij, and Maarten de Rijke. Dynamic collective entity representations for entity ranking. In Paul N. Bennett, Vanja Josifovski, Jennifer Neville, and Filip Radlinski, editors, *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA, February 22-25, 2016*, pages 595–604. ACM, 2016.

[64] Tias Guns. Declarative pattern mining using constraint programming. phd, KU Leuven, Leuven, Belgium, 2012.

[65] Tias Guns, Siegfried Nijssen, and Luc De Raedt. Itemset mining : A constraint programming perspective. *Artif. Intell.*, 175(12-13) :1951–1983, 2011.

[66] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. pages 1–12, 2000.

[67] Jiawei Han, Jianyong Wang, Ying Lu, and Petre Tzvetkov. Mining top-k frequent closed patterns without minimum support. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan*, pages 211–218. IEEE Computer Society, 2002.

[68] Sariel Har-Peled, Dan Roth, and Dav Zimak. Constraint classification : A new approach to multiclass classification. In Nicolò Cesa-Bianchi, Masayuki Numao, and Rüdiger Reischuk, editors, *Algorithmic Learning Theory, 13th International Conference, ALT 2002, Lübeck, Germany, November 24-26, 2002, Proceedings*, volume 2533 of *Lecture Notes in Computer Science*, pages 365–379. Springer, 2002.

[69] Robert M. Haralick and Gordon L. Elliott. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14(3) :263–313, 1980.

[70] Warwick Harvey and Peter J. Stuckey. Improving linear constraint propagation by changing constraint representation. *Constraints An Int. J.*, 8(2) :173–207, 2003.

[71] Mohammad Al Hasan and Mohammed J. Zaki. Output space sampling for graph patterns. *Proc. VLDB Endow.*, 2(1) :730–741, 2009.

[72] Emmanuel Hebrard, Brahim Hnich, Barry O'Sullivan, and Toby Walsh. Finding diverse and similar solutions in constraint programming. In *AAAI*, volume 5, pages 372–377, 2005.

[73] Hannes Heikinheimo, Jouni K. Seppänen, Eino Hinkkanen, Heikki Mannila, and Taneli Mielikäinen. Finding low-entropy sets and trees from binary data. In Pavel Berkhin, Rich

Caruana, and Xindong Wu, editors, *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007*, pages 350–359. ACM, 2007.

[74] Mannila Heikki and Toivonen Hannu. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1 :241–258, 1997.

[75] Pascal Van Hentenryck and Jean-Philippe Carillon. Generality versus specificity : An experience with AI and OR techniques. In Howard E. Shrobe, Tom M. Mitchell, and Reid G. Smith, editors, *Proceedings of the 7th National Conference on Artificial Intelligence, St. Paul, MN, USA, August 21-26, 1988*, pages 660–664. AAAI Press / The MIT Press, 1988.

[76] R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 1, pages 97–102 vol.1, 1999.

[77] A. Hien, S. Loudni, N. Aribi, Y. Lebbah, M. Laghzaoui, A. Ouali, and A. Zimmermann. Approche basée sur la relaxation pour la fouille de motifs fermés et diversifiés. In *Journées Francophones de Programmation par Contraintes (JFPC 2021)*, Nice, France, 2021.

[78] A. Hien, S. Loudni, N. Aribi, Y. Lebbah, M. Laghzaoui, A. Ouali, and A. Zimmermann. Fouille de motifs fermés et diversifiés basée sur la relaxation. In *Conférence Internationale Francophone sur la Sciences des Données (CIFSD2021)*, Marseille, France, 2021.

[79] A. Hien, S. Loudni, N. Aribi, Y. Lebbah, M. El Amine Laghzaoui, A. Ouali, and A. Zimmermann. A relaxation-based approach for mining diverse closed patterns. In Frank Hutter, Kristian Kersting, Jefrey Lijffijt, and Isabel Valera, editors, *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2020, Ghent, Belgium, September 14-18, 2020, Proceedings, Part I*, volume 12457 of *Lecture Notes in Computer Science*, pages 36–54. Springer, 2020.

[80] Robert Hilderman and Howard Hamilton. *Knowledge Discovery and Measures of Interest*, volume 638. Springer, 01 2001.

[81] Willem-Jan van Hoeve and Irit Katriel. Global constraints. In *Handbook of Constraint Programming*, pages 169–208. Elsevier Science Inc., 2006.

[82] Eyke Hüllermeier and Johannes Fürnkranz. On predictive accuracy and risk minimization in pairwise label ranking. *J. Comput. Syst. Sci.*, 76(1) :49–62, 2010.

