Analyse de séquences avec GALACTIC – Approche générique combinant analyse formelle des concepts et fouille de motifs

Salah Eddine Boukhetta

LA ROCHELLE UNIVERSITÉ

ÉCOLE DOCTORALE EUCLIDE

LABORATOIRE L3i

THÈSE présentée par :

Salah Eddine BOUKHETTA

soutenue le : **30/08/2022**

pour obtenir le grade de : **Docteur de La Rochelle Université**

Discipline : **Informatique et Applications**

Analyse de séquences avec GALACTIC – Approche générique combinant analyse formelle des concepts et fouille de motifs

---

RAPPORTEURS	Thomas GUYET	Professeur des universités	Université de Lyon
	Florence LE BER	Professeur des universités	ENGEES Strasbourg
EXAMINATEURS	Sébastien FERRÉ	Professeur des universités	IRISA, Université Rennes I
	Marianne HUCHARD	Professeur des universités	Université Montpellier
	Gaël LEJEUNE	Maître de conférences HDR	Sorbonne Université
	Nathalie GIRARD	Maître de conférences	IRISA, Université Rennes I
INVITÉS	Mehdi KAYTOUE	Maître de conférences HDR	Entreprise Infologic (R&D), Lyon
	Cyril FAUCHER	Maître de conférences	La Rochelle Université
DIRECTION	Karell BERTET	Maître de conférences HDR	La Rochelle Université
	Christophe DEMKO	Maître de conférences	La Rochelle Université

Thèse réalisée au Laboratoire L3i
Faculté des Sciences et Technologies
Avenue Michel Crépeau
17042 La Rochelle cedex 01

Tel : +33 5 46 45 82 62
Web : `http://l3i.univ-larochelle.fr`

Sous la direction de Karell Bertet — Maître de conférences HDR
Christophe Demko — Maître de conférences

Financement Allocation de recherche de la région Nouvelle-Aquitaine

Résumé

Les données de type séquences sont à nos jours utilisées dans beaucoup de domaines dans le but de mieux les analyser et d'en extraire des connaissances. Une séquence est une suite d'éléments ordonnés comme par exemple les trajectoires de déplacement ou les séquences d'achats de produits dans un supermarché. Il existe plusieurs types des séquences, les séquences simples, les séquences temporelles et les séquences d'intervalles. La fouille de séquences est un domaine de la fouille de données qui vise à extraire des motifs séquentiels fréquents à partir d'un ensemble de séquences, où ces motifs sont le plus souvent des sous-séquences. Un motif séquentiel peut être une sous-séquence, et il est fréquent s'il est partagé par un nombre suffisamment représentatif de données. Plusieurs algorithmes ont été proposés pour l'extraction des motifs séquentiels fréquents. Avec l'évolution des capacités de calcul, la tâche d'extraction des motifs séquentiels fréquents est devenue plus rapide. La difficulté réside alors dans le trop grand nombre de motifs séquentiels extraits, qui en rend difficile la lisibilité et donc l'interprétation. On parle de *déluge de motifs*. Une premiere approche pour résoudre ce problème consiste à réduire les motifs fréquents générés aux seuls motifs fermés qui portent la même information. Il a été observé que l'ensemble de tous les motifs fermés peut être organisé dans une structure appelée *treillis*. Cette structure est la base de l'Analyse Formelle de Concepts (AFC) qui est un domaine d'analyse de données permettant d'identifier des relations dans l'ensemble de données. L'AFC est classiquement conçue pour traiter des données décrites par des ensembles d'attributs, donc des données binaires. Le formalisme des structures de motifs et la navigation conceptuelle abstraite étendent l'AFC pour traiter des données complexes comme les séquences. Inspiré des structures de motifs, l'algorithme NextPriorityConcept propose une approche d'extraction de motifs pour des données hétérogènes et complexes. Cet algorithme permet un calcul de motifs génériques à travers des *descriptions* spécifiques d'objets par des *prédicats monadiques*. Il propose également de raffiner un ensemble d'objets en un ensemble plus petit à travers des *stratégies* d'explorations spécifiques choisies l'utilisateur, ce qui permet de réduire le nombre de motifs et ainsi limiter le deluge de motifs générés. La plateforme `GALACTIC` implémente l'algorithme NextPriorityConcept et propose un écosystème d'extensions pour le traitement de données. Dans ce travail, nous nous intéressons à l'analyse de données séquentielles en utilisant GALACTIC. Nous proposons plusieurs descriptions et stratégies adaptées aux séquences simples, temporelles et d'intervalles. Les descriptions sont basées sur les sous-séquences communes maximales avec des descriptions plus spécifiques aux types des séquences. Nous proposons une stratégie naïve permettant la génération de tous les concepts et des stratégies plus élaborées permettant de réduire le nombre de concepts et donc la taille du treillis. Nous proposons des mesures de qualité non supervisées pour pouvoir comparer entre les treillis obtenus. Une analyse qualitative et quantitative est menée sur des jeux de données réels et synthétiques afin de montrer l'efficacité de notre approche.

Mots clés : Analyse Formelle de Concepts, Fouille de séquences, Treillis, Structure de motifs, Séquences temporelles, Séquences d'intervalles, Sous-séquences Communes Maximales.

Sequence analysis using GALACTIC - Generic approach combining Formal Concept Analysis and pattern mining

Abstract

Sequence data are nowadays used in many domains in order to better analyze and extract knowledge. A sequence is a set of ordered elements such as travel trajectories or sequences of product purchases in a supermarket. There are three types of sequences, simple sequences, temporal sequences and interval sequences. Sequence mining is a domain of data mining that aims at extracting frequent sequential patterns from a set of sequences, where these patterns are most often common subsequences. A sequential pattern can be a subsequence, and it is frequent if it is shared by a sufficiently representative number of objetcs in the data. Several algorithms have been proposed for frequent sequential pattern mining. With the evolution of computing capabilities, the task of frequent sequential pattern mining has become faster. The difficulty then lies in the large number of extracted sequential patterns, which makes it difficult to read and therefore to interpret. We speak about "deluge of patterns". A first approach to solve this problem is to reduce the generated frequent patterns to the only closed patterns that carry the same information. It has been observed that the set of all closed patterns can be organized in a structure called a lattice. This structure is the basis of Formal Concept Analysis (FCA) which is a field of data analysis for identifying relationships in the data set. FCA is classically designed to deal with data described by sets of attributes, thus binary data. The pattern structure formalism and the abstract conceptual navigation extend FCA to handle complex data such as sequences. Inspired by pattern structures, the NextPriorityConcept algorithm proposes a pattern mining approach for heterogeneous and complex data. This algorithm allows a generic pattern computation through specific descriptions of objects by monadic predicates. It also proposes to refine a set of objects into a smaller set through specific exploration strategies defined by the user. This allows to reduce the number of patterns and thus to limit the deluge of generated patterns. The GALACTIC platform implements the NextPriorityConcept algorithm and offers an ecosystem of extensions for data processing. In this work, we are interested in the analysis of sequential data using GALACTIC. We propose several descriptions and strategies adapted to simple, temporal and interval sequences. The descriptions are based on the maximum common subsequences with descriptions more specific to the types of sequences. We propose also a naive strategy allowing the generation of all concepts and more elaborate strategies allowing to reduce the number of concepts, i.e. the size of the lattice. We propose unsupervised quality measures to compare between the obtained lattices. A qualitative and quantitative analysis is conducted on real and synthetic datasets to show the efficiency of our approach.

Keywords : Formal Concept Analysis, Sequence Mining, Lattice, Pattern Structure, Temporal Sequences, Interval Sequences, Maximum Common Subsequences.

Remerciements

C'est grâce à l'aide de nombreuses personnes que j'ai pu mener cette thèse à son terme.

Je voudrais tout d'abord remercier grandement mes directeurs de thèse, Karell Bertet et Christophe Demko, pour leurs aides, leurs suivis, leurs disponibilités, et leurs conseils qu'ils m'ont apportés toute au long de ces quatre années de thèse. Leurs conseils m'ont amené à faire les bons choix et de prendre les bonnes décisions.

Je voudrais remercier Florence Le Ber et Thomas Guyet d'avoir accepté de rapporter ma thèse et pour l'intérêt qu'ils ont accordé à mon travail. Leurs remarques m'ont permis d'envisager mon travail sous différentes angles. Je tiens également à remercier les autres membres du jury, Marianne Huchard, Nathalie Girard et Sébastien Ferré pour avoir accepté d'évaluer mon travail, et en particulier Cyril Faucher, Gaël Lejeune et Mehdi Kaytoue pour les discussions enrichissantes que nous avons eu pendant ma thèse.

Mes remerciements vont également aux membres de l'équipe Modèles et Connaissances du L3i, et en particuliers les membre du groupe GALACTIC pour leur soutien durant mes travaux de thèse et l'envéronement de travail agréable qu'ils ont su m'offrir.

Je remercie toutes les personnes avec qui j'ai partagé mes études et notamment ces années de thèse : Nour, Marwa, Muzzamil, Lionel, Jérémy, Thomas, Youcef, Amine, Viviana, ... et en particulier mes amis avec qui j'ai partagé le bureau : Damien, Jordan, Chloé, Marcela, Musab, Antoine, Mohamed, Latifa et Wiam ... Merci à tous pour les discussions, le soutien et les pauses cafés ...

Merci également à tous les enseignants du département informatique de La Rochelle université, avec qui j'ai appris à enseigner durant mon contrat doctoral et durant une année d'ATER.

Je ne remercierai jamais assez mes parents Ahmed et Kheira et mes frères et sœurs pour leur soutien sans faille, leurs encouragements, et sans oublier tous mes chers amis.

Enfin, je remercie infiniment ma femme Nouha pour sa présence et son soutien, son effort et son aide quotidienne, tant moralement que matériellement, qui mon permis de mener à terme mes travaux, et ce, dans les meilleures conditions possibles.

Table des matières

Introduction générale **11**

I État de l'art **19**

1 Définitions et notations **21**
1 Fouille de données 21
1.1 Données 21
1.2 Méthodes 22
2 Données séquentielles 23
3 Conclusion et discussion 26

2 Fouille de séquences **29**
1 Introduction 29
2 Fouille d'*itemsets* fréquents 30
2.1 L'algorithme *Apriori* 31
2.2 Fouille d'*itemsets* fermés fréquents 33
3 Fouille de motifs séquentiels fréquents 34
3.1 Les algorithmes basés sur *Apriori* : Représentation Horizontale . . 36
3.2 Les algorithmes basés sur *Apriori* : Représentation Verticale 39
3.3 Les algorithmes de croissance de motifs 40
3.4 Approches par contraintes 42
4 Fouille de motifs séquentiels fermés 43
5 Fouille de motifs temporels 47
5.1 Fouille de motifs séquentiels avec contraintes temporelles 47
5.2 Fouille d'épisodes fréquents 48
5.3 Fouille de chroniques fréquentes 49
5.4 Fouille de motifs *delta* 51
6 Fouille de motifs séquentiels d'intervalles 51
7 Conclusion et discussion 53

3 Analyse Formelle de Concepts **55**
1 Introduction . . . 55
2 Ensemble partiellement ordonné et théorie des treillis . . . 56
2.1 Ensemble partiellement ordonné . . . 56
2.2 Théorie des treillis . . . 57
3 Analyse Formelle de Concepts . . . 60
3.1 Contexte et concept . . . 60
3.2 Treillis des concepts . . . 61
3.3 Connexion de Galois et treillis de fermés . . . 63
3.4 Algorithmes de génération de treillis des fermés . . . 64
4 Extension de l'AFC à des données non binaires . . . 67
4.1 Extensions dédiées à certains types de données . . . 67
4.2 Extensions dédiées aux données complexes . . . 68
4.3 Fouille de séquences et AFC . . . 69
5 Conclusion et discussion . . . 70

4 L'algorithme NextPriorityConcept **73**
1 Introduction . . . 73
2 Descriptions . . . 74
3 Stratégies . . . 76
4 L'algorithme NextPriorityConcept . . . 77
5 La Plateforme `GALACTIC` . . . 81
6 Cas d'utilisation de l'algorithme NextPriorityConcept . . . 84
7 Conclusion et discussion . . . 86

II Contributions **89**

5 Analyse de séquences avec NextPriorityConcept **91**
1 Introduction . . . 92
2 Séquences simples . . . 93
2.1 Descriptions . . . 94
2.1.1 Description de Sous-séquences Communes Maximales (SCM) 94
2.1.2 Description de Sous-séquences Communes Préfixées (SCP) 95
2.1.3 Description de K-Sous-séquences Communes (KSC) . . . 97
2.2 Stratégies . . . 98
2.2.1 La Stratégie Naïve (SN) . . . 99
2.2.2 La Stratégie Augmentée (SA) . . . 99
2.3 Exemples . . . 99
3 Analyse de séquences temporelles . . . 105
3.1 Descriptions . . . 107
3.1.1 Description de Sous-séquences Distancielle Communes Maximales (SDCM) . . . 108
3.1.2 Description SDCM avec Contrainte de *fenêtre* . . . 108

3.1.3 Description SDCM avec Contrainte d'*écart* 110
3.2 Stratégies 110
3.2.1 Stratégie *Par étape* 111
3.2.2 La stratégie *Naïve* 111
3.2.3 La stratégie *Milieu* 111
3.3 Exemples 112
4 Analyse de séquences d'intervalles 117
4.1 Descriptions 119
4.1.1 Description de Sous-séquences d'intervalles Communes MAXimales (SCMAX). 119
4.1.2 Description de Super-séquences d'intervalles Communes MINimales (SCMIN). 120
4.2 Stratégies 122
4.2.1 Stratégies avec sélecteurs de sous-séquences d'intervalle . 122
4.2.2 Stratégies avec des sélecteurs de super-séquences 123
4.3 Exemples 124
5 Mesures de qualité 129
5.1 Stabilité Logarithmique Normalisée Globale 129
5.2 Représentabilité et Distinctivité 131
6 Les extensions de données séquentielles 132
6.1 Caractéristiques 132
6.2 Descriptions 133
6.3 Stratégies 134
7 Conclusion 136

6 Mesures et Expérimentations 139
1 Introduction 140
2 Jeux de données 141
2.1 Jeux de données synthétiques 141
2.2 La Cité du Vin 141
2.3 GéoLuciole 142
2.4 Daily Actions 144
2.5 Catch me if you can 144
3 Analyse qualitative 145
3.1 Impact de changement de stratégies sur l'analyse 145
3.1.1 Analyse de `GéoLuciole` avec δ_{SCM}-σ_{SN} 145
3.1.2 Analyse de `GéoLuciole` avec δ_{SCM}-σ_{SA} 146
3.2 Impact du changement de la description sur l'analyse 147
3.2.1 Analyse de `Cité-du-vin` avec δ_{SCP}-σ_{SA} 147
3.2.2 Analyse de `GéoLuciole` avec δ_{KSC}-σ_{SN} 149
3.3 Analyse hétérogène de `GéoLuciole` 150
3.3.1 Focus sur une information géographique : La plage 150
3.3.2 Focus sur une information temporelle : La nuit 151

4 Analyse quantitative . . . 152
4.1 Séquences simples . . . 152
4.1.1 Fouille de motifs séquentiels fermés avec δ_{SCM}-$\sigma_{\text{SN,SA}}$ . . . 152
4.1.2 Analyse de `Daily-actions` avec $\delta_{\text{KSC,SCM}}$-σ_{SN} . . . 153
4.1.3 Analyse de `Cité-du-vin` avec $\delta_{\text{SCP,SCM}}$-σ_{SA} . . . 154
4.1.4 Analyse de `Cité-du-vin` avec tous les descriptions et stratégies . . . 155
4.2 Séquences temporelles . . . 155
4.2.1 Analyse de `Catch-me` et `Cité-du-vin` avec δ_{SDCM}-$\sigma_{\text{SDN,SDM}}$ 155
4.2.2 Analyse de `Catch-me` et `Cité-du-vin` avec δ_{SDCM}-σ_{SDP} . . . 158
4.2.3 Analyse de `Cité-du-vin` avec $\delta_{\text{SDCMw,SDCMg}}$-$\sigma_{\text{SDN}}$ . . . 159
4.3 Séquences Intervalles . . . 159
4.3.1 Analyse de `Cité-du-vin` avec δ_{SCMAX}-σ_{CMA} et δ_{SCMIN}-σ_{NCT} 160
4.3.2 Analyse de `Cité-du-vin` avec δ_{SCMIN}- $\sigma_{\text{NCT,BCT,FACT,ACT}}$ . . . 160
4.3.3 Analyse de `GéoLuciole` avec δ_{SCMIN}-$\sigma_{\text{NCT,BCT,FACT,ACT}}$ . . . 161
5 Discussion et conclusion . . . 162

7 Conclusion générale et Perspectives 163
1 Conclusion . . . 163
2 Contributions . . . 163
3 Perspectives . . . 168

Table des figures

2.1 *Itemsets* de la Table 2.1 avec leur support 33
2.2 La construction des *Id-listes* avec l'algorithme *SPADE* pour l'Exemple 1.1 41
2.3 L'arbre de préfixe construit par *PrefixSpan* pour l'Exemple 1.1 42
2.4 L'arbre lexicographique des séquences de la Figure 2.3 44
2.5 L'arbre lexicographique des séquences 45
2.6 Exemple de séquence temporelle et épisodes fréquents 49
2.7 Exemple de séquence temporelle et de chronique frequente. 50
2.8 Les sept relations d'Allen 52

3.1 Exemple de poset 3.1a et son diagramme de Hasse 3.1b 57
3.2 Diagramme de Hasse du treillis $\mathcal{L} = (P(\{x, y, z\}), \subseteq)$ 59
3.3 Exemple de treillis de l'Exemple 3.3 62
3.4 Le treillis réduit du treillis de la Figure 3.3 63
3.5 Treillis des fermés de l'Exemple 3.3 65

4.1 l'architecture de `GALACTIC` 82
4.2 Treillis pour le jeu de données `Lenses` avec la stratégie d'entropie 86
4.3 Relation entre l'algorithme NextPriorityConcept et les deux domaines de fouille de motifs et de l'AFC 87

5.1 Diagramme de Hasse du treillis de concepts généré par la description SCM et la stratégie SN 101
5.2 Diagramme de Hasse du treillis de concepts généré avec la description SCM et la stratégie SA 102
5.3 Diagramme de Hasse du treillis de concepts généré avec la description SCP et la stratégie SA 103
5.4 Diagramme de Hasse du treillis de concepts généré avec la description KSC et la stratégie SN avec $k = 2$ 104
5.5 Diagramme de Hasse du treillis de concepts généré avec la description KSC et la stratégie SN avec $k = 3$ 105
5.6 Exemple de séquences distancielles d'actions quotidiennes 107
5.7 Diagramme de Hasse du treillis de concepts généré par la description SDCM et la stratégie SDN. 113

5.8 Diagramme de Hasse du treillis de concepts généré par la description SDCM et la stratégie SDM. . . . 114
5.9 Diagramme de Hasse du treillis de concepts généré par la description SDCM et la stratégie SDP (pas = 7). . . . 115
5.10 Diagramme de Hasse du treillis de concepts généré par la description SDCMw avec w=7 et la stratégie SDN. . . . 116
5.11 Diagramme de Hasse du treillis de concepts généré par la description SDCMw avec w=6 et la stratégie SDN. . . . 116
5.12 Diagramme de Hasse du treillis de concepts généré par les description SDCMg avec g=7 et la stratégie SDN. . . . 117
5.13 Exemple de séquences d'intervalles 118
5.14 $\delta_{\text{SCMAX}}(\{S_1, S_2, S_5\})$ et S_1, S_2 et S_5. . . . 120
5.15 $\delta_{\text{SCMIN}}(\{S_2, S_3, S_4\})$ et S_2, S_3 et S_4. . . . 121
5.16 Diagramme de Hasse du treillis de concept généré par la description SCMAX et la stratégie CMA. . . . 125
5.17 (a) Diagramme de Hasse du treillis de concepts généré par la description du SCMIN et la stratégie du NCT et (b) zoom sur le concept \$12. . . . 126
5.18 Diagramme de Hasse du treillis de concepts généré par la description du SCMIN et la stratégie du BCT avec cardinalité minimale. . . . 126
5.19 Diagramme de Hasse du treillis de concepts généré par la description du SCMIN et la stratégie du BCT avec cardinalité maximale. . . . 127
5.20 (a) Diagramme de Hasse du treillis de concepts généré par la description du SCMIN et la stratégie du FACT avec $f = 1$ et (b) zoom sur le concept \$1. 128
5.21 Diagramme de Hasse du treillis de concepts généré par la description du SCMIN et la stratégie du ACT. . . . 129
5.22 l'architecture de `GALACTIC` avec les extensions pour les séquences 137

6.1 Plan de la Cité du Vin à Bordeaux, France. . . . 142
6.2 Segmentation de la ville de La Rochelle en quartiers 143
6.3 Une trajectoire enrichie d'un touriste à La Rochelle. . . . 143
6.4 Treillis de concepts généré par δ_{SCM}-σ_{SN} (support 20) pour `GéoLuciole`. . 146
6.5 Treillis de concepts généré par δ_{SCM}-σ_{SA} (support 20) pour `GéoLuciole`. . 147
6.6 Treillis de concepts généré par δ_{SCP}-σ_{SA} (support 5%) pour `Cité-du-vin`. . 148
6.7 Une partie du treillis de concepts généré par δ_{SCP}-σ_{SA} (support 2%) pour `Cité-du-vin`. . . . 149
6.8 Treillis de concepts généré par δ_{KSC}-σ_{SN} (support 26%) pour `GéoLuciole`. . 150
6.9 Nombre de concepts générés par δ_{SCM}-σ_{SN} et δ_{KSC}-σ_{SN} pour les données synthétiques. . . . 154
6.10 Représentabilité et Distinctivité des treillis générés par δ_{SCM}-σ_{SN}, δ_{SCM}-σ_{SA}, δ_{SCP}-σ_{SA} et δ_{KSC}-σ_{SN} pour `Cité-du-vin`. . . . 155
6.11 Temps d'exécution et nombre de concepts et d'irréductibles générés par δ_{SDCM}-σ_{SDN} et δ_{SDCM}-σ_{SDM} pour `Cité-du-vin`. . . . 156

6.12 Représentabilité, Distinctivité et Stabilité logarithmique globale des treillis généré par δ_{SDCM}-σ_{SDN} et δ_{SDCM}-σ_{SDM} pour `Catch-me` et `Cité-du-vin`. . . . 158
6.13 Nombre de concepts obtenu avec différents *pas* en utilisant δ_{SDCM}-σ_{SDP} pour `Catch-me` et `Cité-du-vin`. 159
6.14 Nombre de concepts du treillis généré par δ_{SDCMw}-σ_{SDN} et δ_{SDCMg}-σ_{SDN} pour `Cité-du-vin`. 159
6.15 Temps d'exécution et nombre de concepts des treillis générés par δ_{SCMIN}-$\sigma_{\text{NCT,BCT,FACT,ACT}}$ pour `Cité-du-vin`. 161

Liste des tableaux

1 Les descriptions proposées selon le type de séquence 15
2 Les stratégies proposées selon le type de séquence 16

1.1 Exemple de données simples 22
1.2 Exemple de séquences d'actions quotidiennes 24
1.3 Exemple de séquences temporelles d'actions quotidiennes 24
1.4 Exemple de séquences d'intervalles d'actions quotidiennes 25
1.5 Définitions des séquences simples, temporelles et d'intervalles avec des éléments et des ensembles d'éléments 25

2.1 Exemple d'un ensemble de données 33
2.2 Algorithmes de fouille de motifs séquentiels fréquents 34
2.3 Déroulement de l'algorithme *GSP* pour l'Exemple 1.1 39
2.4 La représentation verticale de l'ensemble de données séquence de l'Exemple 1.2. 40
2.5 Les représentations projetées pour les motifs fréquents de l'Exemple 1.1 . 42

3.1 Exemple de contexte 61
3.2 Exemple de sous-ensembles construits à partir de données binaires 64

4.1 Exemple de données numériques et catégorielles 75

5.1 Quelques descriptions pour les séquences de l'Exemple 1.1 98
5.2 Quelques exemples des stratégies SN et SA pour les séquences 1,2,3 et 5 de l'Exemple 1.1 100
5.3 Combinaisons description/stratégie avec les figures des treillis générés . . . 100
5.4 Sous-séquence distancielle commune de $\{S_1, S_2, S_4, S_5\}$ 107
5.5 Exemples de prédicats de la description SDCM pour l'exemple 1.2. 110
5.6 Exemples de prédicats des descriptions SDCMg et SDCMw pour l'exemple 1.2. 110
5.7 Les stratégies SDN, SDP et SDM pour l'ensemble $A = \{S_1, S_2, S_5\}$. 112
5.8 Combinaisons description/stratégies avec les figures des treillis générés . . 112
5.9 Combinaisons description/stratégies avec les figures des treillis générés . . 124
5.10 Extensions de type caractéristique des données séquentielles avec quelques statistiques 132
5.11 Extensions de type description pour les données séquentielles avec quelques statistiques 133

5.12 Extensions de type stratégie pour les données séquentielles avec quelques statistiques . . . 135
5.13 Résumé des descriptions, stratégies et mesures proposées. . . . 138

6.1 L'ensemble de jeux de données utilisés pour les expérimentations . . . 141
6.2 Les aspects de jeu de données `GéoLuciole` avec leurs types, leurs alphabets et leurs sources . . . 144
6.3 Exemple de prédicats montrant l'impact du climat sur une excursion à la plage. . . . 151
6.4 Exemple de prédicats montrant les habitudes de sommeil par status. . . . 151
6.5 Comparaison du nombre de motifs générés entre CloSpan et δ_{SCM}-σ_{SN} et δ_{SCM}-σ_{SA} pour les données synthétiques . . . 153
6.6 Nombre de concepts et prédicats générés par δ_{SCM}-σ_{SN} et δ_{KSC}-σ_{SN} pour `Daily-actions`. . . . 153
6.7 Nombre de concepts et prédicats générés par δ_{SCM}-σ_{SA} et δ_{SCP}-σ_{SA} pour `Cité-du-vin` . . . 154
6.8 Nombre de concepts, temps d'exécution, sup et inf irréductibles générés par δ_{SDCM}-σ_{SDN} et δ_{SDCM}-σ_{SDM} pour `Catch-me`. . . . 157
6.9 Nombre de concepts et de temps d'exécution des treillis générés par δ_{SCMAX}-σ_{CMA} et δ_{SCMIN}-σ_{NCT} pour `Cité-du-vin`. . . . 160
6.10 Taux de compression et nombre de concepts des treillis générés par δ_{SCMIN}-$\sigma_{\text{NCT,BCT,FACT,ACT}}$ pour `GéoLuciole`. . . . 161

7.1 Résumé de tous les descriptions et stratégies proposées avec les expérimentations et les publications. . . . 165

Introduction générale

De nos jours, d'énormes quantités de données sont produites et générées dans de nombreux domaines. Toutes ces données contiennent des connaissances qui peuvent être précieuses pour mieux comprendre le monde autour de nous et apporter des solutions à des questions pratiques. À la question *"Comment extraire des connaissances à partir des données ?"*, l'analyse de données propose des méthodes nombreuses et variées pour mieux comprendre les données. Un exemple de connaissance extraite à partir de séquences de déplacements de touristes est : *"Les touristes qui visitent la ville avec leurs amis et qui ont participé à notre expérimentation ont tendance à passer leurs nuits dans des quartiers festifs"*.

Il existe plusieurs techniques d'analyse de données, qui se distinguent selon un objectif de description des données ou de prédiction d'une nouvelle donnée à partir des données existantes ; on parle de méthodes descriptives ou prédictives. La *classification non supervisée* est une méthode descriptive qui vise à regrouper les données selon leur similarité, alors que les méthodes de régression visent à prédire la valeur d'une donnée.

Les méthodes d'analyse de données diffèrent également en fonction des données traitées. Nous pouvons distinguer deux catégories des données. Les données dites "simples" s'organisent sous forme tabulaire classique avec des objets en lignes et des attributs en colonnes, en indiquant dans les cases la valeur d'un attribut pour un objets. Ces valeurs peuvent être numériques ou catégorielles. On trouve également des données dites "complexe" lorsqu'elles ne sont pas représentables avec le format classique attribut-valeur. Par exemple, les graphes ou les séquences sont des données complexes.

Les techniques d'analyse classiques de données sont définies pour des données simples tabulaires, et il est difficile de les étendre vers des données complexes. À travers ce travail, nous allons nous intéresser à des méthodes d'analyse descriptives pour les données séquentielles.

Les données de type séquences sont à nos jours utilisées dans beaucoup de domaines dans le but de mieux les analyser et d'en extraire des connaissances. Séquences des mots dans un texte, séquences de trajectoires de déplacement, séquences d'étapes de vie, séquences de navigation sur internet, ou même séquences d'achats de produits dans un supermarché, tous ces cas sont des exemples simples de séquences qui peuvent être analysés afin d'en tirer des connaissances. En fait, nous pouvons observer des séquences partout autour de nous. Il suffit de définir deux axes, un axe temporel (ou autre qui définit une relation d'ordre entre les éléments) et un axe représentant des éléments de

la séquence ; que ce soit des actions ou événements, des mots, des étiquettes ou bien tout autre élément qui apparaît dans un ordre donné. Par conséquent, nous pouvons dire qu'*une séquence est un ensemble d'éléments ordonnés.*

L'importance des séquences réside dans les liaisons entre les éléments de la séquence. C'est un type de données qui raconte une histoire, qui ne donne pas une caractéristique simple pour l'objet ou l'individu, mais donne un parcours entier de changement temporel de cet individu. Cela est très important dans la compréhension des systèmes et des individus.

L'utilisation des séquences a permis l'évolution de plusieurs domaines : la bio-informatique où l'utilisation des séquences pour l'alignement d'ADN est devenue une tâche très importante pour l'analyse de ce dernier [Sanger et al., 1977, Durbin et al., 1998], la sociologie où l'on s'intéresse aux actions et aux événements [Abbott, 1995], l'économie avec l'utilisation des séquences d'achats des consommateurs [Aaker et al., 1986], la fouille de texte où l'analyse de séquence joue un rôle important dans l'extraction des motifs séquentiels des expressions [Munková et al., 2013]. Et avec la variété des applications apparaît également des différences dans la définition des séquences.

Nous pouvons définir plusieurs types de séquences. Les séquences simples sont composées de plusieurs éléments ordonnées. Les séquences temporelles introduisent une information temporelle aux éléments et les séquences d'intervalles lient les éléments à un intervalle de temps.

Il existe plusieurs problèmes liés aux séquences. Parmi ces problèmes on retrouve des problèmes d'extraction des comportements ou des problèmes de similarités. La recherche de la plus longue sous-séquence commune (*Longest Common SubSequences (LCSS)*) est un problème bien connu [Maier, 1978] et considéré comme un problème de similarité. Il s'agit d'extraire les sous-séquences communes les plus longues d'un ensemble de séquences. Ce problème est défini pour deux séquences et peut être résolu par la programmation dynamique. Pour un ensemble de plus de deux séquences le problème est clairement NP-difficile [Maier, 1978]. Le temps d'exécution est exponentiel en nombre de séquences. Ce problème a de nombreuses applications en bio-informatique et en biologie moléculaire. Dans le domaine de la bio-informatique, les molécules d'*ADN*, d'*ARN* et de protéines sont souvent représentées par des séquences. La longueur du *LCSS* est une mesure largement utilisée en biologie computationnelle [Minkiewicz et al., 2015], dans l'édition de textes [Kruskal, 1983] dans la compression des fichiers [Storer, 1987], et la comparaison de fichiers [Bergroth et al., 2000] etc.

La fouille de séquences est un domaine d'analyse de données qui vise à extraire des motifs séquentiels fréquents dans un ensemble de données séquentielles, où ces motifs sont le plus souvent des sous-séquences communes. La fouille de séquences est une extension de la fouille d'*itemsets* (ensemble d'éléments) qui est un domaine de recherche introduit en 1994 par Agrawal et Srikant consistant à extraire des *itemsets* fréquents à partir de données décrites par des attributs binaires, où de façon équivalente par l'ensemble des attributs possédés (valeur binaire positive). Un *itemset* est un ensemble d'éléments identifié comme fréquent lorsqu'il est partagé par un nombre suffisamment représentatif de données. Les algorithmes de fouille d'*itemsets* fréquents utilisent une technique de

génération des candidats niveau par niveau en utilisant une mesure de *support* qui définit la proportion de données partageant un *itemset*.

Ce domaine de recherche a été introduit par Agrawal et al pour étudier le *"panier de la ménagère"* où les éléments sont des articles achetés par les clients et un *itemset* est un ensemble d'articles. Une année plus tard, les mêmes auteurs ont étendu cette approche à des données séquentielles où les *itemsets* sont ordonnées selon la date d'achat pour construire une séquence d'*itemsets*. La fouille de données séquentielles consiste alors à extraire des *motifs séquentiels fréquents*, à savoir des sous-séquences communes dans un ensemble de données séquentielles afin d'identifier les comportements fréquents des clients.

Plusieurs algorithmes ont été proposés pour l'extraction des *itemsets* et des motifs séquentiels fréquents. Avec l'évolution des capacités de calcul, la tâche d'extraction des motifs fréquents est devenue plus rapide. La difficulté réside alors dans le trop grand nombre de motifs extraits, qui en rend difficile la lisibilité et donc l'interprétation. On parle de *déluge de motifs*.

Pour remédier au problème du *déluge de motifs*, une premiere approche a consisté à réduire les motifs fréquents générés aux seuls motifs fermés qui portent la même information [Pasquier et al., 1999]. Plusieurs algorithmes ont été proposés dans ce sens.

Il a été observé que l'ensemble de tous les motifs fermés peut être organisé dans une structure appelée *treillis*. Cette structure est la base de l'Analyse Formelle de Concepts. L'Analyse Formelle de Concepts (AFC) est un domaine d'analyse de données, qui permet d'identifier des relations dans l'ensemble de données. L'AFC a été introduite en 1982 [Wille, 1982], puis dans les travaux de Ganter et Wille en 1999 [Ganter and Wille, 1999] et fait suite aux travaux de Barbut et Monjardet de 1970 [Barbut and Monjardet, 1970]. Elle est basée sur la théorie mathématique des ensembles ordonnés, et sur la théorie des treillis introduite par Birkhoff en 1940 [Birkhoff, 1940]. À partir d'une relation binaire entre un ensemble d'objets et un ensemble d'attributs, les concepts formels sont construits comme des ensembles maximaux d'objets en relation avec des ensembles maximaux d'attributs. L'ensemble des concepts reliés par une relation de spécialisation/généralisation forme un ensemble partiellement ordonné possédant la propriété de treillis appelé treillis de concepts. Ce treillis représente les données où les concepts sont des sous-groupes d'objets similaires, partageant les mêmes attributs, et le treillis des concepts correspond à une hiérarchie où chaque sous-groupe est décrit par un ensemble d'attributs qui correspond a un *itemset* fermé. La propriété du treillis garantit à la fois une hiérarchie de sous-groupes et une structure de navigation complète et consistante pour des approches de recherche d'information [Ferré, 2014].

Telle qu'elle est introduite initialement, l'AFC permet de traiter des données décrites par des ensembles d'attributs, donc des données binaires. Le formalisme des structures de motifs [Ganter and Kuznetsov, 2001, Kaytoue et al., 2015] et la navigation conceptuelle abstraite [Ferré, 2014, Ferré, 2002] étendent l'AFC pour traiter des données non binaires, les objets peuvent être décrits par d'autres types de description que des attributs binaires. Ces extensions reposent sur une propriété importante : la structure de treillis est maintenue lorsque l'espace de description s'organise comme un inf-demi-treillis afin de maintenir une connexion (dite de Galois) entre les objets et leurs descriptions. Les structures de

motifs permettent ainsi de traiter des données non binaires et complexes telles que les séquences. Le trellis ainsi défini correspond à une hiérarchie de sous-groupes où les motifs sont des motifs séquentiels fermés. Les treillis de motifs restent cependant volumineux et souvent impossibles à traiter [Kaytoue, 2020]. Le nombre de concepts est exponentiel en fonction de l'ensemble de données en entrée, et donc la réduction des treillis constitue un défi actuel.

Inspiré des structures de motifs, l'algorithme NextPriorityConcept, introduit dans un article récent [Demko et al., 2020], propose une approche d'extraction de motifs pour des données hétérogènes et complexes en entrée. Cet algorithme permet un calcul de motifs génériques à travers des *descriptions* spécifiques d'objets par des *prédicats monadiques*. Il propose également de raffiner un ensemble d'objets en un ensemble plus petit à travers des *stratégies* d'explorations spécifiques de l'utilisateur, ce qui permet de réduire le nombre de motifs et ainsi limiter le deluge de motifs générés.

L'algorithme NextPriorityConcept introduit ce mécanisme de *description* d'un sous-groupe de données offrant la possibilité de définir des prédicats pour des types de données différents, permettant ainsi d'appréhender des données hétérogènes. Par exemple, les attributs numériques seront décrits par des prédicats de type "est plus grand/petit que x", et des attributs catégoriels par des prédicats de type "contient une valeur dans X". Ces prédicats sont calculés de manière itérative pour chaque sous-groupe.

L'algorithme NextPriorityConcept introduit également la notion de *stratégie* pour générer les sous-groupes du niveau suivant. Une stratégie génère des prédicats permettant de séparer un sous-groupe en plusieurs sous-groupes. Alors qu'une stratégie naïve va considérer tous les sous-groupes possibles d'un sous-groupe, des stratégies plus élaborées permettent d'en réduire le nombre et ainsi de limiter le nombre de sous-groupes global généré, et donc le déluge de motifs. La plateforme `GALACTIC`[1] implémente l'algorithme NextPriorityConcept et propose un écosystème d'extensions pour le traitement de données hétérogènes.

La problématique principale de cette thèse est :

Comment étendre l'algorithme NextPriorityConcept *à des données séquentielles ?*

L'adaptation de l'algorithme NextPriorityConcept pour les séquences passe par la définition de *descriptions* et de *stratégies* spécifiques aux séquences. Il existe plusieurs façons de définir des descriptions pour des séquences. i.e. on peut décrire un ensemble de séquences de plusieurs manières. Par exemple, un ensemble de séquences peut être décrit simplement par l'ensemble de sous-séquences communes maximales mais aussi par leur préfixe maximal commun. La manière dont on décrit un ensemble de séquences depend de l'information que l'on souhaite extraire. Les descriptions sont également liées au type de séquences, selon qu'elles sont simples ou qu'elles contiennent une information temporelle. Ainsi, plusieurs descriptions sont envisageables selon l'information que l'on souhaite extraire et le type de séquences manipulées. Une premiere problématique de cette these est donc :

1. https ://galactic.univ-lr.fr

Comment peut-on décrire un ensemble de séquences selon qu'elles portent ou non une information temporelle ?

Plusieurs stratégies d'exploration peuvent être utilisées pour les séquences. On peut facilement définir des stratégies naïves pour des séquences simples et temporelles, qui génèrent tous les concepts possibles. D'autres stratégies sont également envisageables pour réduire le nombre de concepts. Notre deuxième problématique sera donc :

Comment peut-on définir des stratégies qui réduisent le nombre de concepts ?

Plusieurs strategies étant envisageables générant des concepts et treillis différents, il est nécessaire de pouvoir les comparer. Une dernière problématique de ce manuscrit sera donc :

Comment mesurer la qualité du treillis obtenu ?

Contributions

Dans ce travail, nous nous sommes intéressés à l'analyse de séquences en utilisant l'Analyse Formelle de Concepts et l'algorithme NextPriorityConcept. On considère trois types de séquences, les séquences simples, les séquences temporelles et les séquences d'intervalles.

Descriptions pour les séquences

Nous avons développé plusieurs descriptions pour les données séquentielles. La description avec l'ensemble des sous-séquences communes maximales est une description classique. Dans ce manuscrit, nous proposons et définissons plusieurs descriptions pour chaque type de séquence. Les descriptions que nous proposons sont résumées dans la Table 1.

Type de séquence	Descriptions
Séquences simples	*K-Sous-séquences Communes (KSC)* *Sous-séquences Communes Maximales (SCM)* *Sous-séquences Communes Préfixées (SCP)*
Séquences temporelles	*Sous-séquences Distanciées Communes Maximales (SDCM)* SDCM avec contrainte de *fenêtre* (SDCMw) SDCM avec contrainte d'*écart* (SDCMg)
Séquences d'intervalles	*Sous-intervalles Communs MAXimaux (SCMAX)* *Super-intervalles Communs MINimaux (SCMIN)*

Table 1 – Les descriptions proposées selon le type de séquence

Stratégies pour les séquences

Nous avons développé plusieurs stratégies pour les données séquentielles selon le type des séquences. Une stratégie naïve est définie pour chaque type de séquence, ainsi que d'autres strategies permettant de reduire le nombre de concepts générés. Les stratégies que nous proposons sont résumées dans la Table 2.

Type de séquence	Stratégies
Séquences simples	*Stratégie Naïve (SN)* *Stratégie Augmentée (SA)*
Séquences temporelles	*Stratégie Distancielle Naïve (SDN)* *Stratégie Distancielle Par Étape (SDP)* *Stratégie Distancielle Milieu (SDM)*
Séquences d'intervalles	*Cardinalité Minimale Augmentée (CMA)* *Naïve Cadre Temporel (NCT)* *Bornes des Cadres Temporels (BCT)* *Fenêtre Affixe de Cadre Temporel (FACT)* *Alphabet Cadre Temporel (ACT)*

TABLE 2 – Les stratégies proposées selon le type de séquence

Mesures de qualité non supervisées

Enfin, nous définissions dans ce memoire de nouvelles mesures de qualité des concepts et treillis générés. Nous proposons trois mesures qui évaluent la qualité des treillis générés. La *Stabilité logarithmique Normalisée globale* donne une valeur globale de la stabilité du treillis. La *Représentabilité* mesure la quantité d'information générée. La *Distinctivité* mesure la capacité à distinguer les objets.

Organisation de ce mémoire

Ce manuscrit est organisé en deux parties :

Partie I

Une premiere partie qui résume l'état de l'art sur l'analyse de séquences et l'Analyse Formelle de Concepts avec introduction de l'algorithme NEXTPRIORITYCONCEPT que nous utilisons par la suite. Cette partie est organisée en quatre chapitres. Le premier chapitre est une introduction à l'analyse de données avec les définitions de bases autour des séquences. Le deuxième chapitre présente la fouille de motifs séquentiels. Le troisième chapitre introduit l'Analyse Formelle des Concepts et ses extensions pour des données non binaires. Et le quatrième chapitre présente l'algorithme NEXTPRIORITYCONCEPT et la plateforme `GALACTIC`.

Partie II

La deuxième partie de ce mémoire présente nos contributions majeures, avec deux chapitres. Le premier chapitre porte sur la définition des descriptions, des stratégies et des mesures de qualité pour l'analyse de séquences. Le deuxième chapitre décrit nos expérimentations avec une analyse qualitative et quantitative des résultats.

Première partie

État de l'art

Chapitre 1

Définitions et notations

Sommaire

1 Fouille de données 21
1.1 Données 21
1.2 Méthodes 22
2 Données séquentielles 23
3 Conclusion et discussion 26

1 Fouille de données

Le développement rapide de l'informatique nous a mis devant un immense volume de données qui peuvent être analysées afin d'en tirer des connaissances dans plusieurs domaines, dans l'industrie, l'économie, les découvertes scientifiques, etc. Presque tous les systèmes modernes génèrent des données pour mieux comprendre le comportement du système et de ses utilisateurs. Ces données diffèrent en termes de quantité et/ou de structure. On peut citer de nombreux exemples de données qui s'organisent en séquences : les traces utilisateurs où les actions dans un système informatique sont récoltées, les déplacements des touristes dans les villes et dans les musées, la quantité de textes produits sur internet, les achats sur les sites d'e-commerce. Il s'agit d'exemples qui nécessitent des outils adaptés pour les analyser et mieux les comprendre. Le fouille de donnes est un vaste domaine qui regroupe de nombreuses méthodes, avec des objectifs différents, pour des donnees différentes.

1.1 Données

Les données sont classiquement organisées sous forme tabulaire avec des objets en lignes, des caractéristiques ou attributs en colonnes et la valeur de chaque attribut lié à un objet dans les cases du tableau. Ce type de données simples peut être découpé en trois catégories selon le type de valeur des attributs :

Quantitatives ou numériques : ce sont des attributs à valeurs numériques ; comme l'âge, la taille, la température, etc.

Catégorielles ou discret : ce sont des attributs à valeurs discrètes, elles expriment l'‘appartenance d'‘un objet à une catégorie spécifique, comme la ville, la couleur, etc.

Binaires ce sont des attributs à valeur binaire. Cela est souvent représenté par une table avec des croix dans les cases signifiant qu'un objet possède un attribut. Les données binaires sont un cas particulier de données catégorielles à 2 modalités.

La Table 1.1 représente un exemple de données simples, où *âge* est un attribut quantitatif, *ville* un attribut catégoriel, et *chat* un attribut binaire.

personne	âge	ville	chat
Karell	49	La Rochelle	x
Christophe	45	Esnandes	x
Jérémy	25	La Rochelle	
Cécile	25	Pau	x
Salah	28	La Rochelle	

TABLE 1.1 – Exemple de données simples

Il existe des données plus complexes. Il s'agit de données structurées qui ne peuvent se représenter classiquement sous la forme attribut-valeur. Un des types de données complexes est la séquence. On retrouve des données structurées en séquence dans de nombreux domaines : séquences de mots dans un texte, trajectoires, séquences d'étapes de vie, séquences de navigation sur Internet, ou séquences d'achats de produits dans un supermarché. Une séquence est une liste ordonnée de symboles, d'ensembles ou d'événements.

D'autres types de données complexes sont les graphes ou encore les séries temporelles. Une série temporelle est une série de valeurs numériques indexées dans l'ordre chronologique. Le plus souvent, une série temporelle est une séquence prise à des points successifs et équidistants dans le temps. Il s'agit donc d'une séquence de données à temps discret. Les séries temporelles sont utilisées dans les domaines qui impliquent des mesures temporelles, souvent issues des capteurs telles que les prévisions météorologiques, la prévision des tremblements de terre, etc. Les graphes aussi font partie des données complexes qui décrivent un ensemble de données avec les relations qui les lient. Les graphes peuvent par exemple représenter des réseaux sociaux ou des réseaux biologiques au niveau des molécules, des protéines ou des espèces.

1.2 Méthodes

La grande variété des types de données rencontrés dans les applications réelles est un défi pour la fouille de données. Il existe deux types de méthodes d'exploration et de fouille de données :

Méthodes prédictives Les méthodes prédictives s'appuient sur des données connues pour essayer de prévoir de futures données. On s'intéresse ici à un ou plusieurs attributs définis comme étant les cibles de l'analyse. On construit un modèle à partir de données d'apprentissage, et on cherche à prédire les valeurs des attributs cibles. Il y a deux types d'analyse selon le type de l'attribut cible à prédire. Pour un attribut cible *discret*, on parle de méthode de classification (supervisée). Alors que, pour un attribut cible *numérique*, il s'agit de prédiction. Citons les réseaux de neurones qui sont des méthodes prédictives à partir de données numériques, ou encore l'arbre de décision qui permet de considérer des données catégorielles.

Méthodes descriptives Les méthodes descriptives cherchent à décrire les données ainsi que leurs similarités et leurs différences. L'objectif est de synthétiser les données et d'extraire des relations cachées entre les données. La description a pour but de décrire les tendances et les modèles cachés dans les données. Le *clustering* consiste à créer des sous-ensembles de données similaires encore appelés *clusters* ou sous-groupes afin d'en obtenir une description concise. Il existe plusieurs méthodes de *clustering*. La classification hiérarchique ascendante vise à construire une hiérarchie de cluster à partir de données numériques, ou encore des données catégorielles munies d'une mesure de distance ou de similarité. Les règles d'association permettent d'exprimer des corrélations sous forme de règles pour des données catégorielles. La fouille de motifs vise à construire une hiérarchie de *clusters* pour des données catégorielles ou structurées.

Dans le cadre de notre travail, nous nous concentrons sur la fouille de séquences, il s'agit de méthodes descriptives pour les données séquentielles. Nous nous intéressons plus particulièrement à la génération d'une hiérarchie de sous-groupes pour des données séquentielles.

Nous donnons dans ce chapitre les définitions et notations de base autour des séquences que nous allons utiliser par la suite dans ce manuscrit.

2 Données séquentielles

La fouille de motifs séquentiels consiste à découvrir des sous-séquences fréquentes dans un ensemble de séquences. Les motifs peuvent être des sous-séquences, des préfixes, suffixes, des sous-séquences avec ou sans écarts, etc. Dans cette section nous commençons par des définitions autour des séquences.

Il existe une grande variété de types de séquences, allant des simples séquences d'éléments, de symboles ou de lettres, à des séquences d'ensembles ou d'intervalles. Pour une application donnée, les séquences sont construites à partir d'un ensemble d'éléments appartenant à un alphabet. Un alphabet Σ est un ensemble d'éléments de même type qu'on note $\Sigma = \{e_1, e_2, \dots e_n\}$. Nous définissons une séquence s à partir d'un alphabet Σ :

Définition 1.1 (Séquence simple)**.** Une séquence simple $S = \langle s_i \rangle_{i \leq n}$ est une liste ordonnée d'*éléments* s_i appartenant à un alphabet Σ. n est la taille de la séquence S et on note $|S| = n$.

Dans ce manuscrit, sans precision pour les séquences temporelles ou d'intervalles, on utilise le terme séquence pour les séquences simples.

Un ensemble de séquences $G = \{(j, S_j)|j \leq d\}$ est un ensemble de couples (j, S_j) où S_j est une séquence et j est son identifiant. d est la taille de l'ensemble de séquences G et on note $d = |G|$.

Exemple 1.1. *Prenons l'exemple dans la table 1.2 qui présente cinq séquences d'activités réalisées par des personnes au cours d'une journée, leurs identifiants sont* $1, 2, 3, 4$ *et* 5. *L'alphabet des activités est* $\Sigma = \{$*Travail(***T***), Sieste(***S***), Café(***C***), Sport(***Sp***), Lecture(***L***)*$\}$ *La première séquence* S_1 *raconte que la personne a réalisé successivement cinq actions dans la journée :* **T**, **S**, **C**, **Sp** *et enfin* **L**.

SID	Séquences simples
1	$S_1 = \langle \mathrm{T}, \mathrm{S}, \mathrm{C}, \mathrm{Sp}, \mathrm{L} \rangle$
2	$S_2 = \langle \mathrm{T}, \mathrm{C}, \mathrm{Sp} \rangle$
3	$S_3 = \langle \mathrm{T}, \mathrm{C}, \mathrm{L} \rangle$
4	$S_4 = \langle \mathrm{C}, \mathrm{T}, \mathrm{Sp} \rangle$
5	$S_5 = \langle \mathrm{T}, \mathrm{C}, \mathrm{S}, \mathrm{Sp} \rangle$

TABLE 1.2 – Exemple de séquences d'actions quotidiennes

Nous étendons la définition de la séquence à une séquence temporelle avec l'ajout de l'information temporelle dans la séquence.

Définition 1.2 (Séquence temporelle)**.** Une séquence temporelle $S = \langle (t_i, s_i) \rangle_{i \leq n}$, est une séquence où chaque élément $s_i \in \Sigma$ est associé à un horodatage t_i, avec $t_i < t_{i+1}$.

Exemple 1.2. *Étendant l'Exemple 1.1 pour des séquences temporelles, la Table 1.3 enrichit les séquences avec une information temporelle qui correspond aux heures dans la journée.*

SID	Séquences temporelles
1	$S_1 = \langle (8, \mathrm{T}), (14, \mathrm{S}), (16, \mathrm{C}), (18, \mathrm{Sp}), (21, \mathrm{L}) \rangle$
2	$S_2 = \langle (9, \mathrm{T}), (10, \mathrm{C}), (18, \mathrm{Sp}) \rangle$
3	$S_3 = \langle (9, \mathrm{T}), (11, \mathrm{C}), (14, \mathrm{L}) \rangle$
4	$S_4 = \langle (9, \mathrm{C}), (10, \mathrm{T}), (12, \mathrm{Sp}) \rangle$
5	$S_5 = \langle (10, \mathrm{T}), (11, \mathrm{C}), (14, \mathrm{S}), (18, \mathrm{Sp}) \rangle$

TABLE 1.3 – Exemple de séquences temporelles d'actions quotidiennes

Pour plus de précisions lorsqu'un événement n'est pas ponctuel mais peut se dérouler sur une durée dans le temps, on introduit les séquences d'intervalles [1], qui sont des

1. Par abus de langage, nous utiliserons le terme "intervalle"

séquences où chaque élément se déroule au cours d'un intervalle de temps avec des intervalles qui se succèdent.

Définition 1.3 (Séquence d'intervalles)**.** Une séquence d'intervalles $S = \langle(\underline{t}_i, \overline{t}_i, s_i)\rangle_{i\leq n}$ est une liste de tuples vérifiant $\underline{t}_i < \overline{t}_i$ et $\overline{t}_i \leq \underline{t}_{i+1}$. Ainsi, une séquence basée sur des intervalles est une liste d'intervalles distincts contenant un élément. La taille n de la séquence d'intervalles est le nombre de ses tuples.

Exemple 1.3. *En remplaçant les horodatages par des intervalles on obtient des séquences d'intervalles. La table 1.4 présente un ensemble de* 5 *séquences d'intervalles.*

SID	Séquences d'intervalles
1	$S_1 = \langle(8, 12, \mathrm{T}), (14, 15, \mathrm{S}), (16, 19, \mathrm{C}), (18, 19, \mathrm{Sp}), (20, 21, \mathrm{L})\rangle$
2	$S_2 = \langle(9, 10, \mathrm{T}), (10, 11, \mathrm{C}), (18, 19, \mathrm{Sp})\rangle$
3	$S_3 = \langle(9, 11, \mathrm{T}), (11, 12, \mathrm{C}), (14, 15, \mathrm{L})\rangle$
4	$S_4 = \langle(9, 10, \mathrm{C}), (10, 12, \mathrm{T}), (12, 14, \mathrm{Sp})\rangle$
5	$S_5 = \langle(10, 11, \mathrm{T}), (11, 12, \mathrm{C}), (14, 15, \mathrm{S}), (18, 19, \mathrm{Sp})\rangle$

TABLE 1.4 – Exemple de séquences d'intervalles d'actions quotidiennes

Les séquences présentées sont des séquences d'éléments qui peuvent être étendues facilement aux séquences d'ensembles d'éléments en remplaçant les éléments $s_i \in \Sigma$ par des ensembles d'éléments $S_i \subseteq \Sigma$. La table 1.5 résume les types de séquences avec deux définitions possibles, comme des séquences d'éléments ou comme des séquences d'ensemble d'éléments.

Type de séquence	Éléments $s_i \in \Sigma$	Ensembles $S_i \subseteq \Sigma$
Séquences simples	$\langle s_i\rangle_{i\leq n}$	$\langle S_i\rangle_{i\leq n}$
Séquences temporelles	$\langle(t_i, s_i)\rangle_{i\leq n}$	$\langle(t_i, S_i)\rangle_{i\leq n}$
Séquences d'intervalles	$\langle(\underline{t}_i, \overline{t}_i, s_i)\rangle_{i\leq n}$	$\langle(\underline{t}_i, \overline{t}_i, S_i)\rangle_{i\leq n}$

TABLE 1.5 – Définitions des séquences simples, temporelles et d'intervalles avec des éléments et des ensembles d'éléments

Pour un ensemble de séquences, les sous-séquences fréquentes sont une manière de décrire le comportement commun d'un ensemble de séquences.

Exemple 1.4. *Reprenons l'Exemple 1.1, la sous-séquence* $\langle T, C, Sp\rangle$ *est fréquente car elle se produit trois fois dans l'ensemble de séquences. Cela peut se traduire par : la plupart des personnes travaillent, prennent ensuite un café, puis font du sport. En ajoutant l'information temporelle nous pouvons avoir plus de précisions sur la connaissance extraite, de l'Exemple 1.2 : la plupart des personnes travaillent entre* $8h$ *et* $10h$*, prennent ensuite un café entre* $10h$ *et* $6h$*, puis font du sport le soir à* $18h$*. Pour les séquences d'intervalles (Exemple 1.3), les individus des séquences* 1 *et* 5 *travaillent tous les deux entre* $10h$ *et* $11h$ *et font du sport de* $18h$ *à* $19h$*.*

Ces trois exemples présentent des comportements communs à un sous-ensemble de séquences. L'extraction des sous-séquences permet d'identifier ces comportements.

Définissons maintenant de manière formelle la relation de sous-séquence pour les séquences simples. Pour les séquences temporelles et d'intervalles, la définition de sous-séquence sera discutée dans le Chapitre 5.

Définition 1.4 (Sous-séquence)**.** Une séquence $S = \langle s_1, s_2 \dots s_n \rangle$ est une sous-séquence d'une séquence $R = \langle r_1, r_2 \dots r_m \rangle$ et on écrit $S \sqsubseteq_s R$ s'il existe des entiers $1 \leq i_1 < i_2 < \dots < i_n \leq m$ tels que $s_j = r_{i_j}$ avec $j \leq n$. On dit aussi que R est une super-séquence de S ou R contient S.

Le symbole $\sqsubseteq_s$ désigne la relation de sous-séquence, à distinguer du symbole $\sqsubseteq$ classiquement utilisé pour la subsomption. On peut dire que la séquence S_2 de l'exemple 1.1 est une sous-séquence de la séquence S_1, et on écrit $S_2 \sqsubseteq_s S_1$.

On définit des sous-séquences particulières d'une séquence qui sont les *fenêtres*, *préfixes* et *suffixes* :

Définition 1.5 (Fenêtre, préfixe et suffixe)**.** On appelle fenêtre $F = \langle f_1, f_2, \dots f_m \rangle$, d'une séquence $S = \langle s_1, s_2, \dots s_n \rangle$ une partie de la séquence telle qu'il existe deux entiers k et l ; $1 \leq k \leq l \leq n$ et $f_i = s_k, f_{i+1} = s_{k+1}, \dots, f_n = s_l$. On note cette fenêtre $F = S(k, l)$.

Un préfixe d'une séquence S est une fenêtre de S qui apparaît au début de S, que l'on note $Pre(S) = S(1, l)_{1 \leq l \leq n}$. Un suffixe d'une séquence S est une fenêtre qui se produit à la fin de S, que l'on note $Suf(S) = S(k, n)_{1 \leq k \leq n}$.

Exemple 1.5. *Dans nos exemples, la sous-séquence* $S = \langle T, C \rangle$ *et un préfixe de* S_2, S_3 *et* S_5. *La fenêtre* F *de* $S_1(2, 4)$ *est* $F = \langle S, C, Sp \rangle$.

3 Conclusion et discussion

Dans ce chapitre, nous avons introduit le domaine de la fouille de données en distinguant entre les méthodes descriptives et prédictives selon qu'elles visent à décrire des données existantes ou prédire de nouvelles données. Les méthodes classiques sont des méthodes numériques ou discrètes selon qu'elles s'appliquent sur des données numériques et discrètes, se pose alors la question des données hétérogènes qui peuvent être traitées par des méthodes classiques après un prétraitement de transformation. Citons la discrétisation des données numériques en intervalles, ou encore le plongement de données non numériques dans un vecteur numérique. Le même type de prétraitement est envisageable pour des données plus complexes telles que les séquences mais avec une perte de lisibilité.

