Fouille de motifs : entre accessibilité et robustesse

Yacine Abboud

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Fouille de motifs : entre accessibilité et robustesse

THÈSE

présentée et soutenue publiquement le

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Yacine ABBOUD

Composition du jury

Rapporteurs :	Sandra Bringay	Professeur, Université de Montpellier 3
	Omar Boucelma	Professeur, Université d'Aix-Marseille
Examinateurs :	Vincent Guigue	Maître de Conférences, Sorbonne Université
	François Charoy	Professeur, Université de Lorraine
Directrice :	Anne Boyer	Professeur, Université de Lorraine
Co-encadrante :	Armelle Brun	Maître de Conférences, Université de Lorraine
Invitée :	Malika Smail-Tabbone	Maître de Conférences, Université de Lorraine

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Table des matières

Chapitre 1

Introduction

1.1 Contexte

L'information occupe une place de plus en plus importante dans notre vie quotidienne, que l'on soit une personne ou une organisation. Elle est non seulement omniprésente : nous sommes tous informés, connectés, divertis. Elle est également facile d'accès : via radio, télévision, smartphone, tablette ou même ordinateur... Cette réalité est la conséquence de près de trois décennies de l'ère de la société de l'information et d'ici 2020, chaque seconde et pour chaque être humain, 1,7Mo de données sera créé [1].
Pourtant, la donnée ne représente que la matière brute, c'est l'organisation et la contextualisation de ces données pour répondre à un besoin [Bellinger et al., 2004] qui en fait de l'information. Nous sommes donc face à un problème d'extraction à grande échelle de l'information des données. Ce problème, qui a longtemps été le monopole des grandes organisations, est devenu une question centrale pour toute organisation ou pour tout utilisateur produisant ou ayant accès à des données. La question qui est alors classiquement posée est de savoir comment extraire de l'information des données en optimisant le temps et la mémoire utilisés.

La fouille de données [Fayyad et al., 1996b] est le processus permettant d'extraire de l'information à partir des données. Elle regroupe un grand nombre de méthodes visant à mener à bien cette extraction d'information en fonction de l'objectif de la fouille et des caractéristiques des données. Cependant, à l'heure actuelle, la fouille de données est essentiellement utilisée dans les grandes organisations ayant les moyens financiers et humains pour l'exploiter. En effet, même si de nombreuses méthodes de fouilles de données sont maintenant accessibles à tous [Fournier-Viger et al., 2014b], les résultats de cette fouille sont souvent complexes à exploiter, notamment par un humain, et ne contiennent pas nécessairement l'information que recherche l'utilisateur. Afin de démocratiser l'exploitation des données et donc l'accès à l'information, la fouille de données se doit de proposer des méthodes accessibles à l'utilisateur, c'est-à-dire adaptables à ses besoins avec des résultats appréhendables.

La tâche de la fouille de données est loin d'être évidente, car un grand volume de données ne signifie pas forcément une grande quantité d'informations. Finalement, cette explosion du volume des données ne fait qu'accroître théoriquement la quantité d'informations disponible et c'est bien la capacité à extraire l'information des données qui devient critique. Les méthodes actuelles ne permettent pas toujours de s'adapter aux spécificités des données fouillées, empêchant

1. https ://www.quantium.com/wp-content/uploads/2016/08/BFM_Quantium.pdf trouvé le 1 juin 2018

ainsi d'extraire une partie de l'information. Une spécificité très courante est la présence de bruit dans les données, c'est-à-dire des pertes, des substitutions ou même des additions d'éléments non porteurs d'information. Le besoin est donc d'extraire l'information des données en minimisant l'impact du bruit présent dans ces dernières.

Les méthodes les plus utilisées de la fouille de données sont regroupées en trois principales catégories : la classification [Dietterich, 1998], la clusterisation [Jain and Dubes, 1988] et la fouille de motifs [Han et al., 2011, Aggarwal, 2015a]. Le choix de la méthode à mettre en œuvre est conditionné par l'objectif de la fouille, la nature des données et par les caractéristiques de l'information que l'on souhaite extraire. Les méthodes de classification et de clusterisation permettent essentiellement de structurer les données tandis que les méthodes de fouille de motifs permettent de les synthétiser grâce à l'extraction d'éléments récurrents. C'est pourquoi plusieurs de ces méthodes sont souvent utilisées séquentiellement pour extraire l'information la plus utile possible. En pratique, la fouille de motifs est souvent utilisée conjointement avec la classification ou la clusterisation, soit pour améliorer leurs résultats, soit pour proposer des résultats plus synthétiques. Dans notre démarche de rendre la fouille de données plus adaptable et ses résultats plus appréhendables par l'utilisateur, nous avons choisi de nous focaliser sur la fouille de motifs pour son caractère incontournable dans le processus d'extraction de l'information lors de nombreuses fouilles de données.

La fouille de motifs est l'un des domaines les plus étudiés dans la littérature de la fouille de données. Ce domaine a émergé dans les années 90 avec l'introduction de l'algorithme Apriori [Agrawal et al., 1994]. Depuis, de nombreux autres algorithmes [Zaki, 2001] [Han et al., 2001, Fournier-Viger et al., 2014a, Fahed et al., 2018] ont grandement amélioré le domaine de la fouille de motifs. Toutefois, l'ensemble final des motifs extraits est souvent conséquent et donc non appréhendable par un utilisateur en plus d'être coûteux, voire impossible à extraire [Pei et al., 2007, Abboud et al., 2017]. Dans notre contexte de fouille de motifs, nous définissons l'efficience comme la capacité à extraire l'ensemble des motifs en utilisant le minimum de moyens (temps et matériel, notamment mémoire). En pratique, l'ensemble complet des motifs est souvent compliqué à appréhender pour l'utilisateur. Non seulement l'ensemble est régulièrement très large, mais l'utilisateur n'est généralement intéressé que par un sous-ensemble [Pei et al., 2007]. Le passage de cet ensemble à l'ensemble des motifs que l'utilisateur juge pertinents peut donc fréquemment constituer un véritable frein à l'utilisation de la fouille, il y a donc un réel besoin d'améliorer l'appréhendabilité des résultats en réduisant l'ensemble des motifs extraits sans perdre l'information.
Par ailleurs, les grandes bases de données sont maintenant très communes, la fouille de motifs est donc régulièrement confrontée à des problèmes de temps d'exécution ou de consommation de mémoire. Pourtant, l'utilisateur rencontre en permanence des situations où il souhaite réaliser une fouille de données avec des exigences de temps ou des contraintes matérielles. Une fouille de motifs qui ne peut être effectuée dans un temps raisonnable ou sur un ordinateur accessible est souvent inutile pour l'utilisateur. L'autre enjeu majeur pour l'accessibilité de la fouille de motifs à l'utilisateur est donc l'amélioration de son efficience afin de permettre de réaliser des fouilles avec un minimum de ressources (temps et matériel).
L'utilisation de contraintes est la direction la plus prometteuse adoptée par la littérature [Yan et al., 2003, Pei et al., 2007, Béchet et al., 2015] pour améliorer l'appréhendabilité et l'efficience en réduisant à la fois la taille de l'ensemble des motifs extraits et le temps de calcul. Une contrainte est une fonction booléenne qui permet de focaliser la fouille uniquement sur les motifs qui la satisfont. En pratique, l'utilisateur identifie les caractéristiques de l'infor-

mation qu'il souhaite extraire pour les retranscrire sous forme de contraintes applicables aux motifs. C'est pourquoi l'utilisateur cherche souvent à utiliser une combinaison de contraintes afin d'obtenir l'ensemble de motifs le plus proche de l'ensemble de motifs qu'il considère pertinents. Cependant, la combinaison de contraintes désirée par l'utilisateur est régulièrement problématique. D'abord parce cette combinaison de contraintes est rarement intégrable au sein d'un seul et même algorithme, ce qui entraîne une utilisation séquentielle de plusieurs algorithmes sans garanties de compatibilité. Ensuite parce que cette combinaison de contraintes n'est pas toujours adaptable aux caractéristiques des données fouillées. Ainsi, le bruit, par exemple, impacte négativement de nombreuses contraintes de l'état de l'art comme la clôture ou la contiguïté. La contiguïté est satisfaite par les motifs dont tous les éléments sont consécutifs. La clôture est satisfaite par les motifs qui ne sont inclus dans aucun autre motif avec la même fréquence d'apparition.
L'enjeu est finalement d'identifier et de proposer le plus grand nombre de contraintes permettant de réduire la taille de l'ensemble des motifs extraits tout en conservant au maximum les motifs jugés pertinents par l'utilisateur, et ce, même dans un contexte de données bruitées.

Dans cette thèse, nous posons donc la question scientifique suivante : **"Comment proposer une fouille de motifs efficiente et résistante au bruit permettant d'extraire un ensemble de motifs pertinent et appréhendable par l'utilisateur ?"** Nous nous intéressons à cette question en nous fixant pour objectif de concevoir un algorithme intégrant plusieurs contraintes, en particulier des contraintes résistantes au bruit, permettant ainsi de minimiser les résultats de la fouille tout en conservant l'information recherchée par l'utilisateur.

1.2 Contributions

Les contributions scientifiques de cette thèse consistent en l'introduction de deux nouvelles contraintes liées à des motifs séquentiels et d'un nouvel algorithme générique de fouille de motifs séquentiels C3Ro.

1.2.1 Contrainte de robustesse au bruit

La contrainte de contiguïté est une contrainte classique de la littérature [Fürnkranz, 1998, Chen, 2008] permettant à la fois de réduire le nombre de motifs extraits et le temps d'exécution, tout en conservant une très grande partie de l'information contenue dans les
données [Zhang et al., 2015]. Elle peut être utilisée, quel que soit le contexte applicatif de l'utilisateur. En revanche, la contrainte de contiguïté fausse l'extraction de l'information lorsque la fouille est effectuée sur des données bruitées. Dans une base de séquences, le bruit correspond soit à une perte, soit à une substitution ou soit à une addition d'items au sein des séquences. Ces perturbations induisent une sous-évaluation du support de certains motifs. L'utilisation de *wildcards* avec la contrainte de contiguïté permet de pallier l'addition d'items due au bruit lors de la fouille de données [Abboud et al., 2017]. Une wildcard [Li and Wang, 2008, Wu et al., 2013, Xie et al., 2017] est un joker par rapport à la contrainte de contiguïté. Si $\langle a, b, d, c\rangle$ est un motif, il peut être considéré comme le motif $\langle a, b, c\rangle$ avec une *wildcard* (d). Cependant, l'utilisation de *wildcards* lors d'une fouille de motifs séquentiels contigus ne permet pas de savoir quels motifs ont été extraits grâce aux *wildcards* et dans quelle proportion. Or de nombreux contextes applicatifs requièrent que les motifs soient majoritairement contigus pour être pertinent. Nous proposons donc une nouvelle contrainte : la contrainte de robustesse au bruit. Cette contrainte permet à

l'utilisateur de quantifier la proportion maximale de *wildcards* utilisables dans l'extraction d'un motif, grâce à un paramètre (ratio de robustesse). L'utilisation des *wildcards* et de la contrainte de robustesse permet donc de fouiller des données bruitées en conservant la valeur ajoutée de la contrainte de contiguïté tout en autorisant l'utilisateur à choisir la proportion de contiguïté des motifs extraits.
Cette contribution est en cours de soumission dans un article de revue.

1.2.2 Contrainte de clôture allégée

La contrainte de clôture est une autre contrainte classique de la littérature [Pei et al., 2000, Yan et al., 2003] qui peut être utilisée indépendamment du contexte applicatif de l'utilisateur. Tout comme la contrainte de contiguïté, elle permet à la fois de réduire le nombre de motifs extraits et le temps d'exécution tout en conservant l'information contenue dans les données. Cette contrainte permet d'écarter tous les motifs ayant un super-motif de même support. Le support correspond au nombre d'occurrences du motif. La contrainte de clôture est souvent insuffisante pour réduire l'ensemble des motifs à une taille appréhendable par l'utilisateur [Zhang et al., 2015] particulièrement lorsque les données sont bruitées. En effet, les perturbations engendrées par le bruit entraînent l'apparition de nombreux nouveaux motifs. Nous proposons une nouvelle contrainte de clôture allégée permettant une réduction plus importante de l'ensemble des motifs notamment dans le cas de données bruitées. La contrainte de clôture allégée ne requiert pas que le super-motif ait le même support que ses sous-motifs pour les écarter. Cette contrainte repose sur un ratio d'allègement pour fixer jusqu'à quelle proportion du support d'un super-motif ses sous-motifs sont écartés. Elle a été conçue pour pallier la substitution et la perte d'éléments dans les données bruitées. Ce ratio d'allègement permet donc d'adapter la fouille aux besoins de l'utilisateur en réduisant l'ensemble des motifs extraits au prix d'une partie de l'information. La contrainte de clôture allégée, via la valeur du ratio d'allègement, offre la possibilité d'améliorer l'appréhendabilité de la fouille de motifs selon les besoins de l'utilisateur.
Cette contribution est en cours de soumission dans un article de revue.

1.2.3 C3Ro : un algorithme générique

C3Ro est un algorithme générique de fouille de motifs séquentiels intégrant de nombreuses contraintes au travers de paramètres afin de proposer à l'utilisateur la fouille la plus efficiente possible tout en réduisant au maximum la taille de l'ensemble des motifs extraits. Les contraintes de contiguïté, de robustesse et de clôture allégée sont naturellement intégrées dans le processus de fouille et ajustables par l'utilisateur grâce au paramètre du nombre de wildcards, au ratio de robustesse et au ratio d'allègement.
Une combinaison de contraintes préfixes-monotones [Pei et al., 2007] est également intégrable pendant la fouille en fonction du contexte applicatif de l'utilisateur. L'algorithme C3Ro, grâce à ses nombreux paramètres, est capable de fouiller des motifs séquentiels sans contraintes tout comme des motifs séquentiels très contraints. L'algorithme C3Ro rivalise avec les meilleurs algorithmes de fouille de motifs de la littérature en termes de temps d'exécution tout en consommant beaucoup moins de mémoire. C'est pourquoi contrairement aux autres algorithmes de la littérature, C3Ro est capable d'être exécuté en un temps raisonnable (moins d'une heure), sur de grands jeux de données où le nombre de motifs à extraire est important.
Cette contribution est en cours de soumission dans un article de revue. Nous avons publié l'algorithme CCPM [Abboud et al., 2017] qui est une version antérieure de C3Ro n'incluant pas les contraintes de robustesse et de clôture allégée.

1.3 Application

Cette thèse a été financée par l'entreprise People Compétences qui travaille dans le domaine des ressources humaines et plus particulièrement dans le champ des compétences. Une grande partie de l'activité de People Compétences repose sur le conseil pour diverses organisations concernant la mise en place de la démarche compétence. "La démarche compétences structure et dynamise les étapes de la gestion des ressources humaines en fournissant un cadre méthodologique et une palette d'outils opérationnels." [2]. Cette démarche s'inscrit dans la stratégie de l'entreprise et répond à un triple enjeu : (i) anticiper l'évolution des emplois ; (ii) recruter, gérer et développer le potentiel de l'entreprise ; (iii) consolider la performance de l'entreprise sur son marché. Sachant que la démarche compétence repose sur l'élaboration de référentiels métiers qui répertorient toutes les compétences d'une organisation, en concertation avec l'entreprise, nous avons choisi d'utiliser notre algorithme dans le contexte du marché de l'emploi en ligne pour extraire les compétences contenues dans les offres d'emploi. L'identification des compétences du marché de l'emploi représente une source d'information inestimable pour l'accompagnement dans l'élaboration de référentiels métiers pertinents.
Le marché de l'emploi en ligne constitue un excellent cadre d'application pour notre algorithme de fouille de motifs, car il s'agit d'un secteur avec énormément de données et une forte croissance : pas moins de 7,4 millions d'offres d'emploi ont été diffusées sur le marché de l'emploi en ligne en 2016 contre 5.8 millions en 2015 [3]. Nous avons fait le choix d'extraire les compétences des offres d'emploi, car dans le cadre du marché de l'emploi en ligne, une offre d'emploi est supposée contenir les compétences demandées par l'organisation qui a formulé cette offre. Nous avons mentionné plus haut un premier intérêt dans leur identification, un deuxième intérêt est de mettre en place un encodage des offres d'emploi uniquement avec les compétences demandées par l'emploi. Cet encodage peut permettre d'effectuer des rapprochements automatiques plus pertinents avec les compétences que possèdent les demandeurs d'emploi et ainsi de pourvoir au mieux les emplois.
L'objectif applicatif dans cette thèse est d'exploiter, d'analyser les données issues des sites d'offres d'emploi et de traiter les besoins applicatifs suivants :

- Identifier et extraire les compétences présentes dans les offres du marché de l'emploi en ligne [Abboud et al., 2015].
- Encoder ces offres d'emploi uniquement avec ces compétences.
- Valider la pertinence de cet encodage pour représenter les offres d'emploi.