[83] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pages 133–142. ACM, 2002.

[84] Sung Young Jung, Jeong-Hee Hong, and Taek-Soo Kim. A formal model for user preference. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 235–242, 2002.

[85] Narendra Jussien, Romuald Debruyne, and Patrice Boizumault. Maintaining arc-consistency within dynamic backtracking. In Rina Dechter, editor, *Principles and Practice of Constraint Programming - CP 2000, 6th International Conference, Singapore, September 18-21, 2000, Proceedings*, volume 1894 of *Lecture Notes in Computer Science*, pages 249–261. Springer, 2000.

[86] Toshihiro Kamishima, Shotaro Akaho, and Jun Sakuma. Fairness-aware learning through regularization approach. In Myra Spiliopoulou, Haixun Wang, Diane J. Cook, Jian Pei, Wei Wang, Osmar R. Zaïane, and Xindong Wu, editors, *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on, Vancouver, BC, Canada, December 11, 2011*, pages 643–650. IEEE Computer Society, 2011.

[87] Toshihiro Kamishima, Hideto Kazawa, and Shotaro Akaho. *A Survey and Empirical Comparison of Object Ranking Methods*, pages 181–201. 09 2010.

[88] Matthew A. Kayala, Chloé-Agathe Azencott, Jonathan H. Chen, and Pierre Baldi. Learning to predict chemical reactions. *J. Chem. Inf. Model.*, 51(9) :2209–2222, 2011.

[89] Amina Kemmar, Yahia Lebbah, Samir Loudni, Patrice Boizumault, and Thierry Charnois. Prefix-projection global constraint and top-k approach for sequential pattern mining. *Constraints An Int. J.*, 22(2) :265–306, 2017.

[90] M. Khiari, P. Boizumault, and B. Crémilleux. Constraint programming for mining n-ary patterns. In David Cohen, editor, *Principles and Practice of Constraint Programming - CP 2010 - 16th International Conference, CP 2010, St. Andrews, Scotland, UK, September 6-10, 2010. Proceedings*, volume 6308 of *Lecture Notes in Computer Science*, pages 552–567. Springer, 2010.

[91] Medhi Khiari. *Découverte de motifs n-aires utilisant la programmation par contraintes*. Theses, Université de Caen, June 2012.

[92] D. Kifer, J. Gehrke, C. Bucila, and W. White. How to quickly find a witness. In *Constraint-Based Mining and Inductive Databases*, pages 216–242. Springer Berlin Heidelberg, 2006.

[93] Arno J. Knobbe and Eric K. Y. Ho. Pattern teams. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Knowledge Discovery in Databases : PKDD 2006, 10th European Conference on Principles and Practice of Knowledge Discovery in Databases, Berlin, Germany, September 18-22, 2006, Proceedings*, volume 4213 of *Lecture Notes in Computer Science*, pages 577–584. Springer, 2006.

[94] Arno J Knobbe and Eric KY Ho. Maximally informative k-itemsets and their efficient discovery. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 237–244, 2006.

[95] Edwin M. Knorr, Raymond T. Ng, and Vladimir Tucakov. Distance-based outliers : Algorithms and applications. *The VLDB Journal*, 8(3–4) :237–253, feb 2000.

[96] David Krantz, Duncan Luce, Patrick Suppes, and Amos Tversky. *Foundations of Measurement, Vol. I : Additive and Polynomial Representations*. New York Academic Press, 1971.

[97] N. Lazaar, Y. Lebbah, S. Loudni, M. Maamar, V. Lemière, C. Bessiere, and P. Boizumault. A global constraint for closed frequent pattern mining. In Michel Rueher, editor, *Principles and Practice of Constraint Programming - 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings*, volume 9892 of *Lecture Notes in Computer Science*, pages 333–349. Springer, 2016.

[98] Ching-Pei Lee and Chih-Jen Lin. Large-scale linear ranksvm. *Neural Comput.*, 26(4) :781–817, 2014.

[99] Olivier Lhomme. Consistency techniques for numeric csps. In Ruzena Bajcsy, editor, *Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993*, pages 232–238. Morgan Kaufmann, 1993.

[100] Bing Liu, Wynne Hsu, Lai-Fun Mun, and Hing-Yan Lee. Finding interesting patterns using user expectations. *IEEE Transactions on Knowledge and Data Engineering*, 11(6) :817–832, 1999.

[101] Alan K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1) :99–118, 1977.

[102] Tatiana Makhalova, Sergei O. Kuznetsov, and Amedeo Napoli. Mint : Mdl-based approach for mining interesting numerical pattern sets. *Data Min. Knowl. Discov.*, 36(1) :108–145, 2022.