Il existe des méthodes adaptées à la fouille de séquences dans les domaines de la fouille de motifs et de l'AFC. Notre travail se positionne sur une analyse descriptive des données séquentielles. Nous allons utiliser la plateforme `GALACTIC` qu'est un cadre de travail basé sur l'AFC pour des données hétérogènes et complexes. La plateforme est basée sur l'algorithme NextPriorityConcept qui donne la possibilité de définir des descriptions et des stratégies permettant l'analyse des données séquentielles.

Nous allons commencer par présenter dans le Chapitre 2 les méthodes adaptées aux séquences avec la fouille de motifs séquentiels. Ensuite, dans le Chapire 3 nous présentons l'analyse formelle de concepts avec ses extensions pour les données complexes dont les séquences. Et dans le dernier chapitre de cette partie d'état de l'art, nous présentons la plateforme `GALACTIC` et l'algorithme NextPriorityConcept qui regroupent des notions des deux domaines de la fouille de motifs et de l'AFC.

Chapitre 2

Fouille de séquences

"A sequence of random events isn't really random at all. It's just made to look that way so it will ensnare the unwary."

Anthony T.Hincks

Sommaire

1 Introduction 29
2 Fouille d'*itemsets* fréquents 30
2.1 L'algorithme *Apriori* 31
2.2 Fouille d'*itemsets* fermés fréquents 33
3 Fouille de motifs séquentiels fréquents 34
3.1 Les algorithmes basés sur *Apriori* : Représentation Horizontale 36
3.2 Les algorithmes basés sur *Apriori* : Représentation Verticale . . 39
3.3 Les algorithmes de croissance de motifs 40
3.4 Approches par contraintes 42
4 Fouille de motifs séquentiels fermés 43
5 Fouille de motifs temporels 47
5.1 Fouille de motifs séquentiels avec contraintes temporelles . . . 47
5.2 Fouille d'épisodes fréquents 48
5.3 Fouille de chroniques fréquentes 49
5.4 Fouille de motifs *delta* 51
6 Fouille de motifs séquentiels d'intervalles 51
7 Conclusion et discussion 53

1 Introduction

La fouille de motifs séquentiels consiste à extraire les motifs séquentiels fréquents. La fouille de séquences est une tâche de la fouille de données qui consiste en l'extraction de

connaissances à partir d'un ensemble de séquences. Cette connaissance prend plusieurs formes dont les motifs séquentiels qui sont souvent des sous-séquences communes d'un ensemble de séquences. Les sous-séquences communes donnent une information sur le comportement commun d'un ensemble de séquences. Elles nous aident à décrire ce comportement ou encore à prédire de nouveaux phénomènes. Nous pouvons constater qu'il existe plusieurs types de sous-séquences : les préfixes, les sous-séquences qui portent une information temporelle ou encore les sous-séquences qui respectent certaines contraintes.

Selon le type des séquences et des motifs séquentiels, plusieurs techniques et algorithmes ont vu le jour. Nous allons essayer à travers ce chapitre de couvrir la plupart des types d'analyse et les dernières avancées pour la fouille de séquences simples, de séquences temporelles et de séquences d'intervalles. Le reste de ce chapitre va être organisé en six sections. Dans la deuxième section nous commençons d'abord par introduire la fouille de motifs (*itemsets*) sur laquelle reposent nos travaux et qui présente un cadre similaire de la fouille de données séquentielles mais pour des données binaires. Ensuite nous présenterons les algorithmes de fouille de motifs séquentiels fréquents dans la troisième section, puis leur réduction aux motifs séquentiels fermés fréquents dans la quatrième section. Les cinquième et sixième sections présentent la fouille de motifs temporels et d'intervalles. Et nous finissons ce chapitre par une conclusion et discussion.

2 Fouille d'*itemsets* fréquents

Parmi les problèmes connus sous l'angle de l'analyse de données ou *data mining*, on trouve la fouille d'*itemsets* fréquents qui est parmi les tâches importantes pour la découverte de motifs et évènements fréquents et les relations entre les différents éléments d'un ensemble de données. La fouille d'*itemsets* fréquents consiste à extraire des *itemsets* fréquents, à savoir des groupes d'éléments apparaissant ensemble dans un ensemble de données. Le support mesure la fréquence d'apparition d'un motif dans l'ensemble de données. Le problème est défini formellement comme suit :

Soit $\Sigma = \{a_1, a_2, ...a_k\}$ un *alphabet* d'éléments dit *items*. Soit $\mathfrak{B}$ un ensemble de données constitué d'un ensemble d'*itemsets* $T \subseteq \Sigma$. L'ensemble de données $\mathfrak{B}$ est donc constituée d'un ensemble de couples $(tid_i, T_i)_{i \leq n}$; T_i est un ensemble d'éléments appelé *itemset*, et tid_i est l'identifiant de l'*itemset* $T_i \subseteq \Sigma$. On suppose que les éléments de chaque *itemset* sont triés dans un ordre lexicographique. Pour un sous-ensemble $A \subseteq \Sigma$ d'éléments, le support de A est défini comme la fréquence d'apparition de A dans l'ensemble de données $\mathfrak{B}$:

$$support(A) = \frac{|\{T \in \mathfrak{B} | A \subseteq T\}|}{|\mathfrak{B}|} \tag{2.1}$$

Le numérateur $|\{T \in \mathfrak{B} | A \subseteq T\}|$ de l'equation 2.1 est appelé la *cardinalité* de A ; c'est le nombre d'*itemsets* de $\mathfrak{B}$ contenant l'ensemble A. Les motifs d'*itemsets* fréquents sont donc ceux où le support est supérieur à un support minimal défini par l'utilisateur. Le calcul de support est présenté par l'Algorithme 1 avec la nécessité de parcourir l'ensemble des données. Pour un motif M, la fonction $support(M)$ donne le support de M.

Algorithm 1 Support

Input : Un ensemble de données $\mathfrak{B}$, un motif M
Output : Le support de M

```
1: sup ← 0;
2: for T ∈ 𝔅 do
3:     if M ⊆ T then
4:         sup ← sup + 1;
5:     end if
6: end for
7: Return sup;
```

La fouille de motifs a été introduite pour étudier "le panier de la ménagère" où les données consistent en des transactions de clients dans un supermarché où chaque client achète un ensemble de produits [Agrawal et al., 1993]. L'achat d'un ensemble de produits est appelé *transaction* $T_i \subseteq \Sigma$. Le défi donc est de savoir quels sont les produits achetés fréquemment ensemble, ou plus précisément, les ensembles d'éléments fréquents (*frequent itemsets*).

2.1 L'algorithme *Apriori*

Le premier algorithme qui répond au problème de la fouille de motifs fréquents est l'algorithme *Apriori* [Agrawal et al., 1993]. À partir d'un ensemble de données et d'un seuil de support minimal donné par l'utilisateur, l'algorithme génère l'ensemble des *itemsets* fréquents. C'est un algorithme niveau par niveau qui repose sur la monotonie du support.

Propriété 2.1 (Monotonie)**.** *La mesure de support est une mesure monotone, c'est à dire : Si* $A' \subseteq A$ *alors* $support(A) \leq support(A')$.

Une conséquence directe de cette proposition affirme que si un ensemble d'éléments est fréquent alors tous ses sous-ensembles sont fréquents et si un ensemble d'éléments est non fréquent alors tous ses super-ensembles sont non fréquents.

L'algorithme 2 décrit l'algorithme *Apriori*. Il s'agit d'un algorithme de génération de motifs fréquents niveau par niveau pour un $support_{min}$ en paramètre. Les motifs fréquents de niveau 1 sont tout d'abord générés (lignes 2-6). Il s'agit des motifs 1-*fréquents*, i.e, les motifs singletons dont le support est supérieur à $support_{min}$. Puis à chaque itération k de la boucle (lignes 8-18), les motifs $(k+1)$-*fréquents* sont générés (F_{plus}) à partir des motifs k-*fréquents* du niveau inférieur (F). Tout d'abord les motifs candidats sont générés comme extensions de l'ensemble F, i.e, une jointure entre l'ensemble F avec lui même (ligne 10). Le calcul des motifs candidats est décrit par l'algorithme 3. Puis seuls les motifs fréquents avec un support supérieur au seuil seront générés (lignes 11-17).

Algorithm 2 Apriori

Input : Un ensemble de données $\mathfrak{B}$, un seuil de support $support_{min}$ et un alphabet Σ

Output : L'ensemble de tous les motifs fréquents pour un $support_{min}$

$F \leftarrow \emptyset$;
for $x \in \Sigma$ **do**
  if $support(x) \geq support_{min}$ **then**
    $F \leftarrow F \cup \{\{x\}\}$;
  end if
end for
$Result \leftarrow F$;
while $F \neq \emptyset$ **do**
  $F_{plus} \leftarrow \emptyset$;
  Générer l'ensemble de candidats $C = Candidats(F)$
  for $M \in C$ **do**
    if $support(M) \geq support_{min}$ **then**
      $F_{plus} \leftarrow F_{plus} \cup M$;
    end if
  end for
  $F \leftarrow F_{plus}$;
  $Result \leftarrow Result \cup F_{plus}$;
end while
Return $Result$;

Algorithm 3 Candidats

Input : Un ensemble de motifs F

Output : Le résultat de jointure $F \times F$

$Res \leftarrow \emptyset$;
for $M = \{m_1, m_2...m_k\} \in F$ **do**
  for $N = \{n_1, n_2...n_k\} \in F$ **do**
    if $m_1 = n_1$ Et $m_2 = n_2$ Et ... Et $m_{k-1} = n_{k-1}$ Et $m_k < n_k$ **then**
      $J \leftarrow \{m_1, m_2...m_k\} \cup \{n_k\}$;
      $Res \leftarrow Res \cup J$
    end if
  end for
end for
Return Res;

Exemple 2.1. *Prenons l'exemple de la Table 2.1, qui présente un ensemble de données contenant* 5 *ensembles d'éléments.*

tid	transaction
1	$\{a, b\}$
2	$\{b, c, d\}$
3	$\{a, b, c, d\}$
4	$\{b, c\}$
5	$\{a, b, d\}$

TABLE 2.1 – Exemple d'un ensemble de données

La Figure 2.1 présente le déroulement de l'algorithme. L'algorithme procède par niveau et construit une hiérarchie qui correspond à un treillis (la définition de treillis sera discutée en Chapitre 3). Chaque *itemset* est associé avec son support. Avec un support minimal de $\frac{2}{5}$, les *itemsets* entourés en gris seront générés. Pour un support 0, tous les *itemsets* seront générés.

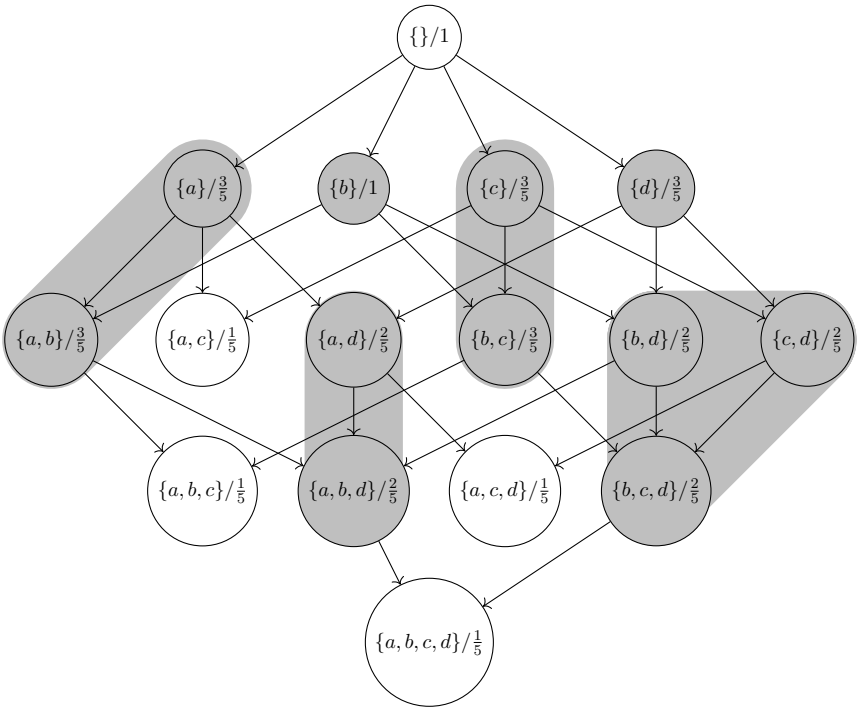


FIGURE 2.1 – *Itemsets* de la Table 2.1 avec leur support

2.2 Fouille d'*itemsets* fermés fréquents

Les améliorations de l'algorithme *Apriori* ont permis la réduction de temps de calcul [Zaki, 2000a, Borgelt, 2012, Zhang et al., 2013, Vo et al., 2016]. Cependant, deux problèmes sont vite apparus : le nombre exponentiel des motifs d'*itemsets* fréquents extraits qui rend plus difficile l'analyse et l'extraction de connaissance et la redondance de l'information.

Le défi revient donc à trouver un ensemble réduit de motifs qui permet de représenter la même information que l'ensemble de tous les motifs fréquents.

On identifie dans la Figure 2.1 des *itemsets "équivalents"* (les zones en gris dans l'arrière-plan) qui sont partagés par les mêmes objets (et donc de même support) et par conséquent qui partagent une information redondante. Le motif $\{a, d\}$ a un support de $\frac{2}{5}$, de même que le motif $\{a, b, d\}$. Ici l'information que $\{a, d\}$ se produit 2 fois dans les données est redondante et donc le motif $\{a, d\}$ peut être négligé des résultats. Chacun de ces groupes d'objets *"équivalents"* possède un seul *itemset* maximal qui est l'*itemset fermé*. Une solution pour réduire le nombre d'*itemsets* fréquents consiste donc à extraire seulement les *itemsets fermés fréquents*. Donc, au lieu de générer les 11 *itemsets* de la Figure 2.1, seuls les 6 fermés seront générés : $\{a, b\}, \{b\}, \{b, c\}, \{d\}, \{a, b, d\}, \{b, c, d\}$

L'algorithme *A-Close* [Pasquier et al., 1999] génère l'ensemble des *itemsets* fermés fréquents. Il utilise un opérateur de fermeture basé sur la connexion de Galois pour générer l'ensemble des *itemsets* fermés et ensuite il sélectionne les *itemsets* fermés fréquents. Les algorithmes *CHARM* [Zaki and Hsiao, 2002], *CLOSET* [Pei et al., 2000], ont également été introduits pour extraire les motifs fermés fréquents.

3 Fouille de motifs séquentiels fréquents

Selon le type de séquences, les algorithmes de fouille de séquences peuvent être divisés en trois catégories, les algorithmes qui extraient des motifs séquentiels, des motifs temporels ou des motifs d'intervalles. La Table 2.2 présente un résumé des algorithmes de fouille de motifs séquentiels.

Séquences	Types	Algorithmes
Séquences	Apriori	*AprioriAll*, *GSP*, *PSP*, etc.
	Apriori : (utilisation d'*Id-listes*)	*SPADE*, *SPAM*, etc.
	Croissance de motifs : Représentation de données projetées	*PrefixSpan*, *FreeSpan*, etc.
Séquence temporelles	Contraintes temporelles	*cSPADE*, *CCSM*, *GTC*, etc.
	Épisodes fréquents	*MINEPU*, *WINEPI*, *UP-Span*, etc.
	Chroniques	*FACE*, *ASTPMiner*, *DCM*, etc.
	Séquences annotées temporellement	*Hirate*, *Delta pattern*, *TAS*
Séquences d'intervalles	Relations d'Allen	*ARMADA*, *TPrefixSpan*, *QTempIntMiner*, etc.

TABLE 2.2 – Algorithmes de fouille de motifs séquentiels fréquents

Le test de sous-séquence est une tache très importante dans la fouille de motifs séquentielles. Nous utilisons l'Algorithme 4 pour effectuer ce test de sous-séquence. Pour

deux séquences S_1 et S_2, l'algorithme retourne *True* si $S_1 \sqsubseteq_s S_2$, *False* sinon. La complexité de cet algorithme est égale à la taille de S_2 : $O(|S_2|)$.

Algorithm 4 L'algorithme IsSubsequence

Entrée : Deux séquences S_1 et S_2
Sortie : *True* si $S_1 \sqsubseteq_s S_2$, *False* sinon.

```
1: R ← S2
2: // parcourir S1
3: for s ∈ S1 do
4:     sub ← False
5:     p ← 0
6:     while p ≤ |R| do
7:         if r_p = s then
8:             sub ← True
9:             break
10:        else
11:            p ← p + 1
12:        end if
13:    end while
14:    if sub = True then
15:        // prendre la sous-séquence de R qui commence à la position p
16:        R ← R(p, |R|)
17:    else
18:        Return False
19:    end if
20: end for
21: Return True
```

La fouille de motifs séquentiels fréquents consiste à extraire tous les motifs fréquents à partir d'un ensemble de séquences, i.e, leur support est supérieur à un support minimal défini par l'utilisateur. Les notions de support et motif fréquent introduits pour *Apriori* s'étendent naturellement aux séquences :

Définition 2.1 (Support)**.** Le support d'une séquence S est définie par :

$$support(S) = \frac{|\{X \in G \mid S \sqsubseteq_s X\}|}{|G|} \tag{2.2}$$

Le support correspond à la proportion des séquences de G contenant S comme sous-séquence. Le calcul du support est similaire à celui pour les *itemsets* (Algorithme 1) en remplaçant le test d'inclusion ensembliste $\subseteq$ en test de sous-séquence $\sqsubseteq_s$. Pour une séquence S, la fonction $support(S)$ donne le support de S.

Définition 2.2 (Motif séquentiel fréquent)**.** Étant donné un seuil de support minimal $support_{min}$, la sous-séquence S est appelée un *motif séquentiel fréquent* si et seulement si $support(S) \geq support_{min}$.

Les motifs séquentiels fréquents représentent des comportements fréquents extraits d'ensembles de séquences. Étant donné un ensemble de séquences et un seuil de support minimal $support_{min}$, la fouille de motifs séquentiels consiste à extraire toutes les sous-séquences S telles que $support(S) \geq support_{min}$.

Les premiers algorithmes de fouille de motifs séquentiels sont basés sur l'algorithme *Apriori* initialement conçu pour la fouille d'*itemset* fréquents. La propriété 2.2 est une extension de la propriété 2.1 de monotonie de support de motifs d'*itemsets* mais pour les motifs séquentiels.

Propriété 2.2. *Pour une séquence S et une sous-séquence S' de S ($S' \sqsubseteq_s S$), nous avons $support(S) \leq support(S')$.*

Cette propriété est la base des algorithmes de fouille de motifs séquentiels fréquents car elle permet la génération des motifs séquentiels fréquents niveau par niveau comme avec *Apriori*. En effet, si S est un motif séquentiel fréquent, alors tous ses sous-séquences sont fréquents $S' \sqsubseteq_s S$, et si une sous-séquence S' n'est pas fréquente, alors tous ses super-séquences $S \sqsupseteq_s S'$ ne le sont pas non plus.

Plusieurs algorithmes ont été proposés pour la fouille de motifs séquentiels, *GSP* [Srikant and Agrawal, 1996], *PSP* [Masseglia et al., 1998], *SPADE* [Zaki, 2001], *PrefixSpan* [Pei et al., 2001], *SPAM* [Ayres et al., 2002], *LAPIN* [Yang and Kitsuregawa, 2005], *CM-SPAM* et *CM-SPADE* [Fournier-Viger et al., 2014], *FreeSpan* [Han et al., 2000], etc. Tous ces algorithmes prennent en entrée un ensemble de séquences et un seuil de support minimal, et génèrent l'ensemble des motifs séquentiels fréquents.

Les algorithmes de fouille de motifs séquentiels sont de deux principales catégories. Les algorithmes basés sur *Apriori*, catégorisés selon le type de représentation de données utilisant une réprésentation classique des données ($G = \{(i, S_i)\}$) dite *horizontale*, qui sont très lents car ils nécessitent plusieurs passages de données pour le calcul de support. Les algorithmes utilisant une représentation de données dite *verticale* proposée par Zaki [Zaki, 2001] qui consiste à organiser les éléments dans des *Id-listes* où chaque élément est associé à une liste des séquences qui le contient. La représentation de données verticale évite le passage multiple sur l'ensemble de données, ce qui améliore énormément le temps de calcul. L'autre catégorie regroupe des algorithmes de croissance de motifs fréquents qui utilisent une autre amélioration de la représentation de l'ensemble de données dite ensemble de données projetées.

3.1 Les algorithmes basés sur *Apriori* : Représentation Horizontale

À partir d'un ensemble de données de séquences, les premiers algorithmes, *AprioriAll*, *AprioriSome* ont été introduits en se basant sur la propriété 2.2. La génération de motifs séquentiels fréquents est similaire de celle d'*Apriori*; l'algorithme procède par niveau, en prenant en considération l'ordre entre les éléments de la séquence au lieu d'un ensemble d'attributs binaires.

Ensuite Agrawal et Srikant ont proposé l'algorithme *GSP* (*Generalized Sequential Pattern algorithm*) [Srikant and Agrawal, 1996] pour améliorer ces algorithmes [Srikant and Agrawal, 1996] avec la possibilité d'introduire des contraintes temporelles, et des

fenêtres glissantes pour réduire le nombre de motifs générés. *GSP* est un algorithme d'exploration en largeur *(Breadth First Search)* qui utilise un ensemble de données horizontale et effectue plusieurs passes sur l'ensemble de données pour extraire tous les motifs séquentiels. Une représentation horizontale est organisée classiquement par $G = \{(i, S_i)|i \leq n\}$ (Exemple 1.1). *GSP* définit un motif *k-fréquent*, comme un motif séquentiel fréquent de taille k. Dans notre exemple la sous-séquence $S = \langle T, C\rangle$ est fréquente, c'est donc un motif 2-*fréquent* car $|S| = 2$.

L'algorithme 5 décrit *GSP*. Cet algorithme est similaire à *Apriori*. La différence porte sur les motifs générés qui ne sont pas des ensembles mais des séquences. Il s'agit d'un algorithme de génération de motifs séquentiels niveau par niveau. L'ensemble F de motifs séquentiels 1-fréquents est tout d'abord généré (lignes 2-6). Il s'agit des singletons avec support supérieur à $support_{min}$. À partir de l'ensemble F des motifs $k-1$-*fréquents*, l'ensemble de candidats C est généré par un produit (jointure) $F \times F$ (ligne 10). De façon similaire à Apriori (Algorithme 3), la jointure entre deux séquences S_1 et S_2 est réalisé si la sous-séquence obtenue en enlevant le premier élément de S_1 est la même que la sous-séquence obtenue en enlevant le dernier élément de S_2. La séquence candidate est donc S_1 étendue par le dernier élément de S_2. L'ensemble des candidats représente les motifs séquentiels de taille k obtenus par le produit des motifs séquentiels fréquents de taille $k-1$. L'ensemble F_{plus} est l'ensemble des séquences $S \in C$ qui ont un $support(S)$ supérieur à $support_{min}$ (ligne 12). L'algorithme s'arrête lorsqu'aucun nouveau motif n'est généré (boucle *while*).

La Table 2.3 illustre le déroulement de l'algorithme *GSP* sur l'Exemple 1.1, avec un support minimum égal à 3. L'algorithme calcule le support des motifs de taille 1, et ne garde que les motifs avec un support supérieur ou égal à 3, on obtient ainsi l'ensemble $F_1 = \{\langle T\rangle, \langle C\rangle, \langle Sp\rangle\}$ des motifs séquentiels 1-*fréquents*. À partir de cet ensemble, l'algorithme génère un ensemble de candidats C_2 (deuxième ligne de la Table 2.3 $k = 2$). En calculant le support, on obtient l'ensemble $F_2 = \{\langle T, C\rangle, \langle T, Sp\rangle, \langle C, Sp\rangle\}$. Le résultat final de l'algorithme est donc :

$$Res = \{\langle T\rangle, \langle C\rangle, \langle Sp\rangle, \langle T, C\rangle, \langle T, Sp\rangle, \langle C, Sp\rangle, \langle T, C, Sp\rangle\}$$

Algorithm 5 L'algorithme *GSP*

Input : Un ensemble de séquences G, un seuil de support $support_{min}$
Output : L'ensemble de tous les motifs séquentiels fréquents

1: $F \leftarrow \emptyset$;
2: **for** $x \in \Sigma$ **do**
3: **if** $support(x) \geq support_{min}$ **then**
4: $F \leftarrow F \cup \{\{x\}\}$;
5: **end if**
6: **end for**
7: $Result \leftarrow F$;
8: **while** $F \neq \emptyset$ **do**
9: $F_{plus} \leftarrow \emptyset$;
10: Générer l'ensembles de candidats $C \leftarrow F \times F$;
11: **for** $S \in C$ **do**
12: **if** $support(S) \geq support_{min}$ **then**
13: $F_{plus} \leftarrow F_{plus} \cup S$;
14: **end if**
15: **end for**
16: $F \leftarrow F_{plus}$;
17: $Result \leftarrow Result \cup F_{plus}$;
18: **end while**
19: **Return** *Result*

k	C_k	support	F_k
1	$\langle T \rangle$	5	$F_1 = \{\langle T \rangle, \langle C \rangle, \langle Sp \rangle\}$ $Res = \{\langle T \rangle, \langle C \rangle, \langle Sp \rangle\}$
	$\langle C \rangle$	5	
	$\langle Sp \rangle$	4	
	$\langle S \rangle$	2	
	$\langle L \rangle$	2	
2	$\langle T, T \rangle$	0	$F_2 = \{\langle T, C \rangle, \langle T, Sp \rangle, \langle C, Sp \rangle\}$ $Res = \{\langle T \rangle, \langle C \rangle, \langle Sp \rangle, \langle T, C \rangle, \langle T, Sp \rangle, \langle C, Sp \rangle\}$
	$\langle T, C \rangle$	4	
	$\langle T, Sp \rangle$	4	
	$\langle C, T \rangle$	1	
	$\langle C, C \rangle$	0	
	$\langle C, Sp \rangle$	4	
	$\langle Sp, T \rangle$	0	
	$\langle Sp, C \rangle$	0	
	$\langle Sp, Sp \rangle$	0	
3	$\langle T, C, Sp \rangle$	3	$F_3 = \{\langle T, C, Sp \rangle\}$ $Res = \{\langle T \rangle, \langle C \rangle, \langle Sp \rangle, \langle T, C \rangle, \langle T, Sp \rangle, \langle C, Sp \rangle, \langle T, C, Sp \rangle\}$

TABLE 2.3 – Déroulement de l'algorithme *GSP* pour l'Exemple 1.1

De nombreuses améliorations de *GSP* ont été proposées pour améliorer le temps de calcul. L'algorithme *PSP* (*Prefix Tree for Sequential Pattern*) [Masseglia et al., 1998] améliore *GSP* en exploitant une structure de données intermédiaire arborescente pour stocker et indexer les motifs. *GSP* utilise des tables de hachage à chaque nœud intermédiaire de l'arbre des candidats, alors que l'algorithme *PSP* organise les candidats dans un arbre de préfixes pour un comptage efficace. *MFS* (*Maximal Frequent Sequence*) [Zhang et al., 2001] est également basé sur *GSP* mais vise à réduire le coût d'accès à l'ensemble de données. Il extrait un ensemble réduit de l'ensemble de données avec *GSP* et l'utilise pour générer des candidats. Ensuite il conserve les motifs fréquents maximaux et réitère le même traitement sur cet ensemble. Le processus s'arrête lorsqu'aucun motif séquentiel ne peut être généré. *MSPS* (*Maximal Sequential Patterns using Sampling*) [Luo and Chung, 2008] utilise des échantillons multiples pour exclure efficacement les candidats peu fréquents. *MSPS* adopte la méthode de génération de candidats de *GSP*. Il utilise un arbre de préfixes pour stocker les candidats comme *PSP*.

3.2 Les algorithmes basés sur *Apriori* : Représentation Verticale

Les algorithmes qui utilisent une représentation verticale évitent les multiples passages sur l'ensemble de données pour calculer le support. Avec une représentation de données de séquences verticale, chaque entrée représente un élément et indique la liste des séquences où l'élément apparaît et sa position dans la séquence. La Table 2.4 est la représentation verticale de l'Exemple 1.2.

Zaki est le premier à avoir utilisé une représentation verticale pour la fouille de motifs

Id de séquence	T	C	S	Sp	L
1	1	3	2	4	5
2	1	2		3	
3	1	2			3
4	2	1		3	
5	1	2	3	4	

TABLE 2.4 – La représentation verticale de l'ensemble de données séquence de l'Exemple 1.2.

séquentiels fréquents avec l'algorithme *SPADE* [Zaki, 2001]. À partir de la représentation verticale de l'ensemble de données, une structure nommée Id-List peut être associée à chaque motif. Les Id-Lists permettent de calculer le support d'un motif rapidement en faisant des opérations de jointure avec les Id-Lists des motifs plus petits. Au plus, trois passes sur l'ensemble de données sont suffisantes pour identifier les motifs séquentiels fréquents. La Figure 2.2 illustre la construction des *Id-listes*. Les *Id-listes* des motifs séquentiels de taille 1 sont construites en premier. Ensuite, les *Id-listes* de taille k sont générées à partir des *Id-listes* de taille $k-1$ avec des opérations de jointures. Le calcul de support devient donc facile : c'est la taille de l'*Id-liste* de motif séquentiel. Et à chaque niveau l'algorithme considère seulement les motifs séquentiels s avec $support(s) \geq support_{min}$.

Le principal avantage de *SPADE* par rapport à *GSP* est l'utilisation des *Id-listes* pour les jointures. Ces listes sont beaucoup plus petites et rapides à générer ce qui permet d'améliorer grandement le temps de calcul. Cependant, *SPADE* nécessite un grand espace mémoire pour transformer l'ensemble de données en une représentation verticale.

SPAM (*Sequential PAttern Mining algorithm*), proposé par Ayres [Ayres et al., 2002], est un algorithme de recherche en profondeur avec une représentation verticale de l'ensemble de données en tableau de bits (*bitmap*). Il nécessite un tableau de bits pour chaque élément représentant les séquences où l'élément est apparu. Cela peut être très coûteux en mémoire, car il faut que l'ensemble de données soit entièrement chargé en memoire. L'algorithme *CCSM* (*Cache-based Constrained Sequence Miner*) recherche des motifs séquentiels par niveau et adopte une méthode basée sur l'intersection pour déterminer le support des k-séquences candidates en utilisant un cache efficace qui stocke les *Id-listes* intermédiaires. *LAPIN-SPAM* (*LAst Position INduction Sequential PAttern Mining*) [Yang and Kitsuregawa, 2005] est une approche qui repose sur le même principe que *SPAM* [Ayres et al., 2002] en exploitant les dernières positions des éléments.

3.3 Les algorithmes de croissance de motifs

Les algorithmes de croissance de motifs utilisent un arbre de préfixe. Les nœuds de l'arbre sont des préfixes où on ajoute à chaque nœud des extensions. C'est toujours le même principe consistant à trouver des motifs par niveaux. *PrefixSpan* (*Prefix-projected Sequential pattern mining*) [Pei et al., 2001] est un algorithme représentatif de cette

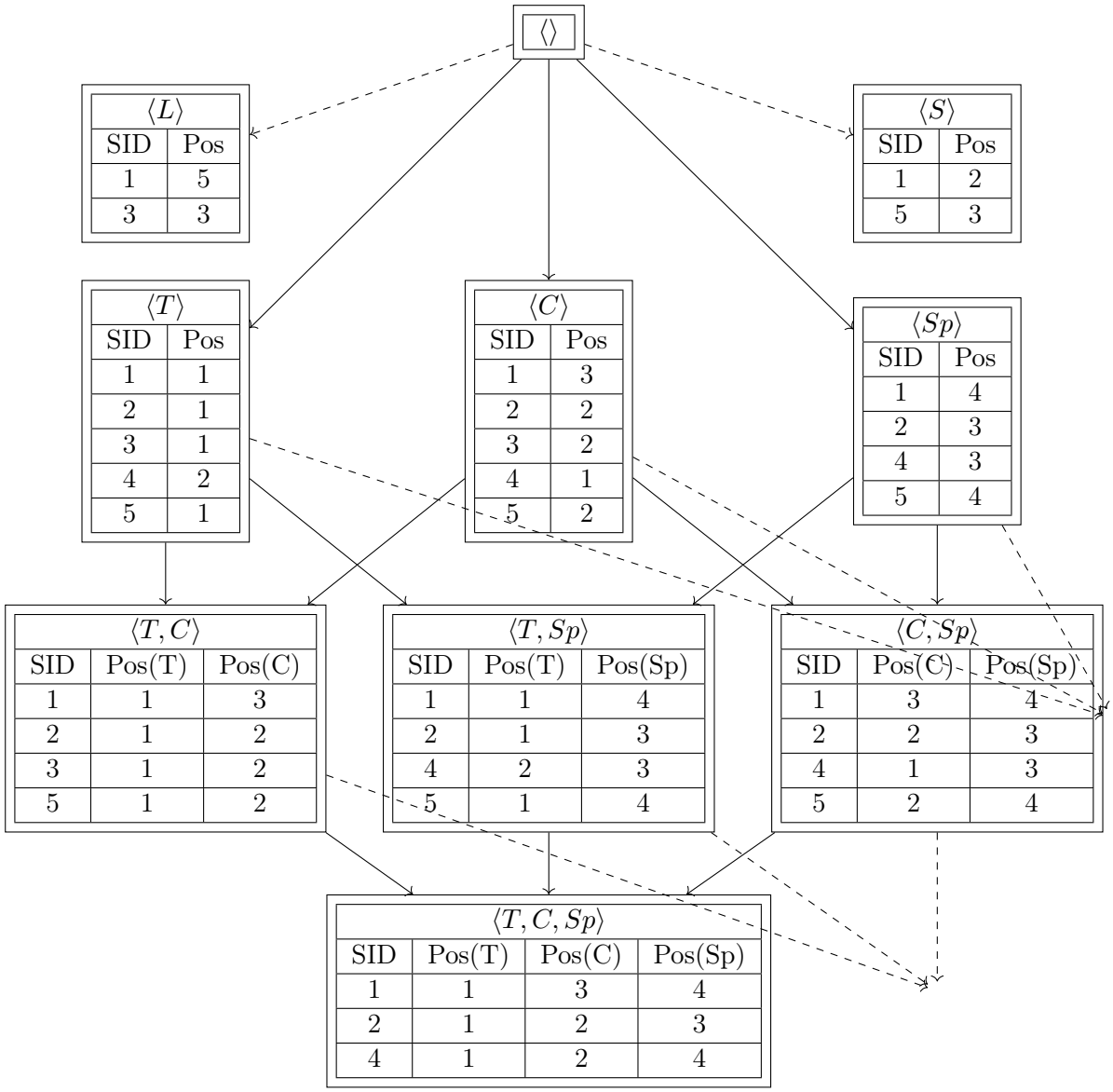


FIGURE 2.2 – La construction des *Id-listes* avec l'algorithme *SPADE* pour l'Exemple 1.1

catégorie, il utilise une représentation projetée de données. À chaque nœud de l'arbre, l'algorithme construit une représentation projetée qui contient des extensions de motifs du nœud actuel.

Soit S un motif séquentiel dans un ensemble de séquences G. La représentation projetée par rapport à S, notée $G|_S$, est la collection des suffixes des séquences dans G par rapport au préfixe S. L'algorithme construit donc les représentations projetées de motifs séquentiels S de taille 1. Ensuite, il parcourt l'ensemble de données projetées pour trouver

des motifs S' tel que l'ajout de S' à S forme un motif séquentiel fréquent. L'algorithme répète récursivement cette opération jusqu'à la découvert de tous les motifs séquentiels.

Prenons l'Exemple 1.1, la Figure 2.3 montre l'arbre de préfixes. Les nœuds en pointillé représentent des nœuds où le support est inférieur au support minimal. La table 2.5 décrit les représentations projetées pour les nœuds des motifs séquentiels fréquents.

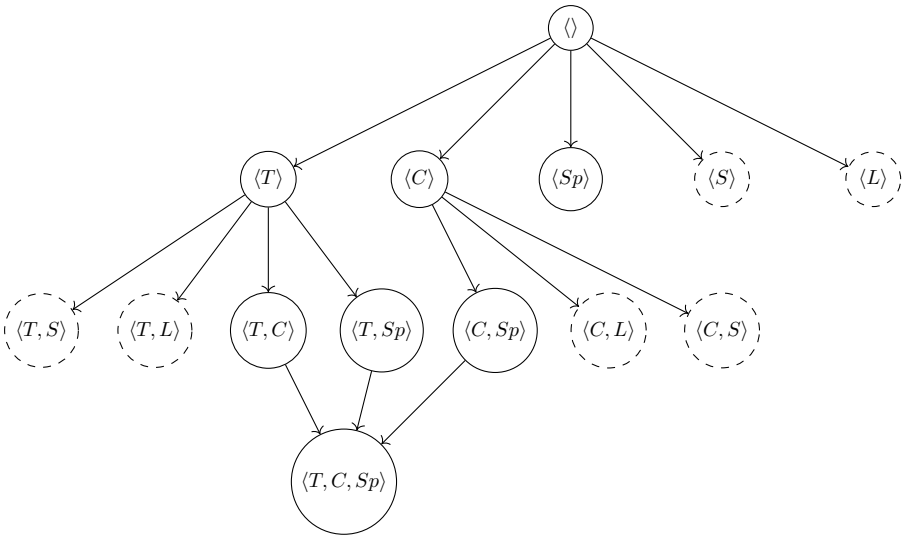


FIGURE 2.3 – L'arbre de préfixe construit par *PrefixSpan* pour l'Exemple 1.1

α	$G\|_\alpha$
$\langle T \rangle$	$\{\langle S, C, Sp, L\rangle, \langle C, Sp\rangle, \langle C, L\rangle, \langle Sp\rangle, \langle C, S, Sp\rangle\}$
$\langle C \rangle$	$\{\langle Sp, L\rangle, \langle Sp\rangle, \langle L\rangle, \langle T, Sp\rangle, \langle S, Sp\rangle\}$
$\langle Sp \rangle$	$\{\langle L\rangle, \langle\rangle, \langle\rangle, \langle\rangle\}$
$\langle T, C \rangle$	$\{\langle Sp, L\rangle, \langle Sp\rangle, \langle L\rangle, \langle S, Sp\rangle\}$
$\langle T, Sp \rangle$	$\{\langle L\rangle, \langle\rangle, \langle\rangle, \langle\rangle\}$
$\langle C, Sp \rangle$	$\{\langle L\rangle, \langle\rangle, \langle\rangle, \langle\rangle\}$
$\langle T, C, Sp \rangle$	$\{\langle L\rangle, \langle\rangle, \langle\rangle, \langle\rangle\}$

TABLE 2.5 – Les représentations projetées pour les motifs fréquents de l'Exemple 1.1

3.4 Approches par contraintes

Pour réduire le nombre de motifs séquentiels extraits et trouver des motifs plus intéressants, les chercheurs ont proposé d'intégrer des contraintes dans le processus d'exploration de motifs séquentiels. Les contraintes sont essentielles pour de nombreuses applications. Elles sont utilisées comme filtre pour réduire le nombre de motifs. Les contraintes de motifs aident à extraire les motifs vérifiant des contraintes d'entrée telles que la taille maximale d'une sous-séquence, la vérification d'une expression régulière,

un écart entre les éléments de la séquence ou un sous-modèle donné [Bonchi et al., 2006, Ugarte et al., 2017].

4 Fouille de motifs séquentiels fermés

Avec la puissance des machines et l'évolution des techniques de fouille de données, les algorithmes de fouille de motifs séquentiels deviennent plus efficace même avec des ensembles de données volumineux. Le problème réside cependant dans le nombre de motifs fréquents qui est trés élevé, particulièrement quand on est face à des séquences denses, et à un support bas. De plus, on a une redondance d'information lorsque deux motifs sont de même support et l'un est inclu dans l'autre. Dans l'Exemple 1.1, le motif $\langle Sp \rangle$ a un support de 4, et de même le motif $\langle T, Sp \rangle$. Ici l'information que $\langle Sp \rangle$ se produit 4 fois dans l'ensemble de données est redondante. Pour répondre à la problématique du déluge de motifs séquentiels et de la redondance d'information extraite, des algorithmes ont été proposés permettant la réduction du nombre de motifs tout en décrivant la même information. Pour faire face à cette problématique, les chercheurs ont introduit la fouille de motifs séquentiels fermés.

Un motifs séquentiels fermés est une séquence qui n'a pas de super-séquence avec le même support. Comme pour les *itemsets*, ces motifs séquentiels fermés sont les motifs maximaux des *classes d'équivalences* dans le treillis de tous les motifs séquentiels. Des algorithmes de fouille de motifs séquentiels fermés sont apparus comme *CloSpan* [Yan et al., 2003], *ClaSp* [Gomariz et al., 2013] et *BIDE* [Wang and Han, 2004].

Formellement, un motif séquentiel fermé est défini de la façon suivante :

Définition 2.3 (Motif séquentiel fermé)**.** Un motifs séquentiel S est fermé, si et seulement si $\nexists S'$ tel que $S \sqsubset_s S'$ et $support(S) = support(S')$.

L'algorithme *CloSpan* [Yan et al., 2003] est basé sur *PrefixSpan* [Pei et al., 2001], il utilise une représentation projetée de données, mais limite l'extraction de motifs pour les motifs séquentiels fermés fréquents. L'élément principal qu'apporte *CloSpan* par rapport à *PrefixSpan* est qu'il élague l'espace de recherche pour éviter de parcourir les branches inutiles de l'arbre, i.e, les motifs qui ne sont par fermés dans l'arbre de recherche. Pour cela, *CloSpan* utilise un arbre lexicographique des séquences [Ayres et al., 2002] defini à partir d'un ordre lexicographique entre les éléments de l'alphabet. On se basant sur cet ordre, l'ordre lexicographique des séquences est définie pour deux séquences $S = \langle s_1, s_2 ... s_n \rangle$ et $R = \langle r_1, r_2 \dots r_m \rangle$ avec, $S <_{lex} R$ si seulement si :

- $S = R(1, k)$ avec $k < m$, ou
- pour un entier l, $1 \leq l \leq min\{n, m\}$, nous avons $s_i =_{lex} r_i$ pour $1 \leq i < l$ et $s_l <_{lex} r_l$.

Par exemple, $\langle a, b, e, f \rangle <_{lex} \langle a, b, e, f, h, i \rangle$ et $\langle a, b, c \rangle <_{lex} \langle a, d, b \rangle$.

L'arbre lexicographique des séquences est construit, avec chaque nœud de l'arbre représente une séquence, en commençant avec la séquence vide $\langle\rangle$. Les fils d'un noeud

sont construits en ajoutant un élément de l'alphabet. Et le frère à gauche est inférieur au frère à droite selon l'ordre lexicographique des séquences. La Figure 2.4 montre l'arbre de *PrefixSpan* représenté comme arbre lexicographique des séquences.

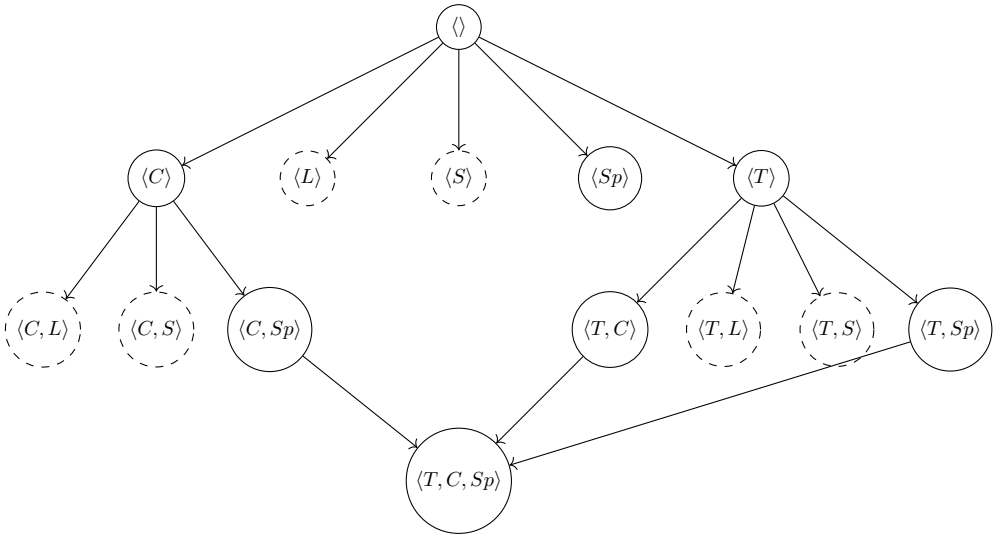


FIGURE 2.4 – L'arbre lexicographique des séquences de la Figure 2.3

Pour élaguer l'espace de recherche, *CloSpan* utilise deux techniques : le *Backward Sub-Pattern* et le *Backward Super-Pattern*.

Backward Sub-Pattern. le *Backward Sub-Pattern* est défini pour deux séquences $S <_{lex} S'$, et $S' \sqsubset_s S$. Si la représentation projetée de S correspond à la représentation projetée de S', on stoppe la recherche des sous-séquences dans la branche de S' dans l'arbre lexicographique des séquences. Par exemple, comme $C <_{lex} Sp$, on considère C avant Sp et on a $G|_{\langle Sp\rangle} = G|_{\langle C,Sp\rangle}$. Donc, pour toute séquence $\langle Sp + \alpha\rangle$ avec α une séquence ajoutée par extension, on aura toujours $G|_{\langle Sp+\alpha\rangle} = G|_{\langle C,Sp+\alpha\rangle}$. On peut conclure alors qu'il n'est pas utile de parcourir la branche de $\langle Sp\rangle$ dans l'arbre. On conserve alors seulement les résultats issus de la branche de C

Backward Super-Pattern. le *Backward Super-Pattern* est défini pour deux séquences $S <_{lex} S'$, et $S \sqsubset_s S'$. Si la représentation projetée de S correspond à la représentation projetée de S', on stoppe la recherche des sous-séquences dans la branche de S' dans l'arbre lexicographique des séquences. Par exemple, comme $Sp <_{lex} T$, on considère Sp avant T et on a $G|_{\langle Sp\rangle} = G|_{\langle T,Sp\rangle}$. Donc, pour toute séquence $\langle Sp+\alpha\rangle$, on aura toujours $G|_{\langle Sp+\alpha\rangle} = G|_{\langle T,Sp+\alpha\rangle}$. On peut conclure alors qu'il n'est pas utile de parcourir la branche de $\langle T, Sp\rangle$ dans l'arbre. On ne conserve alors que les résultats issus de la branche de Sp auxquels on rajoute T au début.

Pour implémenter ces deux techniques de manière efficace *CloSpan* s'appuie sur le théorème suivant :

Théorème 2.1. *Soient S et S' deux séquences telles que $S \sqsubseteq_s S'$. Alors on a : $G|_S = G|_{S'} \Leftrightarrow I(G|_S) = I(G|_{S'})$, où $I(G|_S)$ représente le nombre d'éléments dans $G|_S$.*

La Figure 2.5 illustre l'élagage de *CloSpan* avec le branchement dans l'arbre par *Backward Sub-Pattern* et *Backward Super-Pattern.*

Backward Sub-Pattern Backward Super-Pattern

FIGURE 2.5 – L'arbre lexicographique des séquences

L'algorithme *CloSpan* comporte deux étapes. D'abord, il génère un ensemble de candidats LS qui est un sous-ensemble de tous les motifs séquentiels fréquents FS et un super-ensemble de l'ensemble des motifs séquentiels fermés fréquents CS, $CS \subseteq LS \subseteq FS$. La génération de l'ensemble LS est possible par les deux techniques d'élagage présentées. Ensuite, dans une étape de post traitement, il élimine les motifs séquentiels non fermés.

L'algorithme 6 présente un prétraitement de *CloSpan.* Il construit l'ensemble des motifs $1-$*fréquents* S_1 (ligne 2), ensuite pour chaque élément de S_1, il appelle *CloSpan* decrit par l'algorithme 7. À la fin, il élimine des résultats les motifs non fermés. Pour cela, il utilise la technique de [Zaki and Hsiao, 2002] avec une table de hachage où la fonction de hachage est le support de la séquence. Pour chaque séquence S, il trouve les séquences avec même support. Ensuite, il effectue un test de sous-séquence pour déterminer si la séquence est fermée (si $support(S) = support(S')$ et $S \sqsubseteq_s S'$ alors S est non fermée).

À partir de la représentation projetée de données $G|_S$ d'un motif séquentiel S, l'algorithme *CloSpan* vérifie la présence d'un motif S' avec de possibles *Backward Sub-Pattern/Super-Pattern* (ligne 2 de l'Algorithme 7). Cette vérification est possible avec une table de hachage de la taille de $G|_S$, donc seules les séquences de même taille que $G|_S$ sont conservées. S'il existe un tel motif la recherche sur cette branche est stoppée (lignes 4, 5), sinon S est ajouté à l'arbre (ligne 7). Ensuite, S est étendu avec les éléments fréquents par appel récursif (lignes 9-13).

Nous avons présenté à travers cette section les principaux algorithmes de fouille de motifs séquentiels pour les séquences simples, avec plusieurs améliorations qui ont permis la réduction de temps de calcul. Nous avons vu que ces algorithmes ont rencontré le problème de déluge de motifs séquentiels. Les algorithmes de fouille de motifs séquentiels fermés permettent de réduire le nombre de motifs extraits. Cependant, le deluge de motifs reste encore une problématique majeure. Par la suite, nous explorons l'intégration de l'information temporelle qui permet également de réduire le nombre de motifs générés.

Algorithm 6 ClosedMining

Input : Un ensemble de séquences G et un seuil minimal $support_{min}$
Output : LS : Tous les motifs séquentiels fermés fréquents dans G

$LS \leftarrow \emptyset$
$S_1 \leftarrow$ tous les motifs 1-fréquents.
for $s \in S_1$ **do**
    $CloSpan(s, G|_s, support_{min}, LS)$;
end for
// Éliminer les séquences non fermées de LS
for $(S, S') \in LS$ **do**
    if $S \sqsubseteq_s S'$ **then**
        $DeleteS$
    end if
end for

Algorithm 7 *CloSpan*

Input : Un ensemble de séquences G et un seuil minimal $support_{min}$
Output : Tous les motifs séquentiels fermés fréquents dans G

Vérifié la présence de S' tel que $S \sqsubseteq_s S'$ ou $S' \sqsubseteq_s S$ et $I(G|_S) = I(G|_{S'})$;
▷ $I(G|_{S'})$ est le nombre d'items dans $G|_S$.
if S' exist **then**
    modifier le lien de L ▷ Backward (sub/super pattern)
else
    insert S in L ;
end if
for $S' \in G|_S$ **do**
    if S peut être étendu à $S + S'$; **then**
        $CloSpan(S + S', G|_{S+S'}, support_{min}, L)$;
    end if
end for

Nous allons maintenant décrire les algorithmes de fouille de motifs séquentiels temporels, et les possibilités et les avantages qu'ils proposent.

5 Fouille de motifs temporels

La fouille de motifs temporels consiste à extraire des motifs séquentiels en tenant compte de l'information temporelle. Dans le cas où nous avons des séquences temporelles représentant un déplacement, on extrait par exemple un motif $\langle lieu_1, lieu_2 \rangle$ sans aucune information sur l'écart temporel entre les deux événements de visite des lieux. Il se peut qu'une personne ait visité le $lieu_1$, puis après 2 heures soit passé par le $lieu_2$, tandis qu'une autre les a visités durant deux mois différents. Sans information temporelle la sous-séquence $\langle lieu_1, lieu_2 \rangle$ est commune aux deux personnes mais la visite des lieux n'est pas forcément liée temporellement. L'utilisation de la temporalité dans l'extraction de motifs séquentiels permet de guider l'analyse selon une proximité temporelle en plus de la notion de sous-séquence. La définition de l'écart entre les événements permet d'éviter la génération des motifs inutiles avec des écarts importants non significatifs. L'information temporelle permet donc l'expressivité des motifs et limite le déluge de motifs.

Plusieurs approches permettent l'extraction de motifs temporels. On peut les distinguer en 4 catégories : la fouille de motifs séquentiels avec des contraintes temporelles [Masseglia et al., 2009, Bonchi et al., 2006], la fouille d'épisodes fréquents [Mannila et al., 1997] et la découverte de motifs temporels ou chroniques (*chronicles*) [Dousson and Duong, 1999] et la fouille de motifs *delta* ($\delta - pattern$) [Yoshida et al., 2000]. .

5.1 Fouille de motifs séquentiels avec contraintes temporelles

Le cas le plus trivial pour traiter la temporalité est l'utilisation des contraintes comme filtre pour réduire le nombre de motifs. Les contraintes de motifs aident à extraire les motifs vérifiant des contraintes d'entrée telles que la taille maximale d'une sous-séquence, une expression régulière, un écart entre les éléments de la séquence, un sous-modèle donné [Bonchi et al., 2006, Ugarte et al., 2017].

GSP [Agrawal and Srikant, 1995] est le premier algorithme introduisant la notion de contraintes avec la définition d'écart minimal et maximal (*gaps*) et de fenêtre (*window*). Dans un état de l'art complet sur les séquences [Pei et al., 2007], les auteurs ont défini sept types de contraintes. Nous nous intéressons dans cette partie aux contraintes temporelles.

Une contrainte temporelle est définie principalement dans les ensembles de séquences temporelles. Elle se définit lorsque la différence d'horodatages entre le premier et le dernier élément d'un motif séquentiel est supérieur ou inférieur à une durée donnée. Dans certaines autres applications, l'écart entre les éléments adjacents d'un motif séquentiel peut être important. Pour déterminer si un motif séquentiel satisfait à ces contraintes, il faut examiner l'ensemble de données. [Parthasarathy et al., 1999] propose une approche incrémentale et interactive. L'idée est de mettre en cache les résultats précédemment extraits qui peuvent être utilisés pour répondre aux requêtes des utilisateurs beaucoup plus rapidement que relancer le traitement à chaque nouvelle requête. Zaki propose

cSPADE [Zaki, 2000b], une extension de son algorithme SPADE [Zaki, 2001] mais avec la prise en considération des contraintes, comme la taille des motifs, l'écart minimal et maximal entre deux éléments consécutifs du motif, la définition d'une fenêtre, etc. Il intègre les contraintes dans le processus de fouille et non pas comme un post-traitement. [Pei et al., 2002] proposent l'algorithme *PrefixGrowth* qui prend en compte des contraintes monotones et anti-monotones. [Lee and De Raedt, 2004] introduisent l'algorithme SeqLog pour la fouille de motifs temporels avec des contraintes. Il est basé sur la logique de premier ordre en étendant les predicats aux séquences.[Orlando et al., 2004] introduisent un nouvel algorithme par niveau *CCSM* (*Cache-based Constrained Sequence Miner*), qui prend en compte des contraintes d'écart. La notion de "*k-way intersection*" est introduite pour calculer le support des candidats. Comme *SPADE*, il utilise des *Id-listes* mais avec un système de cache pour réduire le nombre de jointures. [Lin and Lee, 2005] proposent *DELISP* (*DELImited Sequential Pattern*), un algorithme de croissance des motifs. L'algorithme réduit l'ensemble de données projetées par des écarts de temps et une fenêtre glissante. Cette technique génère directement des motifs respectant les contraintes et améliore le temps d'exécution. Masseglia et al, proposent l'algorithme *GTC* pour la fouille de motifs dans des grosses ensembles de données. L'idée est d'utiliser les contraintes temporelles au cours du processus de fouille [Masseglia et al., 2009]. Récemment Kryszkiewicz propose une amélioration de *GSP* [Kryszkiewicz and Skonieczny, 2019] qui consiste à être plus sélective dans le choix des nœuds à visiter lors du parcours de l'arbre contenant les candidats.

5.2 Fouille d'épisodes fréquents

Dans [Mannila et al., 1995] le but est d'extraire des épisodes fréquents dans une longue séquence d'éléments. Mannila a défini l'épisode comme une collection d'événements (ou éléments) qui apparaissait relativement proches [Mannila et al., 1995]. Les épisodes, sont des ensembles d'éléments partiellement ordonnés. Ils peuvent être définie comme un graphe orienté acyclique :

Définition 2.4. Un épisode $\varphi = (V, \leq, g)$ est défini par un ensemble V de nœuds, un ordre partiel $\leq$ sur V et une correspondance $g : V \to \Sigma$ qui associe chaque nœud à un élément de l'alphabet.

L'utilisateur définit une fenêtre et un support minimal. La fenêtre correspond à l'intervalle où les éléments de l'épisode doivent apparaître et le support minimal aux épisodes fréquents. Le calcul du support est différent dans ce cas : le support d'un épisode est le nombre de fenêtres dans lesquelles l'épisode se produit. Deux fenêtres peuvent se chevaucher, mais elles ont une taille fixe définie par l'utilisateur. Il existe deux types d'épisodes : épisodes en série et épisodes en parallèle. Les épisodes en série sont des sous-séquences car les éléments dans l'épisode sont ordonnés. Tandis que les épisodes en parallèle ne présentent aucune contrainte sur l'ordre des éléments de l'épisode.

Exemple 2.2. *La Figure 2.6 (a) donne un exemple de ce type de motifs et de fenêtre, ici, la séquence est* $s = \langle (1, A), (1.5, C), (3, B), (4, A), (4.5, C), (5, D), (5.5, B), (6, C), (7, B) \rangle$.

La partie (b) de la Figure 2.6 représente un épisode en série, qui peut être vue comme une séquence. La partie (c) représente un épisode en parallèle.

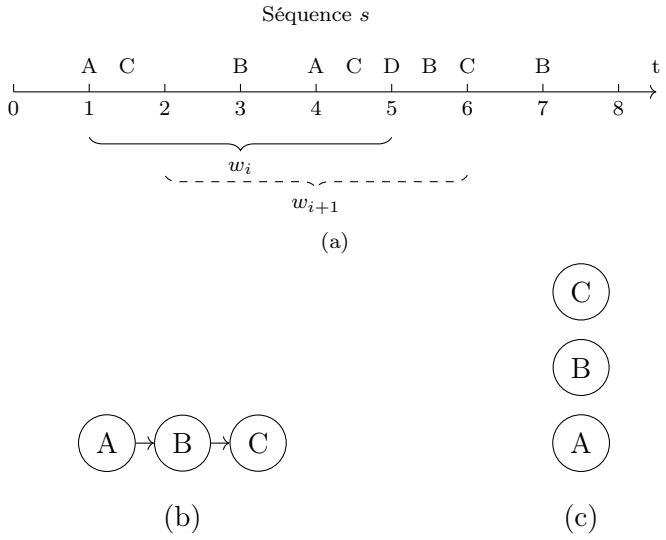


FIGURE 2.6 – Exemple de séquence temporelle et épisodes fréquents

L'algorithme *Winepi* [Mannila et al., 1995] applique *Apriori* pour trouver des épisodes fréquents en considérant la fenêtre glissante comme séquence d'événements. *Minepi* [Mannila et al., 1997] est basé sur les occurrences minimales de l'épisode dans la séquence. Casas-Garriga propose une méthode permettant d'étendre la fenêtre automatiquement au cours du processus de fouille en utilisant un écart maximal entre deux événements consécutifs dans l'épisode [Casas-Garriga, 2003]. Meger propose d'étendre les travaux de Casas-Garriga avec un algorithme complet pour extraire des règles d'épisodes vérifiant un support minimal, une confiance et une contrainte d'écart maximal [Méger and Rigotti, 2004]. Laxman présente un algorithme de fouille d'épisodes fréquents avec un calcul de support basé sur les occurrences non superposées (*non-overlapped*) [Laxman et al., 2007]. [Tatti, 2009] propose une mesure de qualité pour raffiner un ensemble d'épisodes avec une étape de post-traitement pour éliminer les épisodes non intéressants. [Tatti, 2015] divise l'épisode en deux sous-épisodes consécutifs et minimise l'importance de cet épisode si le support observé peut être expliqué par les deux sous-épisodes. [Wu et al., 2013] propose l'algorithme *UP-Span*, pour la fouille d'épisodes utiles par des stratégies d'élagage pour réduire l'espace de recherche. Des travaux ont cherché à réduire le nombre d'épisodes extraits, comme les épisodes stricts [Tatti and Cule, 2012], les épisodes injectifs [Achar et al., 2012] et les épisodes en séries [Ao et al., 2015, Ao et al., 2017, Achar et al., 2013].