En atteignant ces objectifs, nous aidons l'entreprise People Compétences à améliorer ces ressources en vue de leur activité de conseil sur la démarche compétence.

1.3.1 Évaluation

Afin d'évaluer les apports de nos contributions théoriques et applicatives, nous avons mené plusieurs études expérimentales et constitué un corpus de données. Le corpus de données que nous avons constitué est composé de 800 000 offres d'emploi extraites à partir de plusieurs sites d'offres d'emploi français au cours des années 2016 et 2017. Pour l'évaluation de nos contributions théoriques, nous avons d'abord comparé notre algorithme C3Ro à deux algorithmes de l'état de

2. http ://www.ceficem.fr/des-outils-de-developpement-des-competences/une-demarche-conseil/la-demarche-competences/
3. http ://www.pole-emploi.org/statistiques-analyses/entreprises/offres-demploi-et-recrutement/les-offres-demploi-diffusees-p-2.html ?type=article

l'art, en termes de temps d'exécution et de consommation de mémoire, sur plusieurs jeux de données de la littérature ayant des caractéristiques variés. Cette comparaison a pour but de positionner les performances de C3Ro dans le paysage des algorithmes de la fouille de motifs et d'évaluer sa capacité à passer à l'échelle. Nous avons ensuite étudié l'apport de nos deux nouvelles contraintes en évaluant leur impact sur l'ensemble des motifs extraits dans le cadre de l'extraction des compétences sur une partie de notre corpus bruité d'offres d'emploi.
Nous avons finalement évalué notre application par rapport à nos objectifs applicatifs.

1.4 Plan de la thèse

Le reste du manuscrit est organisé comme suit : dans le chapitre 2, nous présentons les concepts et les approches de la littérature liés à nos travaux dans la fouille de données et plus particulièrement dans la fouille de motifs. Puis, nous introduisons et évaluons dans le chapitre 3 nos deux nouvelles contraintes et l'algorithme C3Ro. Ensuite, dans le chapitre 4, nous présentons et évaluons notre application de fouille de compétences dans notre corpus d'offres d'emploi. Nous concluons ce manuscrit et présentons les perspectives dans le chapitre 5.

Chapitre 2

État de l'art

Il y a encore une décennie, les gros volumes de données étaient le monopole des grandes organisations dû à la complexité d'avoir à la fois l'approvisionnement en données et l'infrastructure informatique pour les stocker et en extraire de l'information utile. Aujourd'hui, chaque visite de site web, chaque utilisation d'application mobile, chaque commentaire posté en ligne laissent une trace qui crée de la donnée. De nombreuses organisations et personnes possèdent ou utilisent donc maintenant des applications et sites web qui génèrent des volumes de données suffisamment importants pour justifier à la fois une infrastructure de stockage et une extraction de l'information utile. Extraire cette information peut permettre, par exemple, d'identifier et de mieux comprendre les besoins du client ou de l'utilisateur et donc d'y répondre plus efficacement. La donnée occupe maintenant un rôle central pour toute organisation qui la génère, car elle permet de mettre en évidence certains liens entre l'activité de l'entité et ses utilisateurs. Cette réalité explique la popularité actuelle des méthodes de fouille de données. Pourtant, tout comme l'accès ou la possession de gros volumes de données il y a dix ans, la fouille de données est actuellement l'apanage des grandes structures. En effet, les méthodes de fouille de données sont souvent complexes, opaques, et les résultats durs à interpréter, ce qui les rend très difficiles à utiliser et à adapter par des néophytes du domaine. Afin de proposer un outil de fouille de données à la portée des utilisateurs, j'ai décidé de focaliser mon travail sur la fouille de motifs [Agrawal et al., 1993] dont la caractéristique principale est l'appréhendabilité des résultats par l'utilisateur. Dans ce chapitre, je contextualise la fouille de motifs au sein des méthodes de la fouille de données. Je me focalise ensuite sur la fouille de motifs séquentiels qui est au cœur de mes travaux. Enfin, je présente les solutions apportées par l'état de l'art concernant le problème de l'efficience de la fouille et de l'ampleur de l'ensemble des motifs extraits, via l'utilisation de contraintes. Tout au long de ma revue de la littérature, je mettrai en relief les enseignements (E) et les problématiques scientifiques (PS) soulevés.

2.1 La fouille de données

2.1.1 Principe

La fouille de données, aussi connue sous le nom de *Data mining*, est un domaine de l'informatique qui est apparu au début des années 1990 dont le but est l'extraction d'information voire de connaissances, à partir d'une grande quantité de données [Piateski and Frawley, 1991, Han et al., 1992]. Il y a une divergence dans la communauté scientifique sur la place de la fouille de données dans le processus d'extraction de la connaissance à partir de données : *Knowledge Discovery from Data* (KDD). En effet, pour certains [Fayyad et al., 1996a], la fouille de données

n'est qu'une étape de ce processus : l'extraction (voir figure 2.1). Pour d'autres [Han et al., 2011, Aggarwal, 2015a], la fouille de données et l'extraction de la connaissance à partir de données sont un seul et même domaine. J'adopte ce dernier point de vue et définis la fouille de données comme suit :

Definition 2.1.1. La **fouille de données** est le processus de découverte de connaissances non triviales et utiles à partir de grandes quantité de données.

Ce processus peut-être décomposé en plusieurs étapes [Han et al., 2011, Aggarwal, 2015a] (voir figure 2.1) :

1. **Sélection :** c'est l'étape de la récupération des données pertinentes à analyser.
2. **Préparation :** c'est l'étape de nettoyage des données.
3. **Transformation :** c'est l'étape de transformation des données sous la forme appropriée pour l'extraction.
4. **Extraction :** c'est l'étape centrale du processus où des méthodes de fouille de données sont appliquées pour extraire les motifs/modèles intéressants ou inattendus.
5. **Interprétation et évaluation :** cette étape finale permet d'identifier les motifs ou modèles vraiment intéressants qui représentent des connaissances.

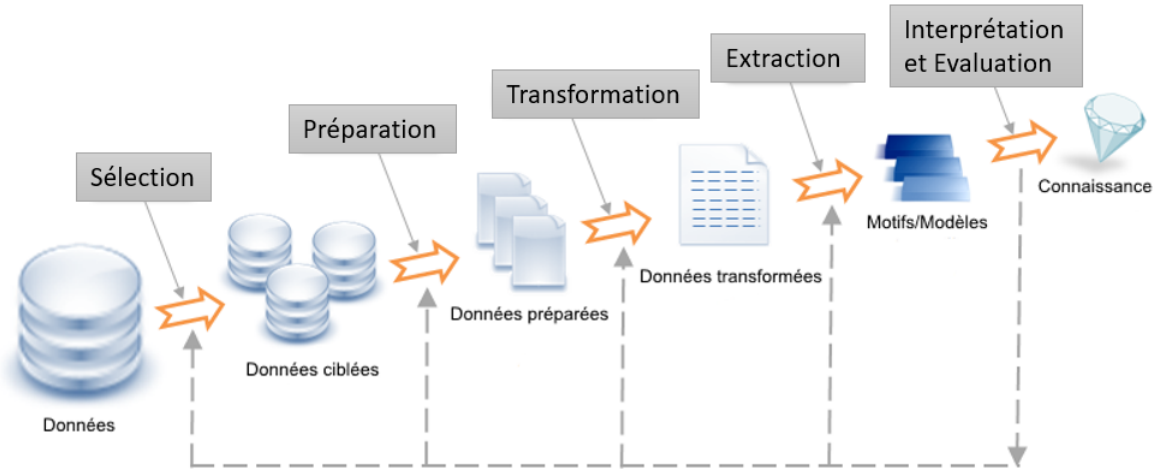


FIGURE 2.1 – Processus de la fouille de données [4]

2.1.2 Type de données

Les trois premières étapes de ce processus sont conditionnées par le type de données que l'on souhaite fouiller. Parmi les types de données standards qui peuvent être fouillés, nous identifions les bases de données [Maier, 1983], les bases de transactions [Tung et al., 1999, Han et al., 2000b] et les entrepôts de données [Kimball and Ross, 2011]. Une base de données est une collection de tables, où chaque table correspond à un ensemble d'attributs (colonnes) lié à un ensemble d'enregistrements (lignes). Nous introduisons une base de données de plusieurs tables liées à la vente de matériel informatique dans une entreprise fictive nommée MeilleureVente (voir table

4. http ://www.zentut.com/data-mining/what-is-data-mining/ trouvé le 4 mai 2018

2.1). Cette base sera utilisée à plusieurs reprises pour illustrer des concepts introduits dans cet état de l'art.

table	attributs
client	$(client\_id, nom, prénom, adresse, age, emploi, revenu, ...)$
produit	$(produit\_id, marque, catégorie, prix, fournisseur, ...)$
magasin	$(magasin\_id, nom, adresse, ...)$
vente	$(vente\_id, client\_id, produit\_id, magasin\_id, date, heure, type\_paiement, ...)$

TABLE 2.1 – Base de données des ventes de MeilleureVente

Une base de transactions est un ensemble de transactions, où chaque transaction a un identifiant unique associé à une liste d'éléments ou *items*. Dans le cas de l'entreprise MeilleureVente, une base de transactions peut regrouper la liste des produits associés à chaque vente (voir la tableau 2.2).

$vente\_id$	$client\_id$	liste des produits
V001	C001	$P1, P4, P10$
V002	C001	$P4, P12, P110$
V003	C002	$P4, P10, P44$
...	...	...

TABLE 2.2 – Exemple de base de transactions des ventes de MeilleureVente

Un entrepôt de données est une structure multidimensionnelle dans laquelle chaque dimension représente un ensemble d'attributs et chaque enregistrement correspond à une agrégation de ces attributs. Dans le cas de l'entreprise MeilleureVente, une agrégation entre les adresses des magasins, le prix et la catégorie des produits vendus donne l'entrepôt visible sur la figure 2.2. La cellule en rouge contient la liste des boîtiers vendus entre 200 et 300 euros dans le magasin de Paris.
Il existe également d'autres types de données, plus versatiles, tels que les
séquences [Mannila et al., 1997], les flux de données [Krempl et al., 2014], les données géographiques [Samet, 1990], les graphes [Likhachev et al., 2008], les données
multimédia [O'Callaghan et al., 2012], etc.

2.1.3 Méthodes de fouille

Toutes les méthodes de fouille de données suivent le même processus, mais diffèrent dans la réalisation de chacune des étapes en fonction du type de données fouillé et de l'objectif de la fouille. Les méthodes de fouille de données sont regroupées en catégories [Han et al., 2011] qui se différencient par les objectifs de la fouille :

— La **Description** vise à synthétiser les caractéristiques des attributs d'enregistrements regroupés selon certains critères. On appelle ce regroupement, **une classe**. Les méthodes de description sont utilisées sur des données où les enregistrements sont déjà assignés à des classes. Pour l'entreprise MeilleureVente, un exemple de description est l'analyse des caractéristiques des clients ayant acheté pour plus de 2000 euros de produits en un an. Le résultat est un profil de client entre 45 et 50 ans avec des revenus élevés.

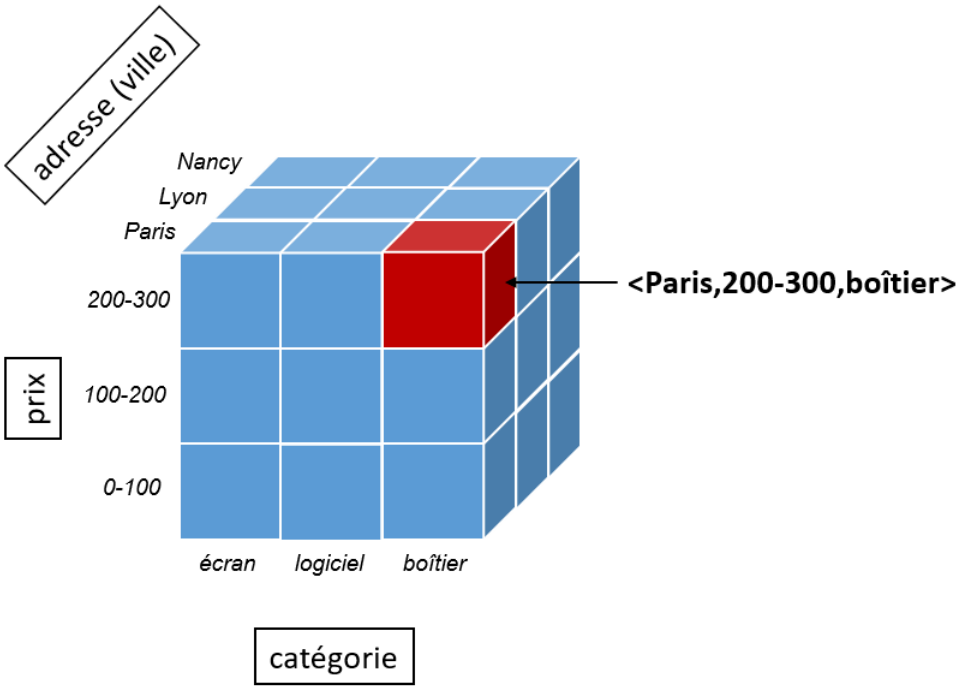


FIGURE 2.2 – Exemple d'entrepôt de données des ventes de MeilleureVente

— La **Classification** [Antonie et al., 2001, Lu and Weng, 2007] vise à identifier la classe d'appartenance d'un enregistrement. Elle consiste à trouver un modèle permettant d'affecter un enregistrement à une classe en fonction de la valeur de ses attributs [Dietterich, 1998]. La classification est généralement réalisée en deux étapes. **L'apprentissage**, qui permet d'entraîner le modèle sur des données où les enregistrements sont déjà affectés à des classes (on dit les que les enregistrements sont étiquetés). À la fin de l'apprentissage, un premier modèle de classification est créé. Vient ensuite la phase de **validation**, effectuée sur d'autres données également classifiées, afin de déterminer la qualité du modèle et de procéder aux ajustements nécessaires pour l'améliorer.
L'entreprise MeilleureVente classifie ses clients en fonction du montant des produits qu'ils ont achetés au cours de l'année pour mieux cibler leur marketing ($C_1 =< 0-100 >; C_2 =< 101-500 >; C_3 =< 501-1000 >; ...$). Un modèle de classification peut permettre d'affecter les nouveaux clients à une classe, à partir des caractéristiques des attributs *age* et *revenu*.
Dans la littérature, il existe plusieurs familles de méthodes de classification, parmi les plus connues nous trouvons les réseaux de neurones [Wan, 1990], les arbres de décisions [Quinlan, 1986], les classifieurs de Bayes [Cheeseman et al., 1988], les machines à support de vecteurs (SVM) [Vapnik, 1999], etc.
Plus généralement, toutes les méthodes de classification souffrent d'une part, du manque de données pour réaliser la phase d'apprentissage et d'autre part, de la présence de données incorrectement classifiées. Dans les deux cas, le modèle de classification créé ne peut être considéré comme étant fiable.