[103] Roger Mohr and Thomas C. Henderson. Arc and path consistency revisited. *Artif. Intell.*, 28(2) :225–233, 1986.

[104] Shinichi Morishita and Jun Sese. Traversing itemset lattices with statistical metric pruning. pages 226–236.

[105] Andrew Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 78, New York, NY, USA, 2004. Association for Computing Machinery.

[106] Raymond T. Ng, Laks V. S. Lakshmanan, Jiawei Han, and Alex Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD '98, page 13–24, New York, NY, USA, 1998. Association for Computing Machinery.

[107] Miho Ohsaki, Shinya Kitaguchi, Kazuya Okamoto, Hideto Yokoi, and Takahira Yamaguchi. Evaluation of rule interestingness measures with a clinical dataset on hepatitis. In *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, PKDD 04, page 362–373, Berlin, Heidelberg, 2004. Springer-Verlag.

[108] Abdelkader Ouali. *Méthodes hybrides parallèles pour la résolution de problèmes d'optimisation combinatoire : application au clustering sous contraintes. (Parallel hybrid methods for solving combinatorial optimization problems : application to clustering under constraints)*. PhD thesis, Normandy University, France, 2017.

[109] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proceedings of the 7th ICDT*, pages 398–416, 1999.

[110] Gilles Pesant. A regular language membership constraint for finite sequences of variables. In Mark Wallace, editor, *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings*, volume 3258 of *Lecture Notes in Computer Science*, pages 482–495. Springer, 2004.

[111] Patrick Prosser. Hybrid algorithms for the constraint satisfaction problem. *Comput. Intell.*, 9 :268–299, 1993.

[112] C. Prud'homme, J.-G. Fages, and X. Lorca. Choco Solver Documentation, 2016.

[113] Philippe Refalo. Impact-based search strategies for constraint programming. In Mark Wallace, editor, *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings*, volume 3258 of *Lecture Notes in Computer Science*, pages 557–571. Springer, 2004.

[114] Jean-Charles Régin. A filtering algorithm for constraints of difference in csps. In Barbara Hayes-Roth and Richard E. Korf, editors, *Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, July 31 - August 4, 1994, Volume 1*, pages 362–367. AAAI Press / The MIT Press, 1994.

[115] J. Rissanen. Paper : Modeling by shortest data description. *Automatica*, 14(5) :465–471, sep 1978.

[116] Francesca Rossi, Peter van Beek, and Toby Walsh. *Constraint Programming*. Elsevier Science Inc., USA, 2006.

[117] Stefan Rüping. Ranking interesting subgroups. In Andrea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman, editors, *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, volume 382 of *ACM International Conference Proceeding Series*, pages 913–920. ACM, 2009.

[118] Daniel Sabin and Eugene C. Freuder. Contradicting conventional wisdom in constraint satisfaction. In Alan Borning, editor, *Principles and Practice of Constraint Programming, Second International Workshop, PPCP'94, Rosario, Orcas Island, Washington, USA, May 2-4, 1994, Proceedings*, volume 874 of *Lecture Notes in Computer Science*, pages 10–20. Springer, 1994.

[119] Thomas Schiex and Gérard Verfaillie. Nogood recording for static and dynamic constraint satisfaction problems. *Int. J. Artif. Intell. Tools*, 3(2) :187–208, 1994.

[120] Bernhard Schölkopf and Alexander Johannes Smola. *Learning with Kernels : support vector machines, regularization, optimization, and beyond*. Adaptive computation and machine learning series. MIT Press, 2002.

[121] Aliaksei Severyn and Alessandro Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In Ricardo Baeza-Yates, Mounia Lalmas, Alistair Moffat, and Berthier A. Ribeiro-Neto, editors, *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, pages 373–382. ACM, 2015.

[122] Shai Shalev-Shwartz and Ambuj Tewari. Stochastic methods for l_1-regularized loss minimization. *J. Mach. Learn. Res.*, 12 :1865–1892, 2011.

[123] Xuehua Shen and ChengXiang Zhai. Active feedback in ad hoc information retrieval. In Ricardo A. Baeza-Yates, Nivio Ziviani, Gary Marchionini, Alistair Moffat, and John Tait, editors, *SIGIR 2005 : Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15-19, 2005*, pages 59–66. ACM, 2005.

[124] Arno Siebes and René Kersten. A structure function for transaction data. In *Proceedings of the Eleventh SIAM International Conference on Data Mining, SDM 2011, April 28-30, 2011, Mesa, Arizona, USA*, pages 558–569. SIAM / Omnipress, 2011.