5.3 Fouille de chroniques fréquentes

Dousson et al, [Dousson et al., 1993] introduit les chroniques dans le cadre de détection d'alarmes où un délai entre deux éléments de la séquence est représenté par un intervalle,

le problème est alors de découvrir toutes les chroniques dans un ensemble de séquences. On les retrouve aussi avec la notion de problème de satisfaction de contraintes temporelles (*Temporal Constraint Satisfaction Problem*) [Dechter et al., 1991]. La découverte de chroniques est devenue par la suite un problème scientifique majeur. Elle est un moyen d'aider les experts à concevoir des représentations du comportement d'un système à partir d'observations directes [Dousson and Duong, 1999]. Les chroniques sont utilisées dans les domaines de trajectoires médicales [Dauxais et al., 2017], les services web [Pencolé and Subias, 2009], les alarmes [Morin and Debar, 2003], etc.

Une chronique est un graphe ou les nœuds sont des éléments ou des événements et les arêtes représentant des contraintes temporelles.

Définition 2.5 (Chronique)**.** Une chronique $\mathcal{C}$ est un graphe $(\mathcal{E}, \mathcal{T})$, ou $\mathcal{E}$ est un ensemble d'éléments de la forme $\{e_1, ..., e_n\}_{(1 \leq i \leq n)}$ et $\mathcal{T}$ est un ensemble de contraintes temporelles $\mathcal{T} = \{t_{ij}\}_{1 \leq i \leq j \leq n}$.

La taille d'une chronique est la cardinalité de $\mathcal{E}$, ici n.

La Figure 2.7 représente une chronique modélisant une partie de la séquence temporelle, où l'élément B doit apparaître entre 1 et 3 unités de temps après A, et l'élément C doit apparaître après B entre 1 et 5 unités, et après A entre 4 et 6.

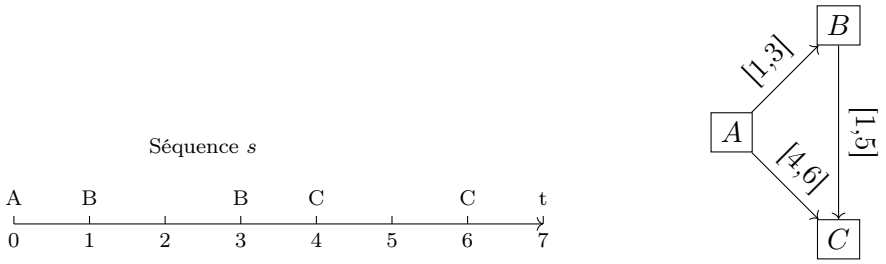


FIGURE 2.7 – Exemple de séquence temporelle et de chronique frequente.

L'algorithme *FACE* [Dousson and Duong, 1999] d'extraction de chroniques adapte des algorithmes classiques de fouille de motifs séquentiels comme *GSP* ou *PrefixSpan* mais cet algorithme n'est pas complet car il extrait seulement une seule chronique.

Cram [Cram et al., 2012] extrait des chroniques d'interactions humaines. Il propose un algorithme complet avec une approche interactive basé sur l'algorithme *APriori*. Cet algorithme demande à l'utilisateur de vérifier le résultat à chaque itération et s'arrête si les chroniques découvertes satisfont l'utilisateur.

ASTPminer (*Apriori simple temporal problem miner*) [Álvarez et al., 2013] étend les travaux de [Álvarez et al., 2011, Álvarez et al., 2010] conçus pour la fouille d'épisodes et de motifs temporels. Il utilise des connaissances préalables du domaine afin de cibler le processus d'extraction. Cela élague l'espace de recherche ; la procédure ne génère que des candidats qui sont des extensions des modèles donnés et le nombre total de candidats est réduit. Les connaissances préalables du domaine sont utilisées pour définir certains paramètres de l'algorithme avant son exécution. L'utilisateur doit fournir l'extension temporelle maximale autorisée pour qu'un motif soit pertinent, la fenêtre temporelle qui

définit la distance entre deux événements pour qu'ils soient considérés comme faisant partie du même motif et un seuil minimal [Álvarez et al., 2013]. Dauxais propose l'algorithme *DCM* (*Discriminant Chronicles Mining*) de fouille de chroniques [Dauxais et al., 2017]. *DCM* n'extrait pas l'ensemble complet des chroniques car il ne les considère pas toutes intéressantes. Les chroniques ayant les même éléments et des contraintes temporelles similaires sont nombreuses et considérées comme redondantes. L'approche proposée permet d'extraire un ensemble incomplet de chroniques discriminantes significatives [Dauxais et al., 2017]. Sahuguède [Sahuguède et al., 2018] propose un algorithme de *clustering* basé sur la densité et est utilisé pour regrouper les motifs en fonction de leur similarité. Il introduit également un ordre entre les motifs.

5.4 Fouille de motifs *delta*

Yoshida [Yoshida et al., 2000] propose la notion de motifs *delta* (*delta pattern*) qui définit un intervalle temporel entre les éléments de la séquence. Il utilise des heuristiques pour extraire ce type de motifs. L'algorithme adopte une méthode avec utilisation d'arbre (*CF-tree*) pour réaliser du regroupement (*clustering*) sur les intervalles de temps [Yoshida et al., 2000]. Un motif *delta* $\mathcal{C} = (\mathcal{S}, \mathcal{T})$ est un couple composé d'une séquence $\mathcal{S} = \{s_1, ..., s_n\}_{(1 \leq i \leq n)}$ et des écarts temporels entre les éléments successifs de la séquence $\mathcal{T} = \{t_{i,i+1}\}_{1 \leq i < n}$.

Des travaux de fouilles de motifs temporels extraient des motifs similaires aux motifs *delta* sont aussi proposés. Hirate et Yamana proposent une généralisation de la fouille de motifs séquentiels avec des intervalles entre les éléments de la séquence. Cela peut être utilisé pour spécifier des mesures temporelles ou des contraintes [Hirate and Yamana, 2006]. De même, des annotations temporelles des séquences ont été introduites par Giannotti dans son algorithme *TAS* (*Temporally-Annotated Sequences*). Les annotations sont des informations temporelles présentées entre les éléments de la séquence [Giannotti et al., 2006]. Yen et Lee [Yen and Lee, 2013] proposent l'algorithme *TGSP* (*mining Time-Gap Sequential Patterns*) qui utilise une technique de *clustering* pour extraire des motifs séquentiels avec un écart temporel entre les éléments.

6 Fouille de motifs séquentiels d'intervalles

Dans la section précédente nous avons vu les séquences temporelles où chaque élément de la séquence se produit à un moment donné. Dans les applications du monde réel, les événements peuvent aussi persister dans le temps ou dans un intervalle de temps. Nous appelons ces séquences des séquences basées sur des intervalles ou des séquences d'intervalles.

La plupart des algorithmes de fouille de motifs séquentiels d'intervalles [Guyet and Quiniou, 2011, Winarko and Roddick, 2007, Kam and Fu, 2000, Moskovitch and Shahar, 2015, Höppner and Klawonn, 2002, Patel et al., 2008] utilisent les relations d'Allen [Allen, 1981]. L'algèbre d'intervalles d'Allen [Allen, 1981] est un ensemble de 7 relations entre des intervalles *(Precedes, Meets, Overlaps, Starts, During, Finishes, Equal)* présentées

dans la Figure 2.8.

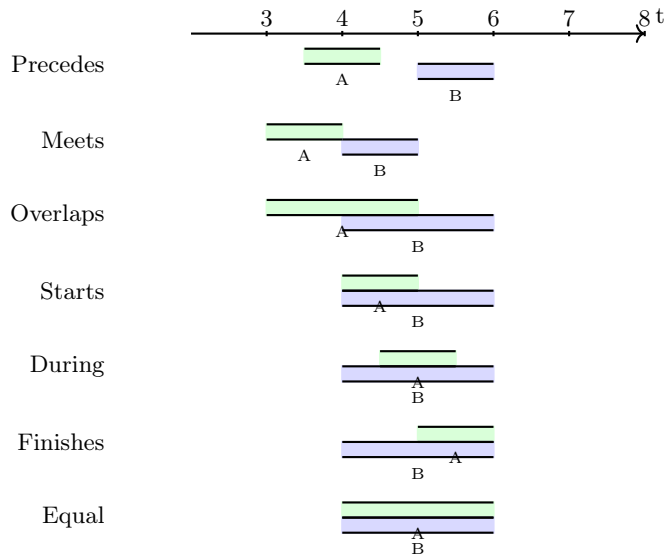


FIGURE 2.8 – Les sept relations d'Allen

Kam et Fu proposent la découverte de séquences d'intervalles [Kam and Fu, 2000]. Les motifs extraits sont du type "*le temps d'occurrence de l'événement A chevauche celui de l'événement B et ces deux événements se produisent avant l'apparition de l'événement C*". Tous comme Höppner [Höppner and Klawonn, 2002], ils étendent *GSP* [Agrawal and Srikant, 1995] pour extraire des motifs d'intervalles avec les relations d'Allen. Wu et Chen proposent *TPrefixSpan* [Wu and Chen, 2007] pour répondre aux problèmes d'ambiguïtés des relations temporelles entre les éléments de la séquence. *QTempIntMiner* [Guyet and Quiniou, 2008] et *QTPSpan* [Nakagaito et al., 2009] prend en compte des séquences d'intervalles. *QTempIntMiner* représente les séquences d'intervalles par des hypercubes et étend *GSP* avec un algorithme de regroupement de séquences temporelles pour extraire des intervalles [Guyet and Quiniou, 2008]. *QTPSpan* est basée sur *QFIMiner* [Washio et al., 2007] qui extrait des motifs avec des contraintes temporelles élaborées par le regroupement des valeurs des intervalles. *QTIPrefixSpan* [Guyet and Quiniou, 2011] étend *PrefixSpan* avec une étape de regroupement multidimensionnel des intervalles pour extraire les intervalles temporelles représentatifs associés aux événements dans les motifs. Winarko et al. [Winarko and Roddick, 2007] proposent *ARMADA*, un algorithme basé sur les algorithmes de fouille de séquences classiques. [Chen et al., 2010] proposent l'algorithme *CTMiner* qui utilise une stratégie d'incision ; il segmente les intervalles dans des tranches disjointes pour simplifier les relations complexes entre les intervalles, les relations entre les tranches sont simplement les deux relations *(Precedes, Equal)*. L'algorithme propose également la représentation de coïncidence pour enlever l'ambiguïté des séquences. L'algorithme *CTMiner* adopte la projection de *PrefixSpan* sur l'ensemble de données transformé dans la représentation de coïncidence. Papapetrou et al. [Papapetrou

et al., 2009] proposent l'algorithme *Hybrid-DFS* pour extraire des arrangements fréquents d'intervalles temporels. La séquence est transformée en une représentation verticale (*Id-listes*). Ensuite des jointures sont réalisées pour générer des motifs temporels. Moskovitch introduit l'algorithme *KarmaLego* [Moskovitch and Shahar, 2015] pour extraire des motifs d'intervalles qu'il nomme *Time Interval Related Patterns (TIRP)*. L'algorithme exploite les relations d'Allen et génère un arbre de *TIRP*. Il extrait en premier lieu tous les motifs fréquents de taille 2, puis les étend récursivement.

Les algorithmes de fouille de motifs d'intervalles sont basés dans leurs majorités sur les algorithmes de fouille de séquences simples (*GSP, PrefixSpan*) et les relations d'Allen avec quelques améliorations d'un algorithme à l'autre.

Les séquences d'intervalles présentées dans la deuxième section avec la contrainte $\underline{t}_i < \overline{t}_i$ correspondent bien à la relation *Precedes* de l'algèbre d'Allen.

7 Conclusion et discussion

Plusieurs algorithmes de fouille de motifs séquentiels fréquents ont été présentés dans ce chapitre, selon qu'il s'agisse de fouiller dans des séquences simples (Section 3), temporelles (Section 5) ou d'intervalles (Section 6). La plupart de ces algorithmes reposent sur la même stratégie que l'algorithme Apriori introduit pour des données binaires (Section 2), à savoir une génération de motifs niveau par niveau.

Les premiers algorithmes tels que *GSP* sont une extension de l'algorithme Apriori adaptée pour les séquences. Ces algorithmes sont coûteux en temps d'exécution et engendrent un déluge de motifs. La manière dont les algorithmes calculent le support joue un rôle important dans la réduction du temps d'exécution. L'algorithme *SPADE* utilise des *Id-listes* et *PrefixSpan* utilise une représentation projetée des données séquentielles. Ces deux approches réduisent le temps d'exécution mais le déluge de motifs reste toujours un problème.

Pour répondre à ce problème, les algorithmes de fouille de motifs séquentiels fermés extraient seulement les motifs fermés fréquents (Section 4). Des algorithmes comme *CloSpan* utilisent deux techniques d'élagage pour détecter les motifs non fermés et arrêter la recherche sur ses branches dans l'arbre de recherche. Mais, même en n'extrayant que des motifs séquentiels fermés, le nombre de motifs résultants reste très élevé surtout avec des jeux de données volumineux et des petits supports. Le déluge de motifs séquentiels reste donc une problématique scientifique majeure.

L'aspect temporel permet également de réduire le nombre de motifs séquentiels résultant en filtrant selon des contraintes temporelles, soit au cours du processus de fouille ou bien en post-traitement pour ne garder que des motifs temporels respectant certaines contraintes. L'utilisation de la temporalité dans la fouille de séquences temporelles diffère selon les données d'entrées, l'application et le type de motif temporel obtenu. Les épisodes, les chroniques, les motifs négatifs ou les motifs d'intervalles sont des exemples de motifs contenant une information temporelle.

Toutes ces approches suivent une technique de génération niveau par niveau pour générer des sous séquences communes qui peuvent également être fermées et maximales.

On trouve un grand nombre d'algorithme qui visent à réduire le temps d'exécution et le nombre de motifs générés. Cependant ces algorithmes sont complexes à apprehender, la problématique de génération par niveau sans doublons et sans "saut de niveau" se mélant à la manipulation complexe des différents types de séquences.

L'analyse formelle de concepts formalise l'approche niveau par niveau avec une structure de treillis définie initialement pour les données binaires et étendue ensuite pour des données complexes (Chapitre 3). L'algorithme NextPriorityConcept et la plateforme `GALACTIC` proposent une extension de l'AFC pour des données complexes et hétérogènes avec des stratégies d'exploration pour limiter le déluge de motifs (Chapitre 4) laissant envisager la possibilité d'une approche simplifiée et générique de fouille de séquences.

Chapitre 3

Analyse Formelle de Concepts

"The analysis of concepts is for the understanding nothing more than what the magnifying glass is for sight."

Moses Mendelssohn

Sommaire

1 Introduction 55
2 Ensemble partiellement ordonné et théorie des treillis 56
2.1 Ensemble partiellement ordonné 56
2.2 Théorie des treillis 57
3 Analyse Formelle de Concepts 60
3.1 Contexte et concept 60
3.2 Treillis des concepts 61
3.3 Connexion de Galois et treillis de fermés 63
3.4 Algorithmes de génération de treillis des fermés 64
4 Extension de l'AFC à des données non binaires 67
4.1 Extensions dédiées à certains types de données 67
4.2 Extensions dédiées aux données complexes 68
4.3 Fouille de séquences et AFC 69
5 Conclusion et discussion 70

1 Introduction

L'Analyse Formelle de Concepts (AFC) est un formalisme introduit par Wille en 1982 [Wille, 1982] qui concerne l'étude et l'analyse des concepts quand ils sont définis dans un contexte donné. Par sa définition dans le dictionnaire Larousse « un concept est une idée générale et abstraite que se fait l'esprit humain d'un objet de pensée concret

ou abstrait, et qui lui permet de rattacher à ce même objet les diverses perceptions qu'il en a, et d'en organiser les connaissances ». L'adjectif *formelle* vient préciser qu'il s'agit d'un type d'analyse qui repose sur des définitions et notions définies de façon théorique et algébrique avec des retombées algorithmiques efficaces qui en favorisent l'analyse.

Si nous parlons de concept et de contexte de manière non formalisée, nous pouvons dire qu'un contexte est un ensemble de données simples présenté sous forme d'objets et de caractéristiques (attributs), avec des relations entre eux. Un concept dans ce contexte est un sous-ensemble d'objets partageant les mêmes caractéristiques ou les mêmes attributs. L'ensemble des concepts possibles forme une hiérarchie qui possède la propriété de treillis. Une propriété qui s'établit grâce à la connexion de Galois entre les objets et leurs attributs permettant de définir un opérateur de fermeture. Cette connexion de Galois s'étend à des descriptions non binaires des objets à condition que l'espace des descriptions possède certaines propriétés ; on parle de treillis de motifs.

Dans ce chapitre nous allons présenter les définitions et notions de l'AFC et leurs extensions à des données non binaires, notamment des séquences. Nous allons commencer dans la deuxième section par des définitions mathématiques de base autour des ensembles partiellement ordonnés et la théorie des treillis. Dans la troisième section, nous détaillons la notion de l'AFC en définissant formellement ce que sont un concept et un contexte. Ensuite, nous présenterons dans la quatrième section les extensions de l'AFC à des données non binaires où nous verrons le lien entre l'AFC et la fouille de séquences. Enfin, nous finissons ce chapitre par une conclusion.

2 Ensemble partiellement ordonné et théorie des treillis

2.1 Ensemble partiellement ordonné

Soit E un ensemble. On appelle relation binaire R sur un ensemble E, toute relation binaire de $E \times E$. Il s'agit d'un ensemble de paires (x, y) inclus dans $E \times E$.

Définition 3.1 (Ensemble partiellement ordonné)**.** Un ensemble partiellement ordonné *(Partially ordered set (poset))* est un couple $(E, \leq)$ où E un ensemble et $\leq$ une relation d'ordre sur E. On dit qu'une relation binaire R est une relation d'ordre si R est réflexive, antisymétrique et transitive pour tout $x,\ y,\ z \in E$:

- réflexivité : xRx
- antisymétrie : xRy et $yRx \Rightarrow x = y$
- transitivité : xRy et $yRz \Rightarrow xRz$

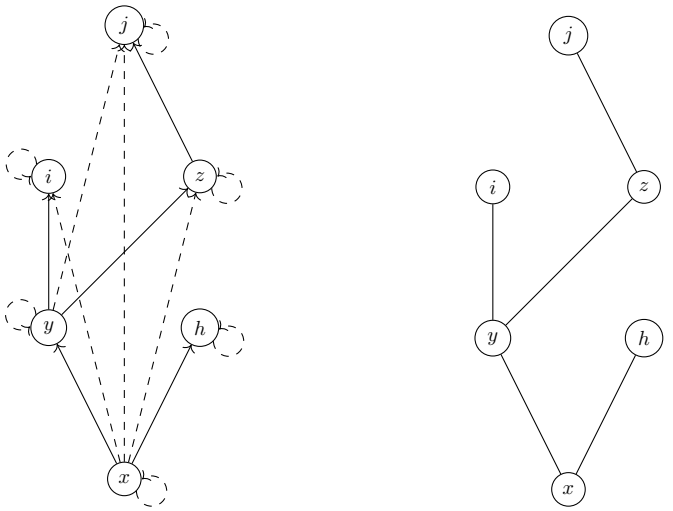
On utilise souvent le symbole $\leq$ pour les relations d'ordre. On écrit $x \leq y$ et on lit "x *est inférieur ou égal à* y". Deux éléments $x, y \in E$ sont *comparables* si $x \leq y$ ou $y \leq x$, sinon, ils sont *incomparables*.

À toute relation d'ordre $\leq$ on associe sa relation d'ordre strict notée $<$ et définie par $x < y$ si $x \leq y$ et $x \neq y$. C'est une relation asymétrique ($x < y \Rightarrow y \not< x$), transitive et irréflexive ($x \not< x$). Elle se déduit de la relation $\leq$ en supprimant la relation de réflexivité.

On associe également à une relation d'ordre sa relation de couverture notée $\prec$, c'est une relation antisymétrique définie par $x \prec y$ tel que $x < y$ et il n'existe pas d'élément z tel que $x < z < y$. On dit alors que y couvre x. Elle se déduit de la relation $\leq$ en supprimant les relations de réflexivité et de transitivité.

Un ensemble ordonné $(E, \leq)$ peut être représenté par un diagramme appelé diagramme de Hasse. Le diagramme de Hasse est une représentation graphique de la réduction transitive et reflexive de l'ordre : on supprime les arcs réflexifs et les arcs de transitivité pour améliorer la lisibilité. Il est possible de retrouver par transitivité les relations initiales. À partir d'un tel diagramme, nous pouvons lire la relation d'ordre $x \leq y$ si et seulement si x est en dessous de y en suivant un chemin.

Exemple 3.1. *La Figure 3.1 représente l'ensemble partiellement ordonné et le diagramme de Hasse pour l'ensemble suivant :* $E = \{x, y, z, i, j, h\}$ *avec* $R = \{(x, y), (x, h) \ldots, etc\}$



(a) Ensemble partiellement ordonné (b) Diagramme de Hasse

FIGURE 3.1 – Exemple de poset 3.1a et son diagramme de Hasse 3.1b

Définition 3.2 (Supremum et Infimum)**.** Soit $(E, \leq)$ un ensemble ordonné et $A \subseteq E$ une partie de E. L'ensemble des majorants de A est l'ensemble $S = \{y \in E \mid \forall x \in A, y \geq x\}$. L'ensemble des minorants de A est l'ensemble $I = \{y \in E \mid \forall x \in A, y \leq x\}$.

S'il existe un plus petit élément dans l'ensemble S des majorants de A, il est appelé le *supremum* ou borne supérieure de A (ou *join*) et on note $sup(A)$ ou bien $\bigvee A$. Dualement, le plus grand élément de l'ensemble des minorants est appelé *infimum* ou borne inférieure de A (ou *meet*), et on note $inf(A)$ ou bien $\bigwedge A$.

2.2 Théorie des treillis

Il existe dans la littérature deux définitions pour un treillis : une définition ordinale et une définition algébrique, qui reposent sur l'existence de *borne supérieure* (ou *supremum*)

et de *borne inférieure* (ou *infimum*) définies comme des opérateurs binaires dans la définition algébrique et comme des éléments particuliers dans la définition ordinale.

La notion de treillis a été introduite de façon algébrique par [Birkhoff, 1940] avec les propriétés algébriques des opérateurs $\vee$ et $\wedge$:

Définition 3.3 (Treillis algébrique)**.** Un treillis est un triplet $\mathcal{L} = (E, \vee, \wedge)$, tel que $\vee$ et $\wedge$ sont deux opérateurs binaires sur l'ensemble E ; on les appelles aussi *sup* et *inf* (*join et meet*). Ces deux opérateurs sont associatifs, commutatifs, idempotents, et vérifient la loi d'absorption :

- associativité : pour tous $x, y, z \in E$, $(x \vee y) \vee z = x \vee (y \vee z)$ et $(x \wedge y) \wedge z = x \wedge (y \wedge z))$
- commutativité : pour tous $x, y \in E, x \vee y = y \vee x$ et $x \wedge y = y \wedge x$
- idempotence : pour tous $x \in E$, $x \vee x = x = x \wedge x$
- loi d'absorption : pour tous $x, y, z \in E, x \vee (x \wedge y) = x = x \wedge (x \vee z))$

La définition ordinale du treillis à été introduite par [Barbut and Monjardet, 1970] avec la relation d'ordre $\leq$:

Définition 3.4 (Treillis ordinale)**.** Un ensemble ordonné $\mathcal{L} = (E, \leq)$ est un treillis si pour chaque paire d'éléments $\{x,\ y\} \in E$, le *supremum* et l'*infimum* existent. Il est dit *sup-demi-treillis* dans le cas où seulement le *supremum* existe. Dualement, il est dit *inf-demi-treillis* si seulement l'*infimum* existe.

Donc, un treillis est à la fois, sup-demi-treillis et inf-demi-treillis. Un treillis est appelé treillis complet si le supremum $\bigvee X$ et l'infimum $\bigwedge X$ existent pour toute partie X de E. Chaque treillis complet possède un plus grand élément $\bigvee E$, noté 1_E ou *top* ($\top$). Dualement, il possède un plus petit élément $\bigwedge E$, noté 0_E ou *bottom* ($\bot$).

Le supremum et l'infimum se définissent également à partir des relations binaires $\vee$ et $\wedge$ sur E par : $x \vee y = \bigvee\{x, y\}$ et $x \wedge y = \bigwedge\{x, y\}$. Il se définissent également à partir de la relation d'ordre $\leq$ par : $x \leq y \Leftrightarrow x \vee y = y$ et $x \leq y \Leftrightarrow x \wedge y = x$.

Exemple 3.2. *Prenons l'exemple de l'ensemble $P(X)$ des parties d'un ensemble X. La relation d'inclusion est une relation binaire sur $P(X)$ qui est à la fois réflexive ($A \subseteq A$), antisymétrique ($A \subseteq B$ et $A \neq B \Rightarrow B \nsubseteq A$) et transitive ($A \subseteq B$ et $B \subseteq C \Rightarrow A \subseteq C$) donc $(P(X), \subseteq)$ est un ensemble ordonné. De plus, tout sous-ensemble de X possède une borne sup et une borne inf, donc $\mathcal{L} = (P(X), \subseteq)$ est un treillis.*

Le treillis $\mathcal{L}$ est représenté par un diagramme de Hasse dans la Figure 3.2. On peut vérifier que toute paire d'éléments admet un supremum ainsi qu'un infimum. De plus, ce treillis possède $\{x, y, z\}$ pour unique élément maximal, également appelé $\top$, et $\{\}$ pour unique élément minimal, également appelé $\bot$.

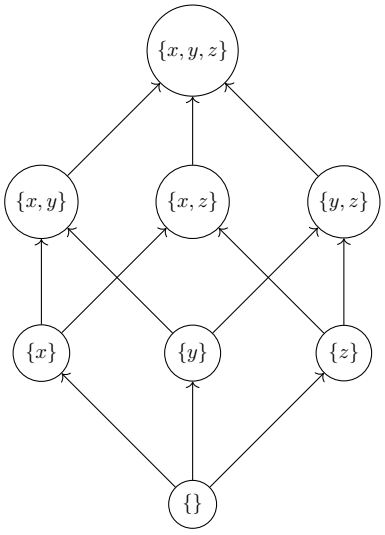


FIGURE 3.2 – Diagramme de Hasse du treillis $\mathcal{L} = (P(\{x, y, z\}), \subseteq)$

Un élément est dit *réductible* s'il est résultat de l'application de l'opérateur *supremum* $\vee$ ou *infimum* $\wedge$ sur au moins deux autres éléments distincts, sinon il est dit *irréductible*.

Définition 3.5 (Éléments irréductibles)**.** Un élément d'un treillis $(E, \leq)$ est dit irréductible s'il ne peut pas être le $sup(join)$ ou le $inf(meet)$ d'un ensemble d'éléments ne le contenant pas :

- Un élément $j \in E$ est appelé sup-irréductible si, pour tout sous-ensemble $X \subseteq E$, $j = \bigvee X$ implique que $j \in X$.
- Un élément $m \in E$ est appelé inf-irréductible si, pour tout sous-ensemble $X \subseteq E$, $m = \bigwedge X$ implique que $m \in X$.

Nous désignons les sup-irréductibles par $J(\mathcal{L})$ et les inf-irréductibles par $M(\mathcal{L})$.

Une caractérisation immédiate des éléments irréductibles à partir de la relation de couverture $\prec$ du treillis établit qu'un élément est un sup-irréductible si et seulement si il ne couvre qu'un seul élément (il n'a qu'un seul successeur immédiat dans le diagramme de Hasse), alors qu'un élément est un inf-irréductible si et seulement si il n'est couvert que par un seul élément (il n'a qu'un seul prédécesseur immédiat dans le diagramme de Hasse) [Bertet et al., 2018].

Tout élément $x \in E$ d'un treillis $\mathcal{L} = (E, \leq)$ est la borne supérieure de l'ensemble de ses prédécesseurs, ainsi que la borne inférieure de l'ensemble de ses successeurs. La définition de treillis permet d'établir facilement qu'il est possible de réduire ces ensembles aux seuls prédécesseurs sup-irréductibles, et successeurs inf-irréductibles :

$$x = \bigvee \{y \text{ sup-irréductible tel que } y \leq x\} \tag{3.1}$$

$$x = \bigwedge \{y \text{ inf-irréductible tel que } y \geq x\} \tag{3.2}$$

Tout élément réductible peut par conséquent s'obtenir par applications successives des opérateurs $\vee$ et $\wedge$ sur un ensemble d'irréductibles.

Définition 3.6 (Table des irréductibles)**.** La table des irréductibles est composée en colonne des sup-irréductibles et en ligne des inf-irréductibles où la case (m, j) dans la table contient $\times$ si $m \leq j$.

3 Analyse Formelle de Concepts

3.1 Contexte et concept

Définition 3.7. Un contexte formel est un triplet (G, M, I), G (en allemand : *Gegenstände*) un ensemble non vide d'objets, M (en allemand : *Merkmale*) un ensemble non vide d'attributs et I la relation binaire entre les objets de G et les attributs de M ($I \subseteq G \times M$). Si nous avons $(g, m) \in I$, avec $g \in G$ et $m \in M$, on dit que l'objet g possède l'attribut m.

Nous pouvons dire que le contexte formel est une autre forme pour définir des données tabulaires ou les attributs sont binaires (cf. 1.1).

Introduisons les deux opérateurs α et β nommés *opérateurs de dérivation* :

- $\alpha : 2^G \to 2^M$ est une application qui associe un sous-ensemble $B \subseteq M$ pour chaque sous-ensemble $A \subseteq G$ tel que :

$$\alpha(A) = \{b \mid b \in M \text{ et } \forall a \in A, (a, b) \in I\} \tag{3.3}$$

  $\alpha(A)$ est l'ensemble des attributs communs d'un ensemble d'objets A.

- $\beta : 2^M \to 2^G$ est une application qui associe un sous-ensemble $A \subseteq G$ pour chaque sous-ensemble $B \subseteq M$ tel que :

$$\beta(B) = \{a \mid a \in G \text{ et } \forall b \in B, (a, b) \in I\} \tag{3.4}$$

  $\beta(B)$ est l'ensemble des objets qui partage les mêmes attributs B.

Dans un contexte donné, on appelle concept un ensemble d'objets partageant les mêmes attributs. Formellement :

Définition 3.8 (Concept)**.** Pour deux sous-ensembles $A \subseteq G$ et $B \subseteq M$, un concept est un couple (A, B) tel que : $A = \beta(B)$ et $B = \alpha(A)$. On appelle A l'*extension*, et B l'*intention* du concept (A, B).

Exemple 3.3. *La Table 3.1 représente un exemple d'un contexte. L'intersection de ligne g avec colonne m désigne l'existence ou non d'une relation entre l'objet g et l'attribut m. Ici la relation binaire est "la séquence contient l'action".*

- $G = \{1, 2, 3, 4, 5\}$

Séquence	T	S	C	Sp	L
1	x	x		x	x
2	x		x	x	
3	x		x		x
4	x		x	x	
5		x	x	x	

TABLE 3.1 – Exemple de contexte

- $M = \{T, S, C, Sp, L\}$

Nous avons : $\alpha(\{2,3\}) = \{T, C\}$ *et* $\beta(\{T, L\}) = \{1, 3\}$.
Le couple $(\{1,3\}, \{T, L\})$ *forme un concept formel car* $\alpha(\{1,3\}) = \{T, L\}$ *et* $\beta(\{T, L\}) = \{1, 3\}$. *Le couple* $(\{3,5\}, \{C\})$ *n'est pas un concept formel car* $\alpha(\{3,5\}) = \{C\}$, *mais* $\beta(\{C\}) = \{2, 3, 4, 5\} \neq \{3, 5\}$.

3.2 Treillis des concepts

Définition 3.9 (Relation de généralisation/spécialisation)**.** La relation sous-concept/super-concept est une relation d'ordre sur l'ensemble des concepts définie pour deux concepts $C_1 = (A_1, B_1)$ et $C_2 = (A_2, B_2)$ par :

$$(A_1, B_1) \leq (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2 \ (\Leftrightarrow B_2 \subseteq B_1). \tag{3.5}$$

On dit que C_1 est un sous-concept de C_2, et C_2 un super-concept de C_1. C'est une relation de généralisation/spécialisation. Le concept C_2 est le concept le plus général, il contient plus d'objets mais peu d'attributs. Par contre, C_1 est plus spécifique, il a moins d'objets mais qui partagent plus d'attributs.

- l'extension de C_1 est incluse dans l'extension de C_2
- l'intention de C_2 est incluse dans l'intention de C_1

L'ensemble ordonné de tous les concepts muni de la relation d'ordre entre les concepts forme un treillis complet appelé le *treillis de concepts.*

Définition 3.10 (Treillis des concepts)**.** Le treillis des concepts d'un contexte (G, M, I) est une paire $(\mathcal{C}, \leq)$ où $\mathcal{C}$ est un ensemble de concepts défini par le contexte (G, M, I) et $\leq$ est la relation de généralisation/spécialisation définie sur l'ensemble $\mathcal{C}$.

Exemple 3.4. *La Figure 3.3 montre le diagramme de Hasse du treillis de concepts généré pour le contexte de l'Exemple 3.3. Chaque nœud de ce diagramme est un concept avec la partie des objets (l'extension) et la partie des attributs (l'intention). Nous avons,* $(\{1,3\}, \{T, L\}) \leq (\{1, 2, 3, 4\}, \{T\})$, *car* $\{1,3\} \subseteq \{1, 2, 3, 4\}$ *et* $\{T\} \subseteq \{T, L\}$. *Les concepts en gris clair sont des concepts sup-irréductibles, et les concepts en gris foncé sont les inf-irréductibles.*

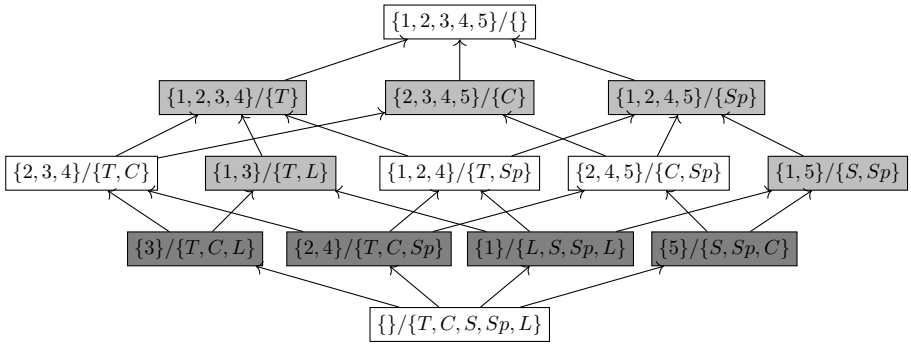


FIGURE 3.3 – Exemple de treillis de l'Exemple 3.3

Avec des treillis volumineux, l'information présentée dans chaque concept peut surcharger le treillis et le rendre moins lisible. Nous pouvons cependant distinguer certains concepts particulier. Un concept-objet (*object-concept*) est un concept C_o qui introduit un nouvel objet o dans le treillis de tel sorte que o soit dans l'extension de C_o mais ne soit pas dans l'extension d'aucun concept $C < C_o$. De même, un concept-attribut (*attribute-concept*) est un concept C_a qui introduit un nouvel attribut a dans le treillis de tel sorte que a soit dans l'intention de C_a mais ne soit pas dans l'intention d'aucun concept $C > C_a$. Les notions de concept-objet et concept-attribut ont été introduites dans [Ganter and Wille, 1999]. Ces concepts sont également définis comme les *concepts introducteurs* [Berry et al., 2014]. Les objets sont introduits de bas en haut et les attributs de haut en bas dans le treillis, ce qui signifie qu'une fois qu'un attribut est introduit, il est hérité par tous les concepts inférieurs, et de la même façon, une fois qu'un objet est introduit, il est hérité par tous les concepts supérieurs.

La *représentation réduite* de treillis consiste à afficher seulement les nouveaux objets des concepts objets et les nouveaux attributs des concepts attributs, et donc permet une meilleure lisibilité. L'AOC-poset (*Attribute/Object Concept poset*) [Petersen, 2001, Osswald and Petersen, 2003] est le sous-ordre du treillis partiel composé des concepts introducteurs dans le treillis des concepts. L'AOC-poset est connu sous différentes terminologies, a savoir : *pruned concept hierarchies* [Godin et al., 1998], *Galois subhierarchy* [Dicky et al., 1994], *knowledge space* [Mineau et al., 1990].

Exemple 3.5. *La Figure 3.4 est la représentation reduite du treillis de la Figure 3.3. Le concept qui contient* $\{L\}$ *est en réalité* $(\{\mathbf{1},\mathbf{3}\},\{\mathbf{T},L\})$

On peut observer qu'un concept sup-irréductible introduit nécessairement au moins un nouvel objet. Tout concept sup-irréductible est donc un concept-objet. À l'inverse, tout concept-objet n'est pas forcément un concept sup-irréductible. De la même façon, tout concept inf-irréductible est un concept-attribut, mais l'inverse n'est pas toujours vrai.

Théorème fondamental [Barbut and Monjardet, 1970] Tout treillis est isomorphe au *treillis de concept* de sa table des irréductibles $(J(\mathcal{L}), M(\mathcal{L}), \leq)$.

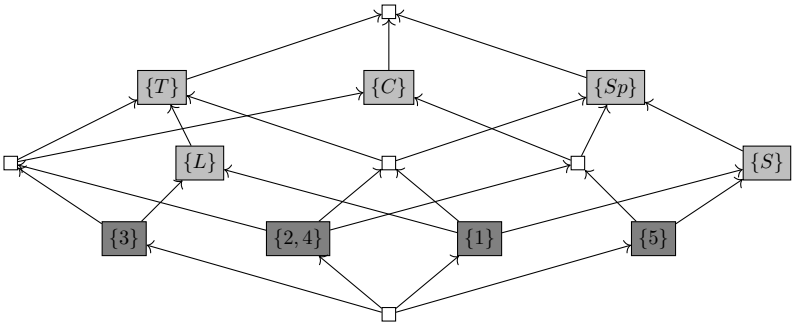


FIGURE 3.4 – Le treillis réduit du treillis de la Figure 3.3

Dans le cadre l'analyse formelle des concepts, la table des irréductibles se retrouve sous la dénomination de *contexte réduit*, à partir duquel est défini son *treillis des concepts* (cf Section 3).

Ce théorème montre l'importance des éléments irréductibles. En effet, les irréductibles suffisent à la reconstruction du treillis, il en portent toute la structure.

3.3 Connexion de Galois et treillis de fermés

Les deux opérateurs de dérivation (α, β) forment une connexion de Galois et le contexte peut se représenter indifféremment par (G, M, I) ou $(G, M, (\alpha, \beta))$.

Définition 3.11 (Connexion de Galois)**.** Une connexion de Galois est définie pour deux ensembles partiellement ordonnées $(2^G, \leq)$ et $(2^M, \leq)$ par le couple (α, β) pour $A_1, A_2 \subseteq G$ et $B_1, B_2 \subseteq M$. (α, β) est une connexion de Galois si les propriétés suivantes sont vérifiées :

- isotone : $A_1 \subseteq A_2 \Rightarrow \alpha(A_2) \subseteq \alpha(A_1)$
- isotone : $B_1 \subseteq B_2 \Rightarrow \beta(B_2) \subseteq \beta(B_1)$
- extensive : $A_1 \subseteq (\beta \circ \alpha)(A_1)$ et $B_1 \subseteq (\alpha \circ \beta)(B_1)$

Un système de fermeture se définie à l'aide d'un opérateur de fermeture φ par :

Définition 3.12 (Système de fermeture)**.** Un système de fermeture est un couple $C = (E, \varphi)$ avec E un ensemble, et φ un opérateur de fermeture définie sur l'ensemble $P(E)$ des parties de E qui soit isotone, extensive, et idempotente (pour $X \subseteq E : \varphi(\varphi(X)) = \varphi(X)$) :

Pour une partie $X \subseteq E$, $\varphi(X)$ est appelée la fermeture de X, ou un fermé de (E, φ).

Définition 3.13 (Treillis de fermés)**.** Un treillis de fermés sur un ensemble E est une paire $(\mathcal{F}, \subseteq)$ où $\mathcal{F}$ est un ensemble de fermés $X = \varphi(X)$ muni de la relation d'inclusion $\subseteq$:

$$\mathcal{F} = \{X \subseteq E \mid \varphi(X) = X\}$$

La composition des deux opérateurs $(\beta \circ \alpha)$ forme un opérateur de fermeture sur les objets. La paire $(G, (\beta \circ \alpha))$ forme donc un système de fermeture sur G, qui est associé au treillis des fermés $(\mathcal{F}_\mathcal{G}, \subseteq)$, où $\mathcal{F}_\mathcal{G}$ est la famille des fermés de G. Dualement, la composition des deux opérateurs $(\alpha \circ \beta)$ forme un opérateur de fermeture sur les attributs, $(M, (\alpha \circ \beta))$ est un système de fermeture sur M, et $(\mathcal{F}_\mathcal{M}, \subseteq)$ est un treillis de fermés sur M.

Le treillis des fermés sur G (M) correspond donc au treillis de concepts mais en ne considérant que les intentions (extensions) des concepts. Le treillis des fermés sur les attributs corresponds aux *itemsets fermés* introduits dans le chapitre précédent. Ce qui signifie que les algorithmes de fouille d'*itemsets fermés fréquents*, génèrent une partie du treillis des fermés sur les attributs en ne conservant que les fermés fréquents : cette partie est appelé l'*iceberg*.

Définition 3.14 (Iceberg)**.** L'*iceberg* est le sous-ordre du treillis des fermés contenant les seuls fermés fréquents pour un seuil de support minimal donné.

Exemple 3.6. *À partir des attributs de la Table 3.1, nous pouvons représenter les données binaires comme des sous-ensembles d'attributs (cf Table 3.2). Le treillis des fermés sur les attribut est représenté dans la Figure 3.5. L'iceberg pour un seuil de support minimal de* $\frac{3}{5}$ *est la partie haute de treillis représenté par des nœuds grisé dans la Figure 3.5.*

Séquence	T	S	C	Sp	L	sous-ensembles
1	x	x		x	x	$\{T, S, Sp, L\}$
2	x		x	x		$\{T, C, Sp\}$
3	x		x		x	$\{T, C, L\}$
4	x		x	x		$\{T, C, Sp\}$
5		x	x	x		$\{S, C, Sp\}$

TABLE 3.2 – Exemple de sous-ensembles construits à partir de données binaires

3.4 Algorithmes de génération de treillis des fermés

Plusieurs algorithmes ont été proposés dans la littérature pour la génération du treillis de fermés. Les premiers algorithmes sont dites non incrémentaux [Chein, 1969, Ganter, 1984, Bordat, 1986, Nourine and Raynaud, 1999]. Citons également des algorithmes incrémentaux pour une mise à jour incrémentale du treillis après ajout d'un objet, attribut ou relation dans le contexte [Norris, 1978, Godin et al., 1991, Carpineto and Romano, 1993].

Nous présentons ici un des algorithmes de base, l'algorithme de Bordat [Bordat, 1986] sur lequel repose l'algorithme NEXTPRIORITYCONCEPT qui sera présenté dans le chapitre suivant. Il est basé sur le théorème de Bordat [Bordat, 1986] pour la génération du treillis de fermés sur les attributs. Nous donnerons une version duale qui génère le treillis des fermés sur M pour un contexte $(G, M, (\alpha, \beta))$:

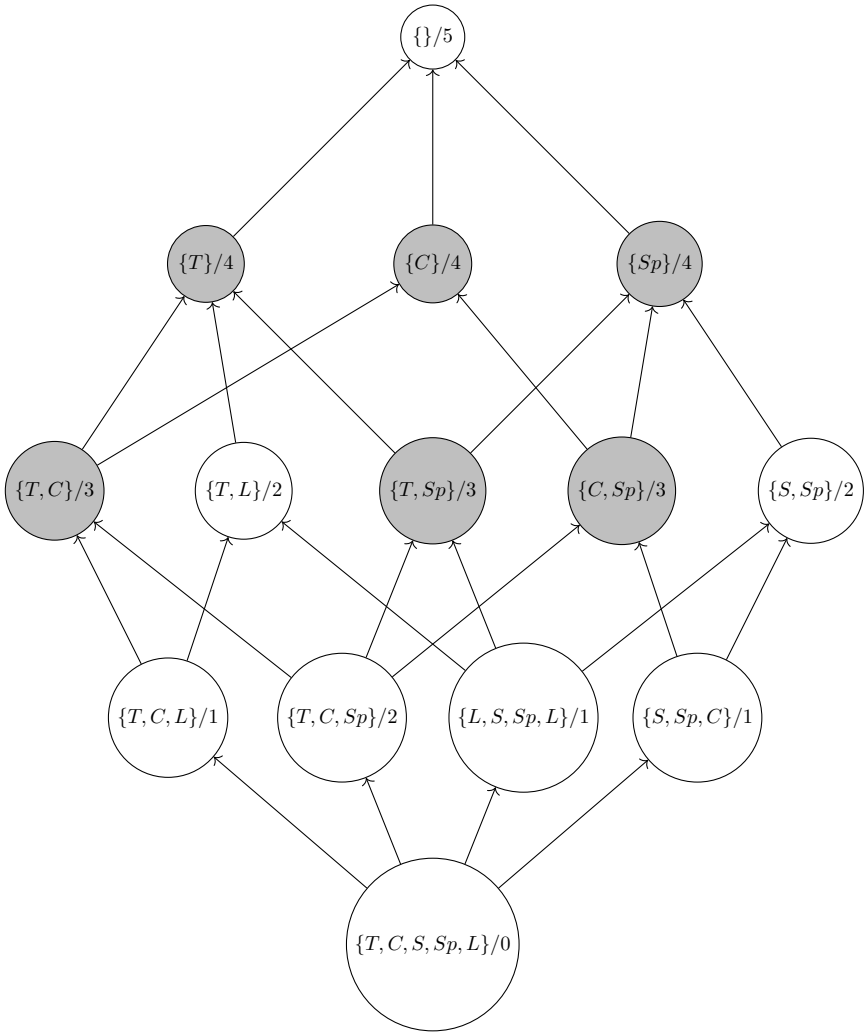


FIGURE 3.5 – Treillis des fermés de l'Exemple 3.3

Théorème 3.1. *Les prédécesseurs immédiats d'un fermé B sont les sous-ensembles maximaux par inclusion de la famille suivante définie pour les attributs de M :*

$$\mathcal{FS}_B = \{(\alpha \circ \beta)(B + b) \mid b \in M \setminus B\} \tag{3.6}$$

L'algorithme de Bordat [Bordat, 1986] est une application directe du théorème de Bordat. L'algorithme 8 présente une initialisation et un premier appel de l'algorithme de Bordat présenté dans l'algorithme 9. Il calcule récursivement le diagramme de Hasse de treillis de concepts d'un contexte $(G, M, (\alpha, \beta))$ en commençant par le concept top $(\beta(\emptyset), \emptyset)$.

L'algorithme procède par niveau et à chaque appel récursif, les prédécesseurs immédiats d'un concept $(\beta(B), B)$ sont calculés. La famille $\mathcal{FS}_B$ est tout d'abord calculée, puis les ensembles maximaux par inclusion sont sélectionnés. $\mathcal{FS}_B$ contient les intentions des

prédécesseurs immédiats possibles de (A, B). L'intention B' associée est $B' = (\alpha \circ \beta)(B+b)$, ou b est un attribut de M qui n'est pas dans B. L'idée est d'augmenter l'ensemble B avec un nouvel attribut de $M \setminus B$ pour générer un nouveau fermé candidat B'. Ensuite, on ne conserve que les fermés maximaux par inclusions qui correspondent aux intensions B' des prédécesseurs immediat de (A, B), les prédécesseurs immediats s'en déduisent avec $(\beta(B'), B')$.

Donc les prédécesseurs immédiats de (A, B) sont les concepts $(\beta(B'), B')$.

Algorithm 8 CalculeTreillis

Input : Un contexte $(G, M, (\alpha, \beta))$
Output : Le treillis L des fermés sur M

1: $B \leftarrow \emptyset$
2: $A \leftarrow \beta(\emptyset)$
3: $Bordat(G, M, (\alpha, \beta), (A, B), L)$
4: $Return\ \ L$

Algorithm 9 Bordat

Input : $(G, M, (\alpha, \beta)), (A, B), L$
Output : Le treillis L des fermés

1: $LM \leftarrow \emptyset$
2: **for** $b \in M/\ B$ **do**
3: $\quad B' \leftarrow (\alpha \circ \beta)(B + b)$
4: $\quad Maximal(LM, B')$;
5: **end for**
6: **for** $B' \in LM$ **do**
7: $\quad L.add(\beta(B'), B')$
8: $\quad Bordat(G, M, (\alpha, \beta), (\beta(B'), B'), L)$
9: **end for**

Algorithm 10 Maximal

Input : Un treillis L et un ensemble d'attributs B
Output : Le treillis L contenant seulement les sous-ensembles maximaux

```
1: add ← True
2: for B' ∈ L do
3:     if B ⊂ B' then
4:         add ← False
5:         break
6:     else
7:         L.remove(B')
8:     end if
9: end for
10: if add then
11:     L.add(B)
12: end if
13: Return L
```

4 Extension de l'AFC à des données non binaires

Nous avons présenté dans la section précédente l'analyse formelle de concepts classique, où les données traitées sont des données binaires organisées sous forme d'un contexte. Les données réelles, les attributs ne sont pas seulement binaires, ils peuvent aussi être catégoriels, numériques, voire plus complexes comme les séquences. Il existe des extensions de l'AFC pour le traitement des données non binaires.

Nous présentons tout d'abord des extensions spécifiques à un type de données (Catégorielles, Relationnelle, Graphes). Puis, le cadre général des structures de motifs pour une extension générique à tout type de données. Et enfin, des approches spécifiques aux séquences.

4.1 Extensions dédiées à certains types de données

Contexte multi-valué. Une extension de l'AFC qui consiste à traiter des données non binaires où les attributs peuvent avoir de multiples valeurs comme les données catégorielles a été introduite par [Wille, 1992] sous le nom de contexte multi-valué (G, M, W, I). Un contexte multi-valué introduit l'ensemble W de valeurs catégorielles des attributs, et la relation I définie pour $g \in G, m \in M, w \in W$, par $(g, m, w) \in I$ signifie que l'objet g a la valeur w pour l'attribut m. La construction du treillis dans ce cas passe par une mise en échelle conceptuelle (*conceptual scaling*) qui est de plusieurs types (ordinale, nominale, dichotomique, etc.) [Ganter and Wille, 1989]. La mise en échelle conceptuelle est une transformation du contexte multi-valué en un contexte binaire. La combinaison de chaque attribut avec ses valeurs correspond à des nouveaux attributs binaires. Le treillis se construit ensuite normalement en utilisant l'AFC classique.

Analyse Relationnelle de Concepts. L'ARC [Rouane-Hacene et al., 2013] est une extension de l'AFC pour des données relationnelles, i.e. plusieurs tables et des associations entre elles, où chaque table décrit une catégorie d'objets par des attributs. Ces tables représentent des contextes dits objets-attributs. La relation entre les tables présente des contextes objets-objets. L'ARC génère une famille de treillis, un par contexte ou catégorie d'objet, dont les concepts sont décrits en faisant référence aux concepts des autres treillis.

Graphe-AFC. Graphe-AFC [Ferré, 2015] est une extension de l'AFC pour les graphes de connaissances. Le contexte dans ce cas est appelé graphe contexte (*graph context*) et est définie par $K = (G, M, I)$ avec G l'ensemble des objets, M l'ensemble des attributs, et I est la relation entre des n-uplet d'objets et les attributs $I \subseteq G^* \times M$. À partir du graphe contexte, les graphes concepts sont construits avec les relations d'objets en extension et les *Projected Graph Patterns* (PGP) en intention. Un *Projected Graph Patterns* est un motif de graphe avec un n-uplet de variables. Les graphes concepts sont ensuite organisés dans un treillis de graphes concepts.

4.2 Extensions dédiées aux données complexes

Le formalisme des structures de motifs [Ganter and Kuznetsov, 2001, Kaytoue et al., 2015] définie un cadre générique d'extension de l'AFC, pour traiter les données non binaires. En effet, il suffit que les espaces de description des attributs non binaires soient organisées comme un inf-demi-treillis pour que la connexion de Galois entre objets et descriptions soit maintenue. La même idée a été introduite avec un formalisme différent en utilisant la logique (Analyse Logical de Concepts (ALC) [Ferré and Ridoux, 2000]).

Analyse Logique de Concepts (ALC) L'ALC est une extension de l'AFC dont laquelle un objet est décrit par une formule logique au lieu d'un ensemble d'attributs. Un contexte dans ce cas est un n-uplet $(G, \mathcal{L}, i)$ ou G est un ensemble d'objet, $\mathcal{L}$ est un treillis (de formules logiques) muni de la relation de déduction $\models$ et i est une correspondance qui associe à chaque objet $g \in G$ une formule dans $f \in \mathcal{L}$. Les deux opérateurs de dérivation sont donc définie par :

- $\alpha : 2^G \to \mathcal{L}$ est une application qui associe une formule logique pour chaque sous-ensemble $A \subseteq G$ tel que :

$$\alpha(A) = \bigvee_{a \in A} i(a) \tag{3.7}$$

  $\alpha(A)$ est la formule logique la plus précise qui subsume toutes les descriptions d'objets de A ($i(a)$).
- $\beta : \mathcal{L} \to 2^G$ est une application qui associe un sous-ensemble $A \subseteq G$ pour chaque formule logique $f \in \mathcal{L}$ telle que :

$$\beta(f) = \{g \mid g \in G \text{ et } i(g) \models f\} \tag{3.8}$$

  $\beta(B)$ est l'ensemble des objets dont la description est subsumée par f.

Un concept logique est donc une paire (A, f) où $A \subseteq G$ et $f \in \mathcal{L}$ tel que $\alpha(A) = f$ et $\beta(f) = A$.

L'ensemble de tous les concepts du contexte $(G, \mathcal{L}, i)$ noté $\mathcal{C}(G, \mathcal{L}, i)$ est muni de la relation d'ordre $\leq^c$ définie pour (A_1, f_1) et (A_2, f_2), par :

$$(A_1, f_1) \leq^c (A_2, f_2) \Leftrightarrow A_1 \subseteq A_2 \Leftrightarrow f_1 \models f_2 \tag{3.9}$$

L'ensemble partiellement ordonné $(\mathcal{C}(G, \mathcal{L}, r), \leq^c)$ forme un treillis de concepts.

Les structure de motifs Les structure de motifs [Ganter and Kuznetsov, 2001] peuvent être vues comme une généralisation de l'AFC où les concepts sont composés d'objets avec leurs descriptions communes. Formellement, une structure de motifs [Ganter and Kuznetsov, 2001] est un triplet $(G, (\mathcal{D}, \sqcap), \delta)$ où G est un ensemble d'objets, $(\mathcal{D}, \sqcap)$ est un inf-demi-treillis de descriptions d'objets potentiel, et $\delta : G \to \mathcal{D}$ associe chaque objet à sa description.

Les éléments de $\mathcal{D}$ sont ordonnés avec la relation de subsomption $\sqsubseteq$. Les relations de dérivation sont définies par :

- $\alpha_{\mathcal{D}} : 2^G \to \mathcal{D}$ est une application qui associe une description d pour chaque sous-ensemble $A \subseteq G$ tel que :

$$\alpha_{\mathcal{D}}(A) = \sqcap_{g \in A} \delta(g) \tag{3.10}$$

  $\alpha_{\mathcal{D}}(A)$ est la description commune d'un ensemble d'objets A.

- $\beta_{\mathcal{D}} : \mathcal{D} \to 2^G$ est une application qui associe un sous-ensemble $A \subseteq G$ pour chaque description $d \subseteq D$ telle que :

$$\beta_{\mathcal{D}}(d) = \{g \in G \mid d \sqsubseteq \delta(d)\} \tag{3.11}$$

  $\beta_{\mathcal{D}}(d)$ est l'ensemble des objets qui partage la description d.

Les concepts de motifs sont des paires (A, d), $A \subseteq G$, $d \in \mathcal{D}$ telle que $\alpha_{\mathcal{D}}(A) = d$ et $A = \beta_{\mathcal{D}}(d)$. La relation d'ordre est définie entre les concepts de motifs par :

$$(A_1, d_1) \leq (A_2, d_2) \iff A_1 \subseteq A_2 (\iff d_2 \sqsubseteq d_1) \tag{3.12}$$

L'ensemble des concepts de motifs forme un treillis appelé treillis de motifs (*pattern lattice*).

4.3 Fouille de séquences et AFC

Liens entre la fouille de séquences et l'AFC Un rapprochement peut être identifié entre le domaine de la fouille de motifs (fermés) fréquents et le domaine de l'AFC et leurs extensions pour les données séquentielles. Dans [Casas-Garriga, 2005], les auteurs présentent les *Ordres Partiels Fermés (OPC)* comme un graphe pour les

motifs séquentiels où chaque chemin dans le graphe représente un motif séquentiel. Les motifs OPC résument les motifs séquentiels fermés communs pour les représenter graphiquement, ce qui facilite l'interprétation. Cette approche utilise l'AFC pour représenter les motifs séquentiels fermés. Les auteurs utilisent les algorithmes de fouille de motifs séquentiels fermés comme BIDE, CloSpan ou TSP pour extraire les motifs séquentiels fermés. Ensuite, dans une opération de post-traitement ils construisent des concepts (S, T) avec S ensemble de sous-séquences fermées communes et T un ensemble de séquences. Et enfin, les séquences $s \in S$ sont représentées dans un OPC. Une approche similaire a été présentée par [Pei et al., 2006] pour l'extraction de motifs partiellement ordonnés pour les chaînes de caractères. Les auteurs proposent l'algorithme *Frecpo* qui extrait des OPC fréquents directement de l'ensemble des chaînes de caractères. L'algorithme *OrderSpan* [Fabrègue et al., 2015] se base sur les préfixes et les suffixes des séquences pour extraire des motifs OPC avec une approche de croissance de motifs.

Analyse de séquences avec l'AFC [Nica et al., 2020] propose une approche *RCA-seq* basé sur l'ARC pour extraire des motifs OPC. À partir d'une base de données relationnelle des séquences, *ARC-seq* extrait les motifs OPC avec la hiérarchie présente grâce à l'ARC.