— La **Clusterisation** vise à regrouper automatiquement les enregistrements au sein de

classes appelées ici clusters. L'ensemble de clusters issus d'un seul processus représente une clusterisation [Jain and Dubes, 1988]. Contrairement à la classification, les classes ne sont ni identifiées ni caractérisées préalablement. La clusterisation s'effectue donc sur des données non classifiées, en se basant sur une mesure de distance ou de similarité [Han et al., 2011]. La constitution d'un cluster repose sur la proximité ou similarité des enregistrements au sein de ce cluster, et de l'éloignement ou de la dissimilarité des enregistrements au sein des autres clusters. Le choix de la mesure de distance ou de similarité est donc primordial pour les méthodes de clusterisation, car il influence la nature et la qualité de la clusterisation. À noter que, chaque domaine d'application possédant ses propres données, il peut également posséder sa propre mesure de distance ou de similarité. Chaque domaine d'application ou chaque type de données peut donc nécessiter de concevoir une mesure différente afin de faire contraster au mieux les différences ou les similarités dans les données.
Dans le cas de l'entreprise MeilleureVente, une clusterisation peut être effectuée sur les données concernant les clients. Les clusters pourraient représenter des profils types de clients pour un marketing ciblé.
Dans l'état de l'art, nous pouvons distinguer trois types principaux de méthodes de clusterisation : (i) les méthodes de clusterisation hiérarchique basées sur la construction d'arbres [Kaufman and Rousseeuw, 2009] ; (ii) les méthodes de clusterisation par partitionnement basées sur un nombre défini de clusters, l'algorithme des k-moyennes (*k-means*) est la méthode de clusterisation par partitionnement la plus
utilisée [MacQueen et al., 1967] ; (iii) les méthodes probabilistes basées sur l'estimation de la densité de probabilité [Dempster et al., 1977].

— La **Fouille de motifs fréquents** vise à identifier des récurrences, que l'on appelle motifs fréquents, parmi les enregistrements. Ces motifs fréquents peuvent être des ensembles d'items fréquents, des sous-séquences (aussi appelées motifs séquentiels) fréquentes ou tout autre type de sous-structures comme les graphes, les arbres, etc. Un ensemble d'items fréquents fait référence à plusieurs items qui apparaissent régulièrement ensemble dans une base de transactions.
Dans la base de transactions des ventes de MeilleureVente (tableau 2.2), les produits $P4$ et $P10$ apparaissent ensemble dans les transactions $V001$ et $V002$. MeilleureVente peut déduire que ces produits sont achetés ensemble par de nombreux clients et proposer des promotions pour l'achat des deux produits.
La fouille de motifs fréquents permet également la découverte d'associations et de corrélations intéressantes entre les enregistrements, on parle d'extraction de **règles d'association**.
Toujours dans la base de transaction des ventes, le produit $P4$ apparaît dans les trois transactions de la base et le produit $P10$ apparaît dans $V001$ et $V002$. MeilleureVente peut déduire la règle d'association suivante : l'achat du produit $P4$ va être accompagné de l'achat du produit $P10$ dans 66% des cas.
— La **Détection d'anomalies** vise à découvrir les enregistrements qui diffèrent significativement du comportement classique des autres enregistrements au sein des données. La plupart des autres méthodes de fouille considèrent ces anomalies comme du bruit ou des exceptions. Pourtant, dans certaines applications (la détection de fraudes), ces anomalies sont plus intéressantes que les comportements classiques. Les méthodes de détection

4. Cet exemple n'est pas statistiquement significatif, mais il illustre l'idée générale

d'anomalies reposent sur des modèles probabilistes ou sur des mesures de distances se focalisant sur les enregistrements éloignés de tout cluster [Aggarwal, 2015b].
L'entreprise MeilleureVente peut utiliser des méthodes de détection d'anomalies pour identifier les achats effectués avec une carte de crédit de montants discordants avec le reste des achats effectués avec cette même carte et ainsi détecter certaines utilisations frauduleuses.

Nous venons de présenter les différentes catégories de méthodes de fouille de données. Les méthodes les plus utilisées de la fouille de données sont la classification, la clusterisation et la fouille de motifs fréquents. Nous avons vu que la classification nécessite d'avoir des classes caractérisées et un jeu de données déjà étiqueté pour la phase d'apprentissage. La clusterisation nécessite de choisir une méthode en fonction des données et des objectifs, ce qui est souvent une tâche complexe. La fouille de motifs fréquents a pour seul prérequis de déterminer le nombre d'apparitions minimums d'un motif pour qu'il soit considéré comme fréquent.
La classification et la clusterisation sont des méthodes visant à structurer les données alors que la fouille de motifs vise plutôt à synthétiser les données. Il n'est donc pas rare d'utiliser séquentiellement plusieurs méthodes de fouille de données pour obtenir des résultats plus pertinents. Ainsi la fouille de motifs peut être exécutée en amont de la classification ou de la clusterisation pour améliorer leurs résultats ou alors en aval pour rendre leurs résultats plus synthétiques pour l'exploitation. Finalement, la fouille de motifs fréquents s'inscrit complètement dans la logique de simplifier les résultats de la fouille de données, car elle permet de mettre en évidence les motifs les plus représentatifs des données en fixant simplement un support minimum.
Dans le cadre de ma problématique de rendre les résultats de la fouille de données appréhendable par les utilisateurs, j'ai choisi de concentrer mon travail sur la fouille de motifs fréquents pour son accessibilité et sa logique de synthétisation.

2.2 La fouille de motifs séquentiels fréquents

Initialement, la fouille de motifs fréquents a été introduite dans le cadre d'études sur les habitudes de consommation des clients de magasins, aussi appelées études sur le panier de la ménagère. L'objectif est alors l'extraction de règles d'association sur les bases de transactions formées à partir de paniers d'achats. Ce panier d'achats correspond aux différents produits achetés (ce que nous avons défini précédemment comme des items) par un client à un moment donné. Les produits achetés dans un panier n'ont pas d'ordre puisqu'ils sont tous achetés en même temps. Dans la littérature, la convention est d'ordonner les items d'un panier alphabétiquement. Une transaction est alors définie comme étant un panier, c'est-à-dire l'ensemble des achats ou items, d'un client à un moment donné. Depuis, la fouille de motifs fréquents dans les bases de transactions est toujours très utilisée et dans de nombreux contextes applicatifs.

Definition 2.2.1. Soit $I = \{i_1, i_2, ..., i_n\}$ un ensemble fini d'items distincts avec $n \in \mathbb{N}^*$. Une **transaction** est un ensemble d'items notée $T = \{i_1, ..., i_j\}$ où $j \in \{1, ..., n\}$.

Definition 2.2.2. Un **itemset** est un ensemble d'items : $I_k = \{i_1, ..., i_k\}$. Si l'itemset I_k est un sous-ensemble d'un itemset I_j, alors I_k est un sous-itemset de I_j et on dit que I_k est inclus dans I_j (noté $I_k \subseteq I_j$). On dit que I_k est de longueur k ou est un k-itemset, car il contient k items ($|I_k| = k$).

Les concepts d'itemset et de transaction sont théoriquement similaires, mais utilisés différemment. L'itemset représente le concept théorique d'ensemble d'items, tandis que la transaction est toujours considérée en tant qu'enregistrement d'une base de transactions dont la spécificité

est de contenir un ensemble d'items. Dans la littérature, lorsque la configuration de la fouille de motifs est la fouille d'items dans une base de transactions, on parlera de fouille d'itemsets [Aggarwal and Yu, 1998, Pasquier et al., 1999]. La fouille d'itemsets fréquents consiste à extraire les items présents dans un nombre de transactions supérieur ou égal à un seuil défini par l'utilisateur. De nombreuses applications ne nécessitent pas d'avoir un ordre parmi les motifs fouillés, comme en attestent les nombreux algorithmes de fouilles d'itemsets de la littérature [Agrawal et al., 1994, Zaki, 2000a, Pei et al., 2001, Han et al., 2004, Uno et al., 2004]. Cependant, une multitude d'applications requiert la conservation de l'ordre entre les motifs, telles la fouille de données textuelles, la fouille de séquences génétiques, etc. Afin de répondre à ce besoin, la fouille de motifs séquentiels est apparue [Agrawal and Srikant, 1995]. Cette fouille s'effectue généralement sur des bases de séquences. Au départ, la base de séquences vient également des études sur le panier de la ménagère. Chaque séquence regroupe les paniers d'achats (transactions) de chaque client en les ordonnant chronologiquement. À partir de la base de transactions des ventes de MeilleureVente, il est donc possible d'obtenir la base de séquences suivante :

client_id	liste des produits
C001	$\langle P1, P4, P10, \rangle, \langle P4, P12, P110\rangle$
C002	$\langle P4, P10, P44\rangle$
...	...

TABLE 2.3 – Base de séquences des ventes de MeilleureVente

À mon sens, la notion d'ordre est souvent présente dans les données des utilisateurs ; il m'apparaît important de conserver cette information supplémentaire lorsqu'elle est disponible. Je choisis donc d'orienter mon travail de recherche vers le domaine de la fouille de motifs séquentiels.

Definition 2.2.3. Une **séquence** est une liste ordonnée d'itemsets notée $S = \langle E_1, ..., E_j\rangle$. On dit que S est de longueur k ou est une k-séquence, si elle contient k items : $k = |E_1| + ... + |E_j|$.

Par convention, nous écrirons toujours les itemsets avec une lettre en majuscule et les items avec une lettre en minuscule. Dans le cas particulier où les itemsets qui constituent les séquences sont réduits à des items uniques, on parle de fouille de *string* [Fischer et al., 2006, Matsui et al., 2013].

Definition 2.2.4. Soit S et S' deux séquences $\langle E_1, E_2, ..., E_n\rangle, \langle E'_1, E'_2, ..., E'_m\rangle$. S est une **sous-séquence** de S' (notée $S \sqsubseteq S'$) s'il existe des entiers $1 \leq j_1 < j_2 < ... < j_n \leq m$ tels que $E_1 \subseteq E'_{j_1}, E_2 \subseteq E'_{j_2}, ..., E_n \subseteq E'_{j_n}$. On dit que S est inclus dans S'.

La fouille de motifs séquentiels fréquents consiste à extraire les sous-séquences ou motifs séquentiels inclus dans un nombre de séquences supérieur ou égal à un seuil défini par l'utilisateur. La fréquence est le critère le plus utilisé pour sélectionner les motifs, mais ce n'est pas le seul, l'intérêt [Sakurai et al., 2007], par exemple, en est un autre. Un **motif séquentiel** a la même définition théorique qu'une séquence. En pratique, le terme de motif séquentiel est utilisé lorsque l'on fait référence à une sous-séquence et le terme séquence est utilisé lorsque l'on fait référence aux enregistrements d'une base de séquences. La base BDS (voir tableau 2.4) est un exemple de base de séquences composée par les items de $I = \{a, b, c, d, e, f\}$.

Le domaine de la fouille de motifs fréquents est un domaine très prisé par la communauté de la fouille de données. Chaque année, des centaines d'algorithmes sont créés pour répondre à des besoins spécifiques. Le tout premier a avoir vu le jour est l'algorithme Apriori [Agrawal et al., 1994] qui est basé sur la contrainte anti-monotone de la fréquence.

sid	séquence
#1	$\langle\{e,b\},\{c\},\{f,d\},\{d\},\{a\}\rangle$
#2	$\langle\{e\},\{c\},\{b\},\{e,b,a,f\}\rangle$
#3	$\langle\{e\},\{b\},\{f,d\},\{a\}\rangle$
#4	$\langle\{b\},\{f,d\}\rangle$

TABLE 2.4 – Base de séquences BDS

Definition 2.2.5. Si une contrainte qui est violée par un élément l'est aussi par tous ses sur-éléments alors on dit que cette contrainte est **anti-monotone** [Uryas' ev, 1988].

Ainsi si un item a n'est pas fréquent, alors tous les motifs contenant l'item a ne peuvent être fréquents [Agrawal et al., 1993]. Le support est la mesure utilisée pour déterminer si un motif est fréquent ou non.

Definition 2.2.6. Le **support** ou **support absolu** d'un motif M dans une base de séquences BDS est défini par le nombre de séquences contenant M (noté $sup_A^{BDS}(M)$). Le **support relatif** de M dans BDS est la proportion de séquences de BDS contenant M (noté $sup_R^{BDS}(M)$). Le **support universel** de M dans BDS est le nombre d'occurrences de M dans BDS (noté $sup_U^{BDS}(M)$).

Par exemple le support du motif séquentiel $\langle\{e\}\rangle$ dans la base de séquences 2.4 est 3, car il est inclus dans les séquences #1, #2 et #3. Son support relatif est de 75% tandis que son support universel est de 4 (il apparaît deux fois dans la séquence #2).

Definition 2.2.7. Un motif M est dit **fréquent** si et seulement si $sup_A^{BDS}(M) \geq min\_sup$, où min_sup est le support minimum.

La fouille de motifs séquentiels fréquents est un problème d'énumération. L'objectif est d'énumérer tous les motifs ayant un support supérieur ou égal au support minimum (min_sup). C'est donc un problème qui a toujours une seule bonne réponse. Il n'en demeure pas moins complexe, car pour une séquence contenant k items, on peut dénombrer jusqu'à $2^k - 1$ sous-séquences distinctes. En raison du temps et de la mémoire nécessaires à l'extraction de l'ensemble complet des motifs séquentiels fréquents dans les bases de séquences utilisées pour les applications réelles, de nombreux algorithmes ont été proposés pour améliorer la fouille de motifs séquentiels.
La très grande majorité des algorithmes de fouille de motifs séquentiels fréquents a en paramètres d'entrée la base de séquences à fouiller et le support minimum. Ils diffèrent uniquement dans **leur façon de découvrir les motifs séquentiels fréquents et de les représenter**. Il existe deux grandes catégories de méthodes de découverte des motifs séquentiels fréquents dans les bases de séquences : **la recherche en largeur** (*breadth-first search*) et **la recherche en profondeur** (*depth-first search*). Le nom de ces deux types de recherche vient de la façon d'explorer l'arbre des motifs (illustrée par la figure 2.3). Les trois types de représentation des données vont souvent de pair avec le type de recherche : **la représentation horizontale**, **la représentation verticale** et **la représentation en motifs croissants** (*pattern-growth*). Ainsi, la représentation horizontale est synonyme de recherche en largeur tandis que les deux autres introduisent une recherche en profondeur. Avant de présenter en détail ces méthodes de recherche, ces représentations et les algorithmes qui les utilisent, nous introduisons quelques notions propres à la découverte des motifs séquentiels. Les algorithmes de la littérature étendent les motifs de deux façons pour découvrir l'ensemble des motifs séquentiels fréquents de la base. Il s'agit des **s-extensions** (s pour séquence) et des **i-extensions** (i pour item). Ces extensions sont utilisées pour découvrir les motifs de taille $k+1$ ayant pour préfixe un motif de taille k.

Definition 2.2.8. Soit M un motif séquentiel $\langle E_1, ..., E_n \rangle$ et e un item. Une **s-extension** de M par e est définie comme : $M \diamond_s \langle e \rangle = \langle E_1, ..., E_n, \{e\} \rangle$. Une **i-extension** de M par e est définie comme : $M \diamond_i \langle \{e\} \rangle = \langle E_1, ..., E_n \cup \{e\} \rangle$. On dit que M est un préfixe et e un suffixe des motifs séquentiels $M \diamond_s \langle \{e\} \rangle$ et $M \diamond_i \langle \{e\} \rangle$. Lorsque les deux extensions peuvent être utilisées, nous indiquons simplement le symbole $\diamond$.