[125] Arno Siebes, Jilles Vreeken, and Matthijs van Leeuwen. Item sets that compress. In Joydeep Ghosh, Diane Lambert, David B. Skillicorn, and Jaideep Srivastava, editors, *Proceedings of the Sixth SIAM International Conference on Data Mining, April 20-22, 2006, Bethesda, MD, USA*, pages 395–406. SIAM, 2006.

[126] Abraham Silberschatz and Alexander Tuzhilin. On subjective measures of interestingness in knowledge discovery. In Usama M. Fayyad and Ramasamy Uthurusamy, editors, *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95), Montreal, Canada, August 20-21, 1995*, pages 275–281. AAAI Press, 1995.

[127] Koen Smets and Jilles Vreeken. Slim : Directly mining descriptive patterns. In *Proceedings of the Twelfth SIAM International Conference on Data Mining, Anaheim, California, USA, April 26-28, 2012*, pages 236–247. SIAM / Omnipress, 2012.

[128] Barbara M. Smith and Stuart A. Grant. Trying harder to fail first. In Henri Prade, editor, *13th European Conference on Artificial Intelligence, Brighton, UK, August 23-28 1998, Proceedings.*, pages 249–253. John Wiley and Sons, 1998.

[129] Arnaud Soulet and Bruno Crémilleux. Adequate condensed representations of patterns. In Walter Daelemans, Bart Goethals, and Katharina Morik, editors, *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML/PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part I*, volume 5211 of *Lecture Notes in Computer Science*, pages 20–21. Springer, 2008.

[130] Arnaud Soulet, Bruno Crémilleux, and François Rioult. Condensed representation of eps and patterns quantified by frequency-based measures. In Bart Goethals and Arno Siebes, editors, *KDID 2004, Knowledge Discovery in Inductive Databases, Proceedings of the Third International Workshop on Knowledge Discovery inInductive Databases, Pisa, Italy, September 20, 2004, Revised Selected and Invited Papers*, volume 3377 of *Lecture Notes in Computer Science*, pages 173–190. Springer, 2004.

[131] Arnaud Soulet, Bruno Crémilleux, and François Rioult. Représentation condensée de motifs émergents. In Georges Hébrail, Ludovic Lebart, and Jean-Marc Petit, editors, *Extraction et gestion des connaissances (EGC'2004), Actes des quatrièmes journées Extraction et Gestion des Connaissances, Clermont Ferrand, France, 20-23 janvier 2004, 2 Volumes*, volume RNTI-E-2 of *Revue des Nouvelles Technologies de l'Information*, pages 265–276. Cépaduès-Éditions, 2004.

[132] Pang-Ning Tan, Vipin Kumar, and Jaideep Srivastava. Selecting the right interestingness measure for association patterns. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD 02, page 32–41, New York, NY, USA, 2002. Association for Computing Machinery.

[133] Nikolaj Tatti and Jilles Vreeken. The long and the short of it : summarising event sequences with serial episodes. In Qiang Yang, Deepak Agarwal, and Jian Pei, editors, *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, pages 462–470. ACM, 2012.

[134] Willy Ugarte, Patrice Boizumault, Samir Loudni, and Bruno Crémilleux. Modeling and mining optimal patterns using dynamic CSP. In *27th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2015, Vietri sul Mare, Italy, November 9-11, 2015*, pages 33–40. IEEE Computer Society, 2015.

[135] T. Uno, Tatsuya Asai, Y. Uchida, and Hiroki Arimura. An efficient algorithm for enumerating closed patterns in transaction databases. *Lecture Notes in Computer Science*, 3245 :16–31, 01 2004.

[136] Shankar Vembu and Thomas Gärtner. Label ranking algorithms : A survey. In Johannes Fürnkranz and Eyke Hüllermeier, editors, *Preference Learning*, pages 45–64. Springer, 2010.

[137] Charles Vernerey, Samir Loudni, Noureddine Aribi, and Yahia Lebbah. Threshold-free pattern mining meets multi-objective optimization : Application to association rules. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 1880–1886. ijcai.org, 2022.

[138] Jilles Vreeken, Matthijs Van Leeuwen, and Arno Siebes. Krimp : mining itemsets that compress. *Data Mining and Knowledge Discovery*, 23(1) :169–214, 2011.

[139] David L. Waltz. Generating semantic descriptions from drawings of scenes with shadows. Technical report, USA, 1972.

[140] J. Wang, J. Han, and Jian Pei. CLOSET+ : searching for the best strategies for mining frequent closed itemsets. In *Proceedings of the Ninth KDD*, pages 236–245. ACM, 2003.