Certaines approches de fouille de séquences se positionnent dans le cadre de l'AFC basée sur le formalisme des structures de motifs [Ganter and Kuznetsov, 2001, Kaytoue et al., 2015]. Les auteurs dans [Buzmakov et al., 2013] sont les premiers à utiliser le formalisme de structures de motifs pour analyser les données séquentielles qui représentent des enregistrement médicaux. Ils extraient en premier lieu les sous-séquences maximales pour construire un inf-demi-treillis de description séquentiel. Les séquences contiennent des éléments complexes composés d'éléments taxonomiques et d'ensembles d'attributs. Les auteurs ont décidé de simplifier le problème en considérant des sous-séquences ne contenant que des éléments contigus. Nous citons aussi les travaux d'extraction de séquences pour découvrir des motifs rares [Codocedo et al., 2017]. Les auteurs utilisent une représentation des séquences par un graphe orienté acyclique *directed acyclic graph* inspiré de [Pei et al., 2006, Fabrègue et al., 2015] d'extraction de motifs OPC. Dans [Gizdatullin et al., 2017], les auteurs utilisent les structures de motifs pour étudier des séquences démographiques. Les motifs obtenus sont des préfixes contigus fréquents des séquences d'entrée.

5 Conclusion et discussion

Nous avons présenté à travers ce chapitre l'Analyse Formelle de Concepts, qui est un cadre de travail pour l'analyse de données. L'AFC est en premier lieu introduite pour des données binaires. L'AFC permet la construction d'un treillis de concepts où les concepts sont des sous-groupes d'objets partageant des attributs communs. La partie d'attributs dans chaque concept correspond à un *itemset* fermé.

Plusieurs extensions de l'AFC ont été introduites, qui visent à traiter des données non

binaires. Les structures de motifs ont permis d'étendre l'AFC à des données complexes, comme les séquences ou les graphes. Dans ce cas, les concepts sont des sous-groupes d'objets partageant des motifs communs, par exemple des motifs séquentiels communs pour des données séquentielles.

L'AFC formalise et caractérise tous les motifs fermés avec une structure de treillis qui garantie la présence de bornes sup et inf. Dans la fouille de motifs (fermés) fréquents, l'extraction des motifs (fermés) s'inspire le plus souvent de l'algorithme *Apriori* pour génèrer les motifs (fermés) fréquents niveau par niveau. L'algorithme de Bordat caractérise cette notion de niveau par niveau à l'aide d'un opérateur de fermeture.

La fouille de séquences introduite dans le chapitre précédent est en lien avec l'AFC et les structures de motifs. Des techniques d'analyse de séquences avec l'AFC et les structures de motifs ont été introduites. Les structures de motifs génèrent un treillis avec des concepts composés d'objets avec leurs descriptions communes. L'espace de description doit être organisée dans un inf-demi-treillis de descriptions potentielles.

La difficulté d'étendre l'AFC aux séquences réside dans l'espace de description des sous-séquences maximales communes qui est trop volumineux. Un calcul préalable est nécessaire dans le but de construire un inf-demi-treillis. Des sous-séquences contiguës sont considérées dans beaucoup de cas afin de simplifier les calculs. Le déluge de motifs déjà identifié dans le Chapitre 2 se retrouve dans cette approche, avec un nombre exponentiel de motifs séquentiels fermés. De plus, à notre connaissance il n'existe pas d'approches basées sur l'AFC pour l'extraction des motifs séquentiels intégrant l'aspect temporel et/ou d'intervalle.

L'algorithme NextPriorityConcept et la plateforme `GALACTIC` étendent l'AFC pour des données hétérogènes et complexes avec des descriptions calculées *"à la volée"* pour chaque sous-groupes d'objets (Chapitre 4) et des stratégies d'exploration pour limiter le déluge de motifs.

Chapitre 4

L'algorithme NextPriorityConcept

Sommaire

1	**Introduction**	**73**
2	**Descriptions**	**74**
3	**Stratégies**	**76**
4	**L'algorithme** NextPriorityConcept	**77**
5	**La Plateforme** `GALACTIC`	**81**
6	**Cas d'utilisation de l'algorithme** NextPriorityConcept	**84**
7	**Conclusion et discussion**	**86**

1 Introduction

Inspiré par l'AFC et les structures de motifs, l'algorithme NextPriorityConcept introduit dans un article récent [Demko et al., 2020] propose une approche de fouille de motifs guidée par l'utilisateur. Cette approche permet l'analyse de données hétérogènes et complexes. L'algorithme utilise la notion de *caractéristique* pour représenter plusieurs types de données, de *description* pour décrire un ensemble de données de manière générique, et de *stratégie* pour réduire un concept en sous-concepts du niveau suivant. Les descriptions et les stratégies sont définies par des *prédicats* monadiques ce qui permet de traiter plusieurs types de données de façon générique. Des *mesures* peuvent aussi être utilisées pour filtrer la génération des sous-concepts à ceux vérifiant un seuil pour ces mesures, tels que le support par exemple.

Bordat [Bordat, 1986] calcule les successeurs immédiats d'un concept niveau par niveau à partir du concept plus bas $(\beta(M), M)$. L'algorithme NextPriorityConcept repose sur une version dual de l'algorithme de Bordat présenté dans le chapitre précédent avec un calcul des prédécesseurs immédiats niveau par niveau en commençant par le concept le plus haut $(G, \alpha(G))$. Pour chaque concept (A, B), ses prédécesseurs sont obtenus en

testant un nouvel attribut potentiel $b \in M \setminus B$ permettant d'obtenir un sous-ensemble A' de A. De cette façon, A diminue tandis que B augmente, ce qui correspond à la relation de généralisation/spécialisation entre concepts.

La récursion de l'algorithme de Bordat est remplacée par une file de priorité utilisant le support des concepts. Les concepts sont donc générés niveau par niveau, en commençant par le concept top $(G, \alpha(G))$. Pour permettre le traitement des données complexes et hétérogènes, les notions de *caractéristiques*, *prédicats*, *descriptions* et *stratégies* ont été introduites. Nous définissons d'abord les trois notions de bases, *prédicats, descriptions, et stratégies*, ensuite nous décrirons l'algorithme NEXTPRIORITYCONCEPT.

Le reste de ce chapitre va être organisé en 6 sections. Nous commençons par présenter les notions de descriptions et les stratégies dans les sections 2 et 3. Ensuite, nous présenterons l'algorithme NEXTPRIORITYCONCEPT dans la section 4. La plateforme `GALACTIC` qui implémente l'algorithme NEXTPRIORITYCONCEPT sera présentée en section 5. Des cas d'utilisation de l'algorithme vont être présentés en chapitre 6. Et nous finissons ce chapitre par une discussion et une conclusion.

2 Descriptions

L'algorithme NEXTPRIORITYCONCEPT introduit la notion de *prédicat* pour décrire les objets indépendamment de leurs types. Le terme *attribut* utilisé classiquement pour les données binaires est remplacé par *caractéristique* qui peut désigner des attributs binaires, mais aussi numériques, catégoriels ou plus complexes. L'algorithme NEXTPRIORITYCONCEPT considère un jeu de données hétérogène (G, S) en entrée où chaque *caractéristique* $s \in S$ peut être vue comme une application $s : G \to R_s$ où R_s est le domaine de s. Les prédicats monadiques décrivent les caractéristiques de manière générique. Pour une caractéristique s et un objet $g \in G$, on désigne par p_s un prédicat, et on écrit $p_s(g)$ si $s(g)$ vérifie p_s.

Par exemple, une caractéristique numérique peut être décrite par des prédicats de la forme *"est plus petit/plus grand que c"* où c est une valeur numérique. Autre exemple, à une caractéristique catégorielle est associé un prédicat de la forme *"est contenue dans X"* où X est un ensemble de valeurs de la catégorie.

Les caractéristiques différentes doivent être traitées séparément alors que des caractéristiques similaires peuvent être traitées ensemble ou séparément, et certaines caractéristiques peuvent ne pas être considérées, ou être considérées plusieurs fois dans différents groupes S_i. Les caractéristiques sont données par une famille $S = (S^i)_{i \leq d}$, où chaque S^i contient des caractéristiques du même domaine. Les prédicats sont une manière de décrire les données selon leur type.

Exemple 4.1. *Prenons l'exemple dans la Table 4.1. L'exemple présente cinq personnes avec leurs Age et Ville de résidence, leurs identifiants sont* $1, 2, 3, 4$ *et* 5. *On a deux groupes de caractéristiques de types différents :* $S^1 = \{Age\}$ *et* $S^2 = \{Ville\}$

SID	Age	Ville
1	25	Paris
2	32	La Rochelle
3	45	Lyon
4	29	Paris
5	38	La Rochelle

TABLE 4.1 – Exemple de données numériques et catégorielles

La caractéristique S^1 va être traitée par des prédicats de type numérique ("est plus petit/plus grand que c"). Et la caractéristique S^2 va être traitée par des prédicats de type catégoriel ("est contenue dans X").

L'algorithme NEXTPRIORITYCONCEPT introduit les *description* δ fournissant des prédicats décrivant un ensemble d'objets A.

Définition 4.1 (Description)**.** Une description δ^i est une application $\delta^i : 2^G \to 2^P$ qui définit un ensemble de prédicats $\delta^i(A)$ décrivant les caractéristiques de S^i pour un sous-ensemble A de G.

Les *prédicats* décrivant les objets A sont calculés localement par un traitement spécifique pour chaque groupe de caractéristiques S^i, et la description finale δ est l'union de ces prédicats. La description $\delta(A)$ d'un sous-ensemble A d'objets est définie par :

$$\delta(A) = \bigcup_i \delta^i(A)$$

Les descriptions permettent de construire les concepts $(A, \delta(A))$. Un concept $(A, \delta(A))$ est alors composé d'un sous-ensemble d'objets A avec un ensemble de prédicats $\delta(A)$ qui les décrivent. Pour éviter toute confusion, nous désignerons un tel concept (A, D) au lieu de (A, B).

Exemple 4.2. *Donnons les descriptions pour un sous ensemble d'objets $A \subseteq G$ de l'Exemple 4.1 :*

- *Pour $A = \{1, 2, 3\}$ et l'attribut numérique $S^1 = \{Age\}$:*
  - $\delta^1(\{1, 2, 3\}) = \{$" est plus grand que 25 "," est plus petit que 45 "$\}$
- *Pour $A = \{1, 2, 3\}$ et l'attribut catégoriel $S^2 = \{Ville\}$:*
  - $\delta^2(\{1, 2, 3\}) = \{$" est contenu dans {Paris, La Rochelle, Lyon}"$\}$

La description finale est donc donnée par :

$$\delta(\{1, 2, 3\}) = \big\{\text{" est plus grand que 25","," est plus petit que 45",}\\ \text{" est contenu dans \{Paris, La Rochelle, Lyon\}"}\big\}$$

Le concept est donc $(\{1, 2, 3\}, \delta(\{1, 2, 3\}))$. Cela se traduit par, tout objet de $A = \{1, 2, 3\}$, a un age plus grand que 25 et plus petit que 45 et habite à Paris, La Rochelle ou Lyon.

3 Stratégies

Pour un ensemble d'objets A décrit par sa description $\delta(A)$, une stratégie propose une façon de "raffiner" A en sous-groupes $A' \subset A$. Le principe des stratégies introduites par NextPriorityConcept est d'ajouter une information élémentaire aux descriptions de $\delta(A)$ qui ne sera vérifiée que par certains objets de A. Par exemple, pour des caractéristiques binaires, une description classique est l'ensemble des caractéristiques communes à un ensemble d'objets, et ajouter une information élémentaire consiste alors à ajouter un attribut à cet ensemble. La difficulté principale pour générer de nouveaux concepts/motifs est de garantir que l'ajout d'une information élémentaire va bien permettre de réduire l'ensemble des objets A à un sous-ensemble strict $A' \subset A$, mais sans trop le réduire car des concepts pourraient être oubliés. Une des spécifité de NextPriorityConcept est de définir des stratégies qui ne génèrent que des "candidats" pour le niveau suivant, ce qui permet de séparer le traitement des données, avec des stratégies spécifique à chaque type de donnée, du traitement global pour garantir une structure de treillis maintenue.

L'algorithme NextPriorityConcept utilise à nouveau des prédicats génériques, dits sélecteurs, pour définir les stratégies. Ces sélecteurs sont donc construit par ajout d'une information élémentaire aux prédicats de $\delta(A)$, et servent à sélectionner des sous-ensembles $A' \subset A$ pour générer un concept $(A', \delta(A'))$ candidat pour être prédécesseur de $(A, \delta(A))$.

Définition 4.2 (Stratégie)**.** Une stratégie σ^i est une application $\sigma^i : 2^G \to 2^P$ qui définit un ensemble de prédicats $\sigma^i(A)$ (sélecteurs) pour les caractéristiques de S^i à partir desquelles les prédécesseurs immédiats d'un concept (A, D) sont générés.

La stratégie finale σ est l'union des sélecteurs σ^i pour chaque caractéristique.

$$\sigma(A) = \bigcup_i \sigma^i(A)$$

Les sélecteurs peuvent proposer une restriction sur la description $\delta(A)$ ou sur l'ensemble A. Ils sont définis en fonction de chaque caractéristique de groupe S^i.

Plusieurs stratégies sont possibles pour générer les prédécesseurs immédiats d'un concept, allant de la stratégie naïve classiquement utilisée dans l'AFC qui considère tous les prédécesseurs immédiats possibles (tous les autres attributs pour le cas binaire), à des stratégies réduisant le nombre de prédécesseurs immédiats ce qui permet d'obtenir des treillis plus petits et qui oriente l'exploration dans une certain direction d'analyse en fonction de la stratégie utilisée.

Exemple 4.3. *Donnons les stratégies pour un sous ensemble d'objets $A \subseteq G$ de l'Exemple 4.1 :*

- *Pour $A = \{1, 2, 3\}$ et l'attribut numérique $S^1 = \{Age\}$:*
  - $\sigma^1(\{1, 2, 3\}) = \{$" est supérieur à 25.71"," est inférieur à 42.28"$\}$ *où* 25.71 *et* 42.28 *sont la moyenne plus ou moins l'écart type des valeurs de Age.*
- *Pour $A = \{1, 2, 3\}$ et l'attribut catégoriel $S^2 = \{Ville\}$:*

– $\sigma^2(\{1,2,3\}) = \{$" est contenu dans {Paris, La Rochelle}",
" est contenu dans {Paris, Lyon}",
" est contenu dans {La Rochelle, Lyon}"$\}$

Où les catégories sont obtenues en enlevant une valeur possible de la caractéristique "Ville". Ainsi, le sélecteur "est contenu dans {Paris, La Rochelle}" *permet de sélectionner de A les objets 1 et 2 alors que le sélecteur* "est contenu dans {Paris, Lyon}" *permet de sélectionner les objets 1 et 3.*

Il est également possible d'introduire une *méta-stratégie* (ou un filtre) sur les sélecteurs. Une méta-stratégie utilise une mesure pour filtrer les sélecteurs. La notion de mesure permet de sélectionner/filtrer les sélecteurs qui maximisent/minimisent une mesure définie sur l'ensemble A' des objets sélectionnés. Par exemple le support ou encore l'entropie qui nécessite la présence d'un attribut catégoriel de classe.

Exemple 4.4. *La méta-stratégie LimitFilter (LF) permet de ne garder que les sélecteurs avec un support supérieur à un seuil* $support_{min}$ *:*

$$\sigma_{LF}^{Sup}(A, support_{min}) = \{p \in \delta(A) \;\; tel\ que \;\; |\{a \in A | p(a), p \in \delta(A) \cup \{p\}\}| \; \geq support_{min}\}$$

4 L'algorithme NEXTPRIORITYCONCEPT

L'algorithme NEXTPRIORITYCONCEPT est donc un algorithme niveau par niveau qui va générer les prédécesseurs immédiats d'un concept $(A, \delta(A))$, où chaque prédécesseur immédiat potentiel $(A', \delta(A'))$ sera défini par un sélecteur. L'objectif est de calculer le treillis des concepts du contexte

$$\langle G, P, I_P \rangle$$

où P est l'ensemble des prédicats décrivant les caractéristiques, et $I_P = \{(a, p) \mid p(a)\}$ est la relation entre les objets et les prédicats. La difficulté principale est de garantir une structure de treillis, donc l'existance de borne inf. Pour cela, NEXTPRIORITYCONCEPT introduit un mécanisme de propagation de contraintes qui distingue entre les contraintes croisées entre concepts frères pour garantir l'existence de leurs bornes inf respectives, puis ces contraintes sont propagées d'un concept à ses prédécesseurs par les contraintes résiduelles.

Définition 4.3 (Mécanisme de propagation de contraintes)**.** Le mécanisme de propagation de contraintes est une application $\mathcal{C}[A]$ définie pour l'*intention* d'un concept (A, D) vers 2^P par $\mathcal{C}[A] = C_{residual} \cup C_{cross}$ où :

- $C_{residual}$ est l'ensemble des contraintes résiduelles issues du concept (A', D') qui a généré (A, D) :

$$C_{residual} = \mathcal{C}[A'] \backslash D$$

- C_{cross} est l'ensemble des contraintes croisées issues des autres prédécesseurs immédiats (A_i, D_i) de (A, D) :

$$C_{cross} = (\bigcup_i A_i \cap \sigma(A'))\backslash D$$

- et $\mathcal{C}[G] = \emptyset$

Les prédécesseurs immédiats d'un concept $(A, \delta(A))$ sont calculés en considérant les sélecteurs de $\delta(A)$ ainsi que ceux issus des contraintes de A, i.e, les sélecteurs $p \in \sigma(A) \cup C[A]$. Et, par extension du théorème de Bordat les prédécesseurs immédiats de $(A, \delta(A))$ sont les sous-ensembles maximaux par inclusion de :

$$\mathcal{FD}_{(A,D)} = \{\{a \in A \mid p(a)\} \mid p \in (\sigma(A) \cup \mathcal{C}[A])\} \tag{4.1}$$

L'algorithme 11 décrit l'algorithme NEXTPRIORITYCONCEPT. Il considère en entrée un ensemble de données $\langle G, S\rangle$ avec des caractéristiques organisées en groupes S^i, une description δ et une stratégie σ, et appelle l'algorithme PRÉDÉCESSEURS-IMMÉDIATS (Algorithme 12) qui calcule les prédécesseurs immédiats d'un concept (A, D).

La file de priorité stocke les concepts générés par ordre décroissant du support ; ce qui garantie une génération niveau par niveau, chaque concept sera ainsi généré avant ses prédécesseurs (lignes 3, 4 et 15). À chaque itération, un concept de support maximal est produit (ligne 10), puis ses prédécesseurs immédiats sont calculés (ligne 11). Les concepts sont donc générés niveau par niveau, en commençant par le concept maximal $(G, \delta(G))$. Chaque concept $(A, \delta(A))$ est composé d'un sous ensemble d'objets A avec leur description commune $\delta(A)$ (lignes 13 et 14 de l'Algorithme 12).

L'algorithme considère les sélecteurs $p \in \sigma(A) \cup C[A]$ (cf Equation 4.1, ligne 2 de l'Algorithme 12), et les prédécesseurs immédiats d'un concept $(A, \delta(A))$ sont calculés comme des sous-ensembles maximaux par inclusion (lignes 4-7 de l'Algorithme 12). La fonction INCLUSION-MAX présenté par l'Algorithme 13 permet de garder uniquement ces sous-ensembles maximaux.

Le théorème principal de [Demko et al., 2020] établie que NEXTPRIORITYCONCEPT calcule le treillis des concepts du contexte formel $\langle G, P, I_P\rangle$.

Théorème 4.1. *Si la description δ vérifie $\delta(A) \sqsubseteq \delta(A')$ pour $A' \subseteq A$ alors l'algorithme* NEXTPRIORITYCONCEPT *calcule les concepts du contexte formel $\langle G, P, I_P\rangle$.*

Le temps d'exécution de l'algorithme NEXTPRIORITYCONCEPT a une complexité temporelle de $O(|\mathcal{B}|\,|G|\,|P|^2\,(c_\sigma + c_\delta))$, où $\mathcal{B}$ est le nombre de concepts, c_σ est le coût de génération des sélecteurs de la stratégie et c_δ est le coût de génération des prédicats de la description. L'algorithme a une complexité en espace de $O(w\,|P|^2)$ où w est la largeur du treillis de concepts et P est l'ensemble des tous les prédicats.

L'algorithme NEXTPRIORITYCONCEPT est une approche de découverte de motifs où les prédicats sont découverts "à la volée", de manière locale pour chaque concept. Ceci est rendu possible par l'utilisation de la file de priorité pour garantir que chaque concept

Algorithm 11 NextPriorityConcept

Input : Un ensemble de données (G, S), une famille de characteristiques $(S^i)_{i \leq d}$, une description δ et une stratégie σ

Output : Les concepts (A, D)

1: $Result \leftarrow \emptyset$
2: *// la file de priorité*
3: $Q \leftarrow []$
4: $Q.push((|G|, (G, \delta(G))))$
5: *// la structure de données pour les contraintes*
6: $C[G] \leftarrow []$
7: *// la génération des prédécesseurs immédiats niveau par niveau*
8: **while** $Q \neq \emptyset$ **do**
9: $(A, D) \leftarrow Q.pop()$
10: $Result \leftarrow Result \cup (A, D)$
11: $LP \leftarrow$ Prédécesseurs-Immédiats$((A, D), C, \delta, \sigma)$
12: *// mettre à jour la file de priorité*
13: **for** $(A', D') \in LP$ **do**
14: **if** $(A', D') \notin Q$ **then**
15: $Q.push((|A'|, (A', D')))$;
16: **end if**
17: **end for**
18: *delete* $C[A]$
19: **end while**
20: *return Result*

Algorithm 12 PRÉDÉCESSEURS-IMMÉDIATS

Input : Un concept (A, D), l'ensemble des prédicats P, les contraintes C, la description δ et la stratégie σ

Output : L'ensemble des prédécesseurs immédiats LP

$L \leftarrow \emptyset$
for $p \in (\sigma(A) \cup C[A]) \backslash D$ **do**
    // p is a new potential selector
    $A' \leftarrow \{a \in A \mid p(a)\}$
    if $A' \subset A$ **then**
        $L \leftarrow$ INCLUSION-MAX$(L, (A', p))$;
    end if
end for
// initialise l'ensemble des nouvelles contraintes N
$N \leftarrow \{p \mid (A', p)\}$
$LP \leftarrow \emptyset$
for $(A', p') \in L$ **do**
    $D' \leftarrow \delta(A')$;
    $LP.add((A', D'))$;
    // calcule des contraintes
    $X \leftarrow \{p'' \in N \mid p''(a),\ \forall a \in A'\}$
    $C[A'] \leftarrow C[A'] \cup C[A] \cup N \backslash X$
end for
return LP

Algorithm 13 INCLUSION-MAX

Input : Un ensemble des extensions des prédécesseurs potentiels L, un ensemble d'objets A

Output : Les ensembles d'inclusion maximale $L \cup A$

$add \leftarrow True$
for $A' \in L$ **do**
    if $A \subset A'$ **then**
        $add \leftarrow False$
        break
    else
        $L.remove(A')$
    end if
end for
if *add* **then**
    $L.add(A)$
end if
Return L

est généré avant ses prédécesseurs et le mécanisme de propagation de contraintes pour s'assurer que les inf seront calculées et donc que la propriété de treillis est maintenue. Les prédicats sont donc bien adaptés aux différentes données et les treillis sont souvent plus petits, avec des concepts plus pertinents. De plus, l'utilisation de prédicats hétérogènes avec des stratégies et des descriptions spécialisées sur chaque type de caractéristique permet l'exploration de données complexes et hétérogènes.

Dans les approches classiques, les treillis de concepts sont souvent très volumineux. Bordat considère tous les attributs possibles restants ($b \in M \backslash B$) alors que NextPriorityConcept propose de sélectionner certains attributs ce qui permet de réduire le nombre de concepts générés. Les stratégies sont une manière de ne générer qu'un sous-ensemble de prédécesseurs parmi tous les prédécesseurs d'un concept, et donc d'obtenir un treillis de concepts plus petit.

Le résultat de l'algorithme NextPriorityConcept peut être interprété comme une structure de motifs sur chaque domaine de caractéristiques S^i. Il dispose d'un espace de description implicite $\mathcal{D}^i$ par δ^i pour chaque sous-ensemble S^i de caractéristiques, où la description $\delta^i(A)$ d'un ensemble $A \subseteq G$ est une traduction directe par les prédicats de sa description dans $\mathcal{D}^i$. Comme dans les structures de motifs, les descriptions doivent vérifier $\delta^i(A) \sqsubseteq \delta^i(A')$ pour $A' \subseteq A$ afin d'obtenir un treillis. Alors que les structures de motifs introduites dans le Chapitre 3 calculent les motifs globalement lors d'une étape de prétraitement, les prédicats sont découverts "à la volée", de manière locale pour chaque concept par NextPriorityConcept.

5 La Plateforme GALACTIC

La plateforme GALACTIC [1] (**GA**lois **LA**ttices, **C**oncept **T**heory, **I**mplicational systems and **C**losures) est écrite en python [2] et met en œuvre l'algorithme NextPriorityConcept qui permet de construire un treillis de concepts pour des données complexes et hétérogènes. La plateforme est architecturalement organisée avec un cœur contenant l'algorithme de base NextPriorityConcept, des extensions pour traiter plusieurs types de données, et des applications servant comme interface d'utilisation de la plateforme.

La plateforme GALACTIC est organisée comme suit :

1. https://galactic.univ-lr.fr
2. https ://www.python.org/

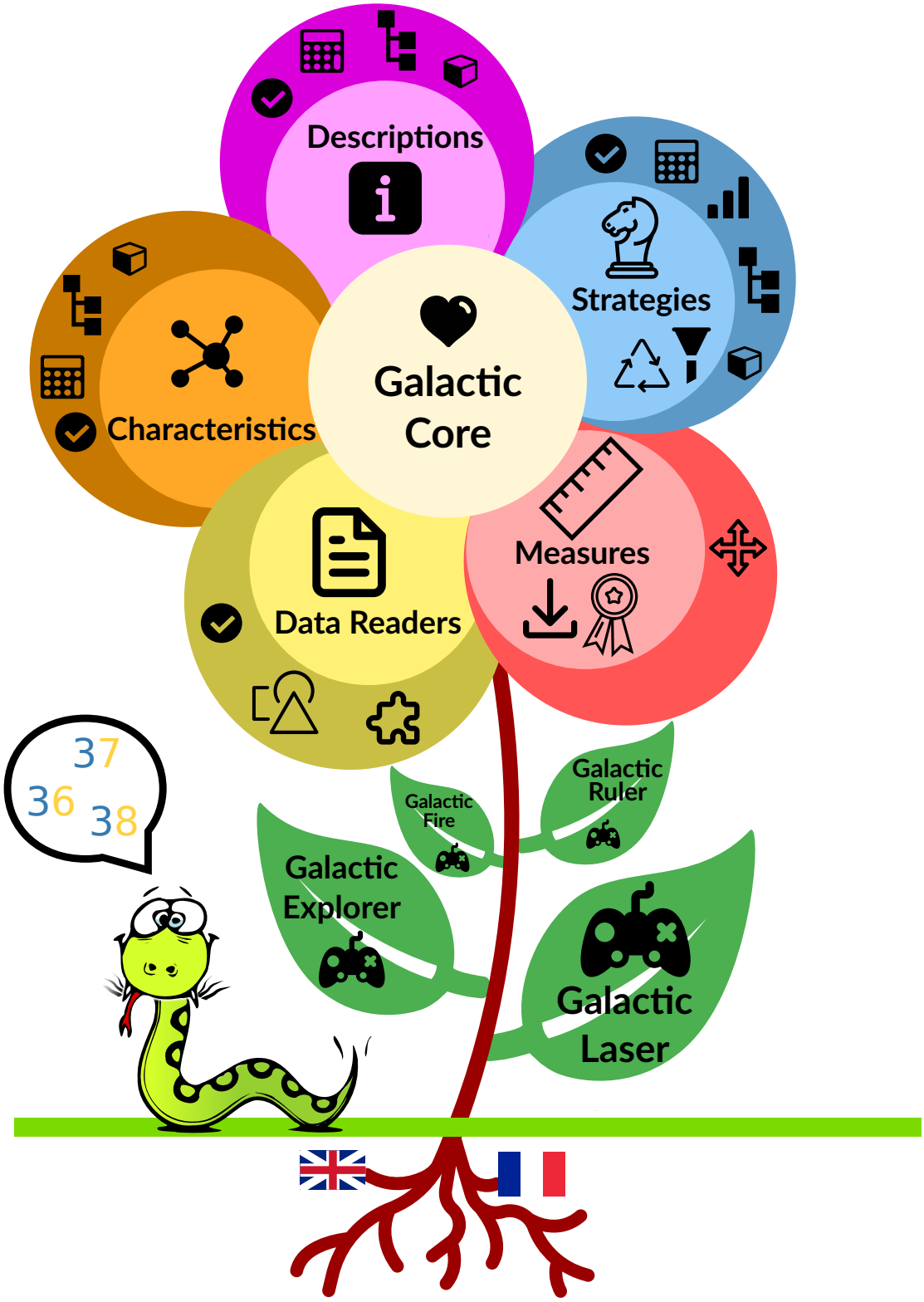


FIGURE 4.1 – l'architecture de GALACTIC

- Le cœur de la plateforme (`Galactic Core`) qui contient l'implémentation de l'algorithme NextPriorityConcept et plusieurs outils pour la manipulation algébrique de treillis.
- Des extensions de *caractéristiques* (pétale `Characteristics`) qui définissent des types de données.
- Des extensions de *descriptions* (pétale `Descriptions`) qui définissent des descriptions par des prédicats pour chaque caractéristique.
- Des extensions de *stratégies* (pétale `Strategies`) qui définissent des stratégies pour la génération des sélecteurs pour chaque caractéristique. Deux méta-stratégies sont définies :
  - La méta-stratégie *SelectionFilter* permet de prendre les sélecteurs qui maximisent/minimisent une mesure.
  - La méta-stratégie *LimitFilter* permet de garder les sélecteurs qui génèrent de prédécesseurs respectant un seuil minimal d'une mesure.
- Des extensions de *mesures* (pétale `Measures`) qui définissent des mesures utilisées par les méta-stratégies. Trois extensions de mesures sont définies :
  - La mesure de la confiance.
  - La mesure de support.
  - La mesure d'entropie.
- Des extensions de lecture de données (pétale `Data Readers`) qui permettent la lecture de plusieurs types de fichiers de données :
  - Extensions pour les données binaires (compatible avec les anciens formats de fichiers du l'AFC).
  - Extensions pour les données hétérogènes (*CSV*, *TOML*, *INI*).
  - Extensions pour les données complexes (*JSON*, *YAML*).
- Des applications qui utilisent le cœur et les extensions (feuilles).
  - `galactic-laser` permet de dessiner un diagramme de Hasse d'un treillis de concepts générés à partir d'un fichier de stratégies et un fichier de données.
  - `galactic-ruler` permet de générer les règles d'associations.
- Des extensions de localisation utilisées pour la traduction des applications en deux langues pour l'instant, anglais et français (racines).

Avec ce système d'extensions, la plateforme peut analyser plusieurs type de données. `GALACTIC` propose actuellement des extensions pour des données binaires, numériques et catégorielles :

Extensions pour des données binaires :

- une description classique décrivant un ensemble d'objets par un ensemble d'attributs communs.
- une stratégie consiste à générer des sélecteurs à l'aide de formules logiques impliquant un groupe de plusieurs attributs booléens et leurs négations [Quine, 1952].

Extensions pour des données numériques :

- une description numérique décrivant une collection d'objets par une enveloppe convexe [van de Vel, 1993] sur $\mathbb{R}^n$, c'est-à-dire un groupe de n caractéristiques numériques.
- deux types de stratégies possibles :

  La stratégie *Normal* consiste à restreindre l'ensemble d'objets en utilisant $m \pm \alpha\sigma$ sur la composante principale (à l'aide de techniques d'analyse en composantes principales). m est la moyenne, σ est l'écart-type et α est un paramètre d'entrée de la stratégie.

  La stratégie *Quantile* affine l'ensemble d'objets en utilisant un sous-groupe pour chaque quantile.

Extensions pour des données catégorielles :

- une description par le sous-ensemble minimal contenant les valeurs des catégories.
- une stratégie qui consiste à retirer une valeur de ce sous-ensemble permettant de sélectionner un ensemble d'objets plus petit.

6 Cas d'utilisation de l'algorithme NEXTPRIORITYCONCEPT

Considérons le jeu de données `Lenses` provenant de la plateforme d'apprentissage automatique de l'UCI [3]. Ce jeu de données est composé de 24 objets/patients décrits par 4 attributs catégoriels. Nous les considérons comme des attributs binaires, c'est-à-dire 9 attributs binaires.

- `âge du patient` (`G`) : `jeune` (`y`) ; `présbyte` (`pp`) ; `presbyopic` (`p`)
- `spectacle prescription` (`P`) : `myope` (`m`) ; `hypermétrope` (`h`)
- `Astigmate` (`A`) : `non` (`n`) ; `oui` (`y`)
- `Taux de production de larmes` (`T`) : `réduit` (`r`) `normal` (`n`)

3. `https://archive.ics.uci.edu`

et classés en 3 classes (`C`) :

- le patient doit être équipé de lentilles de contact rigides (`h`)
- le patient doit être équipé de lentilles de contact souples (`s`)
- le patient n'a pas à être équipé de lentilles de contact (`n`).

`Lenses` avec la mesure d'entropie

Nous considérons la description binaire classique d'un sous-ensemble A de patients comme un ensemble de prédicats. Pour chacun des 9 attributs binaires s la description utilisée est la description binaire classique d'un sous-ensemble A de patients par l'ensemble de ses attributs :

$$\delta(A) = \{s == x \mid s(a) = x \; \forall \text{ attribut } s, \forall a \in A\}$$

Nous utilisons l'entropie comme mesure avec la méta-stratégie *SelectionFilter*, une stratégie supervisée utilisant les informations de la classe $C(h, s, n)$:

$$\sigma_{\text{entropy}}(A) = \{s = b \mid b \in \bigcup_{S^i} \delta^i(A) \; : \; H_{\text{class}}(\beta(b)) \text{ minimal}\}$$

Afin de considérer l'entropie d'un prédécesseur A' de A, mais aussi l'entropie du complément de A' dans A, nous avons défini $H_{\text{class}}(\beta(b))$ par :

$$H_{\text{class}}(\beta(b)) = \theta H_{A'} + (1 - \theta) H_{A \setminus A'}$$

$H_{A'}$ et $H_{A \setminus A'}$ sont les entropies des ensembles A' et $A \setminus A'$. Pour un sous-ensemble A', la fonction $H_{A'}$ est définie par :

$$H_{A'} = \sum_{b \in C} p(A', b) log_2(p(A', b))$$

$p(A', b)$ est la probablilité d'apparition de la classe b dans A', définie par :

$$p(A', b) = \frac{|\{a \in A' \mid C(a) = b\}|}{|A'|}$$

L'approche AFC classique consisterait à considérer tous les attributs possibles comme une stratégie, et le treillis classique de concepts contient 110 concepts. Avec la stratégie d'entropie utilisant l'information de classe en ne conservant que les deux meilleures mesures d'entropie (avec $\theta = \frac{1}{2}$) pour les prédécesseurs, nous obtenons un treillis plus compact de 28 concepts présentés dans la Figure 4.2. Dans cette figure, \$ désigne le numéro de concept et # le support de concept. La représentation du treillis est réduite, affichant seulement les nouveaux prédicats et les nouveaux objets des concepts (cf Chapitre 3). Le concept \$0 contient 24 objets, avec comme prédicat : $C <= \{s, n, h\}$ signifiant que tous les patients sont dans l'une des trois classes possibles s, n ou h. Le concept \$3 contient

12 patients et deux prédicats de types binaires et catégoriel : $\{T == r\}$ et $C <= \{'n'\}$ signifiant que ces 12 patients ont un taux de production de larmes réduit (`r`), et sont tous de la classe n, c'est à dire qu'ils ne sont pas équipés de lentilles de contact.

Les 9 concepts \$3, \$9, \$15, \$17, \$18, \$10, \$14, \$23 et \$24 sont composés d'objets appartenant à la même classe, et peuvent être interprétés comme un regroupement des données en 9 groupes ; chaque concept (A, D) parmi ces 9 concepts correspondant à un cluster (c, D) où c est la classe des objets de A et signifiant que les objets vérifiant les prédicats D appartiennent à la classe c.

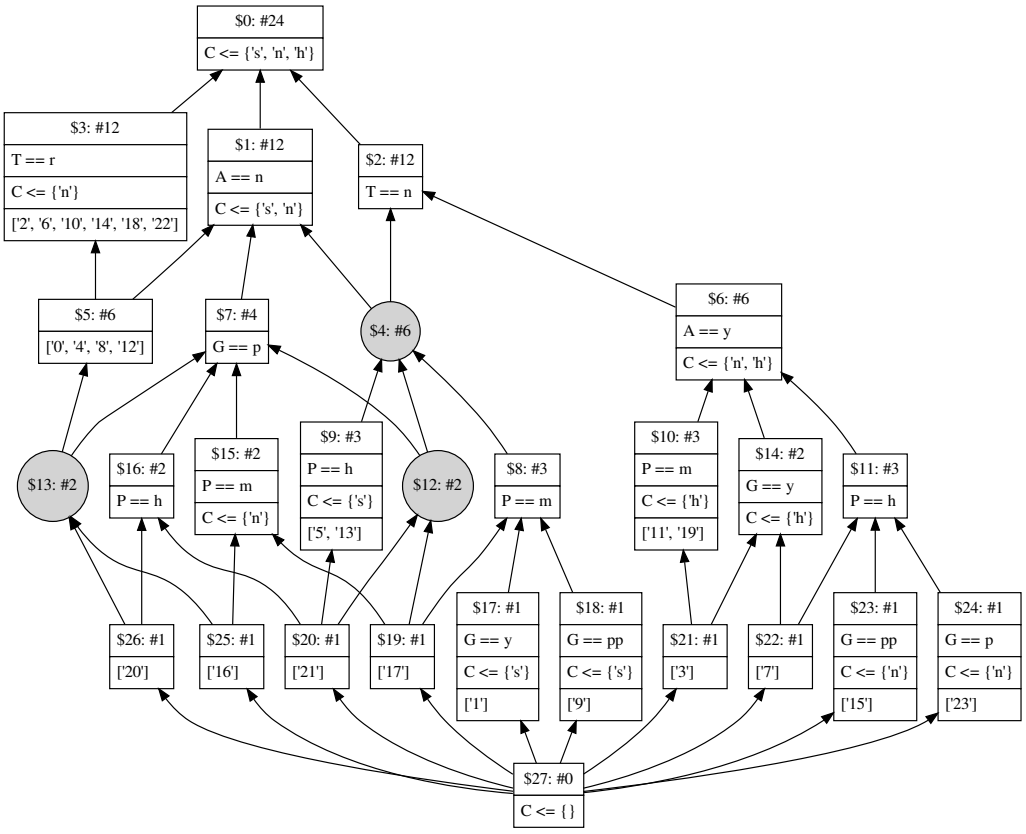


FIGURE 4.2 – Treillis pour le jeu de données `Lenses` avec la stratégie d'entropie

7 Conclusion et discussion

Nous avons présenté dans ce chapitre l'algorithme NEXTPRIORITYCONCEPT. Il est utilisé pour la génération de treillis de concepts avec des données complexes et hétérogènes en entrée. Cet algorithme rend l'analyse de données complexes et hétérogènes possible avec la définition des caractéristiques qui correspond aux attributs de tous types de données

qu'ils soient binaires ou non. Les prédicats monadiques décrivent ces données selon leurs types. Les descriptions génériques produisent un ensemble de prédicats pour chaque concept et les stratégies génériques fournissent des prédicats (sélecteurs) pour pouvoir générer les prédécesseurs d'un concept. Le cœur de l'algorithme est inspiré de l'algorithme de Bordat, avec utilisation d'une file de priorité et un mécanisme de propagation de contraintes pour assurer la génération de treillis. Cet algorithme va constituer la base de nos travaux.

L'algorithme NextPriorityConcept peut être également considéré comme une approche de fouille de motifs fermés où les motifs peuvent être de tous types. Alors que les algorithmes en fouille de motifs fermés sont souvent complexes et dédiés à un type de données, l'algorithme NextPriorityConcept généralise cette approche en séparant la génération niveau par niveau du traitement de chaque type de données de façon générique.

En résumé, nous avons présenté dans cette partie d'état de l'art les principaux algorithmes de fouille de motifs et de l'AFC pour les séquences. L'algorithme NextPriorityConcept et la plateforme `GALACTIC` combinent ces deux approches en résolvant deux verrous scientifiques. Le calcul `"à la volée"` des descriptions communes et la limitation du déluge de motifs avec les stratégies d'exploration.

La Figure 4.3 présente la relation entre l'algorithme NextPriorityConcept et les deux domaines de fouille de motifs et de l'AFC.

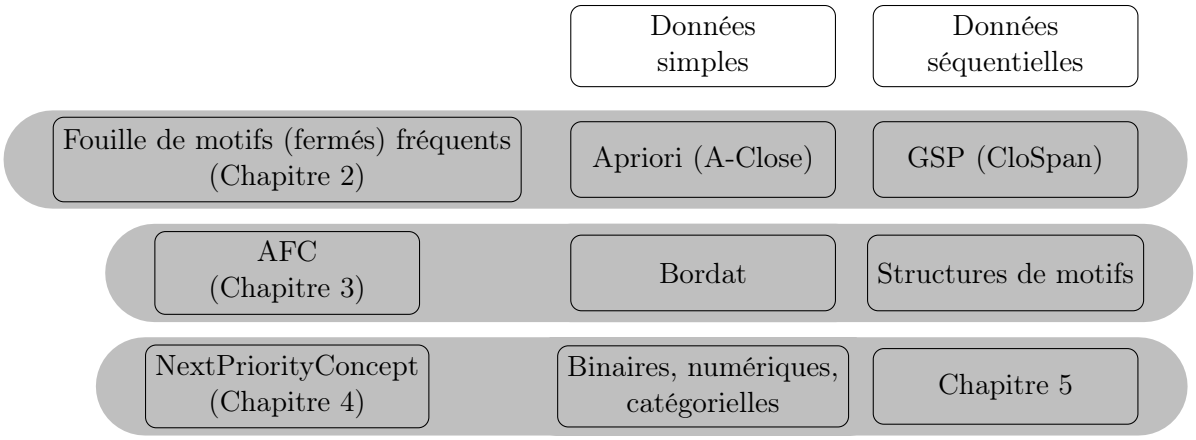


Figure 4.3 – Relation entre l'algorithme NextPriorityConcept et les deux domaines de fouille de motifs et de l'AFC

Nos contributions que nous allons présenter dans la partie suivante représentent une extension de l'algorithme NextPriorityConcept pour analyser des données séquentielles. Notre but est donc de définir des descriptions et des stratégies pour analyser des données séquentielles. Bien que l'algorithme NextPriorityConcept permette l'analyse de données complexes, cela nécessite un travail de formalisation et de définition des caractéristiques, prédicats, descriptions et stratégies qui est indispensable avant de proposer des méthodes d'analyse.

Dans la deuxième partie de ce manuscrit, nous détaillerons nos contributions qui

consistent à définir un formalisme pour analyser des séquences simples, des séquences temporelles et des séquences d'intervalles avec l'algorithme NextPriorityConcept (Chapitre 5). Une analyse quantitative de ces expérimentations seront menées avec de nouvelles mesures de qualité non supervisées pour mesurer la qualité des concepts/motifs/treillis obtenus (Chapitre 6).

Deuxième partie

Contributions

Chapitre 5

Analyse de séquences avec NextPriorityConcept

Sommaire

1 Introduction 92
2 Séquences simples 93
2.1 Descriptions 94
2.1.1 Description de Sous-séquences Communes Maximales (SCM) 94
2.1.2 Description de Sous-séquences Communes Préfixées (SCP) 95
2.1.3 Description de K-Sous-séquences Communes (KSC) . 97
2.2 Stratégies 98
2.2.1 La Stratégie Naïve (SN) 99
2.2.2 La Stratégie Augmentée (SA) 99
2.3 Exemples 99
3 Analyse de séquences temporelles 105
3.1 Descriptions 107
3.1.1 Description de Sous-séquences Distancielle Communes Maximales (SDCM) 108
3.1.2 Description SDCM avec Contrainte de *fenêtre* 108
3.1.3 Description SDCM avec Contrainte d'*écart* 110
3.2 Stratégies 110
3.2.1 Stratégie *Par étape* 111
3.2.2 La stratégie *Naïve* 111
3.2.3 La stratégie *Milieu* 111
3.3 Exemples 112
4 Analyse de séquences d'intervalles 117
4.1 Descriptions 119

4.1.1 Description de Sous-séquences d'intervalles Communes MAXimales (SCMAX). 119
4.1.2 Description de Super-séquences d'intervalles Communes MINimales (SCMIN). 120
4.2 Stratégies 122
4.2.1 Stratégies avec sélecteurs de sous-séquences d'intervalle 122
4.2.2 Stratégies avec des sélecteurs de super-séquences . . . 123
4.3 Exemples 124
5 Mesures de qualité 129
5.1 Stabilité Logarithmique Normalisée Globale 129
5.2 Représentabilité et Distinctivité 131
6 Les extensions de données séquentielles 132
6.1 Caractéristiques 132
6.2 Descriptions 133
6.3 Stratégies 134
7 Conclusion 136

1 Introduction

Nous présentons dans ce chapitre nos contributions principales en analyse de données séquentielles qui reposent à la fois sur la fouille de motifs séquentiels et l'AFC tels qu'ils ont été introduits dans l'état de l'art (Figure 4.3). Nous nous intéressons à des séquences de trois types : les séquences simples qui sont une succession d'éléments, les séquences temporelles où un horodatage est associé à chaque élément, et les séquences d'intervalles de temps, où les éléments de la séquence apparaissent dans un intervalle de temps. Nous introduisons aussi des mesures de qualité non supervisées pour montrer l'efficacité de notre approche.

Nous utilisons l'algorithme NextPriorityConcept pour analyser les données séquentielles. Afin d'analyser des séquences avec l'algorithme NextPriorityConcept, nous devons définir des descriptions et des stratégies pour un ensemble de séquences.

Une description δ est une application fournissant un ensemble de prédicats décrivant un ensemble de séquences A. Classiquement, un ensemble de séquences peut être décrit par l'ensemble des sous-séquences communes maximales. Cependant, d'autres descriptions sont envisageables selon le type de séquences et le besoin d'analyse. Formellement, une description δ est définie par :

Une description δ est une fonction $\delta : 2^G \rightarrow 2^P$ qui définit un ensemble de prédicats $\delta(A)$ décrivant un sous-ensemble $A \subseteq G$ de séquences. Les prédicats sont principalement basés sur la notion de sous-séquence commune maximale.

L'algorithme NextPriorityConcept raffine chaque concept $(A, \delta(A))$ avec des sous-ensembles de séquences $A' \subset A$ et des descriptions plus spécifiques $\delta(A')$ qui constituent

les prédécesseurs $(A', \delta(A'))$ du concept. Pour cela, l'algorithme introduit la notion de stratégie qui permet de réduire l'ensemble des objets du concept. Formellement, une stratégie σ est définie par :

Une stratégie σ est une fonction $\sigma : 2^G \rightarrow 2^P$ qui définit un ensemble de sélecteurs $\sigma(A)$ pour sélectionner des sous-ensembles stricts $A' \subset A$ candidats pour être prédécesseurs du concept $(A, \delta(A))$ dans le treillis des concepts.

Les prédicats et les sélecteurs sont calculés en utilisant la relation de sous-séquence $\sqsubseteq_s$ de la forme *"est une sous-séquence de"* pour les séquences simples, *"est une sous-séquence distancielle de"* pour les séquences temporelles et *"est une sous-séquence d'intervalles de"* pour les séquences d'intervalles. A noter que pour les séquences d'intervalles nous introduisons la relation inverse de super-séquence, ou la forme des prédicats/sélecteurs sera *"est une super-séquence d'intervalles de"*.

Les ensembles $\delta(A)$ et $\sigma(A)$ seront traités soit comme des ensembles de prédicats/sélecteurs : $\delta(A) = \{\text{prédicats } p = "S' \text{ est sous-séquence de } S" \text{ pour toutes } S \in A\}$ soit comme des ensembles de sous-séquences (super-séquences) : $\delta(A) = \{S' \mid \forall S \in A, S' \sqsubseteq_s S\}$ ($\delta(A) = \{S' \mid \forall S \in A, S \sqsubseteq_s S'\}$), ils peuvent réciproquement être déduits les uns des autres.

Selon le théorème 4.1, afin de garantir que l'algorithme NextPriorityConcept génère un treillis de concepts, toute description doit vérifier $\delta(A) \sqsubseteq \delta(A')$ pour $A' \subseteq A$.

La relation de subsomption entre deux ensembles de séquences (descriptions) étend la relation de sous-séquence entre deux séquences simples définie dans le Chapitre 1 (cf Définition 1.4). Rappelons qu'une séquence $S = \langle s_1, s_2 \dots s_n \rangle$ est une sous-séquence d'une séquence $R = \langle r_1, r_2 \dots r_m \rangle$ et on écrit $S \sqsubseteq_s R$ s'il existe des entiers $1 \leq i_1 < i_2 < \cdots < i_n \leq m$ tels que $s_j = r_{i_j}$ avec $j \leq n$.

La relation de subsomption entre deux ensembles de séquences est donc définie par :

Définition 5.1 (Subsomption entre ensembles de séquences)**.** Soient A et B deux ensembles de séquences, $A \sqsubseteq B \iff \forall S \in A : S \in B$ ou $\exists S' \in B$ tel que $S \sqsubseteq_s S'$.

Nos contributions portent sur la definition de descriptions et stratégies pour chacune des types de séquences. Nous avons également proposé des mesures de qualité des treillis générés afin de pouvoir comparer les différentes approches. Il s'agit de mesures non supervisées.

Ce chapitre est divisé en cinq sections ; trois sections pour l'analyse des trois types de données séquentielles (séquences simples, temporelles et d'intervalles), une section pour les mesures de qualité des concepts et des treillis et une section qui présente la partie implémentation dans l'outil GALACTIC. Enfin, nous finissons le chapitre par une discussion et conclusion.

2 Séquences simples

Dans cette section, nous proposons d'étendre l'algorithme NextPriorityConcept pour traiter des séquences simples. Rappelons la définition d'une séquence simple présentée

en Chapitre 2. Une séquence simple est définie par $S = \langle s_i \rangle_{i \leq n}$ où les $s_i (1 \leq i \leq n)$ sont des éléments appartenant à un alphabet Σ.

Nous proposons trois descriptions et deux stratégies pour des séquences simples.

2.1 Descriptions

La description la plus triviale pour un ensemble de séquences est l'ensemble des sous-séquences communes maximales. D'autres descriptions sont aussi envisageables. Nous définissons également une description par préfixe qui génère le préfixe commun maximal d'un ensemble de séquences afin de pouvoir déterminer des événements qui apparaissent en premier dans un ensemble de séquences. Les préfixes permettent par exemple d'identifier des directions dans des séquences de déplacements à partir d'un même point de départ, comme dans un musée par exemple. On peut définir de la même manière une description par suffixe, où nous nous intéressons aux fins des séquences plutôt qu'au début. Les préfixes et les suffixes sont des cas spéciaux des sous-séquences. Nous introduisons également une description par sous-séquences de taille fixée Nous proposons donc trois descriptions pour les séquences simples. La description *Sous-séquences Communes Maximales (SCM)*, la description *Sous-séquence Communes Préfixées (SCP)* et la description *K-Sous-séquences Communes (KSC).*

2.1.1 Description de Sous-séquences Communes Maximales (SCM)

Définition 5.2 (description SCM)**.** Pour un ensemble de séquences $A \subseteq G$, la description SCM est définie par :

$$\delta_{\text{SCM}}(A) = \{X \in \Sigma^* \mid \forall S \in A, X \sqsubseteq_s S, \text{ X est maximale } \} \tag{5.1}$$

Où X est maximale dans le cas où si $\exists X' \in \delta_{\text{SCM}}(A)$ tel que $X \sqsubseteq_s X'$ alors $X = X'$.

Cette description génère l'ensemble des sous-séquences communes maximales pour un ensemble de séquences A. Les prédicats sont de la forme *"X est une sous-séquence de S"* pour toute sous-séquence (prédicat) $X \in \delta_{\text{SCM}}(A)$ et toute séquence $S \in A$.

L'algorithme de génération des sous-séquences communes maximales est décrit dans l'Algorithme 14. Cet algorithme sélectionne d'abord les sous-séquences de taille 1 communes à toutes les séquences de A (Σ^+) (lignes 2-13). Ensuite, il initialise un ensemble de candidats avec la sous-séquence vide $\langle\rangle$ (ligne 15). Tant que il y a encore des candidats (ligne 17), il considère chaque candidat (ligne 18) et il teste si l'ajout d'un élément de Σ^+ (lignes 20-33) à chaque candidat (ligne 24) génère une sous-séquence commune à toutes les séquences de A (lignes 22-28), sinon il arrête (ligne 26) et la sous-sequence ainsi obtenue devient une sous-séquence commune maximale (lignes 34-36). Au final, l'ensemble F contient les sous-séquences communes à toutes les séquences dans A. Un autre parcours de F est nécessaire pour ne conserver que les maximaux, la fonction *Maximal* parcourt tous simplement l'ensemble F et supprime les séquences non maximaux.

Complexité 5.1. *La complexité de la description SCM est* $c_\delta^{SCM} \leq O(n|\delta(A)||\Sigma||A|)$ *où* n *est la longueur de la sous-séquence de taille maximale dans* $\delta(A)$.

La complexité de la description SCM dépend du résultat généré, à savoir le nombre de sous-séquences communes maximales qui est exponentiel. Nous pouvons constater que pour générer une sous-séquence commune maximale de taille k, il faut passer par $k-1$ sous-séquences qui sont obtenues niveau par niveau à partir de la séquences de taille 1. Donc le nombre de candidats est bornée par le nombre de sous-séquences communes maximales multiplié par la taille de la sous-séquence la plus longue de $\delta(A)$ noté n, i.e. $|C| = n|\delta(A)|$. En ajoutant les deux boucles internes on obtient la complexité de la description SCM.

Proposition 5.1. *Pour $A' \subseteq A \subseteq G$, nous avons $\delta_{SCM}(A) \sqsubseteq \delta_{SCM}(A')$*

Preuve 5.1. *Soit A et A' deux sous-ensembles de G tels que $A' \subseteq A$. Soit $C \in \delta_{SCM}(A)$ une sous-séquence maximale de A. Comme $A' \subseteq A$, on peut déduire que C est aussi une sous-séquence des séquences de A', mais pas nécessairement maximale. Si C est une sous-séquence maximale dans A' alors $C \in \delta_{SCM}(A')$. Sinon, il existe $C' \in \delta_{SCM}(A')$ tel que C est une sous-séquence de C'. Alors chaque sous-séquence dans $\delta_{SCM}(A)$ est soit présente dans $\delta_{SCM}(A')$ soit elle est sous-séquence d'une séquence de $\delta_{SCM}(A')$. Dans ces deux cas, on peut en déduire que, $\delta_{SCM}(A) \sqsubseteq \delta_{SCM}(A')$.*

2.1.2 Description de Sous-séquences Communes Préfixées (SCP)

Définition 5.3 (description SCP)**.** Pour un ensemble de séquences $A \subseteq G$, la description SCP est définie par :

$$\delta_{\text{SCP}}(A) = \{X \in \Sigma^* \mid \forall S \in A, X \text{ préfixe de } S,\ \text{X est maximale } \} \tag{5.2}$$

La description génère le préfixe commun maximal pour l'ensemble de séquences A. Les prédicats sont donc de la forme *"S commence par la séquence X "* pour tout $X \in \delta_{\text{SCP}}(A)$ et $S \in A$.

L'algorithme 15 décrit l'algorithme de calcul de cette description. On prend la séquence Q la plus courte de l'ensemble A (lignes 1-6). On commence par un préfixe vide $P = \langle\rangle$ (ligne 7) et à chaque itération (ligne 9 à 16) on vérifie si un élément de Q est dans la même position dans toutes les séquences de A (lignes 10 à 14), sinon l'algorithme retourne le préfixe P.

Complexité 5.2. *La complexité de la description SCP est $c_\delta^{SCP} = O(n|A|)$, où n est la longueur de la séquence de taille minimale dans A.*

Proposition 5.2. *Pour $A' \subseteq A \subseteq G$, nous avons $\delta_{SCP}(A) \sqsubseteq \delta_{SCP}(A')$*

La preuve est similaire à celle de la description de la sous-séquence commune maximale δ_{SCM}, les préfixes étant des sous-séquences particulières.

Algorithm 14 L'algorithme SCM

Entrée : un ensemble de séquences A définies sur un alphabet Σ
Sortie : l'ensemble de prédicats de la description SCM $\delta_{\text{SCM}}(A)$

1: *// parcours de Σ pour sélectionner seulement les éléments (sous-séquences de taille 1) communes à toutes les séquences de A*
2: $\Sigma^+ \leftarrow \emptyset$
3: **for** $a \in \Sigma$ **do**
4: $\quad subsequence \leftarrow True$
5: $\quad$ **for** $S \in A$ **do**
6: $\quad\quad$ **if** $a \not\sqsubseteq_s S$ **then**
7: $\quad\quad\quad subsequence \leftarrow False$
8: $\quad\quad\quad Break$
9: $\quad\quad$ **end if**
10: $\quad$ **end for**
11: $\quad$ **if** *subsequence* **then**
12: $\quad\quad \Sigma^+ \leftarrow \Sigma^+ \cup a$
13: $\quad$ **end if**
14: **end for**
15: *// Σ^+ contient seulement les sous-séquences communes de taille 1*
16: $C \leftarrow \{\langle\rangle\}$
17: $F \leftarrow \emptyset$
18: **while** $C \neq \emptyset$ **do**
19: $\quad SSeq \leftarrow C.pop()$
20: $\quad forward \leftarrow False$
21: $\quad$ **for** $a \in \Sigma^+$ **do**
22: $\quad\quad subsequence \leftarrow True$
23: $\quad\quad$ **for** $S \in A$ **do**
24: $\quad\quad\quad$ *// $(SSeq + a)$ est la concaténation de a à $SSeq$*
25: $\quad\quad\quad$ **if** $(SSeq + a) \not\sqsubseteq_s S$ **then**
26: $\quad\quad\quad\quad subsequence \leftarrow False$
27: $\quad\quad\quad\quad Break$
28: $\quad\quad\quad$ **end if**
29: $\quad\quad$ **end for**
30: $\quad\quad$ **if** *subsequence* **then**
31: $\quad\quad\quad C \leftarrow C \cup (SSeq + a)$
32: $\quad\quad\quad forward \leftarrow True$
33: $\quad\quad$ **end if**
34: $\quad$ **end for**
35: $\quad$ **if** $forward = False$ **then**
36: $\quad\quad F \leftarrow F \cup SSeq$
37: $\quad$ **end if**
38: **end while**
39: **Return** $Maximal(F)$

Algorithm 15 L'algorithme SCP

Entrée : un ensemble de séquences A définies sur un alphabet Σ
Sortie : la description SCP $\delta_{\text{SCP}}(A)$

```
Q ← première séquence de A
for S ∈ A do
    if |S| < |Q| then
        Q ← S
    end if
end for
P ← ⟨⟩
n ← |Q|
for i ≤ n do
    for S ∈ A do
        if q_i ≠ s_i then
            Break
        end if
    end for
    P ← P + q_i
end for
Return {P}
```

2.1.3 Description de K-Sous-séquences Communes (KSC)

Définition 5.4 (description KSC)**.** Pour un ensemble de séquences $A \subseteq G$ et une longueur k, la description KSC est définie par :

$$\delta_{\text{KSC}}(A, k) = \{X \in \Sigma^* \mid \forall S \in A, X \sqsubseteq_s S \text{ et } |X| = k\} \tag{5.3}$$

Cette description génère l'ensemble des k-sous-séquences communes maximales pour l'ensemble des séquences A. Les prédicats sont donc de la forme *"X est une sous-séquence de S"* pour tout $X \in \delta_{\text{KSC}}(A)$ et $S \in A$.