Comme dit précédemment, dans la littérature, les items sont classiquement ordonnés alphabétiquement au sein d'un même itemset. Cette nécessité d'ordonner vient de la façon d'étendre les motifs, qui se fait toujours dans un seul sens, le sens de la séquence. Lors des i-extensions, si les items ne sont pas ordonnés, certains motifs peuvent être sous-évalués ou même manqués.

Nous allons maintenant présenter les deux catégories de méthodes pour la découverte des motifs séquentiels fréquents en précisant à chaque fois le type de représentation des données des algorithmes qui les utilisent. Nous utilisons la base de séquences BDS (tableau 2.4) avec un $min\_sup = 2$ pour illustrer les différentes recherches et représentations.

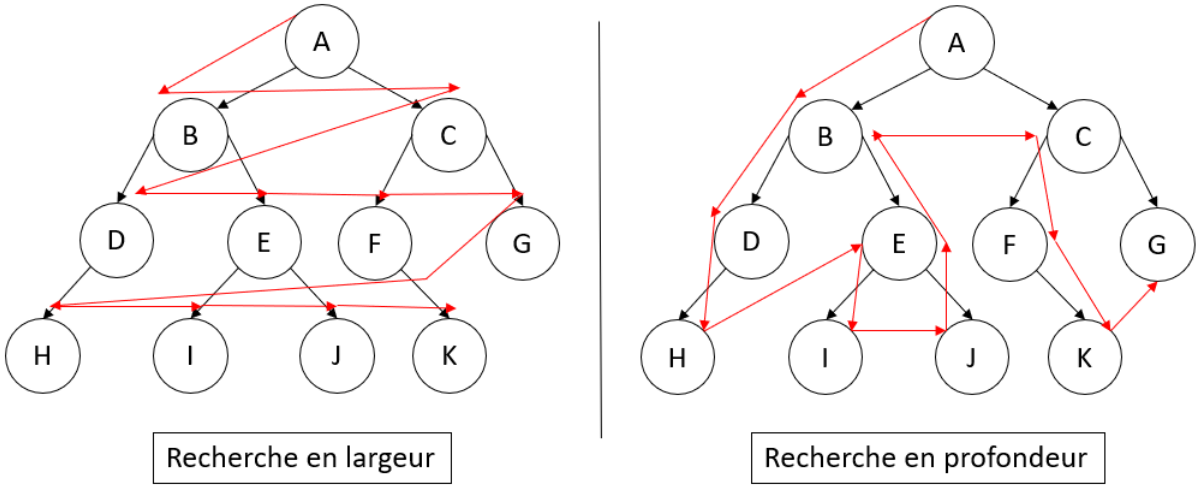


FIGURE 2.3 – Types de recherche

2.2.1 Recherche en largeur avec représentation horizontale

La recherche de motifs séquentiels en largeur a été introduite avec l'algorithme AprioriAll [Agrawal and Srikant, 1995], cette recherche est scindée en 4 étapes :

1. **Initialisation :** L'objectif de cette phase est d'identifier les items (motifs de taille $k = 1$) fréquents de la base de séquences. Un passage complet sur la base est donc effectué pour évaluer le support de chaque item et ne garder que ceux ayant un support supérieur ou égal au support minimum.
   Dans BDS, nous trouvons l'ensemble suivant d'items fréquents : $\{a : 3, b : 4, c : 2, d : 3, e : 3, f : 4\}$.
2. **Génération des candidats :** L'objectif de cette phase est d'identifier les motifs séquentiels de taille $k + 1$ candidats pour être fréquents, à partir de l'ensemble des motifs séquentiels fréquents de taille k. Cette liste de motifs candidats est créée à partir des motifs séquentiels fréquents de taille k dont la jointure est un motif de taille de $k + 1$. L'opération de jointure se fait avec des s-extensions et des i-extensions entre les motifs.

Par exemple, dans BDS avec les motifs de taille 1 : $\langle\{a\}\rangle$ et $\langle\{b\}\rangle$. Nous avons l'ensemble suivant de candidats de taille 2 à générer : $\langle\{a\},\{a\}\rangle$, $\langle\{a\},\{b\}\rangle$, $\langle\{a,a\}\rangle$, $\langle\{a,b\}\rangle$, $\langle\{b\},\{b\}\rangle$, $\langle\{b\},\{a\}\rangle$, $\langle\{b,b\}\rangle$, $\langle\{b,a\}\rangle$. Pour avoir la liste complète des candidats de taille 2 de BDS, il faut appliquer le même principe avec tous les motifs fréquents de taille 1.

3. **Évaluation du support des candidats :** L'objectif de cette phase est de conserver les motifs séquentiels fréquents parmi la liste des candidats. Cette phase se déroule en un passage complet sur la base. À chaque fois qu'un motif séquentiel de la liste de candidats est rencontré dans une séquence, son support est augmenté de 1. Une fois le passage terminé, uniquement les candidats fréquents sont conservés.
   Si nous évaluons les supports de notre liste de candidats de taille 2 : $\langle\{a\},\{a\}\rangle$:0, $\langle\{a\},\{b\}\rangle$:0, $\langle\{a,a\}\rangle$:0, $\langle\{a,b\}\rangle$:0,$\langle\{b\},\{b\}\rangle$:1, $\langle\{b\},\{a\}\rangle$:3, $\langle\{b,b\}\rangle$:0,$\langle\{b,a\}\rangle$:1.
4. **Mise à jour des motifs séquentiels fréquents :** Lors de cette phase, l'ensemble des motifs séquentiels fréquents est actualisé avec les candidats fréquents. La phase de génération des candidats est relancée en incrémentant la taille des motifs de 1. L'itération s'arrête lorsque la liste des candidats fréquents est vide.
   Dans notre exemple, nous ajoutons seulement le motif séquentiel $\langle\{b\},\{a\}\rangle$ à notre ensemble, car c'est le seul motif séquentiel fréquent issu des motifs $\langle\{a\}\rangle$ et $\langle\{b\}\rangle$. Au final, l'ensemble des motifs séquentiels fréquents de BDS est le suivant :
   $\langle\{a\}\rangle, \langle\{b\}\rangle, \langle\{b\},\{a\}\rangle, \langle\{b\},\{d\}\rangle, \langle\{b\},\{d\},\{a\}\rangle, \langle\{b\},\{f\}\rangle, \langle\{b\},\{f\},\{a\}\rangle, \langle\{b\},\{f,d\}\rangle,$
   $\langle\{b\},\{f,d\},\{a\}\rangle, \langle\{c\}\rangle, \langle\{c\},\{a\}\rangle, \langle\{c\},\{f\}\rangle, \langle\{d\}\rangle, \langle\{d\},\{a\}\rangle, \langle\{e\}\rangle, \langle\{e\},\{a\}\rangle, \langle\{e,b\}\rangle,$
   $\langle\{e\},\{b\}\rangle, \langle\{e\},\{b\},\{a\}\rangle, \langle\{e\},\{b\},\{f\}\rangle, \langle\{e\},\{c\}\rangle, \langle\{e\},\{c\},\{a\}\rangle, \langle\{e\},\{c\},\{f\}\rangle,$
   $\langle\{e\},\{d\}\rangle, \langle\{e\},\{d\},\{a\}\rangle, \langle\{e\},\{f\}\rangle, \langle\{e\},\{f\},\{a\}\rangle, \langle\{e\},\{f,d\}\rangle, \langle\{e\},\{f,d\},\{a\}\rangle,$
   $\langle\{f\}\rangle, \langle\{f\},\{a\}\rangle, \langle\{f,d\}\rangle, \langle\{f,d\},\{a\}\rangle$

Nous remarquons que l'ensemble des motifs séquentiels fréquents de la base BDS contient 33 motifs séquentiels. Pourtant, la base BDS ne contient que 4 séquences avec une longueur moyenne de 5,5. Il est donc facile d'imaginer que **l'ensemble des motifs séquentiels fréquents puisse être impossible à appréhender pour un utilisateur sur des bases de séquences plus conséquentes.**

L'algorithme AprioriAll est le premier algorithme de fouille de motifs séquentiels fréquents. Il est basé sur l'algorithme Apriori, qui est le premier algorithme de fouille d'itemsets fréquents. AprioriAll utilise une **représentation horizontale des données** : à chaque séquence est associée une liste de motifs séquentiels, comme dans notre exemple de base de séquences 2.4. L'algorithme le plus connu, utilisant cette méthode et cette représentation, est l'algorithme GSP [Srikant and Agrawal, 1996]. Il s'appuie sur l'algorithme AprioriAll en ajoutant la possibilité d'intégrer plusieurs contraintes : (i) de taxonomie ; (ii) de temps entre les itemsets ; (iii) de fenêtre glissante. En pratique, AprioriAll, GSP et plus globalement, les algorithmes de recherche en largeur souffrent de plusieurs défauts :

— **Passages sur la base trop nombreux.** L'un des plus gros problèmes est l'obligation de faire un passage complet sur la base pour chaque taille de motifs. C'est extrêmement coûteux sur les grandes bases de séquences où l'on trouve de grands motifs.
— **Génération de candidats fictifs.** La particularité de cette phase est la génération de candidats sans prendre en compte la base de séquences fouillée. En conséquence, un grand nombre de candidats non présents dans la base sont créés, utilisant ainsi inutilement du temps et de la mémoire.
— **Poids des candidats en mémoire.** Le dernier défaut vient de la conservation de tous les motifs de tailles k en mémoire pour générer les motifs de taille $k+1$. Ce qui engendre

un gros coût en mémoire.

Ces défauts expliquent pourquoi les algorithmes de fouille de motifs séquentiels avec recherche en largeur sont actuellement peu utilisés. Afin de remédier à certains de ces défauts, des algorithmes utilisant d'autres méthodes de découverte et de représentation des motifs séquentiels fréquents sont apparus.

2.2.2 Recherche en profondeur avec représentation verticale

Le processus de la recherche en profondeur est très proche du processus de la recherche en largeur. Nous indiquons maintenant les étapes de ce processus sans les détailler, car leur déroulement varie énormément en fonction de la représentation de l'algorithme :

1. **Mise en place de la représentation des données :** L'objectif de cette phase est de mettre en place la représentation des données "en profondeur" qui va conditionner le déroulement du reste de la recherche. C'est cette étape qui différencie principalement la recherche en profondeur de la recherche en largeur, puisque cette dernière utilise toujours la même représentation que la base de séquences en entrée (horizontale).
2. **Génération des candidats :** L'objectif de cette phase est de s'appuyer sur la représentation pour générer les motifs candidats dans la même représentation.
3. **Évaluation du support des candidats :** L'objectif de cette phase est de s'appuyer sur la représentation des motifs candidats pour évaluer leur support.
4. **Mise à jour des motifs séquentiels fréquents :** Lors de cette phase, l'ensemble des motifs séquentiels fréquents est actualisé avec les candidats fréquents. La phase de génération des candidats est relancée en incrémentant la taille des motifs de 1. L'itération s'arrête lorsque la liste des candidats fréquents est vide.

La recherche de motifs séquentiels en profondeur a été introduite avec l'algorithme SPADE [Zaki, 2001]. Tout comme AprioriAll avec Apriori, SPADE se base sur l'algorithme de fouille d'itemsets Eclat [Zaki, 2000a], qui est le premier algorithme à avoir utilisé une **représentation verticale des données** et donc **la recherche en profondeur**. Dans la représentation verticale, chaque item est associé à la liste des séquences et des itemsets dans lesquels il apparaît. Pour chaque item fréquent de la base de séquences, une $IDList$ est créée. Cette $IDList$ contient toutes les "coordonnées" d'apparition de l'item, c'est-à-dire le numéro de la séquence et de l'itemset dans lesquels il est présent. Ainsi, avec l'item b de la base de notre exemple nous obtenons : $IDList(\{b\}) = \{(1,1),(2,3),(2,4),(3,2),(4,1)\}$. Cette $IDList$ nous indique que b apparaît dans la séquence #1 et dans le premier itemset de cette séquence, mais également dans le troisième et quatrième itemset de la séquence #2... Formellement, l'$IDList$ d'un item ou d'une séquence est définie de la façon suivante :

Definition 2.2.9. Soit S et S' deux séquences $\langle E_1, ..., E_n\rangle, \langle E'_1, ..., E'_m\rangle$ deux séquences telles que $S \sqsubseteq S'$. La position des items de S dans S' notée $pi(S; S')$ est l'ensemble des couples (S, j_n) où j_n est un entier tel que $1 \leq j_1 < j_2 < ... < j_n \leq m$ tels que $E_1 \subseteq E'_{j_1}, E_2 \subseteq E'_{j_2}, ..., E_n \subseteq E'_{j_n}$. L'$IDList$ de S est l'union des $pi(S; S_k)$ pour chaque S_k dans la base de séquences.

Le tableau 2.5 est la représentation verticale de la base de séquences BDS. Comme cette représentation est coûteuse en mémoire, SPADE sépare l'espace de recherche en sous-espaces pouvant être explorés indépendamment les uns des autres. Cette méthode permet d'utiliser la mémoire uniquement pour le sous-espace en cours. La génération des candidats est très simplifiée par la représentation verticale. En effet, il suffit d'opérer une jointure entre l'$IDList$ du

a

sid	itemsets
#1	5
#2	4
#3	4

b

sid	itemsets
#1	1
#2	3,4
#3	2
#4	1

c

sid	itemsets
#1	2
#2	2

d

sid	itemsets
#1	3,4
#3	3
#4	2

e

sid	itemsets
#1	1
#2	1,4
#3	1

f

sid	itemsets
#1	3
#2	4
#3	3
#4	2

TABLE 2.5 – Représentation verticale de la base 2.4

préfixe en cours et celle de l'extension (s ou i) sans refaire de passage sur la base de séquences. Ainsi, pour obtenir l'$IDList$ du motif séquentiel $\langle\{b\},\{a\}\rangle$ il suffit de joindre $IDList(\{b\}) = \{(1,1),(2,3),(2,4),(3,2),(4,1)\}$ et $IDList(\{a\}) = \{(1,5),(2,4),(3,5)\}$ pour obtenir $IDList(\{b\},\{a\}) = \{(1,5),(2,4),(3,5)\}$. Ensuite pour la phase de l'évaluation du support des candidats, il suffit de calculer le nombre de séquences distinctes dans l'$IDList$. Par exemple, dans $IDList(\{b\}) = \{(1,1),(2,3),(2,4),(3,2),(4,1)\}$, il y a 4 séquences distinctes 1,2,3 et 4 donc $sup_A(\{b\}) = 4$.

Au final, la recherche en profondeur avec représentation verticale ne nécessite pas de passages supplémentaires sur la base de séquences lors de la génération des candidats, contrairement à la recherche en largeur. Par contre, tout comme avec la recherche en largeur, la génération de candidats ne garantit pas leur existence dans la base. On retrouve donc les défauts de la génération de candidats fictifs. Le poids des candidats en mémoire est diminué par la séparation de l'espace de recherche utilisé dans l'algorithme SPADE.