[141] Tino Werner. A review on instance ranking problems in statistical learning. *Mach. Learn.*, 111(2) :415–463, 2022.

[142] Dong Xin, Xuehua Shen, Qiaozhu Mei, and Jiawei Han. Discovering interesting patterns through user's interactive feedback. In Tina Eliassi-Rad, Lyle H. Ungar, Mark Craven, and Dimitrios Gunopulos, editors, *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, pages 773–778. ACM, 2006.

[143] Zuobing Xu, Ram Akella, and Yi Zhang. Incorporating diversity and density in active learning for relevance feedback. In *Proceedings of the 29th European Conference on IR Research*, ECIR'07, page 246–257, Berlin, Heidelberg, 2007. Springer-Verlag.

[144] Hong Yao, Howard J. Hamilton, and Cory J. Butz. A foundational approach to mining itemset utilities from databases. In *SDM*, 2004.

[145] Yiyu Yao, Yan Zhao, Jue Wang, and Suqing Han. A model of machine learning based on user preference of attributes. In Salvatore Greco, Yutaka Hata, Shoji Hirano, Masahiro Inuiguchi, Sadaaki Miyamoto, Hung Son Nguyen, and Roman Slowinski, editors, *Rough Sets and Current Trends in Computing, 5th International Conference, RSCTC 2006, Kobe, Japan, November 6-8, 2006, Proceedings*, volume 4259 of *Lecture Notes in Computer Science*, pages 587–596. Springer, 2006.

[146] M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In *Proceedings of KDD 1997, Newport Beach, California, USA, August 14-17, 1997*, pages 283–286. AAAI Press, 1997.

[147] Ning Zhong, Y.Y.Y. Yao, and M. Ohishima. Peculiarity oriented multidatabase mining. *IEEE Transactions on Knowledge and Data Engineering*, 15(4) :952–960, 2003.

[148] Albrecht Zimmermann. *Mining Sets of Patterns (Zoeken naar verzamelingen van patronen)*. PhD thesis, Katholieke Universiteit Leuven, Belgium, 2009.

Titre : Cadre Interactif de Fouille de motifs avec Prise en compte des Préférences Utilisateurs

Résumé

La recherche de motifs intéressants a fait évoluer la fouille de motifs vers un modèle centré sur l'utilisateur. Dans ce cadre, la fouille interactive de motifs permet de prendre en compte les préférences de l'utilisateur afin de guider la recherche vers les motifs pertinents. Elle consiste à alterner entre phases d'extraction de motifs et phases d'apprentissage sur les motifs intéressants en introduisant un mécanisme de « feedback » permettant à l'utilisateur d'exprimer ses préférences. Les préférences exprimées sont alors exploitées pour apprendre et mettre à jour un modèle de préférences qui sera utilisé pour extraire de nouveaux plus intéressants.

Cette démarche soulève de nombreux défis méthodologiques à relever parmi lesquelles la nécessité de produire rapidement des résultats diversifiés et le choix du modèle de représentation et d'exploitation des préférences exprimées. Les contributions de cette thèse concerne ainsi les point suivants :

1. la proposition d'un cadre générique de fouille de motifs diversifiés exploitant la programmation par contrainte ;
2. l'exploitation de ce cadre pour l'échantillonnage de motifs diversifiés ;
3. la proposition d'une nouvelle classe de descripteurs exploitant les motifs discriminants ;
4. la proposition d'une nouvelle méthode d'apprentissage exploitant les motifs discriminants pour apprendre les préférences de l'utilisateurs.

Mots-clés : fouille interactive de motifs, programmation par contraintes, contraintes globales, relaxations, échantillonnage, motifs discriminants, descripteurs, apprentissage.

Abstract

The search for interesting patterns has evolved pattern mining into a user-centric model. For this purpose, interactive pattern mining allow one to exploit user preferences to ease the mining of interesting patterns. It consists of alternating between phases of pattern mining and learning phases on interesting patterns by introducing a feedback mechanism allowing the user to express his preferences. The preferences expressed are then exploited to learn and update a model of preferences which will be used to extract new ones that are more interesting.

This approach raises many methodological challenges, including the need of an efficient miner of diverse patterns and the choice of an accurate preference model. This thesis thus aims to propose some contibutions to the following points :

1. a generic model for mining diverse patterns using constraint programming ;
2. a method for sampling diverse patterns exploiting the generic model ;
3. a new class of features using discriminant patterns ;
4. a new learning method using discriminant patterns for user preferences learning.

Keywords : Interactive pattern mining, constraint programming, global constraints, relaxations, sampling, discriminant patterns, features, learning.