L'algorithme 16 décrit le calcul des k-sous-séquences communes. L'algorithme est simple car nous avons opté par l'approche naïve, nous générons toutes les sous-séquences possibles de la séquence s de taille minimale de A. Ensuite, nous prenons les sous-séquences de taille k et nous vérifions s'il s'agit des sous-séquences de toutes les séquences dans A.

Complexité 5.3. *La complexité de la description KSC est $c_\delta^{KSC} = O(|A|.|sub|)$, ou sub est l'ensemble de toutes les sous-séquences de s.*

Proposition 5.3. *Pour $A' \subseteq A \subseteq G$ et une longueur k, nous avons $\delta_{KSC}(A, k) \sqsubseteq \delta_{KSC}(A', k)$*

Preuve 5.2. *Soit A et A' deux sous-ensembles de G tels que $A' \subseteq A$. Pour δ_{KSC} la relation de subsomption peut être affinée par la relation d'inclusion car il est évident que*

Algorithm 16 L'algorithme KSC

Entrée : un ensemble de séquences A définies sur un alphabet Σ et un longueur k
Sortie : l'ensemble de prédicats de la description KSC $\delta_{\text{KSC}}(A, k)$

```
1: F ← ∅
2: seq ← Minimale(A)
3: K ← ∅
4: for sub ∈ AllSubsequences(seq) do
5:     if |sub| = k then
6:         isSub ← True
7:         for s ∈ A/seq do
8:             if sub ∉ s then
9:                 Break
10:                isSub ← False
11:            end if
12:        end for
13:        if isSub then
14:            F ← F ∪ sub
15:        end if
16:    end if
17: end for
18: Return F
```

les k-sous-séquences d'un ensemble A de séquences sont également des k-sous-séquences de tout sous-ensemble A' de A. Alors $\delta_{KSC}(A) \subseteq \delta_{KSC}(A')$ et donc $\delta_{KSC}(A) \sqsubseteq \delta_{KSC}(A')$.

Exemple 5.1. *La Table 5.1 présente quelques exemples des trois descriptions SCM, SCP et KSC, pour les séquences de l'Exemple 1.1.*

A	$\delta_{\text{KSC}}(A, k=2)$	$\delta_{\text{KSC}}(A, k=3)$	$\delta_{\text{SCP}}(A)$	$\delta_{\text{SCM}}(A)$
$\{1,2,3\}$	$\{\langle T, C\rangle\}$	$\{\}$	$\{\langle T\rangle\}$	$\{\langle T, C\rangle\}$
$\{1,2\}$	$\{\langle T, C\rangle, \langle T, Sp\rangle, \langle C, Sp\rangle\}$	$\{\langle T, C, Sp\rangle\}$	$\{\langle T\rangle\}$	$\{\langle T, C, Sp\rangle\}$
$\{4,5\}$	$\{\langle T, Sp\rangle, \langle C, Sp\rangle, \}$	$\{\}$	$\{\langle\rangle\}$	$\{\langle T, Sp\rangle, \langle C, Sp\rangle\}$

TABLE 5.1 – Quelques descriptions pour les séquences de l'Exemple 1.1

2.2 Stratégies

Une stratégie raffine chaque concept $(A, \delta(A))$ en des concepts prédécesseurs contenant moins de séquences et donc des descriptions plus spécifiques. Nous introduisons ici deux stratégies pour les séquences simples. La stratégie *Naïve (SN)* permet la génération de tous les prédécesseurs d'un concept, tandis que la stratégie *Augmentée (SA)* vise à augmenter les descriptions $\delta(A)$ par ajout d'un élément de l'alphabet.

2.2.1 La Stratégie Naïve (SN)

La stratégie *Naïve* génère toutes les sous-séquences possibles ayant un support inférieur à 1.

Définition 5.5 (stratégie SN)**.** On désigne par $SS(A)$ l'ensemble des sous-séquences de A. La stratégie *Naïve* est définie pour un ensemble de séquences $A \subseteq G$ par :

$$\sigma_{\text{SN}}(A) = \{x \mid x \in SS(A), support(x) < 1\} \tag{5.4}$$

2.2.2 La Stratégie Augmentée (SA)

La stratégie *Augmentée* consiste à ajouter un élément de l'alphabet aux sous-séquences communes actuelles de la description.

Définition 5.6 (stratégie SA)**.** La stratégie *Augmentée* est définie pour un ensemble de séquences $A \subseteq G$ et une description δ de séquences par :

$$\sigma_{\text{SA}}(A, \delta) = \{x + a \mid x \in \delta(A),\ a \in \Sigma\} \tag{5.5}$$

La stratégie *Naïve* est utilisée seulement avec les descriptions SCM et KSC. Elle génère toute sous-séquence permettant de réduire un ensemble de séquences A et un ensemble plus petit A'. Avec la description SCM, la stratégie *Naïve* génère tous les prédécesseurs possible d'un concept et donc elle permet de générer tous les concepts possibles du treillis. Tandis qu'avec la description KSC nous obtenons les concepts avec des sous-séquences communes de taille k. Par contre, cela n'aurait pas de sens d'utiliser la stratégie *Naïve* avec la description SCP car nous avons seulement besoin d'augmenter la description par l'ajout d'un élément de l'alphabet pour générer tous les prédécesseurs possibles d'un concept. L'utilisation de la description SCP avec la stratégie *Naïve* donne le même treillis si nous utilisons la stratégie *Augmentée* mais avec une complexité algorithmique plus grande.

La stratégie *Augmentée* est principalement utilisée avec les descriptions SCM et SCP. Avec la description SCM, elle génère moins de concepts car les sélecteurs sont obtenues en ajoutant un élément de l'alphabet à des sous-séquences. Avec la description SCP, la stratégie *Augmentée* est efficace car elle permet une exploration des chemins. Cette stratégie n'est pas utilisé avec la description KSC car les tailles des sous-séquences obtenues avec la description KSC sont fixes.

Exemple 5.2. *La Table 5.2 donne quelques prédicats générés pour les séquences* $\{1, 2, 3, 5\}$ *prisent de l'Exemple 1.1 avec les deux stratégies* σ_{SA} *et* σ_{SN}*.*

2.3 Exemples

Déroulons maintenant nos descriptions et stratégies d'analyse de séquences simples sur l'Exemple 1.1. Chaque combinaison description/stratégie est à choisir en fonction de l'analyse souhaitée.

A	$\delta_{\text{SCM}}(A) = \{\langle T, C\rangle\}$	$\delta_{\text{KSC}}(A, k=2) = \{\langle T, C\rangle\}$
$\{1, 2, 3, 5\}$	$\sigma_{\text{SA}}(A)$	$\sigma_{\text{SN}}(A)$
	$\{\langle T, C, T\rangle, \langle T, C, C\rangle, \langle \mathbf{T}, \mathbf{C}, \mathbf{L}\rangle, \langle T, C, S\rangle\}$ $\langle T, C, Sp\rangle\}$	$\{\langle \mathbf{C}, \mathbf{L}\rangle, \langle C, S\rangle, \langle C, Sp\rangle,$ $\langle S, C\rangle, \langle S, L\rangle, \langle S, Sp\rangle, \langle Sp, L\rangle,$ $\langle \mathbf{T}, \mathbf{L}\rangle, \langle T, S\rangle, \langle T, Sp\rangle\}$

TABLE 5.2 – Quelques exemples des stratégies SN et SA pour les séquences 1,2,3 et 5 de l'Exemple 1.1

La description SCM et la stratégie SN correspondent à l'analyse classique de la fouille de motifs séquentiels fermés proposée par exemple par les algorithmes CloSpan [Yan et al., 2003], BIDE [Wang and Han, 2004], etc. Avec ce type d'analyse, nous obtenons un treillis volumineux avec beaucoup de concepts. Une réduction du nombre de concepts est possible en conservant la description SCM et en utilisant la stratégie SA. Avec les descriptions SCP ou KSC, nous obtenons d'autres types de prédicats et donc de motifs. C'est à l'analyste de données de choisir la combinaison description/stratégie dans une approche de découverte de connaissances qui soit plus adaptée à ses attentes.

La Table 5.3 présente un résumé des figures contenant les diagrammes de Hasse des treillis générés donnés sous forme réduite (cf Chapitre 3) pour chaque combinaison description/stratégie.

Descriptions / Stratégies	δ_{SCM}	δ_{SCP}	δ_{KSC}
σ_{SN}	Fig 5.1		Fig 5.4 5.5
σ_{SA}	Fig 5.2	Fig 5.3	

TABLE 5.3 – Combinaisons description/stratégie avec les figures des treillis générés

Description SCM avec la stratégie SN. La Figure 5.1 donne le treillis de concepts généré avec la description SCM et la stratégie SN. 10 concepts/motifs sont générés, correspondant à des sous-groupes de séquences décrits par leurs sous-séquences communes maximales. Le concept \$1 contient 4 objets (séquences $1, 2, 4$ et 5), décrits par les deux prédicats *sequence match* $\langle T, Sp\rangle$ et *sequence match* $\langle C, Sp\rangle$ qui se traduit par : Ces 4 séquences possèdent comme sous-séquences communes maximales $\{\langle T, Sp\rangle, \langle C, Sp\rangle\}$. Le concept \$2 (séquences $1, 2, 3$ et 5) contient $\langle T, C\rangle$ comme sous-séquence commune maximale.

Ce type d'analyse permet d'identifier tous les concepts possibles, tous les sous-groupes de séquences partageant les mêmes sous-séquences maximales. Il est clair que les prédicats du treillis correspondent à toutes les sous-séquences maximales, la description SCM génère des sous-séquences maximales et la stratégie SN considère tous les sous-ensembles possibles de séquences.

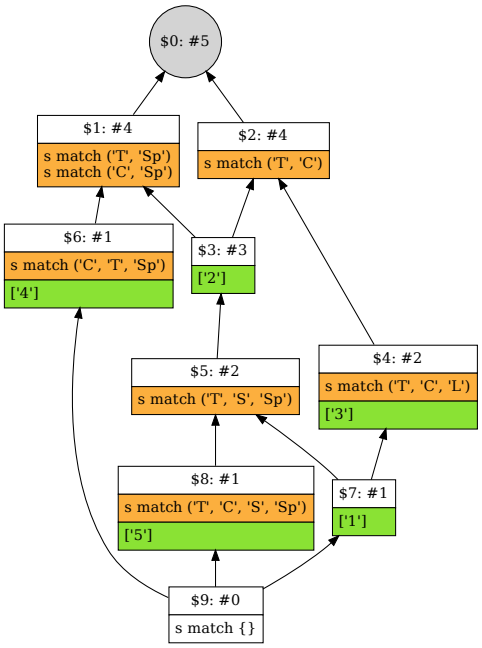


FIGURE 5.1 – Diagramme de Hasse du treillis de concepts généré par la description SCM et la stratégie SN

Description SCM avec la stratégie SA. La Figure 5.2 montre le treillis de concepts généré avec la description SCM et la stratégie SA. Seulement 6 concepts/motifs sont générés. On retrouve les concepts \$0, \$1, \$2, \$3, \$4 et \$7 de la Figure 5.1. Les concepts \$5, \$6, \$8, \$9 ont disparus car la stratégie SA ne génère pas des prédicats les décrivants.

En gardant la même description SCM avec la stratégie SA, cela permet de réduire le nombre de concepts et de motifs et d'obtenir des treillis réduit plus faciles à lire.

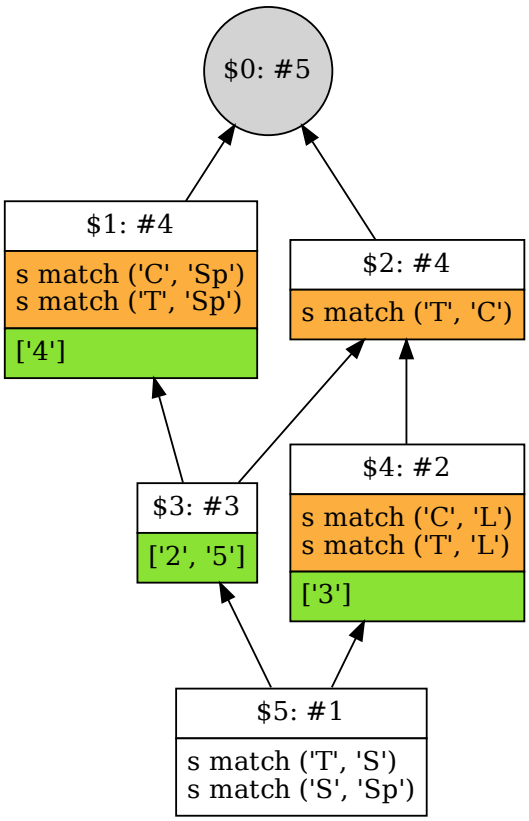


FIGURE 5.2 – Diagramme de Hasse du treillis de concepts généré avec la description SCM et la stratégie SA

Description SCP avec la stratégie SA. La Figure 5.3 montre le treillis de concepts généré avec la description SCP et la stratégie SA. 6 concepts/motifs sont générés. Le concept \$1 montre que les séquences $2, 3, 5$ et 1 commencent toutes par le préfixe T, cela se traduit par le prédicat $s\,starts\,with\,\langle T\rangle$. Pour le concept \$2, les séquences $2, 3$ et 5 commencent par $\langle T, C\rangle$.

La description SCP représente un ensemble de séquences avec leur préfixe commun. Chaque concept contient un seul préfixe car un sous-ensemble de séquences contient un seul préfixe commun maximal ; le concept top contient le préfixe vide $\langle\rangle$. Le treillis ressemble à un arbre de décision : toute chaine de concept entre le top et le bottom correspond à une incrémentation du préfixe commun. Et deux concepts non comparables, qui correspondent à deux prefixes différents, possédent nécessairement le bottom comme borne inférieure.

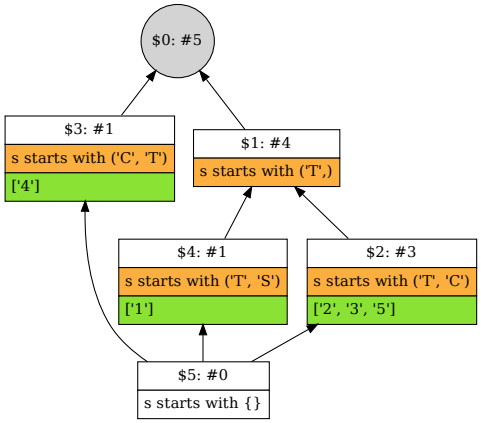


FIGURE 5.3 – Diagramme de Hasse du treillis de concepts généré avec la description SCP et la stratégie SA

Description KCS avec la stratégie SN. Les Figures 5.4 et 5.5 montrent les deux treillis de concepts générés avec la description KSC et la stratégie SN avec $k = 2$ et $k = 3$ respectivement. Nous pouvons observer que nous obtenons moins de concepts avec $k = 3$ (8 concepts) qu'avec $k = 2$ (10 concepts). Le concept \$2 de la Figure 5.5 contient la sous-séquence de taille 3 : $\langle T, C, L\rangle$ qui est présente dans le concept \$4 de la Figure 5.4 avec trois sous-séquences de taille 2 : $\{\langle T, C\rangle, \langle T, L\rangle, \langle C, L\rangle\}$.

Cette description permet de cibler l'analyse vers des sous-séquences de taille fixe ce qui présente un avantage dans les cas où on cherche des motifs d'une taille précise. L'augmentation de k réduit le nombre de concepts mais augemnte le nombre de prédicats.

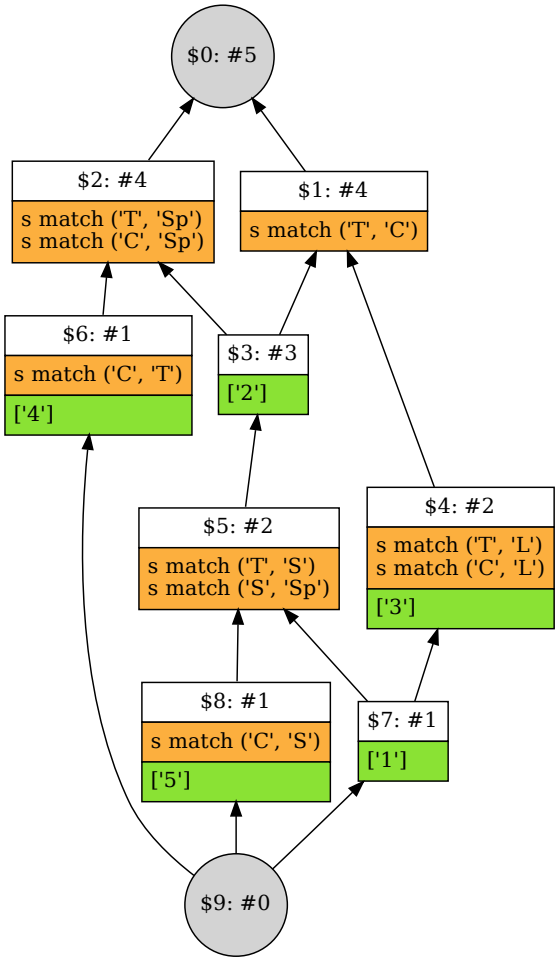


FIGURE 5.4 – Diagramme de Hasse du treillis de concepts généré avec la description KSC et la stratégie SN avec $k = 2$

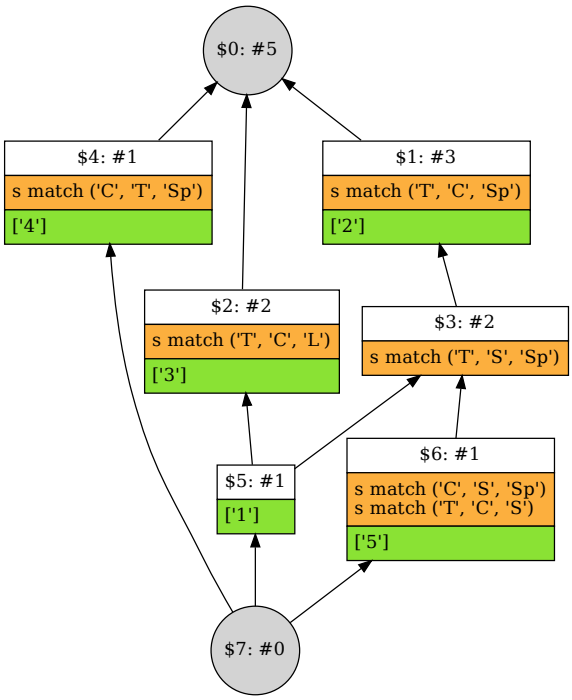


FIGURE 5.5 – Diagramme de Hasse du treillis de concepts généré avec la description KSC et la stratégie SN avec $k = 3$

3 Analyse de séquences temporelles

Dans cette section nous utilisons l'algorithme NEXTPRIORITYCONCEPT pour analyser des séquences temporelles. Les descriptions des séquences temporelles sont également basées sur la notion de sous-séquences communes maximales, mais avec en plus une information temporelle. L'information temporelle est introduite par des intervalles représentant un écart ou un retard entre les éléments des sous-séquences.

Rappelons la définition d'une séquence temporelle présenté en Chapitre 2. Une séquence temporelle est définie pour un alphabet Σ par $S = \langle (t_i, s_i) \rangle_{i \leq n}$, où chaque élément $s_i \in \Sigma$ est associé à un horodatage t_i. Dans ce manuscrit nous considérons $t_i \in \mathbb{N}^+$, mais dans le cas générale nous pouvons considérer $t_i \in \mathbb{R}$.

Nous définissons une séquence distancielle à partir d'une séquence temporelle comme une séquence qui contient l'écart temporel entre deux éléments consécutifs de la séquence (Figure 5.6). Cette définition est similaire à la notion de motif *delta* introduite dans le Chapitre 2.

Définition 5.7 (Séquence distancielle)**.** La séquence distancielle $d(S)$ d'une séquence temporelle S est définie par $d(S) = \langle (s_i, \lfloor d_i^s \rceil), (s_n) \rangle_{i<n}$ avec $d_i = t_{i+1} - t_i$.

La relation de sous-séquence distancielle se définit alors :

Définition 5.8 (Sous-séquence distancielle)**.** Pour deux séquences temporelles $A = \langle (t_i^a, a_i) \rangle_{i \leq n}$, et $B = \langle (t_i^b, b_i) \rangle_{i \leq m}$, la séquence distancielle $d(A) = \langle (a_i, \lfloor d_i^a \rceil), (a_n) \rangle_{i<n}$ est une sous-séquence distancielle de $d(B) = \langle (b_i, \lfloor d_i^b \rceil), (b_m) \rangle_{i<m}$ et on écrit $d(A) \sqsubseteq_s d(B)$ s'il existe des entiers $1 \leq i_0 < \cdots < i_n \leq m$ tels que $a_k = b_{i_k}$ et $d_k^a \leq t_{i_{k+1}}^b - t_{i_k}^b$ pour $k < n$.

La définition de subsomption entre deux ensembles de séquences (Définition 5.1) s'étend naturellement aux séquences distancielles.

Pour un ensemble de séquences temporelles A, l'ensemble de toutes les sous-séquences distancielles communes peut être très volumineux car les distances entre les éléments varient d'une séquence à l'autre, et donc chaque valeur possible de distance entre les éléments va former une nouvelle sous-séquence distancielle. Pour faire face à ce problème, nous introduisons l'ensemble des **Sous-séquences Distancielles Communes** $SDC(A)$ avec un intervalle $\lfloor d_i^{min}, d_i^{max} \rceil$ de toutes les distances possibles entre les éléments consécutifs des sous-séquences :

Définition 5.9 (Sous-séquences Distancielles Communes $SDC(A)$)**.** Un ensemble de sous-séquences distancielles communes SDC est définie pour un ensemble de séquences temporelles A par :

$$\begin{aligned} SDC(A) = \{R \mid R = \langle (r_i, \lfloor d_i^{r,min}, d_i^{r,max} \rceil), (r_{n_r}) \rangle_{i<n_r} \text{ où }, \\ \forall S \in A, \exists \text{ des entiers } 1 \leq i_0 < \cdots < i_n \leq |S| \text{ tels que} \\ r_k = s_{i_k} \text{ et } d_k^{r,min} = min(t_{i_{k+1}}^s - t_{i_k}^s) \text{ et } d_k^{r,max} = max(t_{i_{k+1}}^s - t_{i_k}^s) \text{ pour } k < n\} \end{aligned} \quad (5.6)$$

Exemple 5.3. *La Figure 5.6 présente les séquences distancielles d'actions quotidiennes de l'Exemple 1.2. Pour* $S_5' = \langle (11, T), (13, S), (16, Sp) \rangle$ *nous avons* $d(S_5') = \langle (T, \lfloor 2 \rceil), (S, \lfloor 3 \rceil), (Sp) \rangle$ *qui est une sous-séquence distancielle de* $d(S_5)$.

La séquence $R = \langle (T, \lfloor 2, 10 \rceil), (Sp) \rangle$ *est une sous-séquence distancielle commune des séquences* S_1, S_2, S_4 *et* S_5*. Cela signifie que* $\langle T, Sp \rangle$ *est sous-séquence des quatre séquences* S_1, S_2, S_4 *et* S_5 *avec distance minimale de* 2 *et maximale de* 10 *entre* T *et* Sp *(Figure 5.4). La sous-séquence distancielle* R *peut aussi s'écrire sous la forme* $R = \langle T \lfloor 2 - 10 \rceil Sp \rangle$.

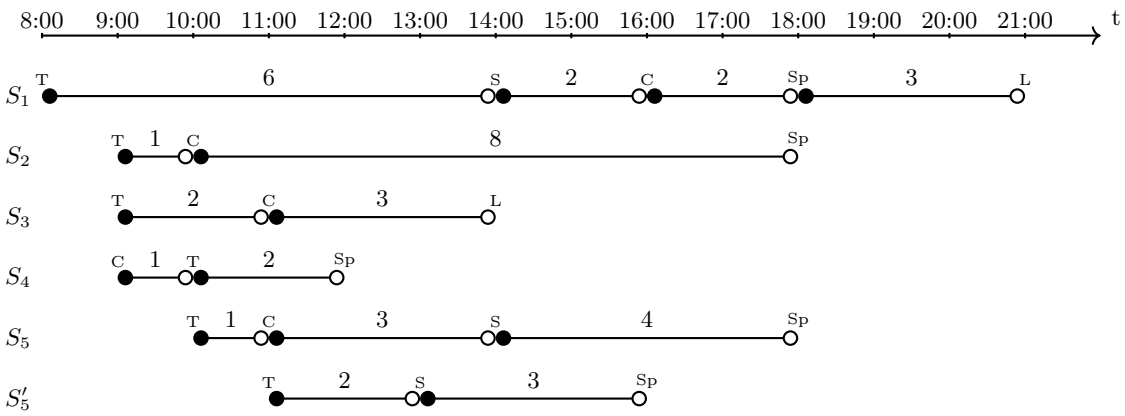


FIGURE 5.6 – Exemple de séquences distancielles d'actions quotidiennes

Séquences temporelles	Sous-séquence distancielle de $\langle T, Sp \rangle$
S_1	$\langle (T, \lfloor \mathbf{10} \rceil), (Sp) \rangle$
S_2	$\langle (T, \lfloor 9 \rceil), (Sp) \rangle$
S_4	$\langle (T, \lfloor \mathbf{2} \rceil), (Sp) \rangle$
S_5	$\langle (T, \lfloor 8 \rceil), (Sp) \rangle$
$SDC(\{S_1, S_2, S_4, S_5\})$	$\langle (T, \lfloor 2, 10 \rceil), (Sp) \rangle$

TABLE 5.4 – Sous-séquence distancielle commune de $\{S_1, S_2, S_4, S_5\}$

3.1 Descriptions

Nous présentons la description distancielle d'un ensemble A de séquences temporelles données par les *Sous-séquences Distancielles Communes Maximales (SDCM)*. Cette description peut ensuite être paramétrée par deux contraintes. Une contrainte de *fenêtre* glissante, et une contrainte d'*écart* maximal.

La contrainte de *fenêtre* est utilisée pour limiter la recherche de sous-séquences à une partie spécifique des séquences temporelles. Ce paramètre permet de réduire l'espace de recherche. Par exemple, lorsque nous voulons analyser des données sur des déplacements de touristes, nous pouvons utiliser une fenêtre d'un jour pour nous concentrer sur les activités d'un voyage sur la journée car il se peut que les touristes passent plusieurs jours dans la même ville et que notre objectif soit de connaître les activités essentielles par journée. Dans l'exemple précédent, si nous fixons une fenêtre de 5 heures, nous n'obtenons que des sous-séquences avec une distance maximale de 5 heures entre le premier et le dernier élément des sous-séquences. La contrainte d'*écart* maximal définit la distance maximale autorisée entre deux éléments consécutifs de la sous-séquence.

3.1.1 Description de Sous-séquences Distancielle Communes Maximales (SDCM)

Définition 5.10 (description SDCM)**.** Pour un ensemble de séquences temporelles $A \subseteq G$, la description SDCM est définie par :

$$\delta_{\text{SDCM}}(A) = \{P = \langle (p_i, \lfloor d_i^{p,min}, d_i^{p,max} \rceil), (p_n) \rangle_{i<n_p} \mid$$
$$P \in SDC(A) \text{ tel que } P \text{ maximale (si } \exists P' \in SDC(A), P \sqsubseteq_s P' \text{ alors } P = P')\} \quad (5.7)$$

La description génère l'ensemble de sous-séquences distancielles communes maximales pour l'ensemble de séquences A. Les prédicats sont donc de la forme *"X est une sous-séquence distancielle de S"* pour tout $X \in \delta_{\text{SDCM}}(A)$ et $S \in A$.

L'algorithme 17 présente le calcul de la description SDCM. Pour calculer la description SDCM d'un ensemble A de séquences temporelles, nous calculons d'abord les sous-séquences communes maximales simples avec le même algorithme de calcul de la description SCM pour les séquences simples en enlevant les horodatages des séquences temporelles pour avoir des séquences simples (ligne 1). Ensuite, pour chaque sous-séquence *sub* (ligne 3), nous initialisons une liste des distances (lignes 7-9) et nous itérons dans A (ligne 10) et prenons les positions de chaque élément de *sub* (lignes 12-15). Enfin, nous mettons à jour les distances des sous-séquences résultantes de $\delta_{\text{SDCM}}(A)$ avec les distances entre les éléments (lignes 17-28). Les deux conditions des lignes 19 et 22 permettent la mise à jour des distances.

Complexité 5.4. *La complexité de la description SDCM est* $c_\delta^{SDCM} = c_\delta^{SCM} + O(|\delta_{SCM}|.|A|) = c_\delta^{SCM} \leq O(n|\delta(A)||\Sigma||A|)$.

Proposition 5.4. *Pour* $A' \subseteq A \subseteq G$*, Nous avons* $\delta_{SDCM}(A) \sqsubseteq \delta_{SDCM}(A')$

Preuve 5.3. *Soit A, A' deux sous-ensembles de séquences temporelles tels que* $A' \subseteq A$*. Soit* $c \in \delta_{SDCM}(A)$*, c'est-à-dire , c est une sous-séquence distancée commune maximale de A. De* $A' \subseteq A$ *on peut déduire que c est aussi une sous-séquence des séquences dans* A'*, mais c n'est pas nécessairement une sous-séquence distancielle commune maximale pour* A'*. Si c est une sous-séquence maximale dans* A' *alors* $c \in \delta_{SDCM}(A')$*. Sinon, il existe* $c' \in \delta_{SDCM}(A')$ *tel que c est une sous-séquence de* c'*. Dans ces deux cas, on peut déduire que,* $\delta_{SDCM}(A) \sqsubseteq \delta_{SDCM}(A')$.

3.1.2 Description SDCM avec Contrainte de *fenêtre*

Définition 5.11 (description SDCMw)**.** Pour un ensemble de séquences temporelles $A \subseteq G$, la description distancielle avec contrainte de *fenêtre* $SDCMw$ est définie par :

$$\delta_{\text{SDCMw}}(A, w) = \{P \mid P \in \delta_{\text{SDCM}}(A), \forall S \in A, |S| = m,$$
$$\exists \text{ des entiers } 1 \leq i_0 \leq ... \leq i_n \leq m, p_k = s_{i_k} \text{ et } (t_{i_m}^s - t_{i_0}^s) \leq w\} \quad (5.8)$$

La description SDCMw génère les sous-séquences distancielles communes maximales qui respectent une contrainte de *fenêtre*. Les prédicats sont donc de la même forme avec

Algorithm 17 L'algorithme SDCM

Entrée : un ensemble de séquences temporelles A sur l'alphabet Σ deux paramètres facultatifs de *fenêtre* w et d'*écart* g

Sortie : l'ensemble de prédicats de la description SDCM $\delta_{\text{SDCM}}(A)$

$subsequences \leftarrow \delta_{\text{SCM}}(A)$
$F \leftarrow \emptyset$
for $sub \in subsequences$ **do**
  $l \leftarrow |sub|$
  $distances \leftarrow \emptyset$
  $subbool \leftarrow True$
  for $i \leq l$ **do**
    $distances[i] = (maximal(\{S \in A\}), 0)$
  end for
  for $S \in A$ **do**
    $templist \leftarrow \emptyset$
    for $element \in sub$ **do**
      $templist.add(t) \mid (t, element) \in S$
      $S \leftarrow S[t, |S|]$
    end for
    $win \leftarrow 0$
    for $i < l(templist) - 1$ **do**
      $win \leftarrow win + (templist[i+1] - templist[i])$
      if $(templist[i+1] - templist[i]) < distances[i][0]$ **then**
        $distances[i][0] = (templist[i+1] - templist[i])$
      end if
      if $(templist[i+1] - templist[i]) > distances[i][1]$ **then**
        $distances[i][1] = (templist[i+1] - templist[i])$
      end if
      if $(templist[i+1] - templist[i]) > g$ or $win > w$ **then**
        $subbool \leftarrow False$
      end if
    end for
  end for
  if $subbool$ **then**
    Add $distances$ to sub to construct a distanciel subsequence
    $F \leftarrow F \cup sub$
  end if
end for
Return F

ajout de l'information sur la *fenêtre*. Le mécanisme permettant la vérification de la *fenêtre* est inclus dans l'algorithme 17. Le paramètre *win* représente la *fenêtre* et la deuxième partie de la condition dans la ligne 25 permet de ne pas prendre en compte les motifs qui ne respectent pas la contrainte de *fenêtre* w.

3.1.3 Description SDCM avec Contrainte d'*écart*

Définition 5.12 (description SDCMg)**.** Pour un ensemble de séquences temporelles $A \subseteq G$, la description distancielle avec contrainte d'*écart* $SDCMg$, est définie par :

$$\delta_{\text{SDCMg}}(A, g) = \{P : P \in \delta_{\text{SDCM}}(A) \text{ et } d_i^{p,max} \leq g, \text{ pour } 0 \leq i < n\} \tag{5.9}$$

La description SDCMg génère les sous-séquences distancielles communes maximales qui respectent une contrainte d'*écart*. Les prédicats sont donc de la même forme avec ajout de l'information sur l'*écart*. Comme pour la *fenêtre*, la vérification d'*écart* est incluse dans l'algorithme 17. Le paramètre g représente l'*écart* et la première partie de la condition de la ligne 25 permet de vérifier si la distance entre deux éléments successifs de la sous-séquence est bien supérieure au paramètre g.

Exemple 5.4. *les Tables 5.5 et 5.6 présentent des exemples de prédicats générés par les trois descriptions SDCM, SDCMg et SDCMw pour les séquences de l'Exemple 1.2.*

A	$\delta_{\text{SDCM}}(A)$
$\{S_1, S_2, S_4, S_5\}$	$\{\langle(T\ \lfloor 2-10\rceil\ Sp\ \rangle, \langle C\ \lfloor 2-8\rceil\ Sp\rangle\}$
$\{S_1, S_3\}$	$\{\langle T\ \lfloor 2-8\rceil\ C\ \lfloor 3-5\rceil\ L\rangle,$
$\{S_1, S_2, S_5\}$	$\{\langle T\ \lfloor 1-8\rceil\ C\ \lfloor 2-8\rceil\ Sp\rangle\}$

TABLE 5.5 – Exemples de prédicats de la description SDCM pour l'exemple 1.2.

A	$\delta_{\text{SDCMw}}(A, 10)$	$\delta_{\text{SDCMg}}(A, 8)$
$\{S_1, S_2, S_4, S_5\}$	$\{\langle(T\ \lfloor 2-10\rceil\ Sp\ \rangle, \langle C\ \lfloor 2-8\rceil\ Sp\rangle\}$	$\{\langle C\ \lfloor 2-8\rceil\ Sp\rangle\}$
$\{S_1, S_3\}$		$\{\langle T\ \lfloor 2-8\rceil\ C\ \lfloor 3-5\rceil\ L\rangle,$
$\{S_1, S_2, S_5\}$	$\{\langle T\ \lfloor 1-8\rceil\ C\ \lfloor 2-8\rceil\ Sp\rangle\}$	$\{\langle T\ \lfloor 1-8\rceil\ C\ \lfloor 2-8\rceil\ Sp\rangle\}$

TABLE 5.6 – Exemples de prédicats des descriptions SDCMg et SDCMw pour l'exemple 1.2.

3.2 Stratégies

Pour les séquences temporelles, une première approche consiste à ajouter des éléments de l'alphabet aux sous-séquences communes maximales sur le même modèle que les stratégies des séquences simples. Une seconde approche est d'exploiter l'information temporelle en réduisant les distances des sous-séquences communes.

3.2.1 Stratégie *Par étape*

La stratégie distancielle *Par étape* (SDP) génère des sous-séquences en modifiant la distance entre les éléments ou en ajoutant un élément de l'alphabet.

Définition 5.13 (stratégie SDP)**.** Pour $A \subseteq G$ et un pas pas :

$$\sigma_{\text{SDP}}(A, pas) = X \cup Y \cup Z \text{ où} \tag{5.10}$$

$$X = \{S : \langle s_i \rangle = \langle p_i \rangle, \forall P \in \delta_{\text{SDCM}}(A) \text{ et } d_i^{s,min} = d_i^{p,min} + pas\} \tag{5.11}$$

$$Y = \{S : \langle s_i \rangle = \langle p_i \rangle, \forall P \in \delta_{\text{SDCM}}(A) \text{ et } d_i^{s,max} = d_i^{p,max} - pas\} \tag{5.12}$$

$$Z = \{S + a : \langle s_i \rangle = \langle p_i \rangle, \forall P \in \delta_{\text{SDCM}}(A) \text{ et } a \in \Sigma\} \tag{5.13}$$

La stratégie SDP génère donc trois types de sélecteurs (prédicats) représentés par les parties X, Y et Z de l'équation de la définition 5.13. La partie X génère des sélecteurs en ajoutant un pas aux distances minimales entre les éléments des séquences de la description $\delta_{\text{SDCM}}(A)$. La partie Y génère des sélecteurs en soustrayant un pas aux distances maximales entre les éléments des séquences de la description $\delta_{\text{SDCM}}(A)$. Ces deux derniers sélecteurs minimisent la distance entre les éléments de la description pour avoir moins d'objets et donc pouvoir générer des prédécesseurs. La partie Z génère des sélecteurs en ajoutant un élément de l'alphabet Σ aux séquences de la description $\delta_{\text{SDCM}}(A)$. La stratégie permet donc l'exploration des séquences au niveau temporel par les sélecteurs X et Y, et au niveau des éléments de la séquence par les sélecteurs Z. Plus le pas est grand, moins il y a de sélecteurs, et donc moins il y aura de concepts prédécesseurs.

3.2.2 La stratégie *Naïve*

La stratégie distancielle *Naïve* (SDN) génère toutes les sous-séquences distancielles possibles. C'est un cas particulier de la stratégie *Par étape*, avec un pas égal à 1 elle permet d'explorer toutes les sous-séquences distancielles.

Définition 5.14 (stratégie SDN)**.** Pour $A \subseteq G$, la stratégie Naïve est définie à partir de la stratégie *Par étape* avec un $pas = 1$:

$$\sigma_{\text{SDN}}(A) = \sigma_{\text{SDP}}(A, 1) \tag{5.14}$$

3.2.3 La stratégie *Milieu*

La stratégie distancielle *Milieu* (SDM) génère des sous-séquences en divisant les distances par deux. Pour permettre plus d'élagage, la stratégie divise toutes les distances pour chaque sélecteur.

Définition 5.15 (stratégie SDM)**.** Pour $A \subseteq G$, la stratégie *Milieu* est définie à partir de la stratégie *Par étape* avec un pas égale au milieu des distances :

$$\sigma_{\text{SDM}}(A) = \{S : S \in \sigma_{\text{SDP}}(A, M), M = (d_i^{s,max} - d_i^{s,min})/2 \text{ pour } 0 \leq i < n\} \tag{5.15}$$

Exemple 5.5. *La Table 5.7 donne quelques sélecteurs générés pour $A = \{1, 2, 5\}$ (concept \$4 dans la Figure 5.7) et avec les trois stratégies. Les deux premières lignes de la table présentent l'ensemble de séquences A et leurs description $\delta_{SDCM}(A)$. Ensuite, les trois parties de l'équations 5.10 sont présentés pour chaque stratégie. L'équation 5.13 est commune aux trois stratégies (la partie Z). Les équations 5.12 et 5.11 (les parties Y et X) sont présentées ensuite, et les stratégies sont données dans la dernière ligne.*

	$A = \{S_1, S_2, S_5\}$		
	$\delta_{\text{SDCM}}(A) = \{\langle T \lfloor 1-8 \rceil\ C \lfloor 2-8 \rceil\ Sp \rangle\}$		
Z	$\langle T \lfloor /-/ \rceil\ C \lfloor /-/ \rceil\ Sp \lfloor /-/ \rceil\ C \rangle$		
	$\langle T \lfloor /-/ \rceil\ C \lfloor /-/ \rceil\ Sp \lfloor /-/ \rceil\ L \rangle$		
	$\langle T \lfloor /-/ \rceil\ C \lfloor /-/ \rceil\ Sp \lfloor /-/ \rceil\ S \rangle$		
	$\langle T \lfloor /-/ \rceil\ C \lfloor /-/ \rceil\ Sp \lfloor /-/ \rceil\ Sp \rangle$		
	$\langle T \lfloor /-/ \rceil\ C \lfloor /-/ \rceil\ Sp \lfloor /-/ \rceil\ T' \rangle$		
	Y_1	Y_2	Y_3
Y	$\langle T \lfloor 1-7 \rceil\ C \lfloor 2-8 \rceil\ Sp \rangle$	$\langle T \lfloor 1-4 \rceil\ C \lfloor 2-8 \rceil\ Sp \rangle$	$\langle T \lfloor 1-4.5 \rceil\ C \lfloor 2-5 \rceil\ Sp \rangle$
	$\langle T \lfloor 1-8 \rceil\ C \lfloor 2-7 \rceil\ Sp \rangle$	$\langle T \lfloor 1-8 \rceil\ C \lfloor 2-4 \rceil\ Sp \rangle$	
	X_1	X_2	X_3
X	$\langle T \lfloor 2-8 \rceil\ C \lfloor 2-8 \rceil\ Sp \rangle$	$\langle T \lfloor 5-8 \rceil\ C \lfloor 2-8 \rceil\ Sp \rangle$	$\langle T \lfloor 4.5-8 \rceil\ C \lfloor 5-8 \rceil\ Sp \rangle$
	$\langle T \lfloor 1-8 \rceil\ C \lfloor 3-8 \rceil\ Sp \rangle$	$\langle T \lfloor 1-8 \rceil\ C \lfloor 6-8 \rceil\ Sp \rangle$	
	$\sigma_{\text{SDN}}(A) = Z \cup Y_1 \cup X_1$	$\sigma_{\text{SDP}}(A, 4) = Z \cup Y_2 \cup X_2$	$\sigma_{\text{SDM}}(A) = Z \cup Y_3 \cup X_3$

TABLE 5.7 – Les stratégies SDN, SDP et SDM pour l'ensemble $A = \{S_1, S_2, S_5\}$.

3.3 Exemples

La description SDCM avec ses deux contraintes et les trois stratégies permettent d'envisager plusieurs types d'analyses pour les séquences temporelles. Nous en présentons ici trois. La première utilise la description SDCM avec la stratégie SDN, qui correspond à l'analyse classique de l'AFC car elle génère tous les concepts possibles. La seconde change de stratégie vers SDM et SDP pour réduire le nombre de concepts en conservant la même description. La dernière utilise des contraintes dans la description. La Table 5.8 présente les figures contenant les diagrammes de Hasse décrivant les treillis générés pour l'Exemple 1.2 par chacune de ces analyses.

Stratégie / Descriptions	δ_{SDCM}	δ_{SDCMw} avec $w = 7, w = 6$	δ_{SDCMg} avec $g = 7$
σ_{SDN}	Fig 5.7	Fig 5.11 et Fig 5.10	Fig 5.12
σ_{SDM}	Fig 5.8		
σ_{SDP} avec pas = 7	Fig 5.9		

TABLE 5.8 – Combinaisons description/stratégies avec les figures des treillis générés

Description SDCM avec la stratégie SDN. La Figure 5.7 montre le diagramme de

Hasse du treillis de concepts généré avec la description SDCM et la stratégie SDN. Le treillis contient 17 concepts. Le concept \$1 possède les concepts \$7, \$3 et \$4 comme prédécesseurs. Les descriptions des concepts \$1 et \$3 décrivent la même sous-séquence $\langle T, C\rangle$ mais avec des distances différentes entre T et C, $\lfloor 1, 8\rceil$ pour le concept \$1 et $\lfloor 1, 2\rceil$ pour le concept \$3. Le concept \$7 possède la description $T\ \lfloor 2-8\rceil\ C\ \lfloor 3-5\rceil\ L$ qui est obtenue en ajoutant l'élément L à la sous-séquence $\langle T, C\rangle$. Le concept \$4 possède également la description $\langle T\ \lfloor 1-8\rceil\ C\ \lfloor 2-8\rceil\ Sp\rangle$.

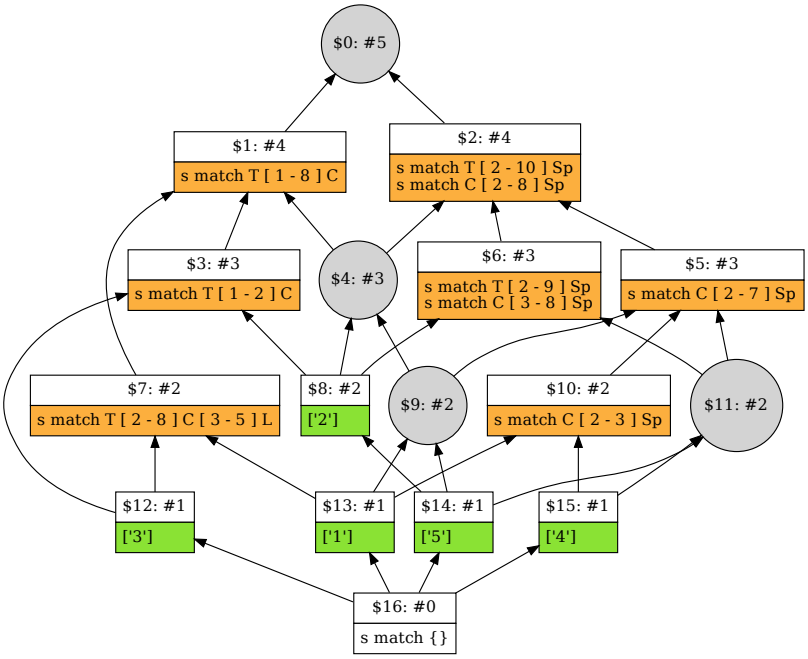


FIGURE 5.7 – Diagramme de Hasse du treillis de concepts généré par la description SDCM et la stratégie SDN.

Description SDCM avec les deux stratégies SDP et SDM. La Figure 5.8 présente le diagramme de Hasse du treillis obtenu avec la description SDCM et la stratégie SDM. Le treillis est composé de 12 concepts donc moins qu'avec la stratégie SDN. Le concept \$2 est un prédécesseur direct du concept maximal, il contient la description $\{\langle T\ \lfloor 2-10\rceil\ Sp\rangle, \langle C\ \lfloor 2-8\rceil\ Sp\rangle\}$. Le concept \$6 est un prédécesseur direct du concept \$2 avec deux séquences $A = \{1, 4\}$ et leur description $\{\langle C\ \lfloor 2-3\rceil\ Sp\rangle\}$. Le concept \$6 correspond au concept \$10 de la Figure 5.7. Avec la stratégie SDN, le concept \$5 de la Figure 5.7 est généré avant, ce qui n'est pas le cas pour la stratégie SDM. Les sélecteurs générés par la stratégie SDM ne permettent pas la génération du concept \$5 car on passe de $\langle C\ \lfloor 2-8\rceil\ Sp\rangle$ à deux sélecteurs $\langle C\ \lfloor 2-6\rceil\ Sp\rangle$ et $\langle C\ \lfloor 5-8\rceil\ Sp\rangle$. La Figure 5.9 montre le diagramme de Hasse du treillis réduit

obtenu avec la description SDCM et la stratégie SDP avec pas égal à 7. Le treillis est composé de 10 concepts seulement donc une réduction du nombre de concepts par rapport à la stratégie SDN.

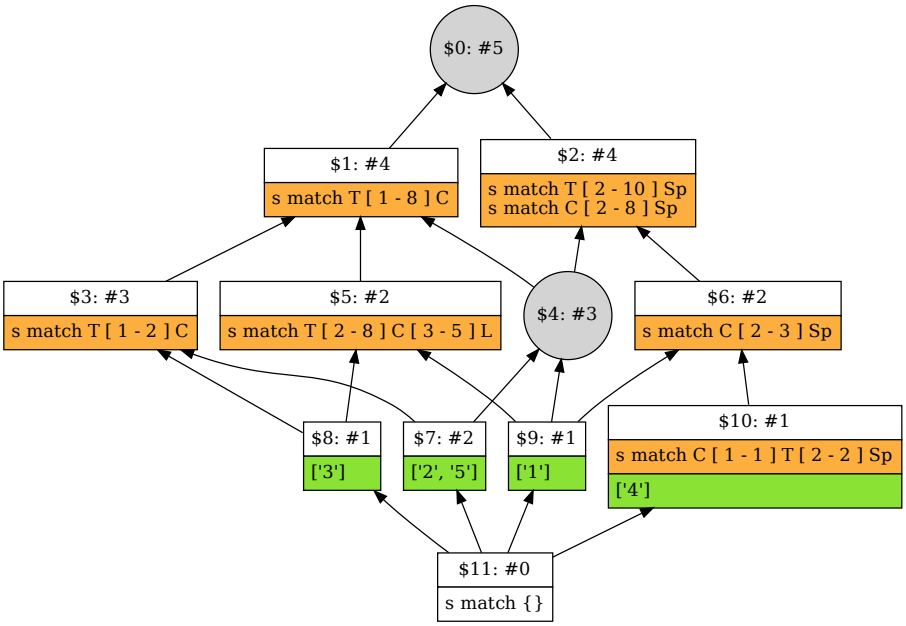


FIGURE 5.8 – Diagramme de Hasse du treillis de concepts généré par la description SDCM et la stratégie SDM.

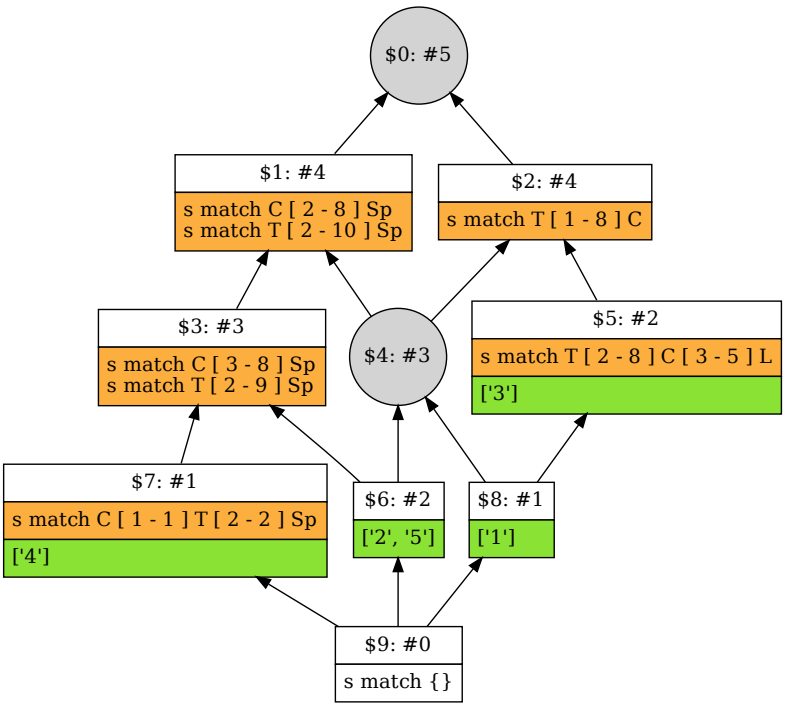


FIGURE 5.9 – Diagramme de Hasse du treillis de concepts généré par la description SDCM et la stratégie SDP (pas = 7).

Description SDCMw et SDCMg avec la stratégie SDN. Changeons maintenant la description pour ajouter les deux contraintes de *fenêtre* et d'*écart*. Les Figures 5.10 et 5.11 présentent les diagrammes de Hasse générés avec une *fenêtre* égale à 7 et 6 respectivement. Nous pouvons voir qu'avec la contrainte de *fenêtre* nous obtenons moins de concepts. Les concepts avec *fenêtre* supérieur à 7 ne sont pas générés. Avec une *fenêtre* égale à 6, nous réduisons également le nombre de concepts.

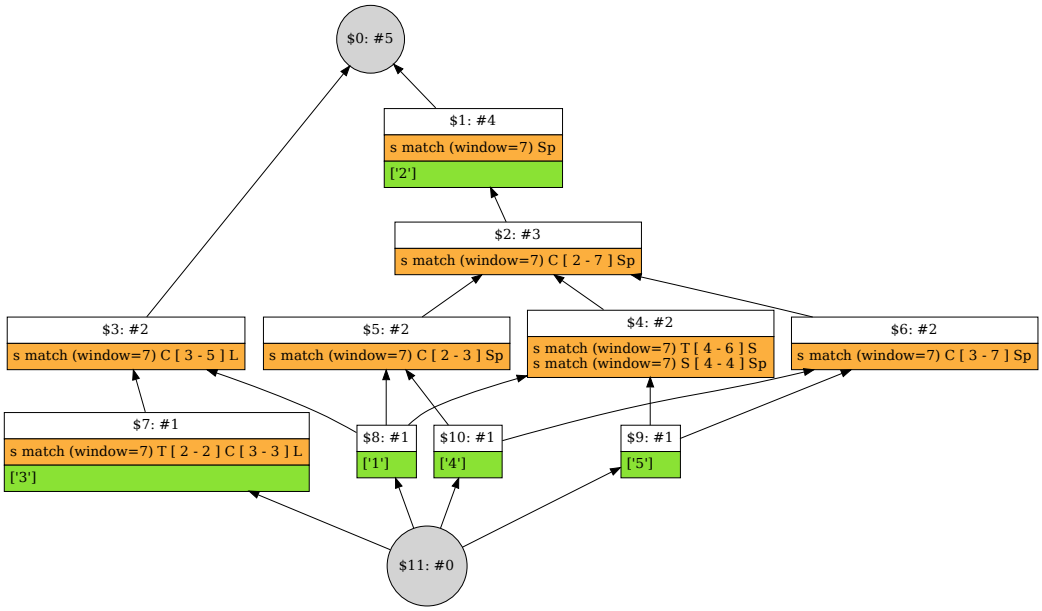


FIGURE 5.10 – Diagramme de Hasse du treillis de concepts généré par la description SDCMw avec w=7 et la stratégie SDN.

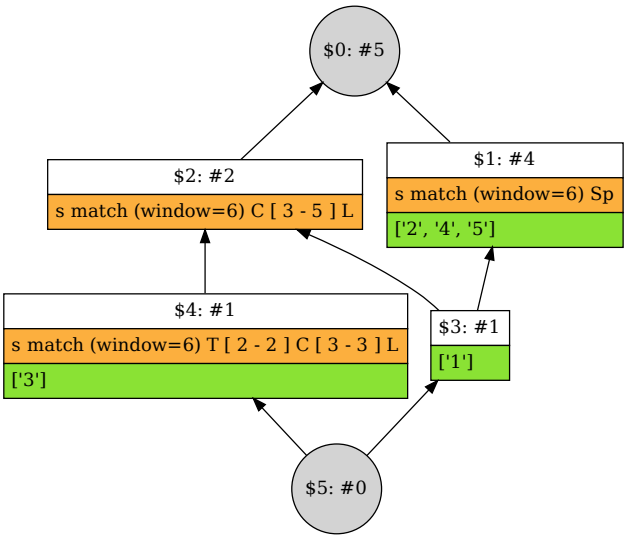


FIGURE 5.11 – Diagramme de Hasse du treillis de concepts généré par la description SDCMw avec w=6 et la stratégie SDN.

La Figure 5.12 présente le diagramme de Hasse généré avec un *écart* égal à 7. De même que la contrainte de *fenêtre*, la contrainte d'*écart* ne permet que la génération des concepts contenant des prédicats respectant la contrainte. Ici, seules les sous-séquences distancielles avec *écart* inférieure à 7 sont générées comme description.

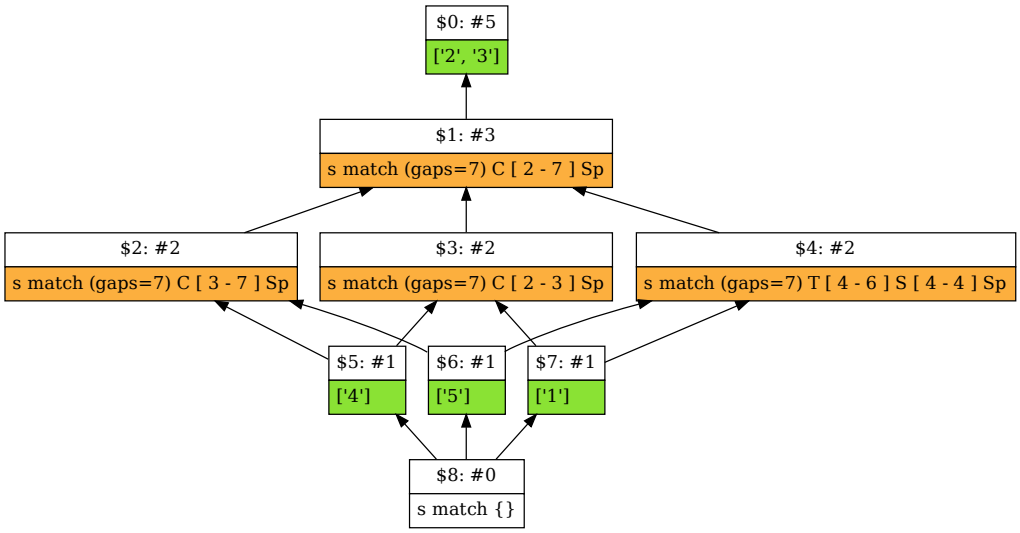


FIGURE 5.12 – Diagramme de Hasse du treillis de concepts généré par les description SDCMg avec g=7 et la stratégie SDN.

4 Analyse de séquences d'intervalles

Cette section est dédiée aux séquences d'intervalles. Nous définissons des descriptions pour les séquences d'intervalles et des stratégies d'exploration de données aidant à bénéficier de l'information temporelle pour mieux analyser et explorer les données.

Rappelons la définition d'une séquence d'intervalles présentée en Chapitre 2. Une séquence d'intervalles est définie pour un ensemble d'alphabet Σ par $S = \langle(\underline{t}_i, \overline{t}_i, s_i)\rangle_{i \leq n}$, où chaque élément $s_i \in \Sigma$ est associé à un intervalle $(\underline{t}_i, \overline{t}_i)$. Chaque tuple $(\underline{t}_i, \overline{t}_i, s_i)$ de la séquence d'intervalles est appelé *Cadre Temporel*. Pour une meilleure lisibilité, nous faisons référence à $(\underline{t}, \overline{t})$ par T. Nous considérons pour cette caractéristique des séquences d'ensemble d'éléments, et donc nous prenons la définition des séquences d'intervalles avec des ensembles : $S = \langle(\underline{t}_i, \overline{t}_i, X_i)\rangle_{i \leq n}$, où chaque élément $X_i \subseteq \Sigma$ est associé à un intervalle $T_i = (\underline{t}_i, \overline{t}_i)$.