Un autre exemple d'algorithme de fouille en profondeur avec représentation verticale est l'algorithme SPAM [Ayres et al., 2002]. Cet algorithme a introduit **l'encodage de l'$IDList$ en vecteur de bits**. Ce changement a pour but de diminuer le temps de jointure entre $IDList$ qui est l'opération la plus coûteuse en temps lors de l'exécution de l'algorithme SPADE. L'encodage en vecteur de bits fonctionne de la manière suivante : soit n le nombre d'itemsets dans la base de séquences entière, chaque item est représenté par un vecteur de n chiffres. Ces chiffres étant des 0 ou 1 en fonction de la présence ou non de l'item dans l'itemset correspondant à chacune des séquences. Par exemple l'item $\{a\}$ de notre base BDS peut être encodé en **000010001000100** comme montré sur la figure 2.4. Le tableau 2.6 représente la verticalisation de la base BDS

sid	séquence
#1	$\langle\{e,b\},\{c\},\{f,d\},\{d\},\{a\}\rangle$
#2	$\langle\{e\},\{c\},\{b\},\{e,b,a,f\}\rangle$
#3	$\langle\{e\},\{b\},\{f,d\},\{a\}\rangle$
#4	$\langle\{b\},\{f,d\}\rangle$

FIGURE 2.4 – Item a encodé en vecteur de bits

encodée en vecteur de bits. L'utilisation de vecteurs de bits devient très pertinente pour la gé-

item	vecteur de bits
a	000010001000100
b	100000011010010
c	010000100000000
d	001100000001001
e	100001001100000
f	001000001001001

TABLE 2.6 – Représentation en vecteur de bits de la verticalisation de Base de séquences BDS

nération des candidats. En effet, l'opération de jointure devient une simple comparaison entre deux vecteurs de chiffres. Ainsi, SPAM a été évalué comme étant un ordre de grandeur plus rapide que SPADE [Ayres et al., 2002]. Cependant, l'algorithme SPAM consomme beaucoup plus de mémoire que l'algorithme SPADE, certainement dû à l'absence de séparation de l'espace de recherche comme dans SPADE.
C'est en partant de ce défaut que l'algorithme bitSPADE [Aseervatham et al., 2006] a été proposé. bitSPADE reprend l'encodage en vecteurs de bits utilisé dans SPAM, et la stratégie de séparation de l'espace de recherche de SPADE. Cet algorithme a montré qu'il était un ordre de grandeur plus rapide que SPADE tout en consommant un ordre de grandeur moins de mémoire que SPAM [Aseervatham et al., 2006]. Le défaut de la génération de candidats fictifs étant toujours présent, les algorithmes CM-SPADE et CM-SPAM [Fournier-Viger et al., 2014a] ont introduit une **matrice de cooccurrence** (CMAP) pour diminuer l'impact de cette génération de candidats fictifs. Pour ce faire, [Fournier-Viger et al., 2014a] a introduit un élagage en se reposant sur cette matrice qui stocke toutes les 2-séquences fréquentes de la base. Ainsi, dès qu'un motif est considéré lors de la génération des candidats, il suffit de vérifier si les 2 derniers items du motif sont dans la matrice CMAP. Si ce n'est pas le cas, le motif peut directement être retiré sans opérer la jointure des $IDLists$. Il a été montré que CM-SPAM et CM-SPADE sont plus rapides que GSP, SPAM et bitSPADE par un ordre de grandeur [Fournier-Viger et al., 2014a].

2.2.3 Recherche en profondeur avec représentation en motifs croissants

La représentation verticale n'est pas la seule représentation dans la recherche en profondeur, FreeSpan [Han et al., 2000a] est le premier algorithme de fouille de motifs séquentiels à avoir utilisé **la représentation en motifs croissants** (*pattern-growth*) qui ne souffre pas de la génération de candidats fictifs. Cette représentation repose sur la notion de base projetée [Han et al., 2000b, Pei et al., 2001, Pei et al., 2004].
Soit BDS une base de séquences, S une séquence $\langle E'_1, E'_2, ..., E'_m \rangle$ de BDS et M un motif séquentiel $\langle E_1, E_2, ..., E_n \rangle$ de BDS tel que $M \sqsubseteq S$. Il existe donc des entiers consécutifs $j_1, j_2, ..., j_n$ tels que : $1 \leq j_1 < j_2 < ... < j_n \leq m$ tels que $E_1 \subseteq E'_{j_1}, E_2 \subseteq E'_{j_2}, ..., E_n \subseteq E'_{j_n}$.

> **Definition 2.2.10. L'ensemble des items projetés de M dans S** est l'ensemble des i-extensions e_i de M telles que $E_n \diamond_i e_i \subseteq E'_{j_n}$ et si $m > n$ combiné à l'ensemble des items du motif séquentiel $\langle E_{j_n+1}, ..., E_m \rangle$.

> **Definition 2.2.11.** L'ensemble des items projetés de M dans S est appelé **la séquence projetée de M dans S**.

Definition 2.2.12. M est appelé **le préfixe**.

Definition 2.2.13. L'ensemble des séquences projetées du préfixe M dans BDS est appelé **la base projetée de M dans BDS**, notée M_BDS.

Finalement, la base projetée d'un motif est l'ensemble des suffixes de la première occurrence du motif de chaque séquence où il apparaît. Nous illustrons cette notion dans le tableau 2.7 avec la base projetée de l'item b dans la base BDS, le processus est mis en évidence sur la figure 2.5. Lorsque le suffixe commence au sein d'un même itemset (i-extension), il est précédé du symbole "_".

sid	séquences
#1	$\langle\{c\},\{f,d\},\{d\},\{a\}\rangle$
#2	$\langle\{e,b,a,f\}\rangle$
#3	$\langle\{f,d\},\{a\}\rangle$
#4	$\langle\{f,d\}\rangle$

TABLE 2.7 – Base projetée de b dans BDS

sid	séquence
#1	$\langle\{\sout{e,b}\},\{c\},\{f,d\},\{d\},\{a\}\rangle$
#2	$\langle\{\sout{e}\},\{\sout{e}\},\{\sout{b}\},\{e,b,a,f\}\rangle$
#3	$\langle\{\sout{e}\},\{\sout{b}\},\{f,d\},\{a\}\rangle$
#4	$\langle\{\sout{b}\},\{f,d\}\rangle$

FIGURE 2.5 – Processus de construction de la base projetée de b

Pour la phase de génération de candidats, FreeSpan fouille chacune des bases projetées qui viennent d'être créées afin de trouver les items candidats à l'extension (s ou i-extension) du préfixe de la base projetée. Les candidats sont donc bien présents dans la base de séquences, contrairement aux autres représentations. La phase d'évaluation du support des candidats consiste simplement à calculer les supports des candidats dans la base projetée pour déterminer s'ils sont fréquents ou non. Par exemple, pour étendre l'item b nous évaluons le support des items de sa base projetée (tableau 2.7) : $\{a:3, b:1, c:1, d:3, e:1, f:4\}$. Les items a et d sont fréquents dans la base projetée, b peut donc être étendu par ces deux items pour former les motifs séquentiels $\langle\{b\},\{a\}\rangle, \langle\{b\},\{d\}\rangle$.
Le processus est répété jusqu'à découvrir récursivement l'ensemble des motifs séquentiels fréquents.
L'avantage de la représentation en motifs croissants vient de son exploration des motifs uniquement présents dans la base de séquences évitant ainsi la génération de candidats fictifs comme avec les autres représentations. Cependant, le principal défaut de cette représentation vient des passages successifs nécessaires pour créer les bases projetées, ce qui est coûteux en temps et en mémoire.
L'algorithme PrefixSpan [Han et al., 2001] a introduit une optimisation des bases projetées avec les pseudo bases projetées afin de réduire ce coût. Contrairement aux bases projetées qui recréent des bases pour chaque préfixe étudié, les pseudo bases projetées ne sont elles que des pointeurs avec les positions du préfixe dans la base de séquences. PrefixSpan [Han et al., 2001] est l'algorithme de fouille de motifs séquentiels fréquents, utilisant cette représentation, le plus performant

en temps d'exécution et en consommation mémoire. Il a été montré que PrefixSpan est à la fois plus rapide et moins coûteux en mémoire que les algorithmes SPADE et GSP [Han et al., 2001]. Plus généralement, l'utilisation de la représentation en motifs croissants accompagnée des pseudo bases projetées permet une consommation inférieure de mémoire que l'utilisation de la représentation verticale.

Nous avons présenté trois grands types d'algorithmes de fouille de motifs séquentiels :

— La recherche en largeur avec représentation horizontale générant des candidats (AprioriAll et GSP)
— La recherche en profondeur avec représentation verticale générant des candidats en utilisant les $IDLists$ ou des variations (SPADE, SPAM, bitSPADE, CM-SPADE, CM-SPAM)
— La recherche en profondeur avec représentation en motifs croissants ne générant pas de candidats fictifs (FreeSpan, PrefixSpan)

La plupart des algorithmes de fouille de motifs séquentiels appartiennent à l'un de ces types. La complexité des algorithmes de fouille de motifs séquentiels dépend donc du nombre de motifs dans l'espace de recherche, ainsi que du coût des opérations pour les générer et les dénombrer. Les algorithmes basés sur la représentation en motifs croissants ont l'avantage de ne considérer que les motifs réellement présents dans la base de séquences. Pourtant, dans les faits, le coût des passages sur la base et de la création des projections font que les derniers algorithmes CM-SPADE et CM-SPAM sont plus rapides que PrefixSpan [Fournier-Viger et al., 2014a] au prix d'une plus grande consommation de mémoire. La fouille de motifs séquentiels fréquents avec recherche en profondeur apparaît donc à la fois plus rapide et moins coûteuse en mémoire qu'une fouille avec recherche en largeur. L'utilisation de la représentation verticale avec ses optimisations (encodage en vecteurs de bits et CMAP) semble plus rapide que la représentation en motifs croissants, mais bien plus coûteuse en consommation de mémoire.
La littérature sur la fouille de motifs séquentiels n'apporte, pour le moment, pas de réponse au problème soulevé en début de section à propos de la taille de l'ensemble des motifs séquentiels fréquents lorsque la fouille est effectuée sur de grandes bases de séquences. Pourtant, un ensemble de motifs trop conséquent est très compliqué à gérer en mémoire et compromet donc l'exécution de la fouille.

Dans le cadre de ma problématique de thèse, j'ai choisi la fouille de motifs séquentiels pour rendre les résultats de la fouille de données appréhendables par l'utilisateur. Afin de pouvoir y répondre, nous devons réussir à diminuer la taille de l'ensemble des motifs séquentiels fréquents tout en conservant les motifs pertinents pour l'utilisateur et proposer une fouille plus efficiente pour permettre l'extraction de motifs à partir de grandes bases de données. Nous proposons de chercher dans la littérature les solutions proposées pour diminuer l'ensemble final de motifs séquentiels fréquents et améliorer l'efficience de la fouille.

Voici les enseignements et la problématique scientifique qui ressortent de cette partie de la revue de la littérature sur la fouille de motifs séquentiels :

- **2.2_E1** : Les algorithmes de fouille de motifs séquentiels utilisant une recherche en profondeur sont plus performants qu'avec une recherche en largeur, autant en temps d'exécution qu'en consommation de mémoire.
- **2.2_E2** : Les algorithmes utilisant une représentation verticale sont les plus rapides, mais plus coûteux en mémoire, contrairement aux algorithmes utilisant une représentation en motifs croissants, qui sont moins rapides, mais bien moins coûteux en mémoire.
- **2.2_E3** : La fouille de motifs séquentiels fréquents est problématique lorsque l'ensemble des motifs extrait est conséquent. D'une part à cause de l'incapacité à le gérer en mémoire. D'autre part pour son exploitation par l'utilisateur.
- **2.2_PS1** : Comment réduire l'ensemble des motifs séquentiels fréquents extraits lorsque cet ensemble est conséquent et améliorer l'efficience de la fouille de motifs ?

2.3 Motifs séquentiels avec contraintes

Dans la section précédente, nous avons présenté les avancées du domaine de la fouille de motifs séquentiels fréquents. Pourtant, dans de nombreux cas, l'extraction de l'ensemble des motifs séquentiels fréquents pose un problème à la fois en termes d'appréhendabilité et en termes d'efficience.
D'un côté, nous avons montré que l'ensemble des motifs séquentiels fréquents dans une grande base de données peut être très conséquent. Comme l'utilisateur n'est souvent intéressé que par une petite partie de cet ensemble, l'obliger à trouver lui-même cette partie dans un ensemble si important rend sa tâche extrêmement complexe, voire impossible. Ce qui pose donc la question de l'appréhendabilité des résultats de la fouille : Comment n'extraire que les motifs qui intéressent l'utilisateur ?
D'un autre côté, malgré les nombreuses optimisations apportées par les derniers algorithmes, l'extraction de l'ensemble des motifs séquentiels fréquents d'une grande base de données demeure une tâche très coûteuse en temps et en mémoire. La possibilité de concentrer la fouille uniquement sur les motifs qui intéressent l'utilisateur permettrait d'économiser les ressources jusqu'à lors utilisées par les motifs qui ne l'intéressent pas. Ce qui crée une opportunité d'amélioration de l'efficience de la fouille : Comment améliorer l'efficience de la fouille de motifs séquentiels fréquents en se focalisant sur les motifs intéressants pour l'utilisateur ?
Afin de répondre à ces deux questions concernant l'appréhendabilité et l'efficience, la littérature propose d'introduire des contraintes dans le processus de la fouille. En effet, l'utilisation de contraintes pendant la fouille permet à la fois de réduire le nombre de motifs séquentiels générés et extraits et donc de réduire par la même occasion le temps d'exécution et le coût en mémoire. Il existe d'autres moyens de réduire le nombre de motifs extraits [Hahsler and Karpienko, 2017] mais ces méthodes interviennent une fois l'ensemble final extrait, elles ne permettent donc pas d'améliorer l'efficience de la fouille. C'est pourquoi je choisis l'intégration de contraintes pour améliorer à la fois l'efficience de la fouille et l'appréhendabilité de ses résultats.

L'enjeu de cette section est d'identifier les contraintes présentes dans la littérature pour évaluer leur impact sur l'ensemble des motifs extraits et leur intégration dans le processus de fouille. Le but étant d'être à même de proposer à l'utilisateur les contraintes les plus intéressantes en termes d'appréhendabilité et d'efficience tout en étant utilisables sur ses données qui peuvent être bruitées.
J'ai regroupé les contraintes de la littérature en deux catégories que j'ai introduite afin de faciliter leur comparaison : les **contraintes globales** portant sur les caractéristiques d'un motif par rapport à d'autres motifs séquentiels de l'ensemble et les **contraintes locales** portant sur les caractéristiques propres d'un motif.

2.3.1 Contraintes globales

Les contraintes que je considère comme globales s'appliquent et s'évaluent par rapport à d'autres motifs séquentiels fréquents et non sur chaque motif pris individuellement. Ces contraintes permettent d'extraire un sous-ensemble de l'ensemble des motifs séquentiels fréquents, mais au prix d'une partie de l'information. En fonction de la contrainte, l'information perdue est plus ou moins importante et plus ou moins préjudiciable, en fonction des applications.
La très populaire contrainte de clôture permet d'extraire l'ensemble des **motifs séquentiels clos** [Pei et al., 2000, Zaki and Hsiao, 2002, Yan et al., 2003, Li and Wang, 2008]
[Gomariz et al., 2013, Fournier-Viger et al., 2014a, Le et al., 2017]

[Rodríguez-González et al., 2017], ce sont les motifs qui ne sont inclus dans aucun autre motif ayant le même support. Cette contrainte est certainement la plus utilisée, car c'est celle qui occasionne le moins de perte d'information. En effet, la partie de l'information perdue se résume à la composition des sous-motifs inclus dans les motifs séquentiels clos.
La contrainte de maximisation, elle, permet d'extraire les **motifs séquentiels maximaux** [Luo and Chung, 2005, García-Hernández et al., 2006, Fournier-Viger et al., 2014d], qui sont les motifs fréquents inclus dans aucun autre motif fréquent. On trouve également la contrainte de génération pour extraire les **motifs séquentiels générateurs** [Gao et al., 2008, Lo et al., 2008, Yi et al., 2011, Pham et al., 2012, Fournier-Viger et al., 2014c], qui sont les motifs n'ayant aucun sous-motif de même support.
Nous allons maintenant présenter ces trois types de motifs séquentiels à contraintes globales, mais nous concentrerons notre analyse sur les motifs séquentiels clos. Nous utilisons la base de séquences du tableau 2.4 avec un support minimum de 2 pour illustrer ces motifs dans le tableau 2.8.