Afin de pouvoir manipuler des séquences d'intervalles, il nous faut tout d'abord définir la relation de sous-séquence pour des séquences d'intervalles, qui elle même repose sur la notion de sous-intervalles. Pour deux intervalles, $T = (\underline{t}, \overline{t})$ et $T' = (\underline{t'}, \overline{t'})$. T est un sous-intervalle de T' si : $\underline{t} \geq \underline{t'}$ et $\overline{t} \leq \overline{t'}$ et on écrit $T \preceq T'$. Notons que cette relation correspond aux relations de contenance des relations d'Allen [Allen, 1981] (*Starts, During, Finishes, Equal*).

Définition 5.16 (Sous-séquence d'intervalles)**.** Une séquence S, est une sous-séquence d'intervalles d'une autre séquence S', $S \sqsubseteq_s S'$ si pour tout $(T_i, X_i) \in S$, il existe $(T'_j, X'_j) \in$

S' telle que $T_i \preceq T'_j$ et $X_i \subseteq X'_j$. On dit aussi que S' est une super-séquence d'intervalles de S.

Définition 5.17 (Affix)**.** Un préfixe/suffixe d'une séquence $S = \langle (T_i, X_i) \rangle_{i \leq n}$ selon une fenêtre w est la sous-séquence de S composée par les premières/dernières w intervalles de S, $\text{prefix}(S, w) = \langle (T_i, X_i) \rangle_{1 \leq i \leq w}$, $\text{suffix}(S, w) = \langle (T_i, X_i) \rangle_{(n-w) < i \leq n}$.

Définissons la projection qui permet de faire un focus sur les intervalles selon les éléments ou sur les *itemsets* selon un intervalle :

Définition 5.18 (Projections)**.** Un opérateur de projection Φ sur un intervalle T est une application qui sélectionne tous les *itemsets* d'une séquence S inclus dans cet intervalle :

$$\Phi_T(S) = \{X' \mid T' \preceq T \text{ et } (T', X') \in S\} \tag{5.16}$$

L'opérateur de projection Φ est définie aussi sur un ensemble d'éléments $X \subseteq \Sigma$ pour sélectionner tous les intervalles où les éléments de X peuvent apparaître :

$$\Phi_X(S) = \{T' \mid X' \subseteq X \text{ et } (T', X') \in S\} \tag{5.17}$$

Notons que $\Phi_\Sigma(S)$ représente un ensemble de tous les intervalles dans S.

Définition 5.19 (Cardinalité)**.** Pour un ensemble de séquences A, un élément $x \in \Sigma$ et un intervalle T, la fonction *card* donne le nombre de séquences $S \in A$ possédant l'élément x dans la projection de S sur T, $x \in \Phi_T(S)$:

$$card(A, T, x) = |\{S \in A \mid x \in \Phi_T(S)\}| \tag{5.18}$$

Lorsque $card(A, T, x)$ est maximale, nous désignons $card(A, T, x)$ par $card_{max}(A, T)$. Nous définissons $card_{min}(A, T)$ de la même manière lorsque $card(A, T, x)$ est minimal.

Exemple 5.6. *Reprenons l'Exemple 1.3, la Figure 5.13 présente les mêmes séquences avec un format plus visuel.*

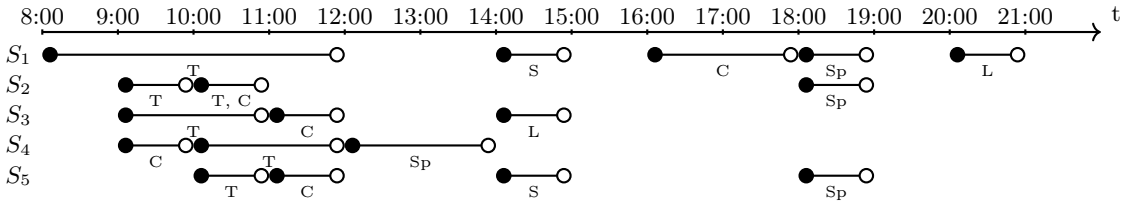


FIGURE 5.13 – Exemple de séquences d'intervalles

Nous avons $\Phi_{(10:00,11:00)}(S_2) = \{T, C\}$*, le préfixe de* S_1 *est* $\langle (08{:}00, 12{:}00), T) \rangle$*. La sous-séquence* $\langle (10{:}00, 11{:}00), T) \rangle$ *est une sous-séquence d'intervalles de* $S1, S2, S3, S4$ *et* $S3$ *décrivant que les cinq personnes travaillait entre 10h00 à 11h00. Pour* $A = \{S1, S2, S3, S4, S5\}$*, nous avons,* $card(A, 09{:}00, 10{:}00, C) = 1$ *et* $card(A, 09{:}00, 10{:}00, T) = 3$.

4.1 Descriptions

Dans cet section nous définissons deux descriptions pour un ensemble de séquences d'intervalles $A \subseteq G$. La description par *Sous-séquences d'intervalles Communes MAXimales (SCMAX)* fait référence à la description classique des sous-séquences communes maximales et correspond à l'ensemble des sous-séquences maximales de toutes les séquences dans A. La description par *Super-séquences d'intervalles Communes MINimales (SCMIN)* contient toutes les super-séquences minimales des séquences de A.

4.1.1 Description de Sous-séquences d'intervalles Communes MAXimales (SCMAX).

La description SCMAX permet de déterminer les séquences d'événements qui se déroulent au même moment. Des touristes qui se retrouvent à un endroit e_1 dans un intervalle de temps T_1, puis dans un autre endroit e_2 pendant l'intervalle T_2 seront détectés, et la séquence $\langle (T_1, e_1), (T_2, e_2) \rangle$ sera définie comme sous-intervalle commun maximal. Plus formellement :

Définition 5.20 (description SCMAX)**.** Pour un ensemble de séquences d'intervalles $A \subseteq G$, la description SCMAX est définie par :

$$\delta_{\text{SCMAX}}(A) = \{\langle (T, X) \rangle \,|\forall S \in A, X \subseteq \Phi_T(S)\} \tag{5.19}$$

La description génère l'ensemble des sous-séquences d'intervalles communes maximales pour l'ensemble de séquences A. Les prédicats sont de la forme *"X est une sous-séquence d'intervalles de S"* pour tout $X \in \delta_{\text{SCMAX}}(A)$ et $S \in A$.

Pour calculer les deux descriptions d'un ensemble A de séquences, nous commençons avec une séquence d'intervalles vide I, nous itérons sur les séquences de A, puis les séquences résultantes de $\delta(A)$ sont mises à jour avec les parties communes (la fonction *Update*). À la fin de la boucle (lignes 2-4) nous obtenons une séquence d'intervalles qui est super-séquences de tous les séquences d'intervalles de A. Ensuite, on supprime les éléments avec une cardinalité inférieure à la taille de A (lignes 5-14).

Complexité 5.5. *La complexité de la description SCMAX est* $c_\delta^{SCMAX} = O(s\,|A|\,log(|A|)) \leq O(s\,|G|\,log(|G|))$ *où* s *est la taille maximale des séquences calculées.*

Proposition 5.5. *Pour* $A' \subseteq A \subseteq G$*, nous avons la propriété suivante :*
$\delta_{SCMAX}(A) \sqsubseteq \delta_{SCMAX}(A')$

Preuve 5.4. *Soit* A *et* A' *sont deux sous-ensembles de* G *tels que* $A' \subseteq A$*. Soit* $c \in \delta_{SCMAX}(A)$*, c'est-à-dire,* c *est une sous-séquence maximale de* A*. De* $A' \subseteq A$ *on peut déduire que* c *est aussi une sous-séquence de séquences dans* A' *mais* c *n'est pas nécessairement une sous-séquence maximale pour* A'*. Si* c *est une sous-séquence maximale dans* A' *alors* $c \in \delta_{SCMAX}(A')$*. Sinon, il existe* $c' \in \delta_{SCMAX}(A')$ *tel que* c *est une sous-séquence de* c'*. Dans ces deux cas, on peut déduire que,* $\delta_{SCMAX}(A) \sqsubseteq \delta_{SCMAX}(A')$*.*

Algorithm 18 L'algorithme SCMAX

Entrée : un ensemble de séquences A et l'alphabet Σ
Sortie : l'ensemble de prédicats de la description SCMAX $\delta_{\text{SCMAX}}(A)$

```
1: I ← ⟨⟩
2: for S ∈ A do
3:     UPDATE(I, S)
4: end for
5: for (T, X) ∈ I do
6:     for x ∈ X do
7:         if card(I, T, x) ≠ |A| then
8:             DELETE(I, T, x)
9:         end if
10:    end for
11:    if X = ∅ then
12:        DELETE(I, T, X)
13:    end if
14: end for
15: return I
```

Exemple 5.7. *La Figure 5.14 représente $\delta_{SCMAX}(A)$ pour $A = \{S_1, S_2, S_5\}$. Nous pouvons observer que SCMAX pourrait être interprété comme une "conjonction" où $(10{:}00, 11{:}00, \{T\})$ signifie que toutes les séquences de S_1, S_2 et S_5 contiennent l'élément T pendant l'intervalle $(10{:}00, 11{:}00)$.*

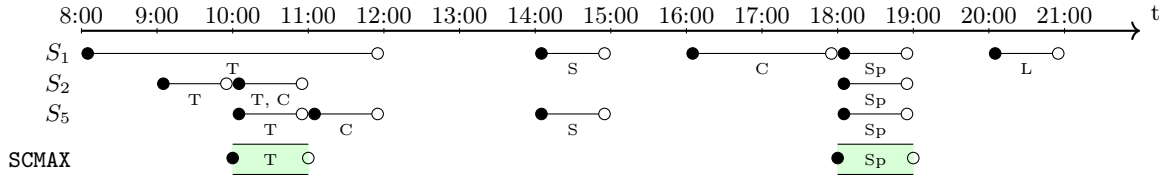


FIGURE 5.14 – $\delta_{\text{SCMAX}}(\{S_1, S_2, S_5\})$ et S_1, S_2 et S_5.

4.1.2 Description de Super-séquences d'intervalles Communes MINimales (SCMIN).

La description SCMIN est une description de la plus courte super-séquence commune. Elle définit l'ensemble des intervalles avec tous les éléments présents dans les séquences d'intervalles. Imaginons des touristes qui se retrouvent à un endroit e_1 dans un intervalle de temps T_1, et un autre groupe de touriste qui se retrouvent dans un autre endroit e_2 pendent l'intervalle T_2. La description sera la séquence $\langle (T_1, e_1), (T_2, e_2) \rangle$ qui contient les deux endroits avec les intervalles de visites.

Définition 5.21 (description SCMIN)**.** SCMIN est définie pour un ensemble de séquences

d'intervalles $A \subseteq G$ par :

$$\delta_{\text{SCMIN}}(A) = \{\langle (T, X) \rangle \mid \forall S \in A, \Phi_T(S) \subseteq X\} \tag{5.20}$$

Cette description génère l'ensemble de super-séquences d'intervalles communes minimales pour l'ensemble de séquences A. Les prédicats sont de la forme *"X est une super-séquence d'intervalles de S"* pour tout $X \in \delta_{\text{SCMIN}}(A)$ et $S \in A$.

L'algorithme est similaire à l'Algorithme 18, sauf qu'on ne supprime pas d'éléments.

Complexité 5.6. *La complexité de la description SCMIN est* $c_\delta^{SCMIN} = O(s\,|A|\,log(|A|)) \leq O(s\,|G|\,log(|G|))$ *où* s *est la taille maximale des séquences calculées.*

Algorithm 19 L'algorithme SCMIN

Entrée : un ensemble de séquences A et l'alphabet Σ
Sortie : l'ensemble de prédicats de la description SCMIN $\delta_{\text{SCMIN}}(A)$

1: $I \leftarrow \langle\rangle$
2: **for** $S \in A$ **do**
3: UPDATE(I, S)
4: **end for**
5: *return* I

Proposition 5.6. *Pour* $A' \subseteq A \subseteq G$, *nous avons la propriété suivante :* $\delta_{SCMIN}(A) \sqsubseteq \delta_{SCMIN}(A')$

Preuve 5.5. *Soit* A *et* A' *sont deux sous-ensembles de* G *tels que* $A' \subseteq A$. *Soit* $s \in \delta_{SCMIN}(A)$, *c'est-à-dire,* s *est la super-séquence de toutes les séquences dans* A. *De* $A' \subseteq A$ *on peut déduire que* s *est aussi une super-séquence de séquences dans* A' *mais pas nécessairement la plus courte. Si* s *est une super-séquence la plus courte dans* A' *alors* $s \in \delta_{SCMIN}(A')$. *Sinon, il existe* $s' \in \delta_{SCMIN}(A')$ *tel que* s *est une super-séquence de* s'. *On peut en déduire que,* $\delta_{SCMIN}(A) \sqsubseteq \delta_{SCMIN}(A')$.

Exemple 5.8. *La Figure 5.15 présente* $\delta_{SCMIN}(A)$ *pour* $A = \{S_2, S_3, S_4\}$. *Nous pouvons observer que SCMIN pourrait être interprété comme une "disjonction" où* $(12{:}00, 14{:}00, \{Sp\})$ *signifie qu'au moins une des séquences* S_2, S_3 *ou* S_4 *contiennent l'élément* P *pendant l'intervalle* $(12{:}00, 14{:}00)$.

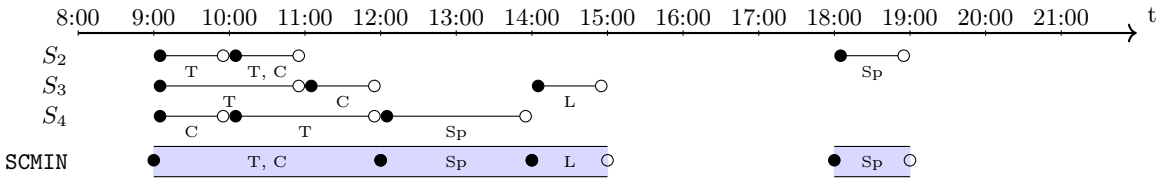


FIGURE 5.15 – $\delta_{\text{SCMIN}}(\{S_2, S_3, S_4\})$ et S_2, S_3 et S_4.

4.2 Stratégies

Les descriptions présentées génèrent des prédicats de deux types différents, les prédicats de sous-intervalles (sous-séquences) et de super-intervalles (super-séquences). La description SCMAX définit les prédicats *"est une sous-séquence d'intervalles de"* alors que la description SCMIN définit les prédicats *"est une super-séquence d'intervalles de"*. Cette différence mène à deux types de stratégies selon les prédicats des descriptions. Une stratégie de sous-séquence d'intervalles pour la description SCMAX et quatre stratégies de super-séquences d'intervalles pour la description SCMIN.

Pour la description SCMAX, la stratégie de *Cardinalité Minimale Augmentée (CMA)* calcule tous les raffinements possibles d'un concept $(A, \delta_{\text{SCMAX}(A)})$ en ajoutant les éléments de cardinalité minimale $card_{min}(A,T)$ dans les intervalles des prédicats de δ_{SCMAX}. La description SCMAX combinée avec la stratégie CMA, correspond à l'analyse classique des séquences adaptée aux séquences d'intervalles. Cette description peut être vue comme une conjonction.

À l'inverse, la description SCMIN correspond à une disjonction. À partir de la super-séquence d'intervalles, les stratégies visent à réduire l'ensemble des objets en réduisant les éléments ou les intervalles de la description. La description SCMIN est très riche en information, et contient plus d'intervalles par rapport à la description SCMAX. Cela donne plusieurs possibilitées d'exploration des séquences d'intervalles. On propose ici quatre stratégies qui génèrent des sélecteurs de super-séquences. La stratégie *Naive Cadre Temporel (NCT)* consiste à générer tous simplement des sélecteurs en supprimant un élément de chaque intervalle de la description SCMIN. La stratégie *Bornes des Cadres Temporels (BCT)* permet de générer des sélécteurs en supprimant des événements de cardinalités maximale ou minimale. La stratégie *Fenêtre Affixe de Cadre Temporel (FACT)* se concentre sur les événements qui apparaissent en premier ou en dernier dans le même intervalle. La stratégie *Alphabet Cadre Temporel (ACT)* se concentre sur les éléments de l'alphabet. La stratégie NCT permet d'obtenir un treillis avec tous les concepts possible, cela correspond à l'approche classique de l'AFC, avec un temps d'exécution et un nombre de concepts importants. Les autres stratégies réduisent le nombre de concepts permettant ainsi de guider l'analyse vers des concepts qui peuvent être plus pertinents. L'utilisation de ces stratégies réduit la complexité de génération des treillis car ils sont plus petits qu'avec la stratégie NCT.

4.2.1 Stratégies avec sélecteurs de sous-séquences d'intervalle

La stratégie de *Cardinalité Minimale Augmentée* (CMA) σ_{CMA} est définie formellement par :

Définition 5.22 (stratégie CMA)**.** Pour un ensemble de séquences d'intervalles $A \subseteq G$ par :

$$\begin{aligned}\sigma_{\text{CMA}}(A) = \{\langle (T,X)\rangle \mid \forall S \in A, \Phi_T(S) \subseteq X \text{ et } \forall x \in X,\\ card(A,T,x) = |A| \text{ ou } card(A,T,x) = card_{min}(A,T)\}\end{aligned} \tag{5.21}$$

Complexité 5.7. *La complexité de la stratégie CMA est la même que la complexité de la description SCMAX :* $c_{\sigma}^{CMA} = c_{\delta}^{SCMAX}$

4.2.2 Stratégies avec des sélecteurs de super-séquences

La stratégie de *Naive Cadre Temporel (NCT)* σ_{NCT} est définie formellement par :

Définition 5.23 (stratégie NCT)**.** Pour un ensemble de séquences d'intervalles $A \subseteq G$ par :

$$\sigma_{\text{NCT}}(A) = \{\langle (T, X \backslash \{x\}) \rangle \mid \forall x \in X,\ (T, X) \in S, S \in \delta_{\text{SCMIN}}(A)\} \tag{5.22}$$

Pour mettre en œuvre cette stratégie, nous devons considérer tout élément de toute séquence de la description SCMIN. Rappelons que $\delta_{\text{SCMIN}}(A)$ contient un seul prédicat.

Complexité 5.8. *La complexité de la stratégie NCT est* $c_{\sigma}^{NCT} = O(|S|)$ *où* S *est le prédicat de* $\delta_{SCMIN}(A)$*.*

La stratégie *Bornes des Cadres Temporels (BCT)* consiste à ne garder que les éléments de cardinalité minimale ou maximale de chaque intervalle, selon le choix de l'utilisateur. De manière générale :

Définition 5.24 (stratégie BCT)**.** Pour un ensemble de séquences d'intervalles $A \subseteq G$ et un nombre entier c :

$$\sigma_{\text{BCT}}(A, c) = \{\langle (T, X \backslash \{x\}) \rangle \mid (T, X) \in S, card(A, T, x) \neq c, S \in \delta_{\text{SCMIN}}(A)\} \tag{5.23}$$

Les deux cas particuliers, $\sigma_{\text{BTF}}(A, card_{max})$ et $\sigma_{\text{BTF}}(A, card_{min})$ permettent de sélectionner des concepts où les éléments sont les plus fréquents ou les moins fréquents. En gardant les éléments avec cardinalité maximale par exemple, on génère des sélecteurs avec des éléments plus fréquents.

Complexité 5.9. *La complexité de la stratégie BCT est* $c_{\sigma}^{BCT} = O(|A|n)$

La stratégie *Fenêtre Affixe de Cadre Temporel (FACT)* utilise un paramètre fenêtre f et génère le préfixe et le suffixe.

Définition 5.25 (stratégie FACT)**.** Pour un ensemble de séquences d'intervalles $A \subseteq G$ et une taille de fenêtre f :

$$\begin{aligned}\sigma_{\text{FACT}}(A, f) = \{\langle (T, X) \rangle \mid (T, X) \in (S - \text{prefix}(S, f)) \\ \cap (S - \text{suffix}(S, f)), S \in \delta_{\text{SCMIN}}(A)\}\end{aligned} \tag{5.24}$$

Où, $S - \text{prefix}(S, f)$ correspond à S sans les f premiers cadres temporels, et $S - \text{suffix}(S, f)$ sans les f derniers cadres temporels.

La stratégie FACT permet de se focaliser sur les événements au milieu des séquences d'intervalles.

Complexité 5.10. *La complexité de la stratégie FACT est* $c_\sigma^{FACT} = c_\delta^{SCMIN}$.

La stratégie *Alphabet Cadre Temporel (ACT)* retire un élément de l'alphabet de tous les cadres temporels.

Définition 5.26 (stratégie ACT)**.** Pour un ensemble de séquences d'intervalles $A \subseteq G$ et un alphabet Σ :

$$\sigma_{\text{ACT}}(A) = \{\langle (T, X \backslash \{x\}) \rangle \ : \forall (T, X) \in S,\ \forall x \in \Sigma,\ S \in \delta_{\text{SCMIN}}(A)\} \tag{5.25}$$

Complexité 5.11. *La complexité de la stratégie ACT est* $c_\sigma^{ACT} = O(|A||\Sigma|)$.

4.3 Exemples

Pour les séquences d'intervalles, les deux descriptions et leurs stratégies donnent deux types de prédicats différents et plusieurs façon d'explorer les intervalles. Nous présenterons ici trois types d'analyses. La première utilise la description SCMAX avec la stratégie CMA, qui correspond à l'analyse classique. La seconde utilise la description SCMIN avec la stratégie NCT, qui correspond également à une analyse classique au sens de l'AFC car tous les concepts possible sont générés. Et enfin, une dernière analyse avec utilisation de stratégies plus fines pour réduire le nombre de concepts avec la description SCMIN. La Table 5.9 présente un résumé des figures contenant les diagrammes de Hasse des treillis générés par ces différentes analyses pour les séquences d'intervalles de l'Exemple 1.2.

Stratégie / Descriptions	δ_{SCMIN}	δ_{SCMAX}
σ_{CMA}		Fig 5.16
σ_{NCT}	Fig 5.17	
σ_{BCT}	Fig 5.18 et Fig 5.19	
σ_{FACT}	Fig 5.20	
σ_{ACT}	Fig 5.21	

TABLE 5.9 – Combinaisons description/stratégies avec les figures des treillis générés

Description SCMAX avec stratégie CMA La Figure 5.16 représente le diagramme de Hasse du treillis généré par la description SCMAX et la stratégie CMA pour les séquence de l'Exemple 1.3. Le diagramme contient 13 concepts. Le concept \$2 contient la description des séquences S_1, S_2 et S_5 (cf Figure 5.14). Le concept \$6 raconte que les deux séquences S_1 et S_5, contiennent la sous-séquence distancielle $\{((10, 11) : T), ((14, 15) : S), ((18, 19), Sp)\})\}$

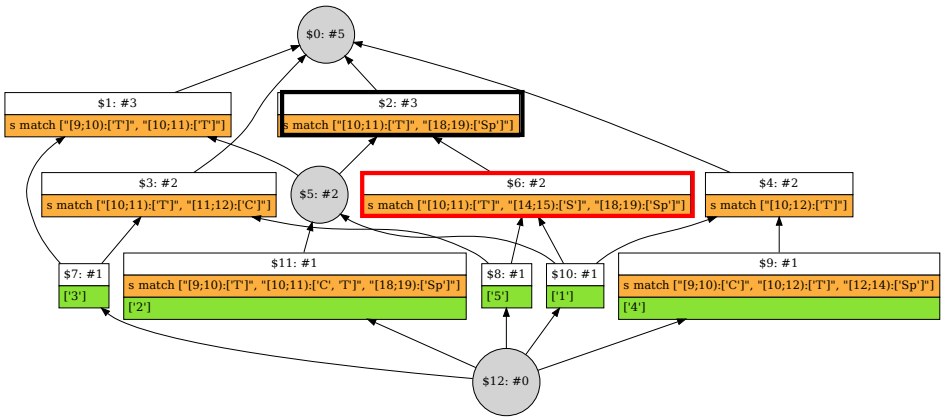
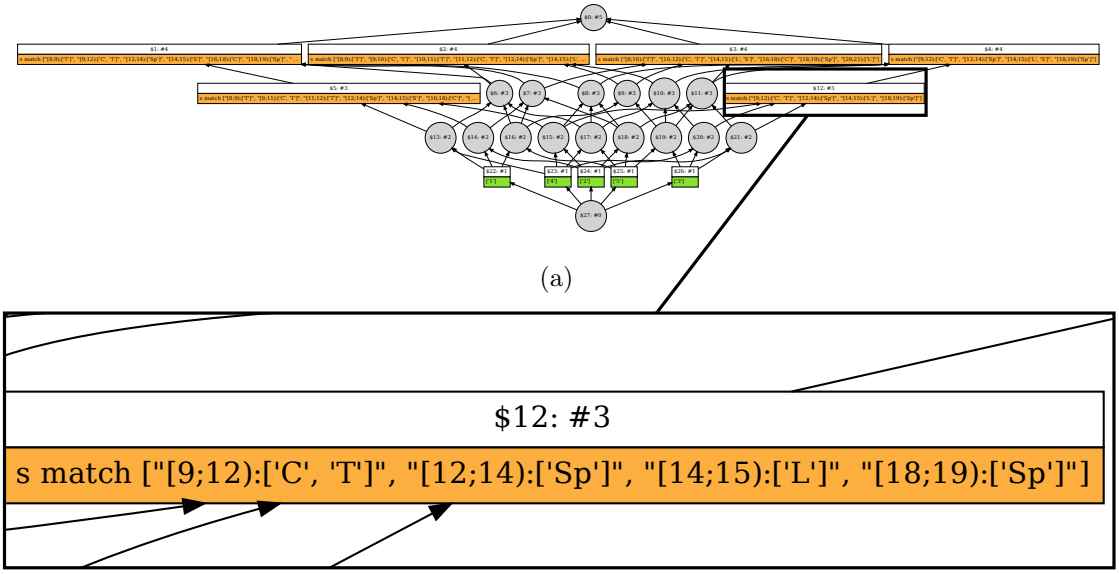


FIGURE 5.16 – Diagramme de Hasse du treillis de concept généré par la description SCMAX et la stratégie CMA.

Description SCMIN avec stratégie NCT La Figure 5.17 représente le diagramme de Hasse du treillis généré avec la description SCMIN et la stratégie NCT. Le treillis généré contient 17 concepts. Le concept maximal $0 contient la description de toutes les séquences, qui est la super-séquence avec tous les cadres temporels. On peut remarquer que les prédicats sont plus longs mais contiennent plus d'informations que les prédicats de sous-séquences d'intervalles. Le concept $12 correspond à la description de l'exemple de la Figure 5.15.



(a)

(b)

FIGURE 5.17 – (a) Diagramme de Hasse du treillis de concepts généré par la description du SCMIN et la stratégie du NCT et (b) zoom sur le concept $12.

Description SCMIN avec les autres stratégies La Figure 5.18 représente le diagramme de Hasse du treillis généré par la stratégie BCT en utilisant $c = card_{min}$. Avec cette stratégie, le nombre de concepts est réduit à 5.

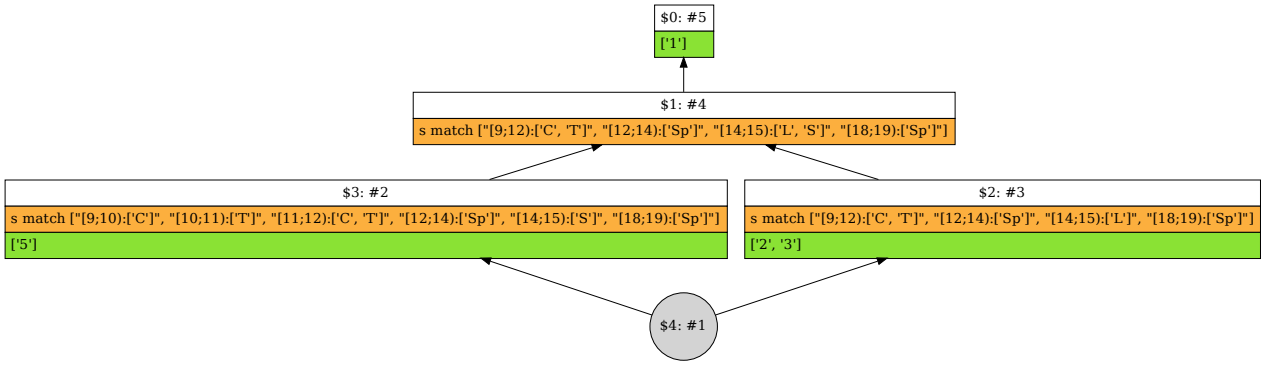


FIGURE 5.18 – Diagramme de Hasse du treillis de concepts généré par la description du SCMIN et la stratégie du BCT avec cardinalité minimale.

Dans la Figure 5.19 nous obtenons un treillis de 18 concepts avec $c = card_{max}$ ce qui s'approche de la stratégie NCT.

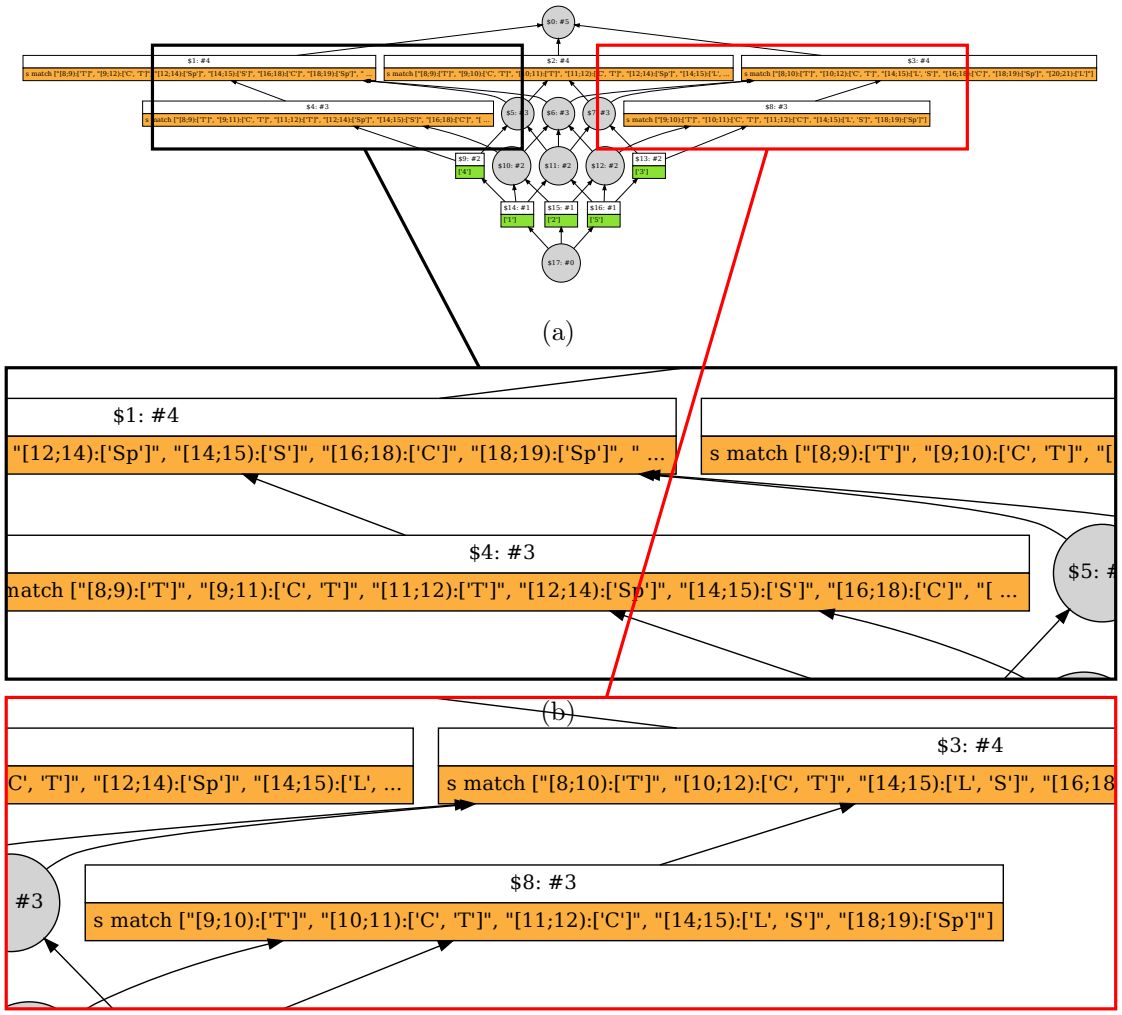


FIGURE 5.19 – Diagramme de Hasse du treillis de concepts généré par la description du SCMIN et la stratégie du BCT avec cardinalité maximale.

Il est clair qu'avec la stratégie BCT $c = card_{min}$ nous obtenons moins de concepts car nous générons directement des concepts de cardinalité minimale, et donc nous allons vers le concept minimal plus rapidement. Avec $c = card_{max}$, nous choisissant à chaque niveau des concepts avec cardinalité maximale des éléments, et donc nous sommes plus proches de la stratégie NCT.

La Figure 5.20 représente le diagramme de Hasse du treillis généré par la stratégie FACT avec $f = 1$. Le treillis contient moins de concepts que la stratégie NCT avec seulement 11 concepts. Le concept \$1 (b) est le seul prédécesseur de concept \$0 contrairement à la stratégie NCT qui génère quatre prédécesseurs. Le prédicat du concept \$1 ne contient pas le premier et le dernier cadre temporel car la stratégie FACT génère des prédécesseurs en supprimant f cadre temporels de la description

SCMIN.

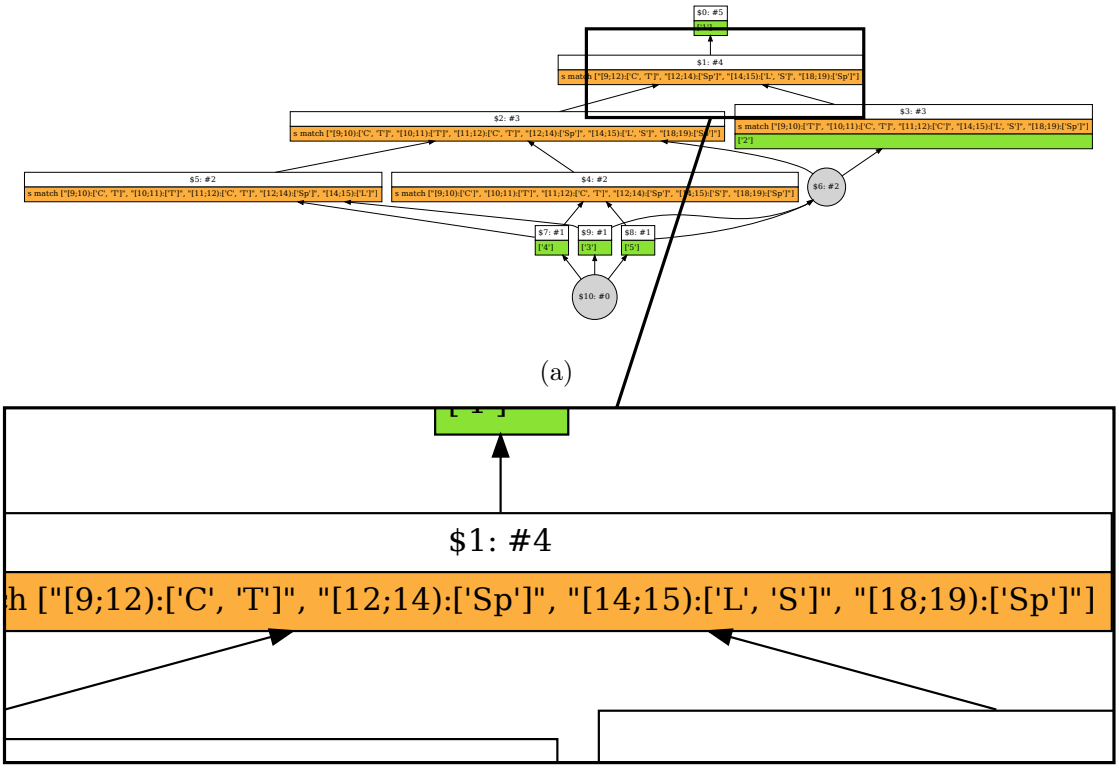


FIGURE 5.20 – (a) Diagramme de Hasse du treillis de concepts généré par la description du SCMIN et la stratégie du FACT avec $f = 1$ et (b) zoom sur le concept $1.

La Figure 5.21 représente le diagramme de Hasse du treillis généré par la stratégie ACT, qui contient moins de concepts. Le concept $1 contient les séquences d'intervalles $2, 3, 4$ et 5, avec leur super-séquence commune minimale. Les deux concepts $2 et $3 prédécesseurs de $1 ne contiennent pas L et S respectivement.

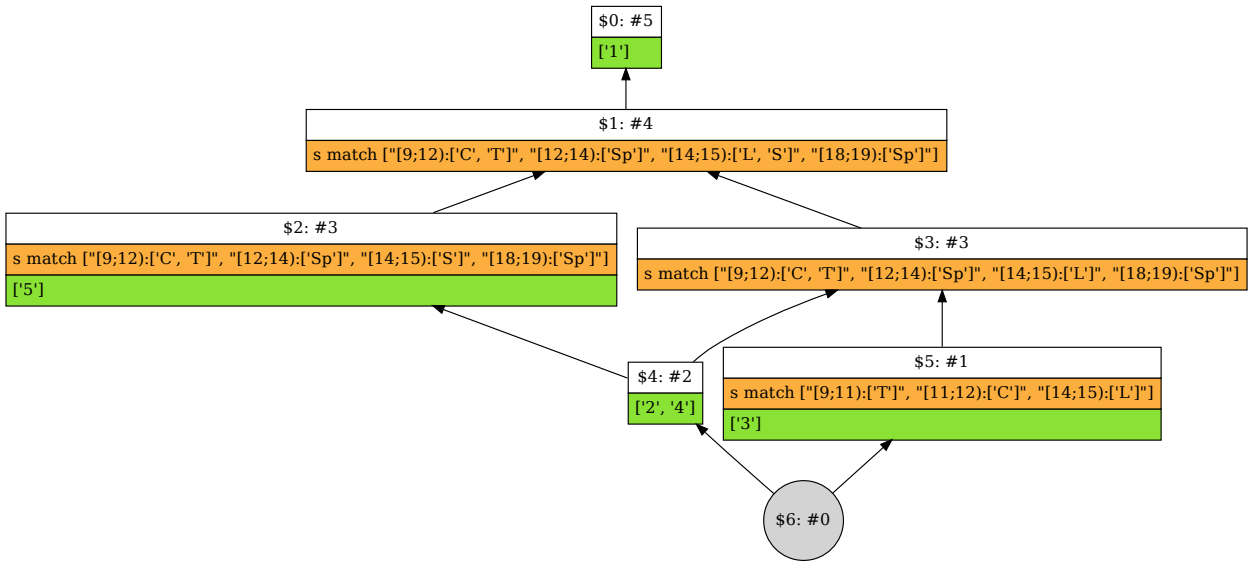


FIGURE 5.21 – Diagramme de Hasse du treillis de concepts généré par la description du SCMIN et la stratégie du ACT.

5 Mesures de qualité

À travers les descriptions et les stratégies définies dans ce chapitre nous pouvons réduire la taille des treillis générés. Les descriptions et stratégies réduisent le nombre de concepts vers les concepts les plus pertinents. Pour mesurer leurs efficacités, nous devons définir des mesures de qualité sur les treillis générés par nos descriptions et stratégies.

5.1 Stabilité Logarithmique Normalisée Globale

Il existe peu de mesures non supervisées de la qualité des concepts ou de motifs. En AFC, la stabilité d'un concept [Kuznetsov, 2007] indique dans quelle mesure un concept dépend de ses objets pour conserver la même description. Un concept est stable si sa description ne dépend pas beaucoup d'objets particuliers. *L'idée derrière la stabilité est qu'un concept stable est susceptible d'avoir une interprétation dans le monde réel même si la description de certains de ses objets est bruitée [Jay et al., 2008].*

Définie pour des données binaires, la stabilité s'étend naturellement aux concepts $(A, \delta(A))$.

La Stabilité. La Stabilité d'un concept $(A, \delta(A))$ est définie comme suit :

$$Stab((A, \delta(A))) = \frac{\kappa(A)}{2^{|A|}} \tag{5.26}$$

avec $\kappa(A)$, l'ensemble des parties de A dont leur fermeture est égale à A.

L'ensemble des parties $\kappa(A)$.

$$\kappa(A) = \left|\left\{\tilde{A} \subseteq A | \delta\left(\tilde{A}\right) = \delta(A)\right\}\right| \tag{5.27}$$

La formule présente le rapport entre le nombre de sous-ensembles de A dont la description est égale à la description de A et le nombre total des sous-ensembles de A. Plus un concept est stable, moins il est dépendant d'objets particuliers. Le calcul de la stabilité avec cette définition nécessite le calcul de toutes les descriptions des sous-ensembles de A, ce qui est un problème NP-Complet [Kuznetsov, 2007]. L'algorithme le plus connu pour le calcul de la stabilité est celui de Roth [Roth et al., 2008], qui propose un parcours du diagramme de Hasse. L'inconvénient majeur de la stabilité est la difficulté de comparer des concepts à fort support puisque la stabilité tend vers 1. Cette question a été discutée dans [Buzmakov et al., 2014, Jay et al., 2008], et la stabilité logarithmique a été proposée :

Stabilité Logarithmique. La stabilité logarithmique pour un concept $(A, \delta(A))$ est définie par :

$$\begin{aligned} LStab((A,\delta(A))) &= -\log_2(1 - Stab((A,\delta(A)))) \\ &= -\log_2\left(1 - \frac{\kappa(A)}{2^{|A|}}\right) \\ &= |A| - \log_2(2^{|A|} - \kappa(A)) \end{aligned} \tag{5.28}$$

Cette formule peut donner une valeur infinie et elle n'est pas normalisée entre les concepts [Buzmakov et al., 2014]. Nous étendons la stabilité logarithmique d'un concept à la **stabilité logarithmique normalisée** $\lambda((A, \delta(A)))$ où les valeurs sont comprises entre 0 et 1.

Stabilité Logarithmique Normalisée. La mesure de stabilité logarithmique normalisée pour un concept $(A, \delta(A))$ est :

$$\begin{aligned} \lambda((A,\delta(A))) &= \frac{-\log_2(1 - Stab((A,\delta(A))) + \frac{1}{2^{|A|}})}{|A|} \\ &= \frac{-\log_2\left(\frac{2^{|A|} - \kappa(A) + 1}{2^{|A|}}\right)}{|A|} \\ &= \frac{|A| - \log_2\left(2^{|A|} - \kappa(A) + 1\right)}{|A|} \\ &= 1 - \frac{\log_2(2^{|A|} - \kappa(A) + 1)}{|A|} \end{aligned} \tag{5.29}$$

$$\lambda((\emptyset, \delta(\emptyset))) = 1 \text{ par convention} \tag{5.30}$$

On peut remarquer que :

- $\lambda((A, \delta(A))) = 1$ quand $\kappa(A) = 2^{|A|}$, ce qui signifie que le concept est très stable.
- $\lambda((A, \delta(A))) = 0$ quand $\kappa(A) = 1$, ce qui signifie que le concept n'est pas stable.
- le cas spécial où $A = \emptyset$ ($2^{|A|} = \kappa((A, \delta(A))) = 1$) donne $\lambda((\emptyset, \alpha(\emptyset))) = 1$ par convention.

La stabilité est définie pour un concept et sert par exemple à sélectionner certains concepts. Pour mesurer l'impact des descriptions et stratégies, nous l'étendons à l'ensemble des concepts d'un treillis afin de pouvoir comparer des treillis. Pour cela, nous étendons la stabilité d'un concept à la **stabilité logarithmique normalisée globale** d'un treillis de concepts L.

Stabilité Logarithmique Normalisée Globale.

$$\lambda(L) = \sum_{(A,\delta(A))\in L} p((A, \delta(A)))\lambda((A, \delta(A))) \tag{5.31}$$

Où $p((A, \delta(A)))$ est la probabilité qu'un concept $(A, \delta(A))$ soit généré :

$$p((A, \delta(A))) = \frac{\kappa(A)}{2^{|G|}} \tag{5.32}$$

Cette mesure peut naturellement être étendue à l'AFC classique pour un concept (A, B) avec $\beta(A)$ au lieu de $\delta(A)$.

5.2 Représentabilité et Distinctivité

La *Représentabilité* et la *Distinctivité* sont deux autres mesures de qualité non supervisées définies pour mesurer la qualité d'un treillis de concepts et plus précisément leur capacité à distinguer les différents objets avec un nombre suffisant de prédicats générés par les descriptions. Ces deux mesures se basent sur deux types de concepts particuliers dans le treillis, qui sont les concepts irréductibles. Un concept inf-irréductible correspond à un prédicat généré par une description. Un concept sup-irréductible correspond à un objet ou groupes d'objets initiaux. Le rapport entre le nombre des concepts sup-irréductibles $|J(\mathcal{L})|$ ou inf-irréductibles $|M(\mathcal{L})|$ et le nombre de concepts totale $|\mathcal{L}|$ donne les deux mesures de qualité :

La représentabilité. montre dans quelle mesure nous pouvons préserver plus d'information représentée par les prédicats générés :

$$R(\mathcal{L}) = \frac{|M(\mathcal{L})|}{\mathcal{L}} \tag{5.33}$$

La distinctivité. est une mesure de séparation d'un treillis de concepts, qui indique à quel point nous distinguons les objets :

$$D(\mathcal{L}) = \frac{|J(\mathcal{L})|}{\mathcal{L}} \tag{5.34}$$

6 Les extensions de données séquentielles

Les méthodes d'analyse de séquences proposées ont été implémentées aux format d'extensions pour la plateforme `GALACTIC`. Ces extensions sont de trois types, les caractéristiques, les descriptions et les stratégies, et permettent d'analyser les trois types de séquences.

La Figure 5.22 présente l'architecture de `GALACTIC` auquel nous ajoutons des extensions de caractéristiques, de descriptions et de stratégies permettant l'analyse des données séquentielles.

6.1 Caractéristiques

Les caractéristiques désignent le type de données utilisé pour l'analyse. Nous avons mis en place quatre caractéristiques pour représenter les données séquentielles. Elles sont résumées dans la Table 5.10, avec la couverture de code et le nombre de lignes de code et de commentaires. La couverture de code est une mesure qui représente le taux de code exécuté suite au lancement des tests.

Type de données	extension	couverture	code	commentaires
Séquences simples	`string`	82%	192	210
	`chain`	81%	198	226
Séquences temporelles	`séquence`	83%	211	220
Séquences d'intervalles	`interval`	88%	306	285

TABLE 5.10 – Extensions de type caractéristique des données séquentielles avec quelques statistiques

`String`. Il s'agit d'un type de séquences qui représente les séquences de caractères. C'est un choix que nous avons fait de considérer ce type à part des séquences simples car pour les chaines de caractères l'analyse sera beaucoup plus rapide en utilisant les algorithmes des expressions régulières. Cette extension renvoie pour chaque objet la chaîne correspondante.

`Chain`. Cette caractéristique permet de représenter une séquence simple par un ensemble d'éléments ordonnées, l'extension vérifie bien que tous les éléments de la séquence appartiennent à l'ensemble d'alphabet Σ.

`Séquence`. Cette caractéristique permet de représenter une séquence par une structure de données composé de plusieurs paires (horodatage, élément).

`Interval`. Comme pour la caractéristique de type séquence, cette caractéristique utilise un dictionnaire pour représenter les séquences d'intervalles, mais la clé sera constituée d'une paire représentant le temps de début et de fin de l'élément. Nous avons utilisé

pour cette caractéristique la bibliothèque `PART` [1] conçue pour traiter les séquences d'intervalles.

6.2 Descriptions

Les descriptions que nous avons proposées sont implémentées comme des extensions contenant les méthodes nécessaires pour décrire un ensemble d'objets défini par une caractéristique. Cinq extensions ont été développées, deux pour les strings, une pour les séquences simples, une pour les séquences temporelles et une pour les séquences d'intervalles. Ces extensions sont des classes python avec les méthodes permettant le calcul des prédicats des descriptions. La Table 5.11 présente les descriptions développées avec quelques statistiques (couverture de code, nombre de lignes de code, nombre de lignes de commentaires).

Type de données	extension	couverture	lignes de code	lignes de commentaires
Séquences simples	`string-match`	90%	615	540
	`string-distance`	90%	434	351
	`chain-match`	84%	909	1006
Séquences temporelles	`sequence-distance`	53%	668	899
Séquences d'intervalles	`interval-sequence`	83%	503	579

TABLE 5.11 – Extensions de type description pour les données séquentielles avec quelques statistiques

`String-Match.` Cette description implémente des descriptions et prédicats pour les séquences simples de type chaines de caractères. La description définie deux prédicats, *SimpleMatch*, pour des prédicats simples de strings de type sous-séquence. Et *PrefixMatch* pour des prédicats de type sous-séquence préfixe. Cette extension définie également trois espaces de descriptions pour les chaines de caractères, *SimpleMatchDescription* qui correspond à la description KSC décrivant un ensemble de chaines de caractères par des sous-séquences de longueur définie par l'utilisateur. *CompleteMatchDescription* décrit un ensemble de chaines de caractères par des sous-séquences maximal en commun (SCM). *PrefixMatchDescription* décrit un ensemble de chaines de caractères par des sous-séquences de types préfixe (SCP).

`String-Distance.` Cette description implémente la description *distancielle* pour les séquences simples de type chaines de caractères, l'horodatage est considéré comme la position du caractère dans la chaîne. La description définie un prédicat appelé *DistanceMatch*, qui donne la distance entre les caractères de la chaines avec la

1. `https://github.com/chdemko/py-part`

cardinalité de leurs apparitions dans la chaîne. L'extension définit aussi un espace de description *DistanceMatchDescription* pour la description *Distancielle* générant les prédicats au format d'expressions régulières.

`Chain-Match.` Cette description définit trois prédicats et quatre descriptions pour les séquences simples. *SimpleMatch* définie un prédicat simple de type sous-séquence. *PrefixMatch* définie un prédicat de type sous-séquence préfixe. *AffixMatch* est une version plus générale de *PrefixMatch* définissant un prédicat d'affixe qui désigne soit des sous-séquences préfixes, soit suffixes ; l'utilisateur peut choisir. Cette extension définie également quatre espaces de descriptions pour les séquences simples générant les sous-séquences communes maximales, les sous-séquences communes de type préfixe et suffixe, et les sous-séquences avec taille fixe. La description *SimpleMatchDescription* décrit un ensemble de séquences par des sous-séquences de longueur définie par l'utilisateur (description KSC). *CompleteMatchDescription* décrit un ensemble de séquences par des sous-séquences communes maximales (SCM). *PrefixMatchDescription* décrit un ensemble de séquences par une sous-séquence préfixe (SCP). *AffixMatchDescription* décrit un ensemble de séquences par une sous-séquence de type préfixe ou suffixe, et accepte un paramètre d'*orientation*, qui définit s'il s'agit de préfixe ou suffixe, et un autre paramètre *start*, qui précise une position de départ. Par exemple, on peut décrire un ensemble de séquences par les préfixes qui commence à partir de deuxième positon de la séquence.

`Sequence-Distance.` Cette description définit le prédicat *DistanceMatch* de type sous-séquence distancielle. Cette extension définie un espaces de descriptions générant des prédicats de type *DistanceMatch*. *DistanceMatchDescription* décrit un ensemble de séquences temporelles par des sous-séquences communes distancielles. Cette description accepte deux paramètres : *fenêtre* et *écart*.

`Interval-Sequence.` Cette description définit deux prédicats et deux descriptions. *IntervalSubSequenceMatch* est un prédicat simple de sous-séquence d'intervalles. *IntervalSupSequenceMatch* est un prédicat simple de super-sequence d'intervalles. Les deux descriptions pour les séquences d'intervalles génèrent les sous-séquences communes maximales et les super-séquences communes minimales. *MaximalCommonIntervalDescription* représente un espace de description de séquences d'intervalles avec les sous-séquences communes maximales (SCMAX). *SharedIntervalDescription* représente un espace de description de séquences d'intervalles avec les super-séquences communes minimales (SCMIN).

6.3 Stratégies

Les stratégies sont également implémentées comme des extensions qui génèrent un ensemble de sélecteurs (ou prédicats). Cinq extensions de stratégies ont été développées, deux pour les chaines de caractères, une pour les séquences simples, une pour les séquences temporelles, et une pour les séquences d'intervalles. Ces extensions sont des classes python avec les méthodes permettant le calcul des sélecteurs. La Table 5.12 présente les stratégies

développées avec quelques statistiques (couverture de code, nombre de lignes de code, nombre de lignes de commentaires).

Type de données	extension	couverture	lines de code	lignes de commentaires
Séquences simples	`string-match`	90%	382	375
	`string-distance`	87%	206	218
	`chain-match`	90%	308	380
Séquences temporelles	`sequence-distance`	35%	634	521
Séquences d'intervalles	`interval-sequence`	89%	472	521

TABLE 5.12 – Extensions de type stratégie pour les données séquentielles avec quelques statistiques

`String-match-basic.` Cette extension implémente trois stratégies. *SimpleMatchStrategy* correspond à la stratégie *Naïve* définie pour la description KSC. *PrefixMatchStrategy* représente la stratégie *Augmentée* pour la description SCP. *CompleteMatchStrategy* représente la stratégie *Augmentée* pour la description SCM.

`String-distance-basic.` Cette extension implémente la stratégie *DistanceMatchStrategy* pour une stratégie de chaines de caractères avec utilisation de distances entre les les éléments de la chaines.

`Chain-match-basic.` Cette extension implémente quatres stratégies. *SimpleMatchStrategy* correspond à la stratégie *Naïve* définie pour la description KSC. *PrefixMatchStrategy* représente la stratégie *Augmentée* pour la description SCP. *NaiveChainMatchStrategy* représente la stratégie *Naïve* définie pour la description SCM. *CompleteMatchStrategy* représente la stratégie *Augmentée* pour la description SCM.

`Sequence-distance-basic.` Cette extension implémente deux stratégies. *NaiveSequenceStrategy* est une stratégie distancielles pour les séquences temporelles, elle définit les deux stratégies *Naïve* et *Par étape*. Elle accepte un paramètre de *pas*, s'il est défini cela correspond à la stratégies *Par étape*, sinon sa valeur par défaut est 1 ce qui correspond à la stratégie *Naïve*. *MiddleSequenceStrategy* définit la stratégie *Milieu* pour les séquences temporelles.

`Interval-sequence-basic.` Cette extension implémente cinq stratégies pour les séquences d'intervalles. *AugmentedMinimumCardinalityStrategy* est une stratégie d'intervalle pour la description *MaximalCommonIntervalDescription*. Pour la description *SharedIntervalDescription*, cette extension définit les quatre stratégies : *SimpleTimeFrameStrategy*, *BoundsTimeFrameStrategy* qui accepte un paramètre *bound* : 0 pour minimale cardinalité et 1 pour maximale cardinalité, *WindowAffixTimeFrameStrategy* avec un paramètre *window*, et *AlphabetTimeFrameStrategy*.

7 Conclusion

À travers ce chapitre nous avons présenté différentes descriptions et stratégies qui offrent plusieurs manières d'analyser des séquences que ce soit des séquences simples, des séquences temporelles ou des séquences d'intervalles. L'analyse présentée ici est une analyse qui positionne l'analyste des données au coeur du processus, en lui proposant plusieurs descriptions et stratégies.

Les descriptions représentent une manière de décrire un ensemble de données répondant à la question *"comment décrire un ensemble de séquences ?"*. Le choix de nos descriptions est basé sur les sous-séquences communes maximales qui peuvent résumer les comportements d'un ensemble de séquences.

Les stratégies permettent l'exploration de l'ensemble de séquences en ajoutant une information élémentaire qui va permettre de générer des prédécesseurs d'un concept. Les stratégies peuvent générer tous les prédécesseurs possibles ou bien se focaliser sur un sous-ensemble de concepts, permettant d'obtenir des treillis plus petit dont le diagramme de Hasse est plus aisé à lire, et ainsi réduire le nombre de motifs générés en fonction des attentes de l'analyste de données.

Nous avons également introduit des mesures de qualité afin de mesurer et comparer l'efficacité des descriptions et stratégies. Les mesures présentées sont des mesures non supervisées utilisant la stabilité et les éléments irréductibles pour déterminer la qualité d'un treillis.

À la fin de chapitre, nous avons présenté la partie implémentation dans l'outil `GALACTIC`. Cette partie décrit la façon dont les caractéristiques, descriptions et stratégies sont implémentées, avec quelques statistiques sur le code. Pour chaque extension, une documentation complète est fournie avec des exemples d'exécutions.

La Table 5.13 résume toutes les descriptions et stratégies présentées.

Le chapitre suivant décrit quelques expérimentations avec une analyse qualitative qui met en avant la sémantique des concepts générés et une analyse quantitative avec des mesures de la qualité des treillis générés.

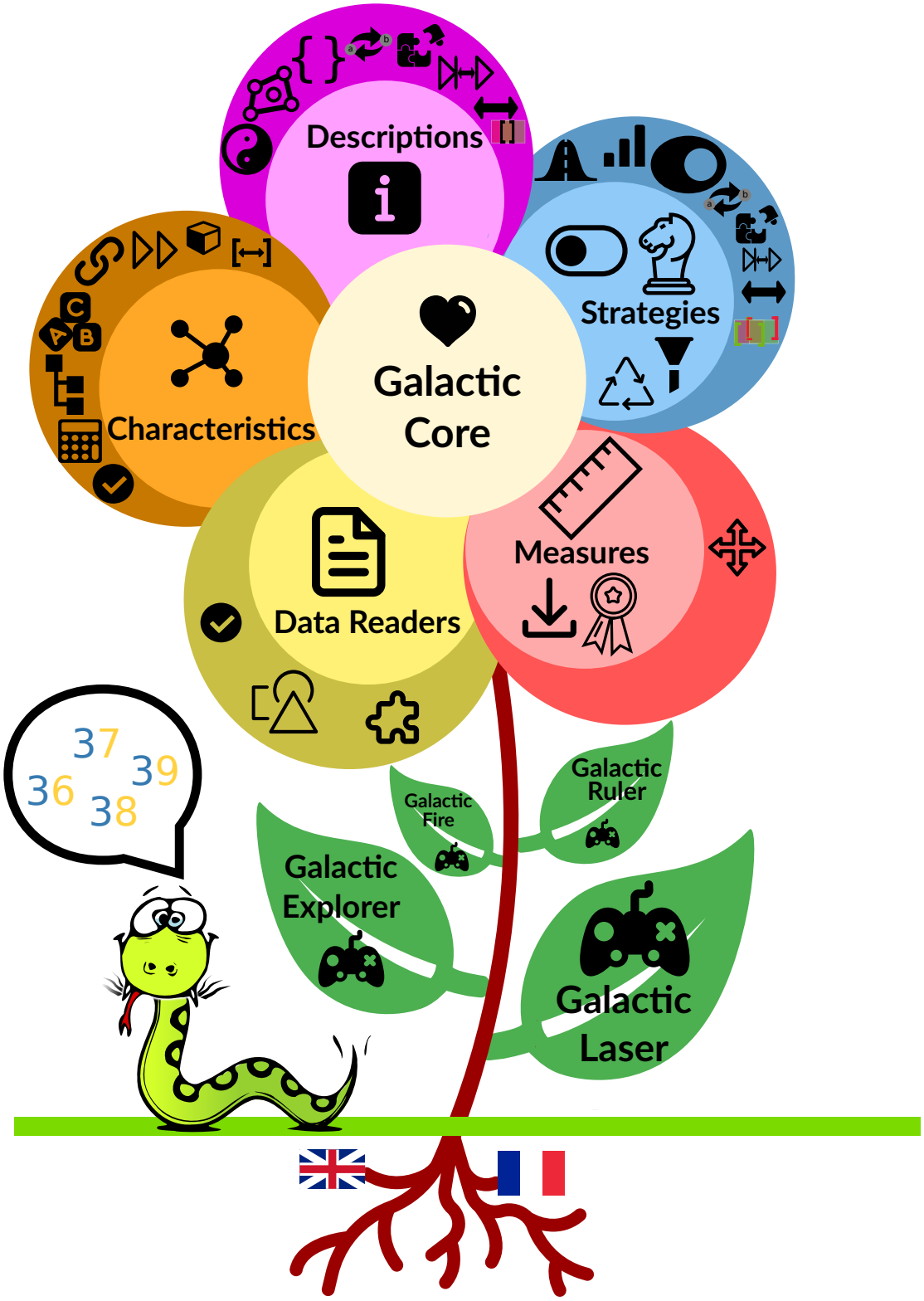


FIGURE 5.22 – l'architecture de GALACTIC avec les extensions pour les séquences

Type de séquence	Descriptions	Stratégies	Mesures
Séquences simples Extension :`Chain`	SCM (Def 5.2, p 94) SCP (Def 5.3, p 95) KSC (Def 5.4, p 97) Extension : `Chain-Match`	SN (Def 5.5, p 99) SA (Def 5.6, p 99) Extension : `Chain-match-basic`	Stabilité Logarithmique Normalisée Globale (eq 5.31, p 131) Représentabilité (eq 5.33, p 131) Distinctivité (eq 5.34, p 131)
Séquences temporelles Extension :`Séquence`	SDCM (Def 5.10, p 108) Extension : `Sequence-Distance`	SDP (Def 5.13, p 111) SDN (Def 5.14, p 111) SDM (Def 5.15, p 111) Extension : `Sequence-distance-basic`	
Séquences d'intervalles Extension :`Interval`	SCMAX (Def 5.20) SCMIN (Def 5.21, p 120) Extension : `Interval-Sequence`	CMA (Def 5.22, p 122) NCT (Def 5.23, p 123) BCT (Def 5.24, p 123) FACT (Def 5.25, p 123) ACT (Def 5.26, p 124) Extension : `Interval-sequence-basic`	

TABLE 5.13 – Résumé des descriptions, stratégies et mesures proposées.

Chapitre 6

Mesures et Expérimentations

Sommaire

1 Introduction 140
2 Jeux de données 141
2.1 Jeux de données synthétiques 141
2.2 La Cité du Vin 141
2.3 GéoLuciole 142
2.4 Daily Actions 144
2.5 Catch me if you can 144
3 Analyse qualitative 145
3.1 Impact de changement de stratégies sur l'analyse 145
3.1.1 Analyse de `GéoLuciole` avec δ_{SCM}-σ_{SN} 145
3.1.2 Analyse de `GéoLuciole` avec δ_{SCM}-σ_{SA} 146
3.2 Impact du changement de la description sur l'analyse 147
3.2.1 Analyse de `Cité-du-vin` avec δ_{SCP}-σ_{SA} 147
3.2.2 Analyse de `GéoLuciole` avec δ_{KSC}-σ_{SN} 149
3.3 Analyse hétérogène de `GéoLuciole` 150
3.3.1 Focus sur une information géographique : La plage . . 150
3.3.2 Focus sur une information temporelle : La nuit 151
4 Analyse quantitative 152
4.1 Séquences simples 152
4.1.1 Fouille de motifs séquentiels fermés avec δ_{SCM}-$\sigma_{\text{SN,SA}}$. 152
4.1.2 Analyse de `Daily-actions` avec $\delta_{\text{KSC,SCM}}$-σ_{SN} 153
4.1.3 Analyse de `Cité-du-vin` avec $\delta_{\text{SCP,SCM}}$-σ_{SA} 154
4.1.4 Analyse de `Cité-du-vin` avec tous les descriptions et stratégies 155
4.2 Séquences temporelles 155
4.2.1 Analyse de `Catch-me` et `Cité-du-vin` avec δ_{SDCM}-$\sigma_{\text{SDN,SDM}}$ 155

4.2.2 Analyse de `Catch-me` et `Cité-du-vin` avec δ_{SDCM}-σ_{SDP} 158
4.2.3 Analyse de `Cité-du-vin` avec $\delta_{\text{SDCMw,SDCMg}}$-$\sigma_{\text{SDN}}$ . . 159
4.3 Séquences Intervalles 159
4.3.1 Analyse de `Cité-du-vin` avec δ_{SCMAX}-σ_{CMA} et δ_{SCMIN}-σ_{NCT} 160
4.3.2 Analyse de `Cité-du-vin` avec δ_{SCMIN}- $\sigma_{\text{NCT,BCT,FACT,ACT}}$ 160
4.3.3 Analyse de `GéoLuciole` avec δ_{SCMIN}-$\sigma_{\text{NCT,BCT,FACT,ACT}}$ 161
5 Discussion et conclusion 162

1 Introduction

Ce chapitre est consacré à la validation expérimentale de notre approche d'analyse de données séquentielles. Nous utilisons la plateforme de développement `GALACTIC` de l'algorithme NextPriorityConcept décrite dans le Chapitre 4 qui propose un système d'extensions permettant d'intégrer facilement de nouveaux types de données, de nouvelles descriptions et stratégies. Nous avons implémenté les descriptions et stratégies introduites dans le chapitre précédent, organisées en trois types de plugins (caractéristiques, descriptions et stratégies) pour les séquences simples, temporelles et d'intervalles.

Deux types d'analyses seront présentées dans ce chapitre :

Premièrement, une analyse qualitative qui présente certains concepts générés qui regroupent des objets dont la description commune s'interprète en fonction de la sémantique des données. Nous présenterons quelques exemples des interprétations que nous avons pu obtenir après l'utilisation de notre méthode avec des séquences simples, temporelles et d'intervalles.

Deuxièmement, dans une analyse quantitative nous faisons varier la taille de l'ensemble de données (nombre de séquences) à l'aide d'une fonction aléatoire, puis chaque expérience est réalisée dix fois, et nous calculons la moyenne de chaque valeur de mesure. Nous utilisons plusieurs mesures pour montrer l'efficacité de nos approches dont la stabilité logarithmique globale, la représentabilité, la distinctivité et le nombre de concepts et prédicats générés.

Dans ce chapitre, les figures présentant des treillis seront des diagrammes de Hasse décrite sous forme réduite (cf Chapitre 3). Nous utilisons la notation δ_{X}-σ_{Y} pour une analyse avec la description X et la stratégie Y. La notation δ_{X}-$\sigma_{Y_1, Y_2, \ldots}$ est utilisée pour des analyses avec la description X et plusieurs stratégies $Y_1, Y_2, \ldots$ et $\delta_{X_1, X_2, \ldots}$-$\sigma_{\text{Y}}$ est utilisée pour des analyses avec les descriptions $X_1, X_2, \ldots$ et la stratégie Y.

Le reste de ce chapitre est organisé en quatre sections. Nous commençons par présenter les jeux de données utilisés. Ensuite, nous présentons dans la deuxième section l'analyse qualitative, et dans la troisième section l'analyse quantitative. Et enfin, nous finissons le chapitre par une discussion et conclusion.

2 Jeux de données

Les expérimentations ont été réalisées sur une machine Intel Core i7 2.20GHz avec 32GB de mémoire principale. Elles ont porté sur des jeux de données synthétiques et réelles. Nous considérons trois groupes des données synthétiques et quatre jeux de données réelles pour évaluer l'efficacité de nos descriptions et stratégies. Les jeux de données sont résumés dans la Table 6.1.

Jeu de données	Taille de données $\|G\|$	Taille de l'alphabet $\|\Sigma\|$	Taille moyenne des séquences $\|S\|$
`Group 1`	$\{25, 50, 75, 100, 200\}$	30	7
`Group 2`	100	$\{10, 20, 30\}$	7
`Group 3`	100	30	$\{5, 6, 7, 8\}$
`Daily-actions`	25	12	8
`Cité-du-vin`	700000	20	9
`GéoLuciole`	192	21	
`Catch-me`	82797	16000	6.4

TABLE 6.1 – L'ensemble de jeux de données utilisés pour les expérimentations

2.1 Jeux de données synthétiques

L'ensemble des jeux de données synthétiques sont générés par le logiciel *Quest Dataset Generator* d'*IBM* hébergé sur le site web de *SPMF*[1] (*Sequential Pattern Mining Framework*). Ces jeux de données ont été générés en faisant varier trois paramètres :

- la taille du jeu de données $x = |G|$.
- la taille de l'alphabet $y = |\Sigma|$.
- la taille des séquences $z = |S|$.

Les jeux de données générés sont nommés $GxAySz$ avec x la taille de jeu de données, y la taille de l'alphabet et z la taille moyenne des séquences. Par exemple $G100A30S7$ correspond à un jeu de données synthétique obtenu avec $x = |G| = 100$, $y = |\Sigma| = 30$ et $z = |S| = 7$.

2.2 La Cité du Vin

Le jeu de données `Cité-du-vin` est issu du musée "La Cité du Vin" à Bordeaux, France[2], recueillie à partir des visites sur une période d'un an (mai 2016 à mai 2017). Le musée est un grand espace ouvert (*open-space*), où les visiteurs sont libres d'explorer le musée comme ils le souhaitent sans parcours prédéterminé. Lorsqu'ils arrivent au musée,

1. http ://www.philippe-fournier-viger.com/spmf/
2. https ://www.laciteduvin.com/en

ils reçoivent un petit dispositif personnel. Ce dispositif permet d'écouter des explications quand l'utilisateur passe devant les lieux d'animations. Le musée est composé de plusieurs espaces dits modules. Chaque module est composé de plusieurs sous-modules qui sont des lieux d'animations. La Figure 6.1 représente le plan du musée.

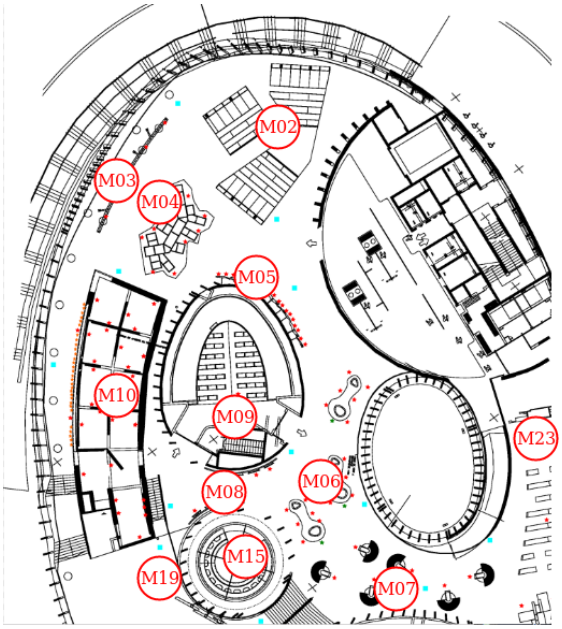


FIGURE 6.1 – Plan de la Cité du Vin à Bordeaux, France.

Le musée contient environ 20 modules représentant l'ensemble d'alphabet Σ. En extrayant les séquences d'activation dans chaque module, nous obtenons une idée assez précise de ce à quoi ressemblait la visite pour chaque visiteur du musée. Les informations extraites permettent d'obtenir des séquences simples de succession de visites de modules. En ajoutant l'information temporelle d'activation du premier sous-module de chaque module on obtient des séquences temporelles. Et en prenant le dernier et le premier sous-module de chaque module on obtient des séquences d'intervalles. Le jeu de données complet contient plus de 700000 séquences, avec une taille moyenne de 9.

2.3 GéoLuciole

Le jeu de données `GéoLuciole` est issu des trajectoires GPS classiques des déplacements de personnes dans la ville de La Rochelle en France. Ce jeu de données a été collecté pendant l'été 2019, auprès de touristes visitant la ville pendant leurs vacances. Il contient 192 trajectoires provenant de différents individus. Leurs positions avaient été obtenues par une application spécifique nommée `GéoLuciole` développée dans le cadre de projet DA3T (Dispositif d'Analyse des Traces numériques pour la valorisation des Territoires Touristiques), donnant leur position toutes les 30 secondes pendant leur visite. Avant d'activer l'application nous leur avons également demandé de remplir un questionnaire

(le nombre de fois qu'ils ont visité la ville, le moyen d'arrivée, s'ils étaient avec des amis ou de la famille, etc.). En faisant correspondre les coordonnées GPS aux quartiers de la ville, comme le montre la Figure 6.2, nous pouvons transformer ces données brutes en séquences d'intervalles interprétées comme des trajectoires sémantiques, où l'alphabet Σ correspond aux quartiers où chaque ensemble de coordonnées GPS dans le même quartier devient un élément de la séquence avec une information temporelle.

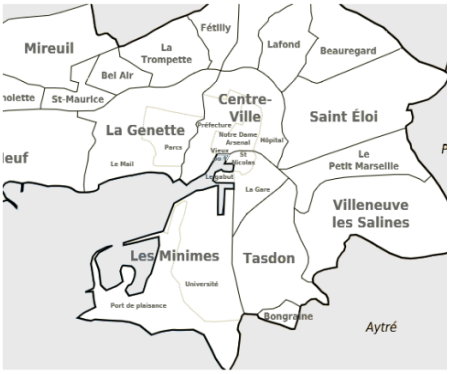


FIGURE 6.2 – Segmentation de la ville de La Rochelle en quartiers

Dans le cadre du projet DA3T, les trajectoires brutes ont ainsi été enrichies en plusieurs séquences sémantiques selon un modèle d'enrichissement multi-niveaux et multi-aspects [Cayèré et al., 2021] (cf Fig 6.3). Les séquences "*beach*" et "*green space*" sont définies de la même façon à partir des plages et des espaces verts de la ville. Les séquences "*weather*" et "*tide*" sont construites à partir d'éléments extraits du web sur les alphabets { *"clear sky", "cloudy", "rainy"* } pour la météo et {*"hight tide", "low tide"*} pour l'information sur la marée.

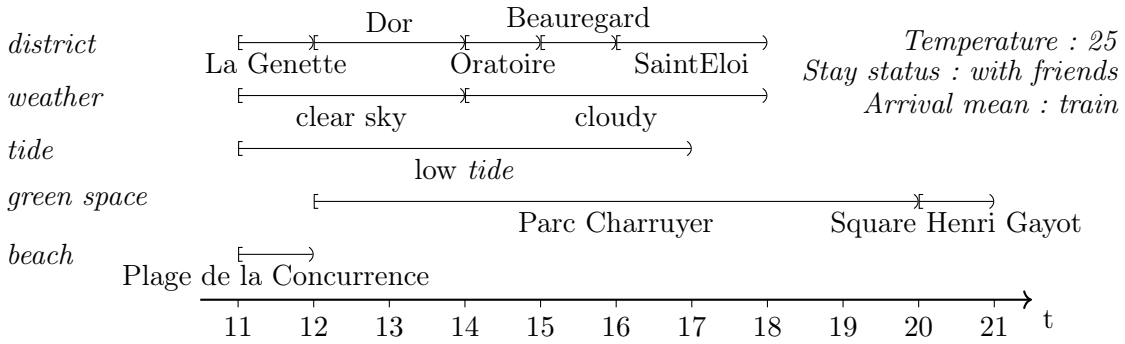


FIGURE 6.3 – Une trajectoire enrichie d'un touriste à La Rochelle.

Des caractéristiques simples de contexte sont également ajoutées :

- *temperature* : attribut numérique
- *stay status* : attribut catégoriel

- *arrival mean* : attribut catégoriel décrivant le moyenne d'arrivée sur place

La Table 6.2 résume les aspects des données présents dans le jeu de données `GéoLuciole` avec leurs types, leurs alphabets et leurs sources.

Aspect	Type	Alphabet	Source
district	Séquence d'intervalles	Σ = l'ensemble des quartiers de la ville (*district*)	Trajectory
weather	Séquence d'intervalles	$\Sigma = \{$"*raining*", "*moderate raining*", "*cloudy*", "*sunny*"$\}$	Web scrapping
beach	Séquence d'intervalles	$\Sigma = \{$"*no beach*", "*Plage de la Concurrence*", "*Plage des Minimes*"$\}$	Trajectory
green space	Séquence d'intervalles	Σ = l'ensemble des parcs de la ville	Trajectory
tide	Séquence d'intervalles	$\Sigma = \{$"*high tide*", "*low tide*"$\}$	Web scrapping
average temperature	numérique		Web scrapping
stay status	catégoriel	$\Sigma = \{$"*with family*", "*with friends*", "*couple*", "*alone*"$\}$	questionnaire
arrival means	catégoriel	$\Sigma = \{$"*train*", "*car*", "*bus*", "*velo*"$\}$	questionnaire

TABLE 6.2 – Les aspects de jeu de données `GéoLuciole` avec leurs types, leurs alphabets et leurs sources

2.4 Daily Actions

Le jeu de données `Daily-actions` est un petit jeu de données de séquences qui représente les actions quotidiennes de 25 individus du laboratoire L3i [3] de *La Rochelle Université*. Pour collecter les données, nous avons envoyé un formulaire à l'ensemble des membres de laboratoire en leur demandant d'indiquer leurs actions quotidiennes avec une indication temporelle. Les actions quotidiennes considérées sont : $\Sigma = \{$ Wakeup, Breakfast, Work, Coffee, Rest, Lunch, Dinner, Read, Nap, Sports, Sleep, Other $\}$. La taille moyenne des séquences récoltées est de 8.

2.5 Catch me if you can

Le jeu de données `Catch-me-if-you-can` (`Catch-me` en abrégé) est composé de séquences de navigation d'utilisateurs sur des pages web. Ce jeu de données est disponible sur le site de Kaggle [4]. Les données sont issues de Open Data Science et proviennent des serveurs proxy de l'Université Blaise Pascal [Kahn et al., 2016]. Le jeu de données contient

3. https ://l3i.univ-larochelle.fr/
4. `https://www.kaggle.com/danielkurniadi/catch-me-if-you-can`

82797 séquences, avec une taille moyenne de 6,4, et une taille d'alphabet d'environ 16000 sites.

3 Analyse qualitative

Dans cette section, nous présenterons des résultats d'analyses de séquences sur un plan qualitatif, qui ont permis de générer des sous-groupes de séquences avec une interprétation sémantique pertinente de leur description.

3.1 Impact de changement de stratégies sur l'analyse

3.1.1 Analyse de `GéoLuciole` avec δ_{SCM}-σ_{SN}

L'utilisation de la description SCM avec la stratégie SN correspond à l'*analyse classique* des approches de fouille de séquences par sous-séquences communes maximales qui génèrent tous les sous-groupes possibles. Ceci correspond aussi aux approches issues de l'AFC qui visent à générer tous les sous-groupes possibles.

Avec 50 séquences de jeu de données `GéoLuciole`, ce type d'analyse a généré 6226 concepts en 50 minutes. Avec cette analyse on obtient un grand nombre de concepts difficilement interprétables, un "déluge de motifs". L'utilisation de filtres telle que le support permet de réduire le nombre de concepts. En limitant la génération des concepts selon une mesure de support à $\frac{2}{5}$, on obtient seulement 212 concepts (Figure 6.4).

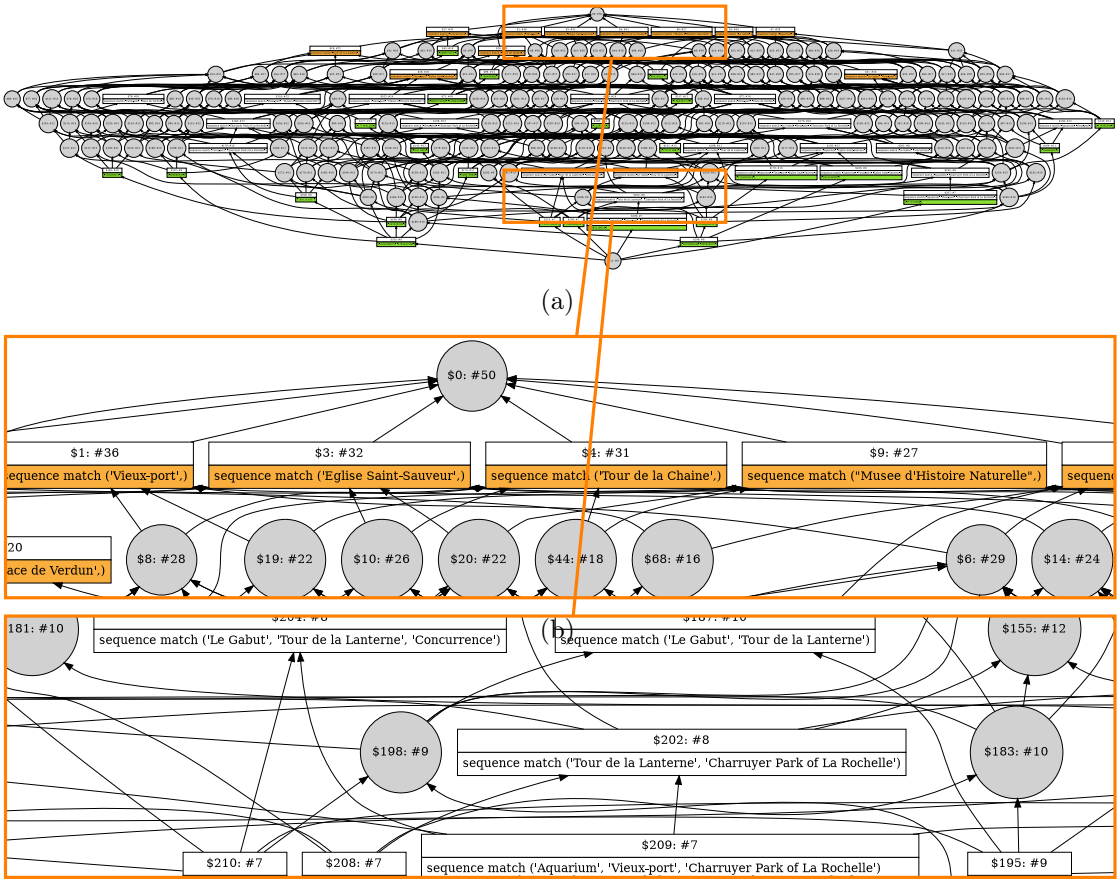


FIGURE 6.4 – Treillis de concepts généré par δ_{SCM}-σ_{SN} (support 20) pour `GéoLuciole`.

Le diagramme présente seulement les concepts avec au moins 20 séquences. La figure présente deux zooms sur la partie haute de treillis et la partie basse. Les premiers concepts contiennent des sous-séquences de taille 1, et plus on descend plus on aura des concepts avec des sous-séquences plus longues. Nous pouvons bien identifier ici les lieux les plus visités, par exemple, le concept $1 montre que 36 personnes parmi 50 ont visité le vieux-port.

3.1.2 Analyse de `GéoLuciole` avec δ_{SCM}-σ_{SA}

`GALACTIC` offre la possibilité de définir des stratégies d'exploration non naïves permettant de réduire le nombre de concepts et les rendre plus facilement interprétables.

La Figure 6.5 présente le treillis de concepts généré en utilisant la description SCM et la stratégie SA et un support de $\frac{2}{5}$. La stratégie SA permet de réduire le nombre de concepts à 169 au lieu de 212 avec la stratégie SN.

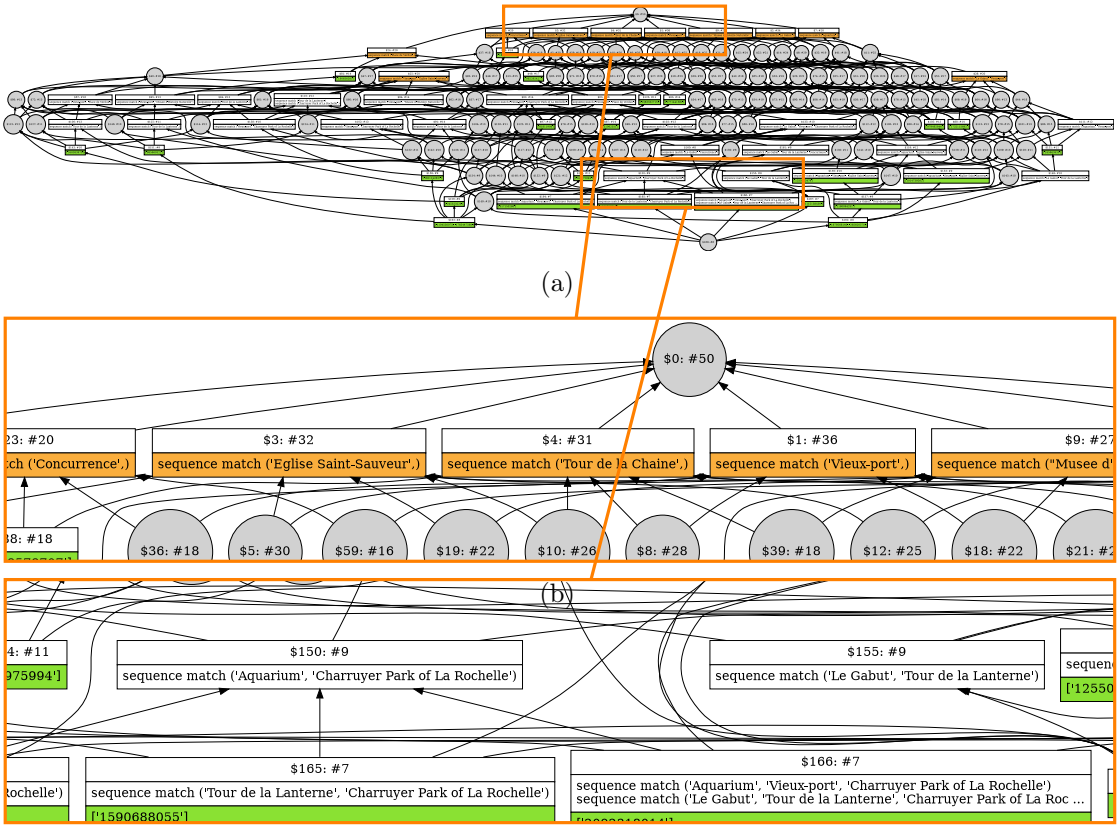


FIGURE 6.5 – Treillis de concepts généré par δ_{SCM}-σ_{SA} (support 20) pour `GéoLuciole`.

3.2 Impact du changement de la description sur l'analyse

La description SCM permet une analyse classique de séquences par sous-séquences communes maximales. Les descriptions SCP et KSC proposent des descriptions de sous-groupes par prefixe commun ou sous-séquences communes de taille fixe, qui peuvent s'envisager dans des analyses spécifiques.

3.2.1 Analyse de `Cité-du-vin` avec δ_{SCP}-σ_{SA}

Avec le jeu de données de la `Cité-du-vin`, une analyse préfixe des parcours des visiteurs ou de leur chemin de visite est pertinente pour mieux appréhender leur comportement de visite. La Figure 6.6 présente le treillis généré par la description SCP et la stratégie SA.

Dans cette figure le concept \$1 correspond au début de la visite qui débute par le module *M00*. Ensuite, les visiteurs partent dans quatre directions : *M03* avec support 25% (concept \$2), *M02* et *M04* avec support de 22% (concepts \$3 et \$4) et enfin *M05* avec 9% de support (concept \$10). Ces différents comportements de visite soulèvent plusieurs questions : pourquoi la majorité des visiteurs préfèrent *M02* à *M03* en début de visite ? Et pourquoi *M05* est moins préféré ? On peut également observer que les visiteurs qui

préfèrent *M02* en premier passent ensuite par *M03* et *M04* mais pas l'inverse. Et ceux qui commencent par *M03* passent ensuite par *M04*.

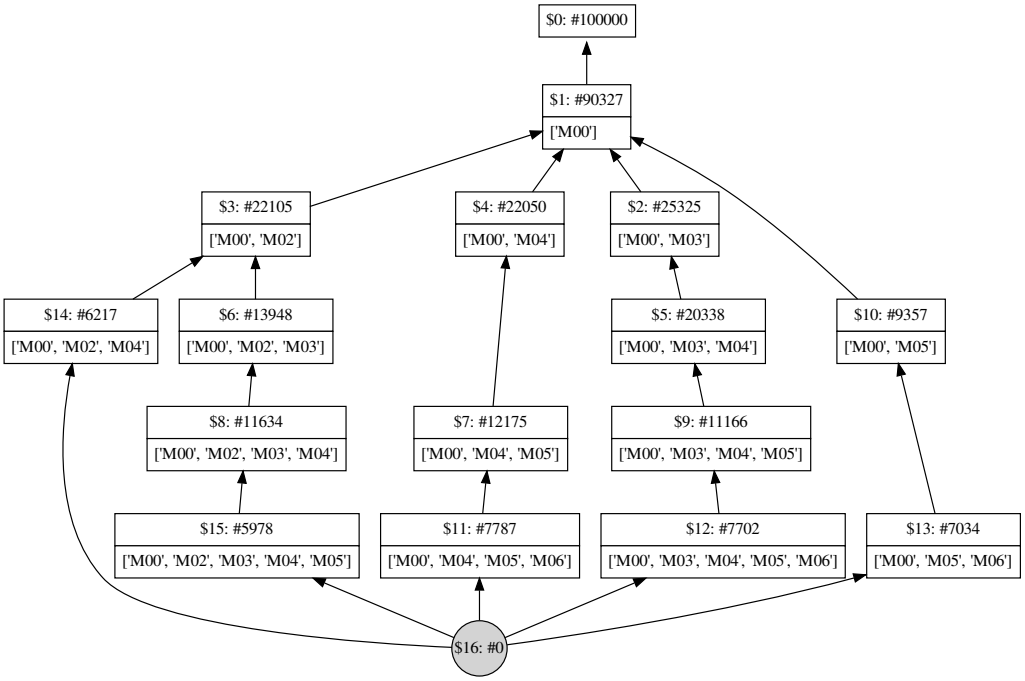


FIGURE 6.6 – Treillis de concepts généré par δ_{SCP}-σ_{SA} (support 5%) pour `Cité-du-vin`.

La Figure 6.7 présente une partie de digramme de Hasse du treillis généré avec la description SCP et la stratégie SA mais avec un support de 2%. Nous voyons dans cette partie que les visiteurs qui commencent leur visite avec *M04* vont ensuite vers *M05* et *M06* en majorité, et une petite partie vers *M10*.

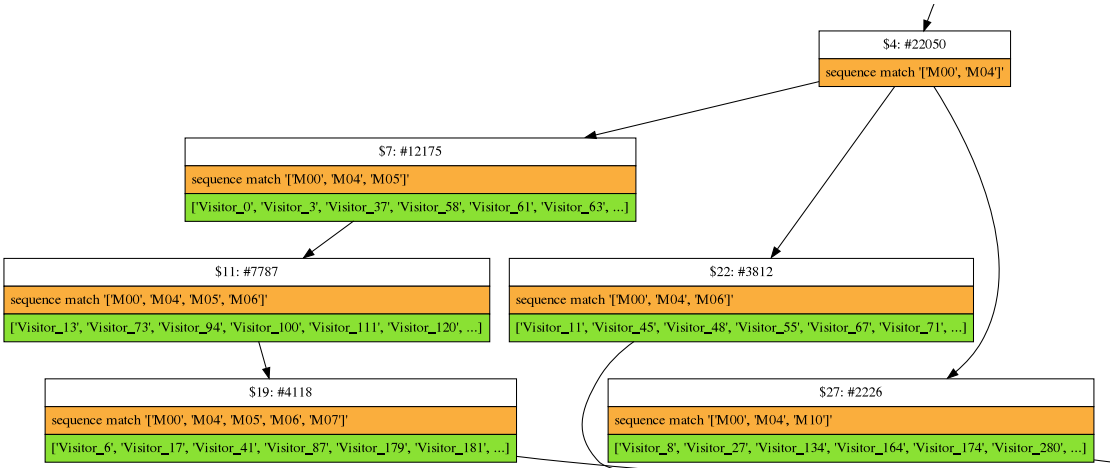


FIGURE 6.7 – Une partie du treillis de concepts généré par δ_{SCP}-σ_{SA} (support 2%) pour `Cité-du-vin`.

3.2.2 Analyse de `GéoLuciole` avec δ_{KSC}-σ_{SN}

Dans l'analyse classique la génération des concepts est effectuée niveau par niveau de telle sorte qu'à chaque niveau les concepts sont composés par des sous-séquences de plus longues tailles. Cela rend l'analyse difficile parfois si nous cherchons des sous-séquences avec une taille spécifique. Par exemple une entreprise de tourisme souhaiterait mettre en place une navette de circulation entre les trois lieux les plus visités par les touristes, nous pouvons dans ce cas générer directement un treillis avec la description KCS en fixant k à 3.

Reprenons le jeu de données `GéoLuciole`, la Figure 6.8 présente le treillis généré par la description KCS avec $k = 3$ et la stratégie SN, mais avec support $\frac{13}{50}$. Nous avons choisi ce support car c'est le support maximal qui permet de générer des concepts, i.e, il n'existe pas un ensemble de 14 séquences avec des sous-séquences communes de taille 3.

On peut dire donc que les chemins les plus visités (13 visites parmi 50) sont :

- $\langle$*Aquarium*, *Le Gabut*, *Vieux-Port*$\rangle$
- $\langle$*Le Gabut*, *Vieux-Port*, *Eglise Saint-Sauveur*$\rangle$
- $\langle$*Le Gabut*, *Vieux-Port*, *Charruyer Park de La Rochelle*$\rangle$

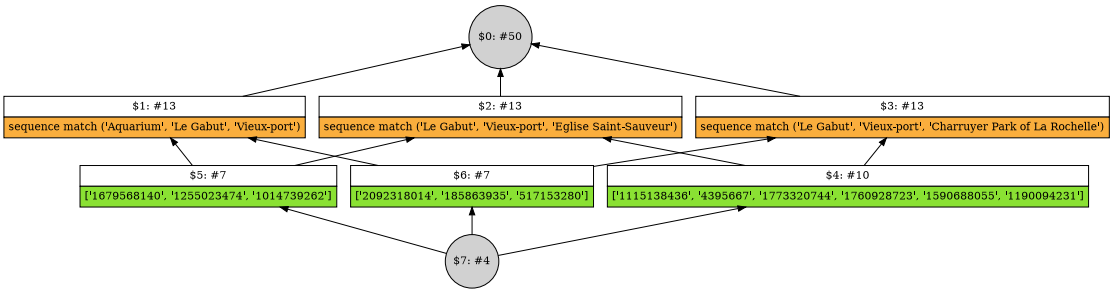


FIGURE 6.8 – Treillis de concepts généré par δ_{KSC}-σ_{SN} (support 26%) pour `GéoLuciole`.

3.3 Analyse hétérogène de `GéoLuciole`

Nous effectuons maintenant une analyse hétérogène en utilisant la temporalité avec les données de `GéoLuciole`. Une façon pour l'analyste de données d'analyser l'ensemble de données est de s'intéresser à une information ciblée des données.

3.3.1 Focus sur une information géographique : La plage

Afin de se concentrer sur les visiteurs allant à la plage, nous analysons la plage avec la météo qui est sémantiquement significative pour cette expérience. Avec 108 trajectoires, 93 concepts sont générés en utilisant la description SCMAX et la stratégie CMA pour les attributs de météo et de visite de plage. La Table 6.3 montre certains concepts contenant des informations précieuses. Elle présente la description du concept avec le support, le nombre de touristes et le support par rapport aux trajectoires contenant la pluie (support *"raining"*). La table montre une forte corrélation entre "pluie" et "pas de plage", ce qui signifie que les gens n'iront pas à la plage s'il pleut. Dans notre jeu de données, nous avons 31 trajectoires avec de la pluie. Sur cette partie de la trajectoire, nous pouvons voir que 8 d'entre elles peuvent être décrites avec un prédicat qui correspond à "pluie" entre 11 et 12 heures et "pas de plage". D'autres prédicats sont également générés qui décrivent également que le mauvais temps a un impact sur un voyage à la plage et ce prédicat semble se concentrer sur le mauvais temps le matin.

Description	support	support *"raining"*	touristes
météo match [[11 ;12) :[pluie]] plage match [[0 ;24) :[pas de plage]]	0.074	0.260	8
météo match [[1 ;2) :[pluie]] plage match [[0 ;24) :[pas de plage]]	0.037	0.130	4
météo match [[8 ;9) :[pluie]] plage match [[0 ;24) :[pas de plage]]	0.028	0.100	3
météo match [[1 ;4) :[pluie]] plage match [[0 ;24) :[pas de plage]]	0.019	0.065	2

TABLE 6.3 – Exemple de prédicats montrant l'impact du climat sur une excursion à la plage.

3.3.2 Focus sur une information temporelle : La nuit

Grâce à l'information temporelle, il est également possible d'analyser une partie de la journée. Une analyse possible pour l'utilisateur consistera à voir si les habitudes de sommeil dépendent du statut de séjour du touriste. Pour ce faire, nous ferons une analyse avec deux attributs hétérogènes, le statut de séjour, qui est une chaîne, et les lieux, qui sont des intervalles. Avec 138 trajectoires, nous avons au total 340 concepts, et 108 pour la période de sommeil (pas de mouvement entre 00h00 et 10h00). La Table 6.4 présente certains prédicats/motifs générés. On peut observer que, les prédicats générés sont soutenus par près de 10 % des trajectoires "en famille" et plus de 10 % des trajectoires "avec des amis" pour localiser les sous-séquences, qui semblent représenter un lieu de sommeil.

Description	support	support *"status"*	touristes
statut de séjour : en famille quartier match [[9 ;10) :[Oratoire]]	0.060	0.098	8
statut de séjour : en famille quartier match [[3 ;5) :[Oratoire], [9 ;10) :[Oratoire]]	0.043	0.073	6
statut de séjour : en famille quartier match [[7 ;8) :[Oratoire], [9 ;10) :[Oratoire]]	0.036	0.061	5
statut de séjour : en famille quartier match [[9 ;13) :[Oratoire]]	0.036	0.061	5
statut de séjour : en famille quartier match [[1 ;8) :[Oratoire], [9 ;10) :[Oratoire]]	0.029	0.049	4
statut de séjour : avec des amis quartier match [[7 ;9) :[Fétilly]]	0.049	0.108	4

TABLE 6.4 – Exemple de prédicats montrant les habitudes de sommeil par status.

En conclusion, la possibilité d'analyser des données hétérogènes permet de cibler des analyses spécifiques en intégrant les données qui permettent de mieux les décrire, telle que la météo pour un déplacement à la plage, et le statut des visiteurs pour le lieu de sommeil.

4 Analyse quantitative

Nous menons à travers cette section une analyse quantitative de nos descriptions et stratégies. Cette section va être divisée en trois sous-sections selon les trois types de séquences que nous traitons. Globalement, nous comparons notre approche naïve avec la fouille de séquences classique et avec l'AFC. Ensuite, nous comparons notre approche naïve avec les descriptions et stratégies plus élaborées qui réduisent le nombre de concepts ou motifs sans perte d'information importante dans la plupart des cas.

4.1 Séquences simples

Avec l'approche *Naïve*, nous nous comparons en premier lieu avec un des algorithmes de fouille de séquences fermées CloSpan, pour montrer que nous générons le même nombre de motifs séquentiels fermés. Nous mesurons ensuite la réduction des prédicats et concepts en changeant la stratégie vers la stratégie *Augmentée*. Puis, nous changeons la description SCM par la description SCP et la description KSC pour générer des treillis de taille réduit. Et enfin, nous utilisons les deux mesures de Représentabilité et de Distinctivité nous observons que nous améliorons la qualité des treillis générés.

4.1.1 Fouille de motifs séquentiels fermés avec $\delta_{\mathbf{SCM}}$-$\sigma_{\mathbf{SN,SA}}$

Commençons par la comparaison avec CloSpan [Yan et al., 2003] (cf Chapitre 2). La comparaison se fait principalement entre le nombre de motifs séquentiels extraits par notre approche et par CloSpan. La Table 6.5 indique le nombre de prédicats (P) et de concepts (C) générés avec la description SCM et les deux stratégies *Naïve* et *Augmentée* pour des jeux de données réels et synthétiques, et le nombre de sous-séquences fermées (M) générées par CloSpan [Yan et al., 2003].

En utilisant la description SCM et la stratégie *Naïve* nous générons clairement toutes les sous-séquences fermées puisque la description SCM génère des sous-séquences fermées, et la stratégie *Naïve* considère tous les sous-ensembles possibles des séquences. Nous pouvons en effet observer que le nombre de prédicats est le même que le nombre de motifs séquentiels fermés avec CloSpan. Le nombre de concepts est plus grand que le nombre de prédicats/motifs séquentiels fermées car beaucoup de concepts sont générés à partir de leurs successeurs sans avoir nécessairement des nouveaux prédicats. Nous pouvons également observer que la stratégie *Augmentée* génère moins de motifs séquentiels fermés puisque seules les sous-séquences augmentées sont considérées. Mais cette différence n'est observée qu'avec plus de 100 séquences dans G et plus de 10 éléments dans A ce qui signifie que la stratégie *Augmentée* est proche de la stratégie *Naïve*, mais avec moins de

concepts et une meilleure complexité temporelle. Ceci illustre la possibilité de réduire les motifs séquentiels fermés selon une stratégie spécifique.

		GxA30S7					G100AyS7		G100A30Sz		
		G25	G50	G75	G100	G200	A10	A20	S5	S6	S8
CloSpan	M	**132**	**331**	**541**	**801**	**1647**	**1094**	**880**	**360**	**552**	**1031**
δ_{SCM}-σ_{SN}	P	**133**	**332**	**542**	**802**	**1648**	**1095**	**881**	**361**	**553**	**1032**
	C	171	520	911	1468	3780	4810	2049	540	898	2182
δ_{SCM}-σ_{SA}	P	**132**	**332**	**541**	785	1639	1075	872	**360**	**552**	**1031**
	C	170	520	910	1458	3771	4788	2037	539	897	2181

TABLE 6.5 – Comparaison du nombre de motifs générés entre CloSpan et δ_{SCM}-σ_{SN} et δ_{SCM}-σ_{SA} pour les données synthétiques

4.1.2 Analyse de `Daily-actions` avec $\delta_{\text{KSC,SCM}}$-σ_{SN}

La Figure 6.9 et la Table 6.6 montrent une comparaison entre les descriptions SCM et KSC pour les ensembles de données synthétiques et `Daily-actions`. Il est clair que la description KSC ne génère pas de motifs séquentiels maximaux car nous fixons la taille des sous-séquences qui ne sont pas maximales. Par conséquent, nous obtenons plus de prédicats, mais moins de concepts. Les séquences sont susceptibles de partager plus de sous-séquences de petite longueur, mais lorsque nous augmentons la longueur, les concepts partageant les mêmes sous-séquences deviennent moins nombreux. Nous pouvons également observer que nous obtenons moins de concepts lorsque le paramètre k augmente. La description KSC est clairement adaptée à l'ensemble de données `Daily-actions`, car les petits motifs séquentiels sont plus pertinents pour analyser les actions quotidiennes que les sous-séquences maximales. La possibilité de fixer k permet d'éviter la génération de toutes les sous-séquences possibles.

		σ_{SN}			
		δ_{SCM}	$\delta_{\text{KSC}}(k=2)$	$\delta_{\text{KSC}}(k=3)$	$\delta_{\text{KSC}}(k=4)$
`Daily-actions`	# predicates	272	107	503	1406
	# concepts	373	275	255	197

TABLE 6.6 – Nombre de concepts et prédicats générés par δ_{SCM}-σ_{SN} et δ_{KSC}-σ_{SN} pour `Daily-actions`.

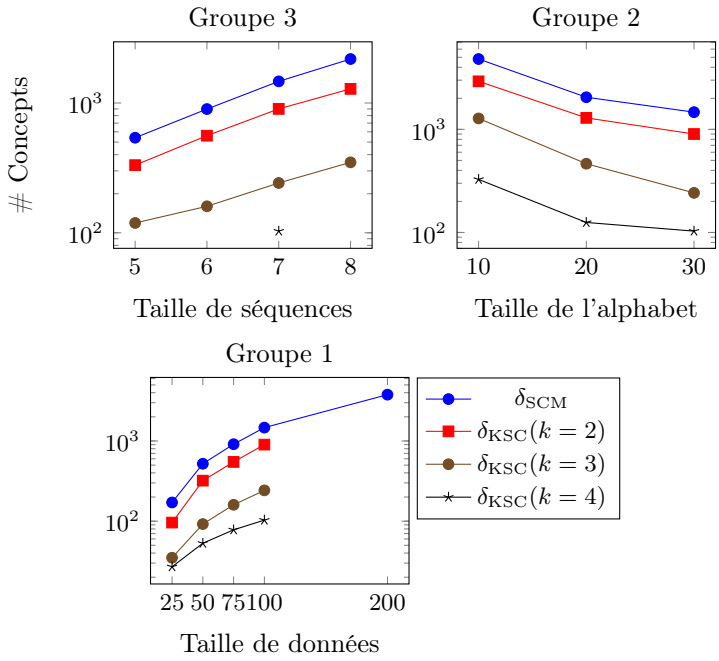


FIGURE 6.9 – Nombre de concepts générés par δ_{SCM}-σ_{SN} et δ_{KSC}-σ_{SN} pour les données synthétiques.

4.1.3 Analyse de `Cité-du-vin` avec $\delta_{\text{SCP,SCM}}$-σ_{SA}

Le Table 6.7 montre la comparaison entre les descriptions SCM et SCP pour le jeu de données `Cité-du-vin`. Pour ce jeu de données particulier, nous avons préféré une description par préfixes puisque nous nous sommes concentrés sur le flux des visiteurs à l'entrée du musée, c'est-à-dire les chemins qui ont été visités en premier. Nous pouvons observer ici qu'avec la description SCP nous obtenons moins de concepts et de prédicats par rapport à la description SCM qui génère un très grand nombre de sous-séquences fermées, et les concepts obtenus correspondent à l'analyse souhaitée des parcours de visite.

Datasets		`Cité-du-vin`			
		25	50	75	100
δ_{SCM}-σ_{SA}	# concepts	576	2172	3580	8233
	# predicats	575	2171	3579	8232
δ_{SCP}-σ_{SA}	# concepts	33	74	114	146
	# predicats	32	75	113	145

TABLE 6.7 – Nombre de concepts et prédicats générés par δ_{SCM}-σ_{SA} et δ_{SCP}-σ_{SA} pour `Cité-du-vin`

4.1.4 Analyse de `Cité-du-vin` avec tous les descriptions et stratégies

La Figure 6.10 représente les mesures de la Représentabilité et la Distinctivité obtenues avec les différentes types d'analyse sur le jeu de données `Cité-du-vin`. Les valeurs de δ_{SCM}-σ_{SN}, δ_{SCM}-σ_{SA} et δ_{KSC}-σ_{SN} sont proches et diminuent avec l'augmentation du nombre de séquences en entrée car le nombre de concepts du treillis augmente rapidement par rapport au nombre de concepts irréductibles, et donc le rapport entre les deux diminue. Avec δ_{SCP}-σ_{SA} nous observons que les mesures restent stables même avec l'augmentation de nombre de séquences. Cela est evident car avec la description SCP, tous les concepts, à part le top et le bottom, sont des inf-irréductibles et la majorité des concepts sont des sup-irréductibles. Donc les deux mesures vont être proche de 1 quelques soit le nombre de concepts. Nous remarquons aussi que la description KSC donne des résultats meilleurs, surtout avec une augmentation du nombre de séquences.

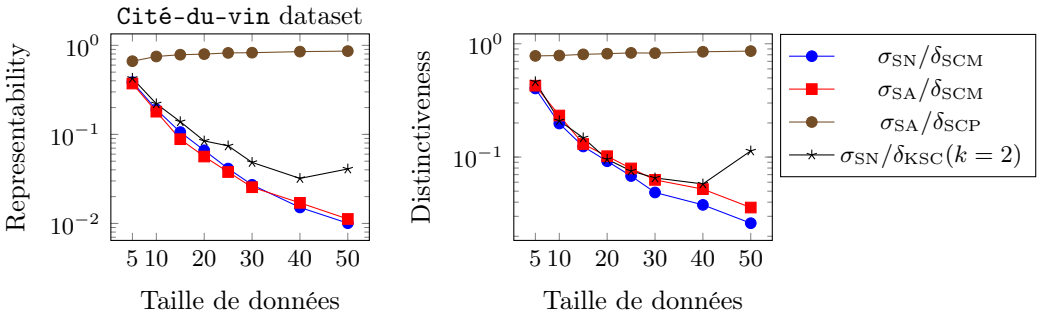


FIGURE 6.10 – Représentabilité et Distinctivité des treillis générés par δ_{SCM}-σ_{SN}, δ_{SCM}-σ_{SA}, δ_{SCP}-σ_{SA} et δ_{KSC}-σ_{SN} pour `Cité-du-vin`.

4.2 Séquences temporelles

Pour les séquences temporelles, trois expérimentations ont été réalisées, la comparaison entre les stratégies en deux étapes, et l'impact de l'utilisation de contraintes. La description SDCM et les stratégies utilisées sont spécifiques et nous n'avons pas trouvé d'études utilisant l'AFC pour analyser des séquences temporelles et explorer les ensembles de données comme nous le faisons. Nous comparons d'abord les deux stratégies SDN et SDM pour voir l'impact de l'utilisation de la stratégie SDM pour réduire le treillis tout en conservant de bonne mesure de qualité. Ensuite, nous faisons varier le paramètre de *pas* de la stratégie SDP pour observer la réduction de la taille du treillis. Enfin, nous utilisons la description SDCM avec les deux contraintes de *fenêtre* et d'*écart* (SDCMw, SDCMg).

4.2.1 Analyse de `Catch-me` et `Cité-du-vin` avec δ_{SDCM}-$\sigma_{\text{SDN,SDM}}$

La Figure 6.11 montre le temps d'exécution, le nombre de concepts et le nombre d'irréductibles générés par la description SDCM et les deux stratégies SDN et SDM avec le jeu de données `Cité-du-vin`.

Dans la Figure 6.11 et la Table 6.8, on peut observer que la stratégie SDM réduit la taille du treillis avec moins d'inf-irréductibles. La stratégie SDN génère plus d'inf-irréductibles car elle calcule tous les concepts possibles. Le nombre de sup-irréductibles est le même avec les deux stratégies, ce qui signifie qu'elles ont la même capacité à distinguer les séquences initiales. Par conséquent, la stratégie SDM est plus rapide que la stratégie SDN, car elle génère moins de concepts. De plus, on peut observer que la stabilité logarithmique globale dans la Table 6.8 varie très peu entre les deux stratégies, même si la stratégie SDM génère moins de concepts. Ce qui signifie que la stratégie SDM réduit la taille du treillis tout en maintenant une bonne stabilité logarithmique globale.

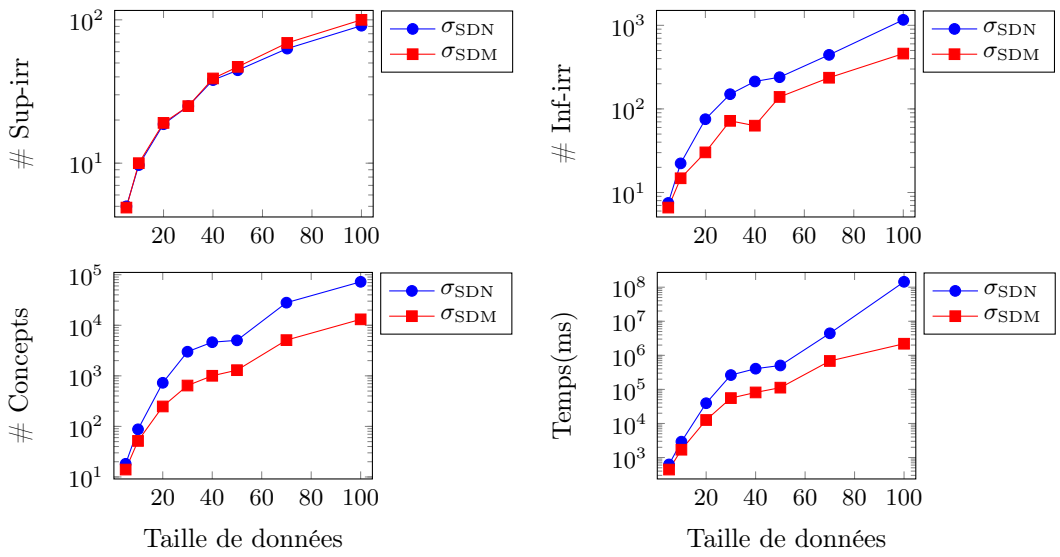


FIGURE 6.11 – Temps d'exécution et nombre de concepts et d'irréductibles générés par δ_{SDCM}-σ_{SDN} et δ_{SDCM}-σ_{SDM} pour `Cité-du-vin`.

Taille de données	stratégie	# concepts	# sup	# inf	temps(ms)	stabilité logarithmique globale
100	σ_{SDN}	172.2	93.8	101.1	5.293.133	0.816
	σ_{SDM}	171	93.9	100.4	5.169.953	0.816
200	σ_{SDN}	368.9	178.6	188.4	26.999.697	0.804
	σ_{SDM}	370.2	180.4	190.6	26.360.502	0.804
300	σ_{SDN}	565.3	261.6	273.6	68.418.978	0.817
	σ_{SDM}	553	263.8	269.9	66.631.920	0.817
400	σ_{SDN}	766.2	339.4	342.8	129.194.954	0.816
	σ_{SDM}	754.5	342.8	340.5	126.650.950	0.816
500	σ_{SDN}	984	416.4	420.8	240.743.394	0.817
	σ_{SDM}	964	421.6	416.5	234.808.014	0.817

TABLE 6.8 – Nombre de concepts, temps d'exécution, sup et inf irréductibles générés par δ_{SDCM}-σ_{SDN} et δ_{SDCM}-σ_{SDM} pour `Catch-me`.

Dans la Figure 6.12, nous pouvons observer que les mesures de Représentabilité et de Distinctivité diminuent lorsque la taille des données augmente car le nombre de concepts du treillis augmente rapidement par rapport au nombre de concepts irréductibles. La Distinctivité est plus élevée avec la stratégie SDM qu'avec la stratégie SDN. Cela signifie que la stratégie SDM maintient un nombre plus élevé d'inf-irréductibles en réduisant la taille du treillis. La représentabilité est légèrement meilleure en utilisant la stratégie SDM pour le jeu de données `Catch-me`. Pour le jeu de données `Cité-du-vin`, comme nous avons augmenté la taille des données, les deux mesures sont plus élevées avec la stratégie SDM par rapport à la stratégie SDN. Ceci est dû au fait que la stratégie SDN génère beaucoup plus de prédicats que la stratégie SDM.

Le nombre de sup-irréductibles correspond au nombre de séquences qui ont des descriptions différentes. On peut observer que l'analyse réussit à produire des descriptions distinctes pour les données d'entrées car le nombre de sup-irréductibles est proche de nombre de séquences.

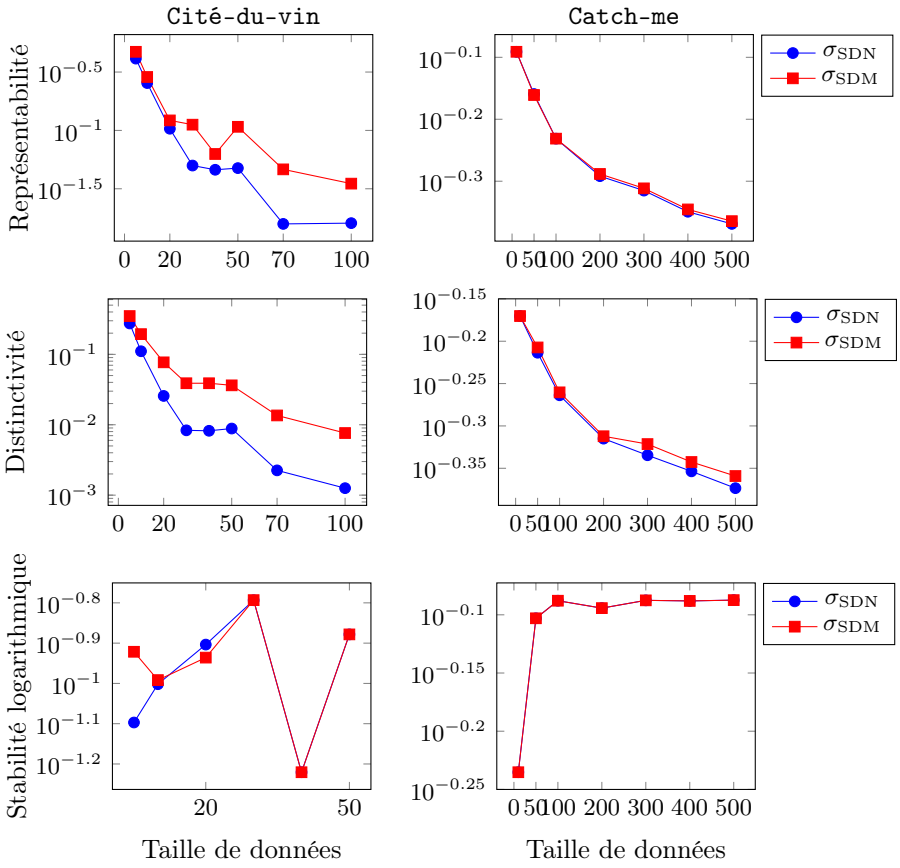


FIGURE 6.12 – Représentabilité, Distinctivité et Stabilité logarithmique globale des treillis généré par δ_{SDCM}-σ_{SDN} et δ_{SDCM}-σ_{SDM} pour `Catch-me` et `Cité-du-vin`.

4.2.2 Analyse de `Catch-me` et `Cité-du-vin` avec δ_{SDCM}-σ_{SDP}

La Figure 6.13 représente le nombre de concepts obtenus avec 3 paramétrages différents de la stratégie SDP : $pas = 1s, 3600s(1h),$ et $21600s(6h)$. Le pas précise la taille de la réduction des distances dans les sous-séquences communes afin de générer des candidats pour l'itération suivante. Par conséquent, plus le pas est élevé, plus le nombre de concepts est faible. À partir de 500 séquences du jeu de données `Catch-me`, nous obtenons 1272 concepts avec un $pas = 1s$, et 861 concepts en utilisant un pas de $1h$ et $6h$. À partir de 70 séquences du jeu de données `Cité-du-vin`, nous obtenons 21916 concepts avec un $pas = 1s$, 1138.5 concepts pour un $pas = 1h$ et seulement 155.75 concepts pour un $pas = 6h$.

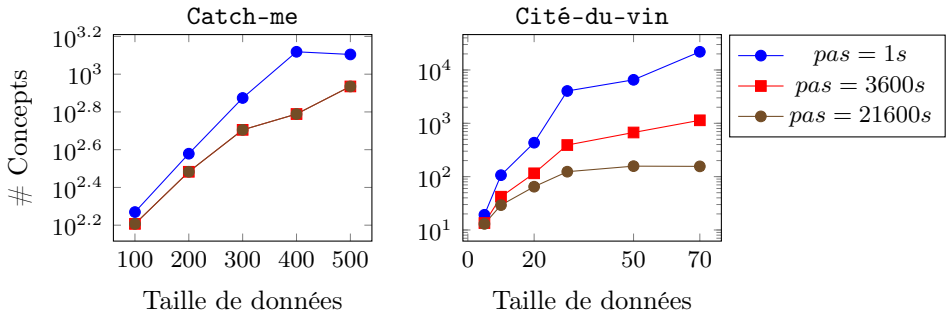


FIGURE 6.13 – Nombre de concepts obtenu avec différents *pas* en utilisant δ_{SDCM}-σ_{SDP} pour `Catch-me` et `Cité-du-vin`.