Definition 2.3.1. Soit MF l'ensemble des motifs séquentiels fréquents, l'ensemble MM **des motifs séquentiels maximaux** est défini comme suit : $MM = \{M | M \in MF \wedge \nexists M' \in MF$ tel que $M \sqsubset M'\}$

La fouille de motifs séquentiels maximaux consiste donc à fouiller les motifs séquentiels fréquents qui ne sont inclus dans aucun autre motif. De nombreux algorithmes permettent d'extraire les motifs séquentiels maximaux, certains utilisent la recherche en largeur comme AprioriAdjust [Lu and Li, 2004], d'autres la recherche en profondeur avec représentation verticale comme VMSP [Fournier-Viger et al., 2014d] et également avec représentation en motifs croissants comme MaxSP [Fournier-Viger et al., 2013]. La stratégie de VMSP pour extraire les motifs séquentiels maximaux repose sur un ensemble de motifs Z qui évolue tout au long de la fouille. Lorsqu'un nouveau motif séquentiel fréquent est extrait, VMSP vérifie si ce motif est inclus ou s'il contient un motif de l'ensemble de Z. En fonction du résultat de cette inclusion, le motif le plus grand est conservé dans l'ensemble Z. Une fois que tous les motifs séquentiels fréquents ont été extraits, l'ensemble Z contient tous les motifs séquentiels fréquents maximaux. Dans l'algorithme MaxSP, la stratégie de vérification de la maximisation d'un motif se fait à la volée lorsqu'un nouveau préfixe est considéré. MaxSP vérifie alors la présence d'extensions fréquentes à gauche et à droite du préfixe dans toutes les séquences où il apparaît. Si une telle extension existe, alors le motif n'est pas maximal.
La fouille de motifs séquentiels maximaux est utilisée dans la recherche des plus grands motifs fréquents dans un corpus de textes [García-Hernández et al., 2006], dans l'analyse des séquences ADN [Bai and Bai, 2009], etc. Le tableau 2.8 montre que parmi les 33 motifs séquentiels fréquents de la base BDS seulement 7 sont des motifs séquentiels maximaux.
Au final, l'ensemble des motifs séquentiels maximaux est une représentation très compacte, c'est-à-dire réduite, de l'ensemble des motifs séquentiels fréquents pouvant être plusieurs ordres de grandeur plus petit [Fournier-Viger et al., 2014d]. L'ensemble des motifs séquentiels maximaux perd toute l'information sur les motifs inclus dans d'autres motifs. Le plus problématique dans cette perte d'information pour l'utilisateur vient de la perte du support de tous les motifs non conservés, car le support est un moyen de mesurer l'importance des motifs dans la base de séquences.

Definition 2.3.2. Soit MF l'ensemble des motifs séquentiels fréquents, l'ensemble MG **des motifs séquentiels générateurs** est défini comme suit : $MG = \{M | M \in MF \wedge \nexists M' \in MF$ tel que $M' \sqsubset M \wedge sup(M) = sup(M')\}$

motif	maximal ?	générateur ?	clos ?
$\langle\{a\}\rangle$		✓	
$\langle\{b\}\rangle$		✓	
$\langle\{b\},\{a\}\rangle$			✓
$\langle\{b\},\{d\}\rangle$			✓
$\langle\{b\},\{d\},\{a\}\rangle$			✓
$\langle\{b\},\{f\}\rangle$			✓
$\langle\{b\},\{f\},\{a\}\rangle$			
$\langle\{b\},\{f,d\}\rangle$			✓
$\langle\{b\},\{f,d\},\{a\}\rangle$	✓		✓
$\langle\{c\}\rangle$		✓	
$\langle\{c\},\{a\}\rangle$			
$\langle\{c\},\{f\}\rangle$			
$\langle\{d\}\rangle$		✓	
$\langle\{d\},\{a\}\rangle$		✓	
$\langle\{e\}\rangle$		✓	
$\langle\{e\},\{a\}\rangle$		✓	✓
$\langle\{e,b\}\rangle$	✓		✓
$\langle\{e\},\{b\}\rangle$		✓	
$\langle\{e\},\{b\},\{f\}\rangle$	✓		✓
$\langle\{e\},\{b\},\{a\}\rangle$	✓		✓
$\langle\{e\},\{c\}\rangle$		✓	
$\langle\{e\},\{c\},\{a\}\rangle$	✓		✓
$\langle\{e\},\{c\},\{f\}\rangle$	✓		✓
$\langle\{e\},\{d\}\rangle$		✓	
$\langle\{e\},\{d\},\{a\}\rangle$			
$\langle\{e\},\{f\}\rangle$			✓
$\langle\{e\},\{f\},\{a\}\rangle$			
$\langle\{e\},\{f,d\}\rangle$			
$\langle\{e\},\{f,d\},\{a\}\rangle$	✓		✓
$\langle\{f\}\rangle$		✓	
$\langle\{f\},\{a\}\rangle$		✓	
$\langle\{f,d\}\rangle$		✓	
$\langle\{f,d\},\{a\}\rangle$			

TABLE 2.8 – Motifs avec contraintes globales dans BDS

De nombreux algorithmes ont été proposés pour la fouille de motifs séquentiels générateurs. La plupart de ces algorithmes utilisent la représentation en motifs croissants comme GenMiner [Lo et al., 2008], FEAT [Gao et al., 2008] et FSGP [Yi et al., 2011]. Ces algorithmes commencent par extraire les motifs séquentiels fréquents puis identifient les motifs séquentiels générateurs en évaluant le support des motifs séquentiels fréquents inclus dans ces derniers. Si aucun de ces motifs inclus n'a le même support, alors le motif peut être considéré comme générateur. L'algorithme VGEN [Fournier-Viger et al., 2014c] basé sur CM-SPAM, apparaît comme étant l'algorithme le plus rapide pour extraire des motifs séquentiels générateurs. VGEN est le premier algorithme à utiliser une représentation verticale pour extraire ces motifs. La stratégie de VGEN pour extraire les motifs séquentiels générateurs repose sur un ensemble de motifs Z qui évolue tout au long de la fouille. Lorsqu'un nouveau motif séquentiel fréquent est extrait, VGEN vérifie si ce motif contient un motif de l'ensemble de Z ayant le même support, si ce n'est pas le cas, il est ajouté à l'ensemble de Z. Dans un second temps, VGEN vérifie si des motifs de l'ensemble Z contiennent ce nouveau motif en ayant le même support, si c'est le cas, ils sont retirés de l'ensemble Z. Une fois que tous les motifs séquentiels fréquents ont été extraits, l'ensemble Z contient tous les motifs séquentiels générateurs.
Dans la base de séquences exemple BDS, il y a 13 motifs séquentiels générateurs (voir tableau 2.8).
Les motifs séquentiels générateurs et clos peuvent être combinés pour générer des règles avec un minimum d'antécédents et un maximum de conséquences [Lo et al., 2008]. Ce type de règles est très utilisé en marketing lors de l'analyse du panier de la ménagère [Lo et al., 2008], puisqu'elles permettent d'identifier les produits impactant le plus sur la vente d'autres produits. En effet, si le motif $\langle a, b\rangle$ est un motif séquentiel générateur de même support que le motif séquentiel clos $\langle a, b, c, d, e\rangle$, on peut extraire la règle $\langle a, b\rangle \rightarrow \langle c, d, e\rangle$. L'ensemble des motifs séquentiels générateurs perd l'information sur tous les super-motifs de même support qu'un de leur sous-motif, cette perte peut être préjudiciable pour l'utilisateur puisqu'il s'agit de motifs ayant la même importance, mais contenant plus d'information dans leur composition.

Definition 2.3.3. Soit MF l'ensemble des motifs séquentiels fréquents, l'ensemble MC **des motifs séquentiels clos** est défini comme suit : $MC = \{M | M \in MF \wedge \nexists M' \in MF$ tel que $M \sqsubset M' \wedge sup(M) = sup(M')\}$

La fouille de motifs séquentiels clos est un domaine très étudié dans la fouille de motifs, de nombreux algorithmes ont donc été proposés pour extraire les motifs séquentiels
clos [Yan et al., 2003, Li and Wang, 2008, Gomariz et al., 2013, Fournier-Viger et al., 2014a]
[Le et al., 2017, Rodríguez-González et al., 2017]. **Clospan** [Yan et al., 2003] est le premier algorithme de fouille de motifs séquentiels clos, il utilise une représentation en motifs croissants. Clospan se base sur PrefixSpan pour extraire les motifs séquentiels fréquents en introduisant un élagage lors de la phase de génération des candidats afin d'empêcher l'extraction de certains motifs non clos. L'idée derrière cet élagage est d'identifier les items fréquents apparaissant toujours après un autre item fréquent et de ne pas les conserver pour étendre le préfixe, car les motifs ainsi générés seront toujours inclus dans les motifs générés avec l'item précédent. Si nous prenons l'exemple de notre base de séquences BDS et du motif $\langle\{b\}\rangle$, nous observons que toutes les occurrences de l'item d apparaissent dans la base projetée du motif $\langle\{b\}\rangle$. Il sera donc inutile d'étendre le motif $\langle\{d\}\rangle$ car tous les motifs générés en l'étendant seront inclus dans les motifs étendus de $\langle\{b\}\{d\}\rangle$. La difficulté de cette méthode réside dans la détection des items apparaissant toujours avant un autre item dans la base de séquences. Clospan a introduit le théorème

suivant pour résoudre cette question :

Theorem 2.3.1 (Théorème de Clospan). *Soit BDS une base de séquences, M et M' deux motifs tels que $M \sqsubseteq M'$.*

$$M\_BDS = M'\_BDS \Leftrightarrow |M\_BDS| = |M'\_BDS|$$

Nous illustrons ce théorème 2.3.1 avec un exemple, toujours dans la base de séquences BDS. Lors de la génération de candidats à partir du préfixe $\langle\{e\}\rangle$, l'item c est un item fréquent de sa base projetée. Sachant que la représentation en motifs croissants fonctionne avec un ordre, ici l'ordre alphabétique, les motifs étendus à partir du motif $\langle\{c\}\rangle$ ont déjà été générés. Nous avons $\langle\{c\}\rangle \subseteq \langle\{e\}\{c\}\rangle$ et $|\langle\{e\}\{c\}\rangle\_BDS| = |\langle\{c\}\rangle\_BDS|$, donc $\langle\{e\}\{c\}\rangle\_BDS = \langle\{c\}\rangle\_BDS$. Ainsi il est inutile d'explorer $\langle\{e\}\{c\}\rangle\_BDS$. On dit que $\langle\{e\}\{c\}\rangle$ peut absorber $\langle\{c\}\rangle$.
La stratégie de vérification de la clôture des motifs utilisée dans Clospan se fait en post-traitement sur l'ensemble des motifs séquentiels fréquents extraits au préalable. Cette stratégie élimine simplement les motifs séquentiels non clos de l'ensemble en vérifiant les inclusions et le support de chaque motif par rapport aux autres motifs de l'ensemble.
Le principal défaut de Clospan vient de la conservation d'une grande partie des motifs séquentiels fréquents, pour finalement tester s'ils sont clos en fin d'exécution de l'algorithme. La conservation de ces candidats est coûteuse en termes de temps et de mémoire. Ce défaut n'empêche pas Clospan d'être plus rapide et moins consommateur de mémoire que PrefixSpan.
Pour résoudre ce problème, l'algorithme **BIDE** [Wang and Han, 2004] propose une méthode de vérification BI-Directionnelle de la clôture d'un motif. Cette nouvelle méthode permet d'identifier les motifs séquentiels clos au moment de leur génération, empêchant ainsi la génération de candidats non clos. BIDE se base lui aussi sur PrefixSpan pour la génération des candidats avec représentation en motifs croissants et utilise la même méthode d'élagage de l'espace de recherche que Clospan.
Le principe derrière la méthode de vérification BI-Directionnelle de la clôture d'un motif est celui des extensions-avant et des extensions-arrière d'un motif. Un item est une extension s'il apparaît dans toutes les séquences où apparaît le motif. Les extensions-avant sont les items de la base projetée ayant le même support que le préfixe. Les extensions-arrière sont les items à l'intérieur ou avant le motif toujours présents dans les séquences d'apparition du motif. La détection des extensions-arrière nécessite donc un traitement supplémentaire par rapport à Clospan mais ce traitement se révèle moins coûteux que de conserver les candidats. Un motif est clos s'il n'a aucune extension. Pour résumer, BIDE détermine à la volée si les motifs candidats générés sont clos ou non sans générer de candidats fictifs, ce qui permet à la fois d'accélérer le temps d'exécution et de diminuer la consommation mémoire par rapport à Clospan.
Plus récemment, les algorithmes **Clasp** [Gomariz et al., 2013] et
CM-Clasp [Fournier-Viger et al., 2014a] ont vu le jour. Clasp se base sur l'algorithme SPADE et une représentation verticale de données pour la génération des motifs séquentiels fréquents tout en conservant les mêmes techniques d'élagage de l'espace de recherche que Clospan. Également comme Clospan, l'algorithme Clasp génère d'abord des motifs séquentiels fréquents candidats et une fois que tous les candidats sont générés, une procédure vérifie quels sont les motifs séquentiels clos. L'algorithme Clasp a montré de meilleures performances en termes de temps d'exécution que l'algorithme Clospan certainement due à l'utilisation d'une représentation verticale pour l'extraction des motifs séquentiels fréquents. CM-Clasp est une version optimisée de Clasp, utilisant la même méthode d'optimisation que CM-SPADE et CM-SPAM : une matrice de cooccurrence pour accélérer la génération des candidats. En conséquence, CM-Clasp semble

s'exécuter plus rapidement que les autres algorithmes de fouille de motifs séquentiels clos, mais également que les autres algorithmes de fouille de motifs séquentiels fréquents de la littérature [Fournier-Viger et al., 2014a]. Cette supériorité s'explique par l'utilisation de la contrainte de clôture pour diminuer l'ensemble des motifs générés et extraits. Même si CM-Clasp apparaît être l'algorithme de fouille de motifs séquentiels clos le plus rapide de la littérature, la génération de candidats demeure un problème pour la consommation de mémoire qui devient même rédhibitoire lorsque la fouille est effectuée sur une grande base de données avec un grand nombre de motifs séquentiels fréquents. L'algorithme BIDE qui s'appuie sur une représentation en motifs croissants et sur une stratégie de vérification de la clôture à la volée, semble être l'algorithme de fouille de motifs séquentiels clos le plus efficient, car il ne génère et donc ne stocke aucun candidat. Malgré tout, la contrainte de clôture ne réduit pas suffisamment la taille de l'ensemble des motifs extraits pour qu'il puisse être maintenu en mémoire lorsque l'ensemble originel est très conséquent.
Dans la base de séquences exemple BDS, parmi les 33 motifs séquentiels fréquents, 14 sont des motifs séquentiels clos (tableau 2.8). Au final, l'ensemble des motifs séquentiels maximaux est bien le plus compact (réduction de presque 80%), et les deux autres ensembles sont de tailles similaires (réduction d'environ 60%). Malgré cette diminution, il reste encore 14 motifs séquentiels clos pour une base de séquences de taille aussi réduite, on imagine aisément que l'ensemble des motifs séquentiels clos sur de grandes bases de séquences soit trop conséquent pour être appréhendé par l'utilisateur. Pourtant, contrairement aux deux autres contraintes globales, l'information perdue par l'utilisation de la contrainte de clôture semble minime, puisqu'elle se résume uniquement à la composition et au support des sous-motifs contenus dans d'autres motifs de l'ensemble des motifs séquentiels clos.
Dans le cadre de ma problématique de thèse, qui est de proposer une fouille de motifs séquentiels dont les résultats sont appréhendables par l'utilisateur, je choisis de privilégier le sous-ensemble des motifs séquentiels clos, car c'est celui qui conserve le plus l'information de l'ensemble initial dont peut avoir besoin l'utilisateur.