4.2.3 Analyse de `Cité-du-vin` avec $\delta_{\text{SDCMw,SDCMg}}$-$\sigma_{\text{SDN}}$

Une autre façon de réduire les motifs est de limiter la description par une *fenêtre* ou un *écart*. La Figure 6.14 montre la taille du treillis généré en utilisant les descriptions SDCMw, SDCMg et la stratégie SDN, où la *fenêtre* et l'*écart* varient. Nous utilisons 3 valeurs de *fenêtre* : $w = 3600s(1h), w = 2700s(45mn),$ et $w = 1800s(30mn)$ et 3 valeurs pour les *écart* : $g = 120s(2mn), g = 600s(10mn), g = 1800s(30mn)$. Nous pouvons voir que le nombre de concepts est significativement réduit. Pour un jeu de données de taille 100, le nombre de concepts est réduit de 72427 (sans *fenêtre*) à 495 avec une *fenêtre* de $1800s$. De manière similaire, l'utilisation d'un *écart* permet de réduire la taille du treillis à 295.75 en moyenne avec un *écart* de $120s$ (2 minutes).

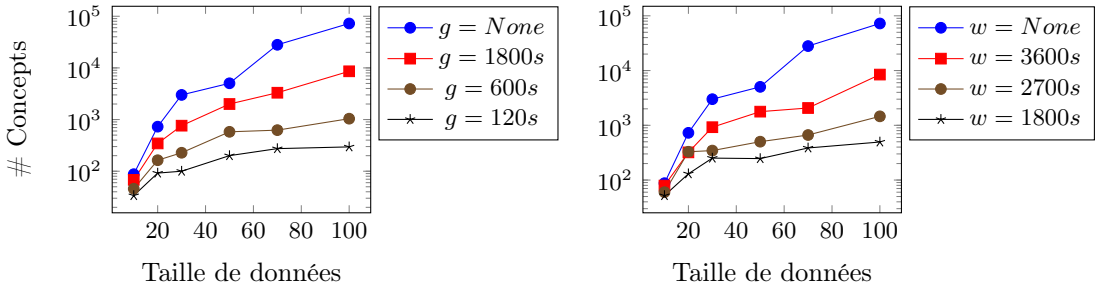


FIGURE 6.14 – Nombre de concepts du treillis généré par δ_{SDCMw}-σ_{SDN} et δ_{SDCMg}-σ_{SDN} pour `Cité-du-vin`.

4.3 Séquences Intervalles

Pour les séquences d'intervalles nous avons proposé deux descriptions et plusieurs stratégies, nos expérimentations vont donc être divisées en deux parties. D'abord nous comparons les deux descriptions avec des strategies naïves, pour identifier les différences entre les deux. Ensuite pour la même description SCMIN, nous comparons les stratégies.

4.3.1 Analyse de `Cité-du-vin` avec δ_{SCMAX}-σ_{CMA} et δ_{SCMIN}-σ_{NCT}

Nous comparons ici nos deux descriptions en termes de temps d'exécution et de taille de treillis. Nous utilisons la description SCMAX avec la stratégie CMA, et la description SCMIN avec la stratégie NCT. Ces deux stratégies génèrent toutes les sous-séquences/superséquences possibles. Les résultats pour le jeu de données `Cité-du-vin` sont donnés dans la Table 6.9. Nous pouvons observer que SCMAX est beaucoup plus rapide que SCMIN. Il génère un treillis de 149 concepts en environ 2 minutes à partir de 500 séquences, alors que la description SCMIN, s'arrête à 1024 concepts générés en environ 30 minutes à partir de seulement 10 séquences. Ces descriptions sont deux façons différentes de représenter les données. La description SCMIN est clairement plus riche que la description SCMAX qui est la description classique des sous-séquences maximales communes étendue aux intervalles, mais SCMIN n'est pas adaptée aux grands ensembles de données.

	taille de données	4	6	10	100	500	1000
δ_{SCMIN}-σ_{NCT}	# concepts	16	33	**1024**			
	temps(ms)	2878	11760	**1927039**			
	$\frac{\text{temps}}{\text{concepts}}$(ms)	179,87	356,36	**1881,87**			
δ_{SCMAX}-σ_{CMA}	# concepts	6	12	13	67	**149**	**132**
	temps(ms)	267	771	838	17198	**141281**	**246437**
	$\frac{\text{temps}}{\text{concepts}}$(ms)	44.5	64.25	64.46	256.68	948.19	1866.94

TABLE 6.9 – Nombre de concepts et de temps d'exécution des treillis générés par δ_{SCMAX}-σ_{CMA} et δ_{SCMIN}-σ_{NCT} pour `Cité-du-vin`.

4.3.2 Analyse de `Cité-du-vin` avec δ_{SCMIN}- $\sigma_{\text{NCT,BCT,FACT,ACT}}$

Nous nous concentrons maintenant sur la description de SCMIN pour comparer les quatre stratégies : NCT, BCT, FACT et ACT. Rappelons que la stratégie NCT génère toutes les super-séquences possibles tandis que les stratégies BCT, FACT et ACT se concentrent sur des super-séquences particulières (préfixe, suffixe, selon une fenêtre). La Figure 6.15 montre le temps d'exécution et le nombre de concepts générés en utilisant le jeu de données `Cité-du-vin`. Par rapport à la stratégie NCT, nous pouvons clairement voir que les autres stratégies sont plus rapides et génèrent moins de concepts. La stratégie FACT est la meilleure dans cet exemple, surtout avec $f = 1$. Avec $f = 2$, le résultat est proche de celui de la stratégie BCT. Nous exécutons la stratégie BCT avec $card_{min}$ et $card_{max}$: nous observons que le temps d'exécution est meilleur, et nous obtenons moins de concepts avec $card_{min}$. La complexité de NEXTPRIORITYCONCEPT dépend de la taille du treillis, donc une réduction à des concepts plus pertinents réduit également le temps d'exécution.

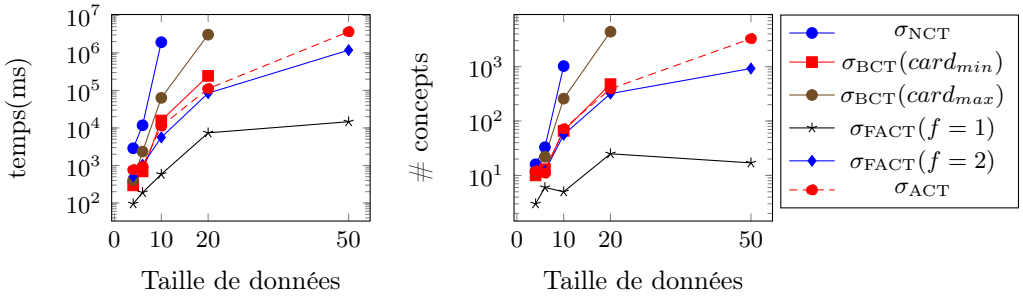


FIGURE 6.15 – Temps d'exécution et nombre de concepts des treillis générés par δ_{SCMIN}-$\sigma_{\text{NCT,BCT,FACT,ACT}}$ pour `Cité-du-vin`.

4.3.3 Analyse de `GéoLuciole` avec δ_{SCMIN}-$\sigma_{\text{NCT,BCT,FACT,ACT}}$

La Table 6.10 présente une comparaison entre les quatre stratégies de description du SCMIN avec le jeu de données `GéoLuciole`. Les stratégies BCT, ACT et FACT sont comparées à la stratégie NCT en termes de taux de compression, c'est-à-dire le rapport entre le nombre de concepts obtenus avec la stratégie NCT et le nombre de concepts obtenus avec chacune des autres stratégies. La Table 6.10 montre l'efficacité de nos stratégies pour réduire le nombre de concepts. Nous pouvons également observer que le taux de compression est amélioré lorsque nous augmentons la taille des données pour toutes les stratégies. La stratégie ACT est plus performante et génère moins de concepts que la stratégie BCT. Le taux de compression est faible avec la stratégie FACT, parce que la taille des séquences est proche de la fenêtre, et donc au fur et à mesure que l'on augmente les fenêtres le nombre de concepts se rapproche de la stratégie NCT. Le comportement de la stratégie FACT est lié à la taille moyenne des séquences. L'analyste de données peut faire varier des paramètres tels que la cardinalité pour BCT, ou la fenêtre pour FACT, afin de ne générer que des concepts pertinents.

Taille de données	4	5	6	7	10	15
	# Concepts					
σ_{NCT}	16	32	64	**128**	**672**	**16640**
	Taux de compression					
$\sigma_{\text{BCT}}(card_{min})$	2.28	**4.57**	4.92	4.74	18.66	96.18
$\sigma_{\text{BCT}}(card_{max})$	**5.33**	2.66	**7.11**	3.45	28	92.96
σ_{ACT}	4	2.13	4	8.53	24.88	130
$\sigma_{\text{FACT}}(f=1)$	1.77	3.2	3.76	**10.66**	**32**	**386.97**
$\sigma_{\text{FACT}}(f=2)$	1.45	2.13	1.42	4.26	7.38	49.52

TABLE 6.10 – Taux de compression et nombre de concepts des treillis générés par δ_{SCMIN}-$\sigma_{\text{NCT,BCT,FACT,ACT}}$ pour `GéoLuciole`.

5 Discussion et conclusion

Dans ce travail, nous avons développé une nouvelle approche d'exploration de séquences utilisant l'algorithme NextPriorityConcept. Cet algorithme permet le calcul d'un motif générique par le biais de *descriptions* et de *stratégies* spécifiques. Nous avons introduit la *Stabilité logarithmique globale*, la *Représentabilité* et la *Distinctivité* pour mesurer la qualité du treillis obtenu.

Trois types de séquences ont été traités, les séquences simples, les séquences temporelles et les séquences d'intervalles. En premier lieu nous avons montré l'efficacité de nos descriptions et stratégies pour les séquences simples, nous avons comparé notre approche avec celle de ClosSpan et nous avons montré que nous générons le même nombre de motifs séquentiels fermées. Ensuite, à travers différentes descriptions et stratégies, nous arrivons à réduire la taille du treillis de concepts qui devient alors plus lisible et pertinent. De plus, les descriptions et stratégies sont plus adaptées aux données à traiter et permettent de cibler l'analyse vers les concepts les plus adéquats.

Nous avons également décrit des analyses pour des séquences temporelles avec la description *Distancielle* qui décrit un ensemble de séquences temporelles par leurs sous-séquences distancielles communes maximales avec des distances entre les éléments. Cette description est étendue avec des contraintes de fenêtre et d'écart. Nous proposons également trois stratégies d'exploration. Les expérimentations ont montré l'efficacité de cette approche et sa flexibilité, avec des paramètres permettant de réduire la taille des treillis, et cibler l'analyse vers les concepts les plus pertinents.

Nous avons présenté deux descriptions et cinq stratégies pour l'analyse de séquences d'intervalles. Les deux descriptions représentent deux approches différentes pour représenter un ensemble de séquences. La première SCMAX est la classique sous-séquence maximale commune, tandis que la seconde SCMIN fournit une description plus riche des séquences d'intervalles. Nous avons présenté une stratégie pour la description SCMAX, et quatre stratégies pour la description SCMIN qui peuvent être testées dans une approche orientée utilisateur afin de générer moins de concepts et plus de données pertinentes. Par conséquent, nous allons nous concentrer sur la réduction de la complexité temporelle de nos extensions, et créer des extensions plus configurables répondant le mieux à la particularité des données que nous voulons traiter.

Ces résultats ont été rendus possibles par l'utilisation de l'algorithme NextPriorityConcept qui permet de se concentrer sur les données (descriptions et stratégies) pour l'exploration et la découverte de motifs. Dans des travaux futurs, nous souhaitons proposer une approche orientée utilisateur qui permettrait à l'analyste de choisir et de tester plusieurs stratégies et descriptions en variant les contraintes d'écart et de fenêtre pour trouver les motifs les plus pertinents. En effet, choisir ou tester plusieurs stratégies à chaque itération est possible puisque les stratégies ne sont pas conservées dans la description finale.

Chapitre 7

Conclusion générale et Perspectives

Sommaire

1 Conclusion 163
2 Contributions 163
3 Perspectives 168

1 Conclusion

À travers ce travail nous avons étendu l'algorithme NextPriorityConcept à des données séquentielles. Notre proposition est basée sur l'analyse formelle de concepts et la fouille de motifs séquentiels. L'algorithme NextPriorityConcept définit les notions de *descriptions* et de *stratégies* pour analyser des données complexes et hétérogènes. Nous avons considéré trois types de séquences : les séquences simples, les séquences temporelles et les séquences d'intervalles. Les descriptions et les stratégies proposées sont adaptées aux données séquentielles selon leurs types et permettent une analyse guidée par l'expert des données. Dans ce chapitre, nous résumons brièvement les principales contributions de nos travaux. Ensuite, nous en présentons une partie importante qui concerne la communication et l'implication dans le processus de promotion de la bibliothèque `GALACTIC`. Enfin, nous présentant quelques perspectives que nous trouvons intéressantes.

2 Contributions

La Table 7.1 présente un résumé de toutes les méthodes que nous avons proposées pour l'analyse de séquences. Les caractéristiques, descriptions et stratégies sont proposées et implémentées sur l'outil `GALACTIC`. Pour chaque type de séquences, des exemples d'analyse ont été effectués et ajouté à la documentation à travers des *notebook jupyter*[1]. Les jeux

1. Notebook jupyter est une application qui permet de créer des programmes contenant du texte en markdown et du code. Lien vers un document généré à partir de nos notebooks : https ://galactic.univ-lr.fr/guides/guide-practice/Galactic-Practice-Guide-latest.pdf

de données et les mesures utilisées pour les expérimentations sont aussi présentés et enfin les publications présentant nos travaux selon le type de séquences.

Characteristics	Descriptions	Strategies	Exemples	Expérimentations	Comparative	Mesures	Publications
String	KSC	SN SA	- Verbes - Formules chimiques - ADN				
	SCM						
	SCP						
	SDCM						
Séquences	KSC		Actions Cité du Vin Geoluciole	Actions Cité du Vin	CloSpan	concepts/ predicats	CLA2020
	SCM						
	SCP						
Séquence temporelles	SDCM	SDN	Cité du Vin Actions Catch-me Géoluciole	Cité du Vin Catch-me		Concepts irreductibles Représentativité Distinctivité Stabilité globale	ICCS2021 Meilleur article d'étudiant
		SDP					
		SDM					
Séquence d'intervalles	SCMIN	NCT	Cité du Vin Géoluciole	Cité du Vin Géoluciole		concepts/ predicats	FCA4AI
		BCT					
		FACT					
		ACT					
	SCMAX	CMA					

TABLE 7.1 – Résumé de tous les descriptions et stratégies proposées avec les expérimentations et les publications.

Descriptions des séquences Nous nous sommes basés sur la notion des sous-séquences communes maximales pour les descriptions qui correspondent à l'analyse classique de séquences. D'autres descriptions plus élaborées ont été aussi introduites dans le but de décrire les données sous différents angles et ainsi donner un sens différent aux concepts générés. Elles donnent à l'expert de données plusieurs possibilités d'analyse et permettent ainsi une analyse basée sur les données. Les descriptions proposées sont adaptées aux types de séquences que nous analysons. La proposition d'une nouvelle description doit prendre en compte le type de séquences en entrée et le type de motifs que l'on souhaite obtenir.

Nous avons proposé trois descriptions pour les séquences simples : la description SCM permettant une analyse classique ; la description SCP décrivant un ensemble de séquences par le préfixe commun et la description KSC permettant de limiter la taille des sous-séquences générées.

Pour les séquences temporelles, nous avons proposé la description SDCM qui correspond à la description SCM avec ajout de l'information temporelle entre les éléments de la sous-séquence. Cette description peut être paramétrée par deux contraintes de *fenêtre* (SDCMw) et d'*écart* (SDCMg) permettant de limiter la génération des sous-séquences à celles respectant ces 2 contraintes.

Nous avons également proposé deux types de descriptions pour les séquences d'intervalles. La première description SCMAX est basée sur la notion de sous-séquence commune maximale qui utilise la sous-séquence d'intervalles commune maximale. La deuxième description SCMIN est basée sur la notion de super-séquence commune minimale qui utilise la super-séquence d'intervalles commune minimale.

Stratégie d'exploration pour les séquences Une stratégie consiste à ajouter des informations élémentaires aux sous-séquences des descriptions, chaque information élémentaire permettant ainsi de sélectionner les sous-groupes de séquences qui la vérifie, et de le proposer comme nouveau sous-groupe candidat.

La stratégie naïve est proposée pour tous les types de séquences ; elle permet de générer tous les sous-groupes possibles, ce qui correspond aux analyses classiques en fouille de séquences et AFC. Cette stratégie combinée avec la description SCM permet de comparer notre approche avec ces algorithmes classiques. Pour les séquences simples, la stratégie naïve avec la description SCM permet d'extraire les mêmes motifs que les algorithmes de fouille de motifs séquentiels fermés comme *CloSpan* ou *BIDE*. D'autres stratégies sont également proposées pour réduire la taille des treillis qui sont souvent de taille exponentielle avec le nombre de séquences. Pour les séquences simples l'information ajoutée consiste en des éléments de l'alphabet, tandis que pour les séquences temporelles et d'intervalles il s'agit d'ajout d'information temporelle.

L'expert de données a ainsi à sa disposition plusieurs combinaisons description/stratégie qu'il peut sélectionner en fonction des séquences manipulées et des motifs qu'il souhaite extraire.

Nous avons proposé deux stratégies pour les séquences simples. La stratégies SN est la stratégie naïve permettant de générer tous les concepts possibles, tandis que la

stratégie SA est une stratégie qui ajoute des éléments de l'alphabet à la fin seulement des sous-séquences de la description permettant ainsi de les étendre, ce qui va générer moins de concepts.

Pour les séquences temporelles, nous avons proposé trois stratégies. Elles sont toutes basées sur l'information temporelle. La stratégie naïve SDN augmente ou diminue le temps entre les éléments avec une unité de temps. La stratégie SDP est définie de manière générique avec un paramètre *pas* de temps à augmenter ou à diminuer entre les éléments. La stratégie SDM est définie avec un *pas* égal à la moitié de temps entre les éléments.

Pour les séquences d'intervalles, cinq stratégies ont été proposées où l'information élémentaire ajoutée est un éléments dans les intervalles.

Mesures de qualité non supervisées Nous avons proposé des mesures de qualité non supervisées dans le but de montrer l'efficacité de notre approche. Elles permettent de mesurer la qualité des treillis générés sous différents angles, et de les comparer. Les mesures sont basées sur la stabilité des concepts générés et les éléments irréductibles qui en codent la structure du treillis. Nous avons introduit trois mesures de qualité d'un treillis de concepts. Les mesures proposées sont : la *Stabilité Logarithmic Normalisé Globale* qui donne une valeur globale de la stabilité du treillis, la *Distinctivité* qui est une mesure de séparation indiquant à quel point nous distinguons les objets, et la *Représentabilité* qui mesure l'information représentée par les prédicats générés.

Cette thèse ayant débuter en même temps que la première version de la plateforme `GALACTIC`, nous avons également beaucoup participé à sa promotion avec des supports de communication et des interventions dans des évènements.

Nous avons construit la première version de la fleur qui illustre l'architecture de la bibliothèque GALACTIC. Nous avons également contribué à la création des guides [2] et de la documentation [3] de l'outil GALACTIC dont le guide pratique, le guide utilisateur et le guide des expérimentations.

Nous avons participé à trois éditions de la fête de la science qui est un événement national de vulgarisation scientifique proposant des rencontres, des animations et des présentations à destination du grand public et des scolaires. Notre participation a consisté à animer un atelier pour des enfants et pour le grand public pour expliquer la fouille de données et présenter nos travaux d'analyse de séquences.

Nous avons participé à trois éditions de l'atelier *AI4industry* qui est un événement dédié à la formation de l'intelligence artificielle en Nouvelle-Aquitaine. Cet atelier propose une sensibilisation des participants aux différents éléments permettant le déploiement opérationnel et efficient de l'IA dans l'industrie au travers des conférences et des *use-cases* (cas d'utilisations) industriels. Dans cet atelier nous avons animé des ateliers en utilisant l'outil `GALACTIC` pour analyser des données séquentielles issues des entreprises ; c'était l'occasion également de tester les extensions des séquences avec des données réelles.

2. https ://galactic.univ-lr.fr/guides
3. https ://galactic.univ-lr.fr/docs

3 Perspectives

Les avancées que nous avons pu réaliser pour l'analyse de séquences nous permettent d'envisager plusieurs perspectives de recherches et d'applications. Dans cette section, nous présenterons quelques perspectives intéressantes qui pourront être explorées par la suite.

Analyse interactive pilotée par l'utilisateur Les stratégies ne sont utilisées que pour réduire un concept afin de générer ses prédécesseurs, les sélecteurs qu'elles définissent ne sont pas conservés dans les descriptions des concepts, ni dans l'ensemble final des prédicats décrivant les données. Par conséquent, il est envisageable de choisir ou tester plusieurs stratégies pour chaque concept, dans une approche de découverte de motifs pilotée par l'utilisateur. En effet, rien ne sert de fouiller dans des données si l'on ne sait pas quel type d'information ou de connaissance on y cherche. Il est ainsi possible de choisir des stratégies à gros grains pour les données initiales, et plus "fines" pour des sous-groupes réduits. Dans un contexte de données hétérogènes, on peut également choisir de n'utiliser ou activer les stratégies de certaines données que pour des sous-groupes spécifiques, comme par exemple une information de marée ou de météo pour un sous-groupes d'individus dont la séquence de déplacement commune est sur une plage. Un mécanisme d'interaction avec l'analyste de données pour qu'il puisse adapter sa stratégie d'exploration des données à chaque étape d'analyse est une perspective qui nous semble des plus prometteuse. Des travaux en ce sens sont amorcés dans notre équipe, avec la mise en place d'une interface de saisie interactive des stratégies, et le développement d'un algorithme incrémental sur la base de l'algorithme NextPriorityConcept.

Enrichissement des extensions pour les séquences Bien que nous ayons introduit plusieurs descriptions et stratégies, il est toujours possible d'en envisager de nouvelles plus adaptées à d'autres cas d'application : par exemple, la stratégie SDM peut être étendue en introduisant d'autres manières de réduire la distance de temps entre les éléments de la séquence. Nous pouvons par exemple imaginer une stratégie *quantile* pour les séquences temporelles qui soit paramétrée par l'utilisateur. La notion de super-séquence commune minimale est exploitée seulement avec les séquences d'intervalles, alors que cette description pourrait également être envisagée pour les séquences simples et les séquences temporelles.

Analyse par niveaux Une autre piste d'amélioration serait de considérer des objets décrits non pas par une seul séquence, mais par un groupe de séquences qui correspondait à plusieurs "axes" ou "aspects". Par exemple, une analyse de plusieurs trajectoirs décrites par des séquences chacune correspondant à un enrichissement sémantique particulier [Cayèré et al., 2021].

Nous avons commencé à exploiter cette piste avec le jeu de données hétérogènes `GéoLuciole` construit à partir de plusieurs axes de séquences sémantiques. L'analyse effectuée a permis le traitement séparé de ces séquences avec nos descriptions et stratégies.

Les descriptions correspondent à l'union des prédicats de description pour chaque séquence, qui sont considérées comme des données hétérogènes

Une perspective intéressante serait de considérer plusieurs séquences comme un groupe de caractéristiques de même type, et de définir des descriptions qui les décriraient de façon commune, et des stratégies qui par exemple ajouterait la même information élémentaire à chaque séquence.

Amélioration de temps de calcul L'utilisation de séquences nous a mis devant le problème de la complexité temporelle et en espace de leur manipulation. Nous avons fait le choix de nous focaliser sur la génération de motifs avec des algorithmes pas toujours optimisés. Les descriptions et les stratégies spécifiques dirigent la génération de concepts vers les plus pertinents au sens de l'analyste des données. Nous sommes persuadés que le temps de calcul de nos descriptions et stratégies pourrait être optimisé.

Publications issues de ces travaux de recherche

2020 Boukhetta Salah Eddine, Christophe Demko, Karell Bertet, Jérémy Richard, and Cécile Cayèré.
Sequence Mining Using FCA and the NextPriorityConcept *Algorithm.*
International Conference on Concept Lattice and Their Applications (CLA)

2020 Boukhetta Salah Eddine, Jérémy Richard, Christophe Demko, et Karell Bertet.
Interval-Based Sequence Mining Using FCA and the NextPriorityConcept *Algorithm.*
Workshop What can FCA do for Artificial Intelligence ? (FCA4AI)
co-located with European Conference on Artificial Intelligence (ECAI)

2021 Boukhetta Salah Eddine, Christophe Demko, Jérémy Richard, et Karell Bertet.
Temporal Sequence Mining Using FCA and `GALACTIC`.
International Conference on Conceptual Structures (ICCS).
Best Student Paper

2022 Boukhetta Salah Eddine, Christophe Demko, Karell Bertet, Jérémy Richard, Cécile Cayèré.
Fouille de séquences temporelles avec l'AFC et l'outil GALACTIC.
Atelier Gestion et Analyse de données Spatiales et Temporelles (GAST)
co-localisé avec la conférence francophone sur l'Extraction et la Gestion des Connaissances (EGC)

2022 Bertet Karell, Christophe Demko, Boukhetta Salah Eddine, Jérémy Richard, Cyril Faucher.
Analysis of Complex and Heterogeneous Data using FCA and Monadic Predicates.
Chapitre de livre : Complex Data Analytics with Formal Concept Analysis.
Éditeurs : Rokia Missaoui, Léonard Kwuida, Talel Abdessalem. et Springer

2022 Richard Jérémy, Guillaume Savarit, Boukhetta Salah Eddine, Cyril Faucher, Karell Bertet and Christophe Demko.
Discover spatio-temporal cluster from trajectory data enriched by heterogeneous contextual knowledge using FCA and the NextPriorityConcept Algorithm.
International Conference on Concept Lattice and Their Applications (CLA).

2022 Christophe Demko, Boukhetta Salah Eddine, Jérémy Richard, Guillaume Savarit, Karell Bertet, Cyril Faucher, and Damien Mondou
`GALACTIC` *: towards a generic and scalable platform for complex and heterogeneous data using Formal Concept Analysis*
Workshop Existing Tools and Applications for Formal Concept Analysis (ETAFCA).

Bibliographie

[Aaker et al., 1986] Aaker, D. A., Stayman, D. M., and Hagerty, M. R. (1986). Warmth in advertising : Measurement, impact, and sequence effects. *Journal of Consumer Research*, 12(4) :365–381.

Voir page 12.

[Abbott, 1995] Abbott, A. (1995). Sequence analysis : new methods for old ideas. *Annual review of sociology*, 21(1) :93–113.

Voir page 12.

[Achar et al., 2013] Achar, A., Ibrahim, A., and Sastry, P. (2013). Pattern-growth based frequent serial episode discovery. *Data & Knowledge Engineering*, 87 :91–108.

Voir page 49.

[Achar et al., 2012] Achar, A., Laxman, S., Viswanathan, R., and Sastry, P. (2012). Discovering injective episodes with general partial orders. *Data Mining and Knowledge Discovery*, 25(1) :67–108.

Voir page 49.

[Agrawal et al., 1993] Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216.

Voir page 31.

[Agrawal and Srikant, 1995] Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. In *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, pages 3–14. IEEE.

Voir pages 47 et 52.

[Allen, 1981] Allen, J. F. (1981). An interval-based representation of temporal knowledge. In *IJCAI*, volume 81, pages 221–226.

Voir pages 51 et 117.

[Álvarez et al., 2011] Álvarez, M. R., Félix, P., and Cariñena, P. (2011). Mining temporal constraint networks by seed knowledge extension. In *Conference on Artificial Intelligence in Medicine in Europe*, pages 250–254. Springer.

Voir page 50.

[Álvarez et al., 2013] Álvarez, M. R., Félix, P., and Cariñena, P. (2013). Discovering metric temporal constraint networks on temporal databases. *Artificial intelligence in medicine*, 58(3) :139–154.

Voir pages 50 et 51.

[Álvarez et al., 2010] Álvarez, M. R., Félix, P., Carinena, P., and Otero, A. (2010). A data mining algorithm for inducing temporal constraint networks. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 300–309. Springer.

Voir page 50.

[Ao et al., 2015] Ao, X., Luo, P., Li, C., Zhuang, F., and He, Q. (2015). Online frequent episode mining. In *2015 IEEE 31st International Conference on Data Engineering*, pages 891–902. IEEE.

Voir page 49.

[Ao et al., 2017] Ao, X., Luo, P., Wang, J., Zhuang, F., and He, Q. (2017). Mining precise-positioning episode rules from event sequences. *IEEE Transactions on Knowledge and Data Engineering*, 30(3) :530–543.

Voir page 49.

[Ayres et al., 2002] Ayres, J., Flannick, J., Gehrke, J., and Yiu, T. (2002). Sequential pattern mining using a bitmap representation. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 429–435. ACM.

Voir pages 36, 40 et 43.

[Barbut and Monjardet, 1970] Barbut, M. and Monjardet, B. (1970). *Ordres et classifications : Algèbre et combinatoire*. Hachette, Paris. 2 tomes.

Voir pages 13, 58 et 62.

[Bergroth et al., 2000] Bergroth, L., Hakonen, H., and Raita, T. (2000). A survey of longest common subsequence algorithms. In *Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000*, pages 39–48. IEEE.

Voir page 12.

[Berry et al., 2014] Berry, A., Gutierrez, A., Huchard, M., Napoli, A., and Sigayret, A. (2014). Hermes : a simple and efficient algorithm for building the aoc-poset of a binary relation. *Annals of Mathematics and Artificial Intelligence*, 72(1) :45–71.

Voir page 62.

[Bertet et al., 2018] Bertet, K., Demko, Ch., Viaud, J.-F., and Guérin, C. (2018). Lattices, closure systems and implication bases : A survey of structural aspects and algorithms. *Theoretical Computer Science*, 743 :93–109.

Voir page 59.

[Birkhoff, 1940] Birkhoff, G. (1940). *Lattice theory*, volume 25. American Mathematical Soc.

Voir pages 13 et 58.

[Bonchi et al., 2006] Bonchi, F., Giannotti, F., Lucchese, C., Orlando, S., Perego, R., and Trasarti, R. (2006). Conquest : a constraint-based querying system for exploratory pattern discovery. In *22nd International Conference on Data Engineering (ICDE'06)*, pages 159–159.

Voir pages 43 et 47.

[Bordat, 1986] Bordat, J.-P. (1986). Calcul pratique du treillis de Galois d'une correspondance. *Mathématiques et Sciences humaines*, 96 :31–47.

Voir pages 64, 65 et 73.

[Borgelt, 2012] Borgelt, C. (2012). Frequent item set mining. *Wiley interdisciplinary reviews : data mining and knowledge discovery*, 2(6) :437–456.

Voir page 33.

[Buzmakov et al., 2013] Buzmakov, A., Egho, E., Jay, N., Kuznetsov, S. O., Napoli, A., and Raïssi, C. (2013). On projections of sequential pattern structures (with an application on care trajectories).

Voir page 70.

[Buzmakov et al., 2014] Buzmakov, A., Kuznetsov, S. O., and Napoli, A. (2014). Scalable estimates of concept stability. In *International Conference on Formal Concept Analysis*, pages 157–172.

Voir page 130.

[Carpineto and Romano, 1993] Carpineto, C. and Romano, G. (1993). Galois : An order-theoretic approach to conceptual clustering. In *Proceedings of ICML*, volume 93, pages 33–40.

Voir page 64.

[Casas-Garriga, 2003] Casas-Garriga, G. (2003). Discovering unbounded episodes in sequential data. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 83–94. Springer.

Voir page 49.

[Casas-Garriga, 2005] Casas-Garriga, G. (2005). Summarizing sequential data with closed partial orders. In *Proceedings of the SIAM International Conference on Data Mining*, pages 380–391. SIAM.

Voir page 69.

[Cayèré et al., 2021] Cayèré, C., Sallaberry, C., Faucher, C., Bessagnet, M.-N., Roose, P., Masson, M., and Richard, J. (2021). Multi-level and multiple aspect semantic trajectory model : Application to the tourism domain. *ISPRS International Journal of Geo-Information*, 10(9) :592.

Voir pages 143 et 168.

[Chein, 1969] Chein, M. (1969). Algorithme de recherche des sous-matrices premières d'une matrice. *Bulletin mathématique de la Société des Sciences Mathématiques de la République Socialiste de Roumanie*, pages 21–25.

Voir page 64.

[Chen et al., 2010] Chen, Y.-C., Jiang, J.-C., Peng, W.-C., and Lee, S.-Y. (2010). An efficient algorithm for mining time interval-based patterns in large database. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 49–58.

Voir page 52.

[Codocedo et al., 2017] Codocedo, V., Bosc, G., Kaytoue, M., Boulicaut, J.-F., and Napoli, A. (2017). A proposition for sequence mining using pattern structures. In *International Conference on Formal Concept Analysis*, pages 106–121. Springer.

Voir page 70.

[Cram et al., 2012] Cram, D., Mathern, B., and Mille, A. (2012). A complete chronicle discovery approach : application to activity analysis. *Expert Systems*, 29(4) :321–346.

Voir page 50.

[Dauxais et al., 2017] Dauxais, Y., Guyet, T., Gross-Amblard, D., and Happe, A. (2017). Discriminant chronicles mining. In *Conference on Artificial Intelligence in Medicine in Europe*, pages 234–244. Springer.

Voir pages 50 et 51.

[Dechter et al., 1991] Dechter, R., Meiri, I., and Pearl, J. (1991). Temporal constraint networks. *Artificial intelligence*, 49(1-3) :61–95.

Voir page 50.

[Demko et al., 2020] Demko, Ch., Bertet, K., Faucher, C., Viaud, J.-F., and Kuznetsov, S. O. (2020). NextPriorityConcept : A new and generic algorithm computing concepts from complex and heterogeneous data. *Theoretical Computer Science*, 845 :1 – 20.

Voir pages 14, 73 et 78.

[Dicky et al., 1994] Dicky, H., Dony, C., Huchard, M., and Libourel, T. (1994). Un algorithme d'insertion avec restructuration dans les hiérarchies de classes. In *Proceedings of the Langages et Modèles à Objets*.

Voir page 62.

[Dousson and Duong, 1999] Dousson, C. and Duong, T. V. (1999). Discovering chronicles with numerical time constraints from alarm logs for monitoring dynamic systems. In *IJCAI*, volume 99, pages 620–626. Citeseer.

Voir pages 47 et 50.

[Dousson et al., 1993] Dousson, C., Gaborit, P., and Ghallab, M. (1993). Situation recognition : Representation and algorithms. In *IJCAI : International Joint Conference on Artificial Intelligence*, volume 93, pages 166–172.

Voir page 49.

[Durbin et al., 1998] Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. (1998). *Biological sequence analysis : probabilistic models of proteins and nucleic acids*. Cambridge Univ. press.

Voir page 12.

[Fabrègue et al., 2015] Fabrègue, M., Braud, A., Bringay, S., Le Ber, F., and Teisseire, M. (2015). Mining closed partially ordered patterns, a new optimized algorithm. *Knowledge-Based Systems*, 79 :68–79.

Voir page 70.

[Ferré, 2014] Ferré, S. (2014). *Reconciling Expressivity and Usability in Information Access - From Filesystems to the Semantic Web*. Habilitation, Univ. of Rennes 1, France.

Voir page 13.

[Ferré, 2014] Ferré, S. (2014). Reconciling expressivity and usability in information access from file systems to the semantic web.

Voir page 13.

[Ferré, 2015] Ferré, S. (2015). A proposal for extending formal concept analysis to knowledge graphs. In *International Conference on Formal Concept Analysis*, pages 271–286. Springer.

Voir page 68.

[Ferré and Ridoux, 2000] Ferré, S. and Ridoux, O. (2000). A logical generalization of formal concept analysis. volume 1867, pages 371–384.

Voir page 68.

[Ferré, 2002] Ferré, S. (2002). *Systèmes d'information logiques : un paradigme logico-contextuel pour interroger, naviguer et apprendre.* Doctorat, Univ. of Rennes 1, France.

Voir page 13.

[Fournier-Viger et al., 2014] Fournier-Viger, P., Gomariz, A., Campos, M., and Thomas, R. (2014). Fast vertical mining of sequential patterns using co-occurrence information. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 40–52. Springer.

Voir page 36.

[Ganter, 1984] Ganter, B. (1984). Two basic algorithms for formal concept analysis. Technical report, Technical Report 831, Technische Hochschule, Darmstadt.

Voir page 64.

[Ganter and Kuznetsov, 2001] Ganter, B. and Kuznetsov, S. O. (2001). Pattern structures and their projections. In *LNCS of International Conference on Conceptual Structures (ICCS'01)*, pages 129–142.

Voir pages 13, 68, 69 et 70.

[Ganter and Wille, 1989] Ganter, B. and Wille, R. (1989). Conceptual scaling. In *Applications of combinatorics and graph theory to the biological and social sciences*, pages 139–167. Springer.

Voir page 67.

[Ganter and Wille, 1999] Ganter, B. and Wille, R. (1999). *Formal Concept Analysis, Mathematical foundations.* Springer Verlag, Berlin.

Voir pages 13 et 62.

[Giannotti et al., 2006] Giannotti, F., Nanni, M., Pedreschi, D., and Pinelli, F. (2006). Mining sequences with temporal annotations. In *Proceedings of the ACM symposium on Applied computing*, pages 593–597.

Voir page 51.

[Gizdatullin et al., 2017] Gizdatullin, D., Ignatov, D., Mitrofanova, E., and Muratova, A. (2017). Classification of demographic sequences based on pattern structures and emerging patterns. In *Supplementary Proceedings of 14th ICFCA*, pages 49–66.

Voir page 70.

[Godin et al., 1998] Godin, R., Mili, H., Mineau, G. W., Missaoui, R., Arfi, A., and Chau, T.-T. (1998). Design of class hierarchies based on concept,(galois) lattices. *Theory and Practice of Object Systems*, 4(2) :117–134.

Voir page 62.

[Godin et al., 1991] Godin, R., Missaoui, R., and Alaoui, H. (1991). Learning algorithms using a galois lattice structure. In *1991 Third International Conference on Tools for Artificial Intelligence*, pages 22–23. IEEE Computer Society.

Voir page 64.

[Gomariz et al., 2013] Gomariz, A., Campos, M., Marin, R., and Goethals, B. (2013). Clasp : An efficient algorithm for mining frequent closed sequences. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 50–61. Springer.

Voir page 43.

[Guyet and Quiniou, 2008] Guyet, T. and Quiniou, R. (2008). Mining temporal patterns with quantitative intervals. In *IEEE International Conference on Data Mining Workshops*, pages 218–227.

Voir page 52.

[Guyet and Quiniou, 2011] Guyet, T. and Quiniou, R. (2011). Extracting temporal patterns from interval-based sequences. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Voir pages 51 et 52.

[Han et al., 2000] Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., and Hsu, M.-C. (2000). Freespan : frequent pattern-projected sequential pattern mining. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 355–359. ACM.

Voir page 36.

[Hirate and Yamana, 2006] Hirate, Y. and Yamana, H. (2006). Generalized sequential pattern mining with item intervals. *JCP : Journal of Computers*, 1(3) :51–60.

Voir page 51.

[Höppner and Klawonn, 2002] Höppner, F. and Klawonn, F. (2002). Finding informative rules in interval sequences. *Intelligent Data Analysis*, 6(3) :237–255.

Voir pages 51 et 52.

[Jay et al., 2008] Jay, N., Kohler, F., and Napoli, A. (2008). Analysis of social communities with ICEBERG and stability-based concept lattices. In *ICFCA*, pages 258–272. Springer.

Voir pages 129 et 130.

[Kahn et al., 2016] Kahn, G., Loiseau, Y., and Raynaud, O. (2016). A tool for classification of sequential data. In *European Conference on Artificial Intelligence (FCA4AI)*.

Voir page 144.

[Kam and Fu, 2000] Kam, P.-s. and Fu, A. W.-C. (2000). Discovering temporal patterns for interval-based events. In *International Conference on Data Warehousing and Knowledge discovery*, pages 317–326. Springer.

Voir pages 51 et 52.

[Kaytoue, 2020] Kaytoue, M. (2020). *Contributions to Pattern Discovery.* Habilitation, Univ. of Lyon, France.

Voir page 14.

[Kaytoue et al., 2015] Kaytoue, M., Codocedo, V., Buzmakov, A., Baixeries, J., Kuznetsov, S. O., and Napoli, A. (2015). Pattern structures and concept lattices for data mining and knowledge processing. In *ECML-PKDD : European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases.*

Voir pages 13, 68 et 70.

[Kruskal, 1983] Kruskal, J. B. (1983). An overview of sequence comparison : Time warps, string edits, and macromolecules. *SIAM review*, 25(2) :201–237.

Voir page 12.

[Kryszkiewicz and Skonieczny, 2019] Kryszkiewicz, M. and Skonieczny, Ł. (2019). Fast discovery of generalized sequential patterns. In *Intelligent Methods and Big Data in Industrial Applications*, pages 155–170. Springer.

Voir page 48.

[Kuznetsov, 2007] Kuznetsov, S. O. (2007). On stability of a formal concept. *Annals of Mathematics and Artificial Intelligence*, 49(1) :101–115.

Voir pages 129 et 130.

[Laxman et al., 2007] Laxman, S., Sastry, P., and Unnikrishnan, K. (2007). A fast algorithm for finding frequent episodes in event streams. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 410–419.

Voir page 49.

[Lee and De Raedt, 2004] Lee, S. D. and De Raedt, L. (2004). Constraint based mining of first order sequences in seqlog. In *Database Support for Data Mining Applications*, pages 154–173. Springer.

Voir page 48.

[Lin and Lee, 2005] Lin, M.-Y. and Lee, S.-Y. (2005). Efficient mining of sequential patterns with time constraints by delimited pattern growth. *Knowledge and Information Systems*, 7(4) :499–514.

Voir page 48.

[Luo and Chung, 2008] Luo, C. and Chung, S. M. (2008). A scalable algorithm for mining maximal frequent sequences using a sample. *Knowledge and Information Systems*, 15(2) :149–179.

Voir page 39.

[Maier, 1978] Maier, D. (1978). The complexity of some problems on subsequences and supersequences. *Journal of the ACM (JACM)*, 25(2) :322–336.

Voir page 12.

[Mannila et al., 1997] Mannila, H., Toivonen, H., and Verkamo, A. I. (1997). Discovery of frequent episodes in event sequences. *Data mining and knowledge discovery*, 1(3) :259–289.

Voir pages 47 et 49.

[Mannila et al., 1995] Mannila, H., Toivonen, H., and Verkamo, I. (1995). Discovering frequent episodes in sequences, in"1st intl. conf. *Knowledge Discovery and Data Mining*.

Voir pages 48 et 49.

[Masseglia et al., 1998] Masseglia, F., Cathala, F., and Poncelet, P. (1998). The psp approach for mining sequential patterns. In *European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 176–184. Springer.

Voir pages 36 et 39.

[Masseglia et al., 2009] Masseglia, F., Poncelet, P., and Teisseire, M. (2009). Efficient mining of sequential patterns with time constraints : Reducing the combinations. *Expert Systems with Applications*, 36(2) :2677–2690.

Voir pages 47 et 48.

[Méger and Rigotti, 2004] Méger, N. and Rigotti, C. (2004). Constraint-based mining of episode rules and optimal window sizes. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 313–324. Springer.

Voir page 49.

[Mineau et al., 1990] Mineau, G., Gecsei, J., and Godin, R. (1990). Structuring knowledge bases using automatic learning. In *[1990] Proceedings. Sixth International Conference on Data Engineering*, pages 274–280. IEEE.

Voir page 62.

[Minkiewicz et al., 2015] Minkiewicz, P., Darewicz, M., Iwaniak, A., Sokołowska, J., Starowicz, P., Bucholska, J., and Hrynkiewicz, M. (2015). Common amino acid subsequences in a universal proteome—relevance for food science. *International journal of molecular sciences*, 16(9) :20748–20773.

Voir page 12.

[Morin and Debar, 2003] Morin, B. and Debar, H. (2003). Correlation of intrusion symptoms : an application of chronicles. In *International Workshop on Recent Advances in Intrusion Detection*, pages 94–112. Springer.

Voir page 50.

[Moskovitch and Shahar, 2015] Moskovitch, R. and Shahar, Y. (2015). Fast time intervals mining using the transitivity of temporal relations. *Knowledge and Information Systems*, 42(1) :21–48.

Voir pages 51 et 53.

[Munková et al., 2013] Munková, D., Munk, M., and Vozár, M. (2013). Data preprocessing evaluation for text mining : transaction/sequence model. *Procedia Computer Science*, 18 :1198–1207.

Voir page 12.

[Nakagaito et al., 2009] Nakagaito, F., Ozaki, T., and Ohkawa, T. (2009). Discovery of quantitative sequential patterns from event sequences. In *2009 IEEE International Conference on Data Mining Workshops*, pages 31–36. IEEE.

Voir page 52.

[Nica et al., 2020] Nica, C., Braud, A., and Le Ber, F. (2020). Rca-seq : An original approach for enhancing the analysis of sequential data based on hierarchies of multilevel closed partially-ordered patterns. *Discrete Applied Mathematics*, 273 :232–251.

Voir page 70.

[Norris, 1978] Norris, E. M. (1978). An algorithm for computing the maximal rectangles in a binary relation. *Revue Roumaine de Mathématiques Pures et Appliquées*, 23(2) :243–250.

Voir page 64.

[Nourine and Raynaud, 1999] Nourine, L. and Raynaud, O. (1999). A fast algorithm for building lattices. *Information processing letters*, 71(5-6) :199–204.

Voir page 64.

[Orlando et al., 2004] Orlando, S., Perego, R., and Silvestri, C. (2004). A new algorithm for gap constrained sequence mining. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 540–547.

Voir page 48.

[Osswald and Petersen, 2003] Osswald, R. and Petersen, W. (2003). A logical approach to data-driven classification. In *Annual Conference on Artificial Intelligence*, pages 267–281. Springer.

Voir page 62.

[Papapetrou et al., 2009] Papapetrou, P., Kollios, G., Sclaroff, S., and Gunopulos, D. (2009). Mining frequent arrangements of temporal intervals. *Knowledge and Information Systems*, 21(2) :133–171.

Voir page 53.

[Parthasarathy et al., 1999] Parthasarathy, S., Zaki, M. J., Ogihara, M., and Dwarkadas, S. (1999). Incremental and interactive sequence mining. In *Proceedings of the eighth international conference on Information and knowledge management*, pages 251–258.

Voir page 47.

[Pasquier et al., 1999] Pasquier, N., Bastide, Y., Taouil, R., and Lakhal, L. (1999). Discovering frequent closed itemsets for association rules. In *International Conference on Database Theory*, pages 398–416. Springer.

Voir pages 13 et 34.

[Patel et al., 2008] Patel, D., Hsu, W., and Lee, M. L. (2008). Mining relationships among interval-based events for classification. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 393–404.

Voir page 51.

[Pei et al., 2000] Pei, J., Han, J., Mao, R., et al. (2000). Closet : An efficient algorithm for mining frequent closed itemsets. In *ACM SIGMOD workshop on research issues in data mining and knowledge discovery*, volume 4, pages 21–30.

Voir page 34.

[Pei et al., 2001] Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., and Hsu, M.-C. (2001). Prefixspan : Mining sequential patterns efficiently by prefix-projected pattern growth. In *icccn*, page 0215. IEEE.

Voir pages 36, 40 et 43.

[Pei et al., 2002] Pei, J., Han, J., and Wang, W. (2002). Mining sequential patterns with constraints in large databases. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 18–25.

Voir page 48.

[Pei et al., 2007] Pei, J., Han, J., and Wang, W. (2007). Constraint-based sequential pattern mining : the pattern-growth methods. *Journal of Intelligent Information Systems*, 28(2) :133–160.

Voir page 47.

[Pei et al., 2006] Pei, J., Wang, H., Liu, J., Wang, K., Wang, J., and Yu, P. S. (2006). Discovering frequent closed partial orders from strings. *IEEE Transactions on Knowledge and Data Engineering*, 18(11) :1467–1481.

Voir page 70.

[Pencolé and Subias, 2009] Pencolé, Y. and Subias, A. (2009). A chronicle-based diagnosability approach for discrete timed-event systems : Application to web-services. *J. Univers. Comput. Sci.*, 15(17) :3246–3272.

Voir page 50.

[Petersen, 2001] Petersen, W. (2001). A set-theoretical approach for the induction of inheritance hierarchies. *Electronic Notes in Theoretical Computer Science*, 53 :296–308.

Voir page 62.

[Quine, 1952] Quine, W. V. O. (1952). The problem of simplifying truth functions. *The American Mathematical*, 59(8) :521–531.

Voir page 84.

[Roth et al., 2008] Roth, C., Obiedkov, S., and Kourie, D. G. (2008). On succinct representation of knowledge community taxonomies with formal concept analysis. *International Journal of Foundations of Computer Science*, 19(02) :383–404.

Voir page 130.

[Rouane-Hacene et al., 2013] Rouane-Hacene, M., Huchard, M., Napoli, A., and Valtchev, P. (2013). Relational concept analysis : mining concept lattices from multi-relational data. *Annals of Mathematics and Artificial Intelligence*, 67(1) :81–108.

Voir page 68.

[Sahuguède et al., 2018] Sahuguède, A., Le Corronc, E., and Le Lann, M.-V. (2018). An ordered chronicle discovery algorithm. In *3nd ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data, AALTD'18*.

Voir page 51.

[Sanger et al., 1977] Sanger, F., Air, G. M., Barrell, B. G., Brown, N. L., Coulson, A. R., Fiddes, J. C., Hutchison, C., Slocombe, P. M., and Smith, M. (1977). Nucleotide sequence of bacteriophage φx174 dna. *nature*, 265(5596) :687.

Voir page 12.

[Srikant and Agrawal, 1996] Srikant, R. and Agrawal, R. (1996). Mining sequential patterns : Generalizations and performance improvements. In *International conference on extending database technology*, pages 1–17. Springer.

Voir page 36.

[Storer, 1987] Storer, J. A. (1987). *Data compression : methods and theory*. Computer Science Press, Inc.

Voir page 12.

[Tatti, 2009] Tatti, N. (2009). Significance of episodes based on minimal windows. In *2009 Ninth IEEE International Conference on Data Mining*, pages 513–522. IEEE.

Voir page 49.

[Tatti, 2015] Tatti, N. (2015). Ranking episodes using a partition model. *Data mining and knowledge discovery*, 29(5) :1312–1342.

Voir page 49.

[Tatti and Cule, 2012] Tatti, N. and Cule, B. (2012). Mining closed strict episodes. *Data Mining and Knowledge Discovery*, 25(1) :34–66.

Voir page 49.

[Ugarte et al., 2017] Ugarte, W., Boizumault, P., Crémilleux, B., Lepailleur, A., Loudni, S., Plantevit, M., Raïssi, C., and Soulet, A. (2017). Skypattern mining : From pattern condensed representations to dynamic constraint satisfaction problems. *Artificial Intelligence*.

Voir pages 43 et 47.

[van de Vel, 1993] van de Vel, M. L. J. (1993). *Theory of Convex Structures*, volume 50. North Holland.

Voir page 84.

[Vo et al., 2016] Vo, B., Le, T., Coenen, F., and Hong, T.-P. (2016). Mining frequent itemsets using the n-list and subsume concepts. *International Journal of Machine Learning and Cybernetics*, 7(2) :253–265.

Voir page 33.

[Wang and Han, 2004] Wang, J. and Han, J. (2004). Bide : Efficient mining of frequent closed sequences. In *Proceedings. 20th international conference on data engineering*, pages 79–90. IEEE.

Voir pages 43 et 100.

[Washio et al., 2007] Washio, T., Nakanishi, K., and Motoda, H. (2007). A classification method based on subspace clustering and association rules. *New Generation Computing*, 25(3) :235–245.

Voir page 52.

[Wille, 1982] Wille, R. (1982). Restructuring lattice theory : an approach based on hierarchies of concepts. *Ordered sets*, pages 445–470. I. Rival (ed.), Dordrecht-Boston, Reidel.

Voir pages 13 et 55.

[Wille, 1992] Wille, R. (1992). Concept lattices and conceptual knowledge systems. *Computers & mathematics with applications*, 23(6-9) :493–515.

Voir page 67.

[Winarko and Roddick, 2007] Winarko, E. and Roddick, J. F. (2007). Armada–an algorithm for discovering richer relative temporal association rules from interval-based data. *Data & Knowledge Engineering*, 63(1) :76–90.

Voir pages 51 et 52.

[Wu et al., 2013] Wu, C.-W., Lin, Y.-F., Yu, P. S., and Tseng, V. S. (2013). Mining high utility episodes in complex event sequences. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 536–544.

Voir page 49.

[Wu and Chen, 2007] Wu, S.-Y. and Chen, Y.-L. (2007). Mining nonambiguous temporal patterns for interval-based events. *IEEE transactions on knowledge and data engineering*, 19(6) :742–758.

Voir page 52.

[Yan et al., 2003] Yan, X., Han, J., and Afshar, R. (2003). Clospan : Mining : Closed sequential patterns in large datasets. In *Proceedings of the SIAM international conference on data mining*, pages 166–177. SIAM.

Voir pages 43, 100 et 152.

[Yang and Kitsuregawa, 2005] Yang, Z. and Kitsuregawa, M. (2005). Lapin-spam : An improved algorithm for mining sequential pattern. In *21st International Conference on Data Engineering Workshops (ICDEW'05)*, pages 1222–1222. IEEE.

Voir pages 36 et 40.

[Yen and Lee, 2013] Yen, S.-J. and Lee, Y.-S. (2013). Mining non-redundant time-gap sequential patterns. *Applied Intelligence*, 39(4) :727–738.

Voir page 51.

[Yoshida et al., 2000] Yoshida, M., Iizuka, T., Shiohara, H., and Ishiguro, M. (2000). Mining sequential patterns including time intervals. In *Data Mining and Knowledge Discovery : Theory, Tools, and Technology II*, volume 4057, pages 213–220. International Society for Optics and Photonics.

Voir pages 47 et 51.

[Zaki, 2000a] Zaki, M. J. (2000a). Scalable algorithms for association mining. *IEEE transactions on knowledge and data engineering*, 12(3) :372–390.

Voir page 33.

[Zaki, 2000b] Zaki, M. J. (2000b). Sequence mining in categorical domains : incorporating constraints. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 422–429.

Voir page 48.

[Zaki, 2001] Zaki, M. J. (2001). Spade : An efficient algorithm for mining frequent sequences. *Machine learning*, 42(1-2) :31–60.

Voir pages 36, 40 et 48.

[Zaki and Hsiao, 2002] Zaki, M. J. and Hsiao, C.-J. (2002). Charm : An efficient algorithm for closed itemset mining. In *Proceedings of the 2002 SIAM international conference on data mining*, pages 457–473. SIAM.

Voir pages 34 et 45.

[Zhang et al., 2013] Zhang, F., Zhang, Y., and Bakos, J. D. (2013). Accelerating frequent itemset mining on graphics processing units. *The Journal of Supercomputing*, 66(1) :94–117.

Voir page 33.

[Zhang et al., 2001] Zhang, M., Kao, B., Yip, C.-L., and Cheung, D. (2001). A gsp-based efficient algorithm for mining frequent sequences. In *Proc. of ic-ai*, pages 497–503.

Voir page 39.

Analyse de séquences avec GALACTIC – Approche générique combinant analyse formelle des concepts et fouille de motifs

Résumé : Une séquence est une suite d'éléments ordonnés comme par exemple les trajectoires de déplacement ou les séquences d'achats de produits dans un supermarché. La fouille de séquences est un domaine de la fouille de données qui vise à extraire des motifs séquentiels fréquents à partir d'un ensemble de séquences, où ces motifs sont le plus souvent des sous-séquences. Plusieurs algorithmes ont été proposés pour l'extraction des motifs séquentiels fréquents. Avec l'évolution des capacités de calcul, la tâche d'extraction des motifs séquentiels fréquents est devenue plus rapide. La difficulté réside alors dans le trop grand nombre de motifs séquentiels extraits, qui en rend difficile la lisibilité et donc l'interprétation. On parle de *déluge de motifs*. L'Analyse Formelle de Concepts (AFC) est un domaine d'analyse de données permettant d'identifier des relations à partir d'un ensemble de données binaires. Les structures de motifs étendent l'AFC pour traiter des données complexes comme les séquences. La plateforme GALACTIC implémente l'algorithme NextPriorityConcept qui propose une approche d'extraction de motifs pour des données hétérogènes et complexes. Il permet un calcul de motifs génériques à travers des *descriptions* spécifiques d'objets par des *prédicats monadiques*. Il propose également de raffiner un ensemble d'objets à travers des *stratégies* d'explorations spécifiques, ce qui permet de réduire le nombre de motifs. Dans ce travail, nous nous intéressons à l'analyse de données séquentielles en utilisant GALACTIC. Nous proposons plusieurs descriptions et stratégies adaptées aux séquences. Nous proposons également des mesures de qualité non supervisées pour pouvoir comparer entre les motifs obtenus. Une analyse qualitative et quantitative est menée sur des jeux de données réels et synthétiques afin de montrer l'efficacité de notre approche.

Mots clés : Analyse Formelle de Concepts, Fouille de séquences, Treillis, Structure de motifs, Séquences temporelles, Séquences d'intervalles, Sous-séquences Communes Maximales.

Sequence analysis using GALACTIC - Generic approach combining Formal Concept Analysis and pattern mining

Abstract: A sequence is a sequence of ordered elements such as travel trajectories or sequences of product purchases in a supermarket. Sequence mining is a domain of data mining that aims at extracting frequent sequential patterns from a set of sequences, where these patterns are most often common subsequences. Support is a monotonic measure that defines the proportion of data sharing a sequential pattern. Several algorithms have been proposed for frequent sequential pattern extraction. With the evolution of computing capabilities, the task of frequent sequential pattern extraction has become faster. The difficulty then lies in the large number of extracted sequential patterns, which makes it difficult to read and therefore to interpret. We speak about "deluge of patterns". Formal Concept Analysis (FCA) is a field of data analysis for identifying relationships in a set of binary data. Pattern structures extend FCA to handle complex data such as sequences. The GALACTIC platform implements the NextPriorityConcept algorithm which proposes a pattern extraction approach for heterogeneous and complex data. It allows a generic pattern computation through specific *descriptions* of objects by *monadic predicates*. It also proposes to refine a set of objects through specific exploration *strategies*, which allows to reduce the number of patterns. In this work, we are interested in the analysis of sequential data using GALACTIC. We propose several descriptions and strategies adapted to sequences. We also propose unsupervised quality measures to be able to compare between the obtained patterns. A qualitative and quantitative analysis is conducted on real and synthetic datasets to show the efficiency of our approach.

Keywords: Formal Concept Analysis, Sequence Mining, Lattice, Pattern Structure, Temporal Sequences, Interval Sequences, Maximum Common Subsequences.

Laboratoire Informatique, Image, Interaction
Faculté des Sciences & Technologie
Avenue Michel Crépeau

17042 LA ROCHELLE CEDEX 1