En conclusion, nous avons montré que la contrainte de clôture permet à la fois de réduire l'ensemble des motifs extraits tout en conservant la quasi-totalité de l'information, mais également d'avoir une fouille plus efficiente qu'une fouille de motifs séquentiels fréquents. J'ai écarté l'utilisation des contraintes de maximisation et de génération, car elles ne me semblent pas garantir systématiquement la présence des motifs contenant l'information importante pour l'utilisateur dans l'ensemble des motifs extraits. La fouille de motifs séquentiels clos permet donc d'améliorer l'appréhendabilité des résultats, mais également l'efficience de la fouille grâce aux techniques d'élagage des motifs séquentiels non clos de l'espace de recherche et parfois de la détection des motifs clos à la volée. Cependant, la contrainte de clôture seule ne suffit pas pour extraire des motifs séquentiels de grandes bases de séquences avec de nombreux motifs séquentiels fréquents, essentiellement à cause de la consommation en mémoire.

Voici les enseignements qui ressortent de cette partie de la revue de la littérature sur la fouille de motifs séquentiels avec contraintes globales :

- **2.3.1_E1** : L'utilisation de la contrainte de clôture améliore l'efficience de la fouille et l'appréhendabilité des résultats en réduisant le nombre de motifs séquentiels générés, extraits et également le temps d'exécution tout en conservant la quasi-totalité de l'information contenue dans les motifs séquentiels fréquents.
- **2.3.1_E2** : La contrainte de clôture n'est pas suffisante pour permettre l'extraction de motifs séquentiels de grandes bases de séquences avec toujours de trop nombreux motifs séquentiels à générer et à extraire.

2.3.2 Contraintes locales

Dans cette section nous présentons les contraintes que je qualifie de locales, qui portent sur les caractéristiques propres d'un motif. L'utilisation de contraintes locales permet de concentrer la fouille sur les motifs les plus pertinents selon les besoins de l'utilisateur, ce qui permet à la fois d'améliorer l'efficience de la fouille de motifs séquentiels et l'appréhendabilité de ses résultats [Srikant and Agrawal, 1996, Mannila et al., 1997, Zaki, 2000b, Pei et al., 2007].

Definition 2.3.4. Soit M un motif séquentiel, une contrainte locale C est une fonction telle que : $C(M) \in \{Vrai, Faux\}$

Le problème de la fouille de motifs séquentiels sous contrainte C est donc le problème de l'extraction de l'ensemble des motifs séquentiels fréquents qui satisfont cette contrainte C. Dans la littérature, il existe un grand nombre de contraintes locales, pour simplifier la discussion nous présentons les catégories introduites dans [Pei et al., 2007] :

- Les **contraintes de longueur** peuvent par exemple être des contraintes sur la longueur des motifs, le nombre d'occurrences d'un item dans les motifs, etc. En analyse de séquences ADN par exemple, l'utilisateur peut être uniquement intéressé par l'extraction de motifs de plus de 20 items : $C_{lon}(M) \equiv (|M| \geq 20)$
- Les **contraintes d'item** spécifient un ensemble d'items I_C qui doivent être présents ou absents dans les motifs séquentiels extraits. Cette contrainte est définie sous la forme :
$$C_{item}(M) \equiv (\varphi i : 1 \leq i \leq |M|, M[i] \theta I_C)$$
ou
$$C_{item}(M) \equiv (\varphi i : 1 \leq i \leq |M|, M[i] \cap I_C \neq \emptyset)$$
avec $\varphi \in \{\forall, \exists\}$ et $\theta \in \{\subset, \subseteq, \supset, \supseteq, \in, \notin\}$
Les contraintes d'items peuvent par exemple être utilisées pour de l'analyse de panier de la ménagère, pour extraire uniquement les motifs contenant tel ou tel produit.
- Les **contraintes de super-motif** spécifient un ensemble de motifs M_C, dont un sous-ensemble doit être inclus dans les motifs séquentiels extraits. Cette contrainte est définie sous la forme :
$$C_{motif}(M) \equiv (\exists m \in M_C, m \sqsubseteq M)$$
Ce type de contraintes peut également être utilisé pour l'analyse marketing du panier de la ménagère, en cherchant cette fois un enchaînement d'achats de produits.
- Les **contraintes d'agrégation** sont des fonctions d'agrégation sur les items des motifs extraits comme la moyenne, la somme, le maximum, etc. Ces contraintes sont encore une fois très utiles pour l'analyse marketing du panier de la ménagère, pour extraire par exemple, les combinaisons d'achats de produits dépassant un certain montant.
- Les **contraintes d'expression régulières** spécifient une expression régulière RE. La contrainte C_{RE} est satisfaite si les motifs extraits sont en accord avec l'expression régulière

RE. Ces contraintes sont par exemple utilisées dans la fouille de logs d'utilisateur sur le Web pour extraire des motifs dans des données plus structurées [Garofalakis et al., 1999].

— Les **contraintes de durée** ne sont possibles que si les bases de séquences comportent des timestamps pour chaque transaction. Ces contraintes portent sur la différence entre le timestamp de deux itemsets du motif séquentiel (fonction Dur). Elle est définie sous la forme :

$$C_{dur}(M) \equiv Dur(M)\theta\Delta t,$$
$$\text{avec } \theta \in \{\leq, \geq\} \text{ et } \Delta t \in \mathbb{N}^*$$

— Les **contraintes de gap** portent sur la différence maximale ou minimale entre deux itemsets d'un motif séquentiel dans la base de séquences (fonction Gap). Cette contrainte est définie sous la forme :

$$C_{gap}(M) \equiv Gap(M)\theta\Delta t,$$
$$\text{avec } \theta \in \{\leq, \geq\} \text{ et } \Delta t \in \mathbb{N}^*$$

De nombreux algorithmes ont été conçus pour extraire les motifs séquentiels soumis à certains de ces types de contraintes locales. La grande majorité de ces algorithmes sont des extensions des algorithmes que nous avons déjà étudiés lors de la revue de la fouille de motifs séquentiels fréquents ou de la fouille de motifs séquentiels clos. L'intégration de ces contraintes locales permet à ces algorithmes de diminuer à la fois le nombre de motifs générés et extraits, ce qui rend ces algorithmes plus efficients que leur version originelle. Le seul inconvénient vient du fait que les contraintes locales sont très situationnelles, il est donc impossible de les intégrer à priori dans un algorithme, mais uniquement de les proposer à l'utilisateur. Nous présentons maintenant plusieurs algorithmes intégrant des contraintes locales.

L'un des tout premiers algorithmes de fouille de motifs séquentiels à intégrer des contraintes locales est GSP [Srikant and Agrawal, 1996]. GSP permet d'utiliser des contraintes de durée et de gap sur l'ensemble des motifs séquentiels qu'il extrait. Ces contraintes sont très utilisées dans la littérature, dans ces deux contributions [Hirate and Yamana, 2006, Fournier-Viger et al., 2008] par exemple, elles sont respectivement intégrées à des extensions des algorithmes avec représentation en motifs croissants : PrefixSpan et BIDE. L'algorithme Pex-SPAM [Ho et al., 2005] a lui introduit les contraintes de gap avec la représentation verticale en les intégrant à une extension de l'algorithme SPAM. Le premier algorithme à avoir intégré les contraintes d'expressions régulières est l'algorithme SPIRIT [Garofalakis et al., 1999], il convertit les contraintes en automate qui élague les motifs pendant une recherche en largeur. Une contrainte de gap très utilisée est la contrainte de contiguïté (gap de 0) [Fürnkranz, 1998, Chen and Cook, 2007], notamment pour le traitement du langage naturel ou encore la classification de documents. Plus récemment [Zhang et al., 2015], il a été montré que l'information contenue dans les motifs séquentiels contigus était presque équivalente à l'information contenue dans les motifs séquentiels fréquents. La contrainte de contiguïté est donc une contrainte locale qui peut être utilisée, quel que soit le contexte applicatif de l'utilisateur pour diminuer le nombre de motifs générés et extraits. Par conséquent, l'utilisation de la contrainte de contiguïté permet d'améliorer à la fois l'efficience de la fouille et l'appréhendabilité des résultats. Cependant, la fouille de motifs contigus a le défaut de ne pouvoir fouiller des données bruitées sans perte d'information. Dans le contexte de la fouille de motifs, le bruit se caractérise par l'apparition "aléatoire" de perturbations dans les motifs comme la perte, la substitution ou l'addition d'items. La répercussion directe est l'apparition de motifs contenant ces items, l'information contenue dans les autres items est donc diluée. En première conséquence, les motifs uniquement constitués d'items porteurs d'information n'ont pas toujours le support reflétant leur fréquence réelle. Leur support ne retranscrit alors plus leur impact dans la base de séquences et l'utilisateur perd de l'information.

La contrainte de contiguïté me semble très pertinente, car elle permet de réduire très efficacement l'ensemble des motifs extraits tout en conservant l'information. Par contre, le problème de la fouille des données bruitées doit être surmonté, car incompatible avec notre objectif de proposer une fouille de motifs à l'utilisateur, quel que soit son contexte applicatif.

Dans les études sur la fouille de motifs séquentiels sous contraintes, les contraintes locales sont caractérisées en se basant sur trois propriétés des contraintes : la monotonie [Agrawal et al., 1993], l'anti-monotonie [Bucilă et al., 2003] et la concision [Lakshmanan et al., 1999]. Ces propriétés permettent de généraliser les méthodes de fouille en regroupant plusieurs types de contraintes locales.

Definition 2.3.5. Une contrainte C_A est **anti-monotone** si lorsqu'un motif M ne satisfait pas C_A alors tout sur-motif de M ne satisfait également pas C_A.

Par exemple, la contrainte de longueur $C_{lon}(M) \equiv (|M| \geq 20)$ est anti-monotone.

Definition 2.3.6. Une contrainte C_M est **monotone** si lorsqu'un motif M satisfait C_M alors tout sur-motif de M satisfait également C_M.

La contrainte de durée, $C_{dur}(M) \equiv Dur(M) \geq 10t$, est un exemple de contrainte monotone.

Definition 2.3.7. Une contrainte C_S est **concise** si lorsqu'un motif M satisfait C_S alors tout sous-motif de M satisfait également C_S.

Cette contrainte de minimum : $C_{min}(M) \equiv (min(M) \geq 20)$ est un exemple de contrainte concise tandis que cette contrainte de somme n'est pas concise : $C_{som}(M) \equiv (somme(M) \geq 20)$.

Nous allons maintenant montrer dans le tableau 2.9 la caractérisation des contraintes introduites précédemment en termes de monotonie, anti-monotonie et concision comme spécifiées dans [Ng et al., 1998] :
L'algorithme PG [Pei et al., 2007] repose sur cette caractérisation pour introduire une nouvelle propriété appelée préfixe-monotone incluant toutes les contraintes monotones, anti-monotones et les expressions régulières. La préfixe monotonie est basée sur ces nouveaux types de contraintes :

Definition 2.3.8. Une contrainte C_{pa} est dite **préfixe anti-monotonique** si pour chaque motif M satisfaisant cette contrainte alors tous les préfixes de M la satisfont aussi.

Definition 2.3.9. Une contrainte C_{pm} est dite **préfixe monotonique** si pour chaque motif M satisfaisant cette contrainte alors tous les motifs ayant pour préfixe M la satisfont aussi.

Nous avons donc la définition suivante :

Definition 2.3.10. Une contrainte est dite **préfixe-monotone** si elle est préfixe anti-monotonique ou préfixe monotonique.

Le lemme suivant a été introduit et prouvé dans [Pei et al., 2007] :

Lemma 2.3.2. *Une contrainte anti-monotone est préfixe anti-monotonique. Une contrainte monotone est préfixe monotonique.*

Nous pouvons illustrer ce lemme avec les contraintes $C_{lon1}(M) \equiv (|M| \leq 20)$ et $C_{lon2}(M) \equiv (|M|) \geq 20)$. Nous savons que C_{len1} est anti-monotone. Nous savons donc que si M satisfait C_{lon1} alors $|M| \leq 20$, et comme tous les préfixes de M sont de taille inférieure à M et donc à 20, alors

	contraintes	**anti-monotone ?**	**monotone ?**	**concise ?**
Longueur	$C_{lon}(M) \equiv (\|M\| \leq l)$	✓		✓
	$C_{lon}(M) \equiv (\|M\| \geq l)$			
Item	$C_{item}(M) \equiv (\forall i : 1 \leq i \leq \|M\|,$ $M[i]\theta I_C), \theta \in \{\subseteq, \supseteq\}$	✓		✓
	$C_{item}(M) \equiv (\forall i : 1 \leq i \leq \|M\|,$ $M[i] \cap I_C \neq \emptyset)$	✓		✓
	$C_{item}(M) \equiv (\exists i : 1 \leq i \leq \|M\|,$ $M[i]\theta I_C), \theta \in \{\subseteq, \supseteq\}$		✓	✓
	$C_{item}(M) \equiv (\exists i : 1 \leq i \leq \|M\|,$ $M[i] \cap I_C \neq \emptyset)$		✓	✓
Super-motif	$C_{motif}(M) \equiv (\exists m \in M_C, m \sqsubseteq M)$		✓	✓
Agrégation	$max(M) \leq n, min(M) \geq n$	✓		✓
	$max(M) \geq n, min(M) \leq n$		✓	✓
	$somme(M) \leq n$	✓		
	$somme(M) \geq n$		✓	
	$moyenne(M)\ \theta\ n, \theta \in \{\leq, \geq\}$			
Exp. rég.	$C_{RE}(M)$			
Durée	$C_{dur}(M) \equiv Dur(M) \leq \Delta t,$	✓		
	$C_{dur}(M) \equiv Dur(M) \geq \Delta t,$		✓	
Gap	$C_{gap}(M) \equiv Gap(M)\theta\Delta t, (\theta \in \{\leq, \geq\})$	✓		

TABLE 2.9 – Caractérisation des contraintes locales

C_{lon1} est bien préfixe anti-monotonique. De même, nous savons que C_{lon2} est monotone. Si le motif M satisfait C_{lon2} alors $|M| \geq 20$, donc tous les motifs ayant pour préfixe M sont de taille supérieure à M et donc à 20, alors C_{lon2} est préfixe monotonique.
Ces nouvelles propriétés permettent même de caractériser les contraintes d'expression régulière puisqu'elles sont préfixe anti-monotonique. En se basant sur les caractéristiques d'une contrainte préfixe-monotone, l'algorithme PG met en place une méthode pour étendre les motifs respectant ce type de contraintes. Voici les règles d'extension d'un motif M avec C une contrainte préfixe-monotone :

1. Quand C est préfixe anti-monotonique, si $C(M) = Faux$ alors il n'existe pas de motif M' ayant M pour préfixe tel que $C(M') = Vrai$.
2. Quand C est préfixe monotonique, si $C(M) = Vrai$ alors chaque motif M' ayant M pour préfixe vérifie $C(M') = Vrai$.

Grâce à ces règles, l'algorithme PG est capable d'extraire l'ensemble des motifs séquentiels fréquents satisfaisant des contraintes préfixes-monotones en se reposant sur une représentation en motifs croissants. Un algorithme de fouille de motifs séquentiels fréquents peut donc intégrer un ensemble de contraintes préfixes-monotones en utilisant une représentation en motifs croissants et les règles de génération de candidats introduites dans l'algorithme PG.

Voici les enseignements qui ressortent de cette partie de la revue de la littérature sur la fouille de motifs séquentiels avec contraintes locales :

- **2.3.2_E1** : L'utilisation de la contrainte de contiguïté améliore significativement l'efficience de la fouille et l'appréhendabilité des résultats en réduisant le nombre de motifs séquentiels générés, extraits et également le temps d'exécution tout en conservant la quasi-totalité de l'information contenue dans les motifs séquentiels fréquents.
- **2.3.2_E2** : La contrainte de contiguïté ne permet pas de fouiller des données bruitées sans perte d'information.
- **2.3.2_E3** : Un ensemble de contraintes préfixes-monotones peut être naturellement intégré à un algorithme de fouille de motifs séquentiels utilisant une représentation en motifs croissants et les règles de génération de candidats de l'algorithme PG.

L'utilisation des contraintes globales et locales permet de diminuer le nombre de motifs séquentiels générés et extraits tout en conservant l'information importante pour l'utilisateur. En plus d'améliorer l'appréhendabilité des résultats, cette utilisation améliore également l'efficience puisque la diminution du nombre de motifs générés et extraits impacte très favorablement le temps d'exécution et la consommation de mémoire. La contrainte globale de clôture et la contrainte locale de contiguïté sont les seules contraintes utilisables sans dépendre du contexte applicatif de l'utilisateur. Une fouille de motifs séquentiels avec une représentation en motifs croissants permet d'intégrer la contrainte de clôture, ainsi qu'un ensemble de contraintes préfixes-monotones, au processus de génération des candidats. Les contraintes locales sont un moyen extrêmement efficace pour réduire l'ensemble des motifs séquentiels générés et extraits lorsque l'utilisateur a un contexte applicatif qui permet leur utilisation.

Voici les enseignements qui ressortent de cette partie de la revue de la littérature sur la fouille de motifs séquentiels avec contraintes :

- **2.3_E1** : L'utilisation des contraintes de clôture et de contiguïté améliore significativement l'efficience de la fouille et l'appréhendabilité des résultats en réduisant le nombre de motifs séquentiels générés, extraits et également le temps d'exécution tout en conservant la quasi-totalité de l'information contenue dans les motifs séquentiels fréquents.
- **2.3_E2** : La représentation en motifs croissants permet d'intégrer naturellement dans le processus de génération des motifs un ensemble de contraintes préfixes-monotones qui est un moyen efficace pour réduire les motifs séquentiels générés et extraits lorsque l'utilisateur a un contexte applicatif qui permet leur utilisation.

Afin de proposer à l'utilisateur une fouille de motifs séquentiels efficiente avec un ensemble de motifs extraits appréhendable, je choisis de me focaliser sur la combinaison des contraintes de clôture et de contiguïté. Nous allons donc maintenant faire la revue de la littérature sur la fouille de motifs séquentiels clos contigus.

2.4 Fouille de motifs séquentiels clos contigus

Nous avons déjà montré que l'ensemble des motifs séquentiels clos est plus réduit que l'ensemble des motifs séquentiels fréquents tout en contenant la même information. La littérature a montré que l'ensemble des motifs séquentiels clos contigus est plus d'un ordre de grandeur plus petit que l'ensemble des motifs séquentiels clos [Zhang et al., 2015]. La fouille de motifs séquentiels clos contigus est donc une fouille bien plus efficiente que la fouille de motifs séquentiels fréquents avec un ensemble de motifs extraits bien plus appréhendable. L'extraction de ces motifs est essentiellement utilisée pour l'analyse de séquences biologiques (ADN et protéines) [Zhang et al., 2015, Zhang et al., 2016], pour l'extraction d'information de corpus de texte [Abboud et al., 2015, Béchet et al., 2015] ou encore pour l'optimisation de trajets [Yang and Gidófalvi, 2017].
Le premier algorithme à avoir spécifiquement fouillé les motifs séquentiels clos contigus est CCSpan [Zhang et al., 2015]. CCSpan est un algorithme de recherche en largeur, son processus est donc très proche de celui que nous avons détaillé lors de la présentation de la recherche en largeur :

1. **Initialisation :** Un passage complet sur la base est effectué pour évaluer le support de chaque item et ne garder que ceux ayant un support supérieur ou égal au support minimum.
2. **Génération des candidats :** L'objectif de cette phase est d'identifier les motifs séquentiels fréquents contigus de taille $k+1$. Cette liste de motifs candidats est créée à partir des sous-motifs contigus de taille $k+1$ inclus dans chaque séquence de la base.
   Dans la séquence #1 de notre base BDS, voici les motifs contigus de taille 2 candidats : $\langle\{e,b\}\rangle, \langle\{b\},\{c\}\rangle, \langle\{c\},\{f\}\rangle, \langle\{f,d\}\rangle, \langle\{d\},\{d\}\rangle, \langle\{d\},\{a\}\rangle$.
3. **Évaluation du support des candidats :** L'objectif de cette phase est de conserver les motifs séquentiels fréquents contigus parmi la liste des candidats. Un passage complet sur la base est effectué, à chaque fois qu'un motif de la liste de candidats est rencontré dans une séquence, son support est augmenté de 1. Un élagage est appliqué afin d'accélérer le traitement de cette étape. La méthode d'élagage de CCSpan considère les deux sous-motifs de taille k obtenus en enlevant le premier item et le dernier item de chaque candidat de

taille $k+1$ de la liste. Si ces deux sous-motifs sont fréquents et contigus alors le candidat de taille $k+1$ correspondant est évalué.

4. **Évaluation de la clôture des candidats :** Pour chaque candidat de taille $k+1$ de la liste, la clôture des mêmes deux sous-motifs de taille k que précédemment est évaluée. Si les sous-motifs sont présents dans la liste des candidats motifs séquentiels contigus potentiellement clos avec le même support alors ils sont retirés. le candidat de taille $k+1$ est lui ajouté à cette liste.
   Lorsqu'il n'y a plus de candidats à évaluer, l'algorithme a extrait l'ensemble des motifs séquentiels fréquents clos contigus.

CCSpan présente les mêmes défauts que les autres algorithmes de recherche en largeur : des passages trop nombreux sur la base de séquences, génération de candidats fictifs, poids des candidats en mémoire. La génération de candidats fictifs et le poids des candidats en mémoire sont bien moins impactant dus à la taille bien plus réduite de l'ensemble des motifs séquentiels clos contigus par rapport à l'ensemble des motifs séquentiels fréquents. De plus, CCSpan est fortement impacté par la longueur des séquences due à sa méthode de génération des candidats [Abboud et al., 2017].
Un autre algorithme permet la fouille de motifs séquentiels clos contigus en combinant des contraintes de clôture et de gap, c'est l'algorithme CloSPEC [Béchet et al., 2015]. Cet algorithme est basé sur l'algorithme BIDE, c'est-à-dire sur une représentation en motifs croissants avec une stratégie de vérification de la clôture à la volée. CloSPEC permet l'intégration de plusieurs contraintes préfixes monotoniques telles que la contrainte de gap, la contrainte de longueur minimum ou la contrainte de super-motif en utilisant les règles de génération de candidats introduites par PG. [Béchet et al., 2015] a montré que CloSPEC pouvait être jusqu'à plus d'un ordre de grandeur plus rapide que Clospan. CloSPEC n'étant pas conçu autour de l'extraction des motifs séquentiels clos contigus, il ne propose aucune stratégie d'élagage ou d'optimisation propre à ces motifs.

Nous remarquons deux manques dans la revue de la littérature sur la fouille de motifs séquentiels clos contigus. Premièrement, nous n'avons pas trouvé de preuve ou d'évaluation de la capacité des algorithmes du domaine à mener une fouille lorsque l'ensemble des motifs séquentiels fréquents à extraire est très conséquent. Deuxièmement, la littérature ne mentionne pas le plus gros défaut de la fouille de motifs séquentiels clos contigus, qui est l'extraction de motifs dans des données bruitées. Or les données bruitées sont maintenant très répandues, surtout pour les utilisateurs qui exploitent des données issues du Web. "Des données sont dites bruitées lorsqu'une large portion de ses données n'apporte aucune information utile supplémentaire" [5]. En deuxième conséquence, l'ensemble des motifs séquentiels clos contigus devient moins compact et beaucoup plus redondant avec l'apparition de motifs parasités par des items bruités qui n'ajoutent aucune information. L'ensemble des motifs séquentiels clos contigus devient donc moins appréhendable pour l'utilisateur.

5. https ://en.wikipedia.org/wiki/Noisy_data

Voici les enseignements qui ressortent de cette partie de la revue de la littérature sur la fouille de motifs séquentiels clos contigus et les deux problématiques qui se posent à la fin de notre revue de la littérature :

- **2.4_E1** : La fouille de motifs séquentiels clos contigus apparaît comme le type de fouille de motifs le plus efficient avec les résultats les plus appréhendables sur les données non bruitées sans dépendre du contexte applicatif de l'utilisateur.
- **2.4_E2** : La fouille de motifs séquentiels clos contigus dans des données bruitées entraîne un biais dans l'information extraite et une augmentation de l'ensemble final sans information supplémentaire.

Voici les enseignements et les deux problématiques qui ressortent de la revue de la littérature sur la fouille de motifs séquentiels :

- **2_E1 : Il n'y a pas de preuves de la capacité de la fouille de motifs séquentiels clos contigus à mener à bien l'extraction lorsque l'ensemble des motifs séquentiels fréquents est très conséquent.**
- **2_PS1 : Comment mener à bien une fouille de motifs séquentiels dans des données bruitées en conservant l'apport de la contrainte de contiguïté ?**
- **2_PS2 : Est-il possible de mettre en place une fouille de motifs séquentiels dans une grande base de données contenant un ensemble de motifs séquentiels fréquents très conséquent ?**

À mon sens, le domaine de la fouille de motifs séquentiels est extrêmement pertinent dans le cadre de notre problématique de thèse de par son accessibilité et l'appréhendabilité des résultats par l'utilisateur. Pourtant, nous avons constaté que ce domaine présente actuellement un frein rédhibitoire pour de nombreuses utilisations : l'incapacité à extraire, de façon efficiente, l'ensemble des motifs séquentiels fréquents de grandes bases de données. L'utilisation de contraintes est la direction la plus prometteuse de la littérature pour répondre à ce problème. Nous devons donc proposer suffisamment de contraintes à l'utilisateur pour rendre les résultats de la fouille appréhendables. J'ai fait le choix de prioriser les contraintes de clôture et de contiguïté pour réduire la taille de l'ensemble des motifs extraits tout en conservant l'information recherchée par l'utilisateur. Cependant, cette combinaison de contrainte n'est pas résistante au bruit dans les données. Dans notre travail, nous souhaitons donc proposer un algorithme générique, capable d'intégrer de nombreuses contraintes, notamment les contraintes de clôture et de contiguïté, pour fouiller des grandes bases de données même bruitées.

Chapitre 3

Contribution scientifique

Dans cette thèse, notre problématique est d'améliorer l'efficience de la fouille de motifs et l'appréhendabilité de ses résultats pour qu'un maximum d'utilisateurs puisse en bénéficier. Dans ce chapitre, nous introduisons nos contributions scientifiques qui constituent notre proposition pour répondre à cette problématique : la contrainte de clôture allégée, la contrainte de robustesse et l'algorithme C3Ro.

Voici le récapitulatif de certains enseignements et problématiques de notre revue de la littérature :

- **2.2_E2** : Les algorithmes utilisant une représentation verticale sont les plus rapides, mais plus coûteux en mémoire, contrairement aux algorithmes utilisant une représentation en motifs croissants, qui sont moins rapides, mais bien moins coûteux en mémoire.
- **2.2_E3** : La fouille de motifs séquentiels fréquents est problématique lorsque l'ensemble des motifs extrait est conséquent. D'une part à cause de l'incapacité à le gérer en mémoire. D'autre part pour son exploitation par l'utilisateur.
- **2.2_PS1** : Comment réduire l'ensemble des motifs séquentiels fréquents extraits lorsque cet ensemble est conséquent et améliorer l'efficience de la fouille de motifs ?
- **2.3_E1** : L'utilisation des contraintes de clôture et de contiguïté améliore significativement l'efficience de la fouille et l'appréhendabilité des résultats en réduisant le nombre de motifs séquentiels générés, extraits et également le temps d'exécution tout en conservant la quasi-totalité de l'information contenue dans les motifs séquentiels fréquents.
- **2.3_E2** : La représentation en motifs croissants permet d'intégrer naturellement dans le processus de génération des motifs un ensemble de contraintes préfixes-monotones qui est un moyen efficace pour réduire les motifs séquentiels générés et extraits lorsque l'utilisateur a un contexte applicatif qui permet leur utilisation.
- **2.4_E1** : La fouille de motifs séquentiels clos contigus apparaît comme le type de fouille de motifs le plus efficient avec les résultats les plus appréhendables sur les données non bruitées sans dépendre du contexte applicatif de l'utilisateur.
- **2.4_E2** : La fouille de motifs séquentiels clos contigus dans des données bruitées entraîne un biais dans l'information extraite et une augmentation de l'ensemble final sans information supplémentaire.
- **2_E1** : Il n'y a pas de preuves de la capacité de la fouille de motifs séquentiels clos contigus à mener à bien l'extraction lorsque l'ensemble des motifs séquentiels fréquents est très conséquent.
- **2_PS1** : Comment mener à bien une fouille de motifs séquentiels dans des données bruitées en conservant l'apport de la contrainte de contiguïté ?
- **2_PS2** : Est-il possible de mettre en place une fouille de motifs séquentiels dans une grande base de données contenant un ensemble de motifs séquentiels fréquents très conséquent ?

La revue de la littérature a montré (**2.2_PS1**) que le principal problème de la fouille de motifs séquentiels est la taille de l'ensemble des motifs extraits. Dès que cet ensemble est trop volumineux, comme dans les grandes bases de données, la fouille devient complexe à exécuter, car l'ensemble est à la fois long à traiter et difficile, voire impossible, à stocker en mémoire. Finalement, même lorsque la fouille peut être exécutée, l'ensemble des motifs est souvent tellement conséquent qu'il devient impossible à appréhender par l'utilisateur. La fouille de motifs séquentiels rencontre donc un problème à la fois d'efficience et d'appréhendabilité de ses résultats par l'utilisateur (**2.2_E3**). La direction proposée par la littérature pour pallier ce problème est l'utilisation de contraintes permettant d'améliorer l'efficience de la fouille et de réduire l'ensemble des motifs extraits.
Nous avons identifié (**2.3_E1**) l'intégration des contraintes de clôture et de contiguïté comme étant un moyen de réduire l'ensemble des motifs extraits tout en conservant l'information recherchée par l'utilisateur sans dépendre de son contexte applicatif et de rendre la fouille plus efficiente. De plus, il apparaît que la représentation en motifs croissants est non seulement le meilleur compromis entre rapidité d'exécution et consommation de mémoire (**2.2_E2**) mais également permet l'intégration naturelle de contraintes préfixes-monotones dans le processus de

la fouille (**2.3_E2**). Nous choisissons donc de proposer **un algorithme de fouille de motifs séquentiels clos contigus basé sur une représentation en motifs croissants** afin de mettre à disposition de l'utilisateur la fouille de motifs la plus efficiente et avec les résultats les plus appréhendables possible (**2.4_E1**).
Notre revue de la littérature nous a permis de mettre en évidence que la fouille de motifs séquentiels clos contigus introduit un biais lorsque la fouille est effectuée sur des données bruitées (**2.4_E2**). Comme nous souhaitons proposer un algorithme de fouille de motifs générique et accessible, il est impératif de permettre à l'utilisateur de fouiller des données bruitées. Afin de répondre à cette problématique scientifique de la littérature (**2_PS1**), nous proposons l'introduction de deux nouvelles contraintes : la contrainte de robustesse et la contrainte de clôture allégée. **La contrainte de robustesse**, combinée à l'utilisation de wildcards, permet de compenser l'addition d'items due au bruit pendant la fouille de motifs séquentiels clos contigus tout en conservant la valeur ajoutée de la contrainte de contiguïté. **La contrainte de clôture allégée** permet de pallier la perte et la substitution d'items dues au bruit lors de la fouille de motifs séquentiels clos contigus en allégeant la contrainte de clôture de la littérature. Ces deux contraintes s'appuient sur les contraintes de clôture et de contiguïté, **l'algorithme C3Ro intègre donc les quatre contraintes dans son processus de fouille de motifs, en plus d'autoriser l'intégration de contraintes préfixes-monotones**.
Nous proposons une évaluation de la performance de l'algorithme C3Ro en le comparant aux algorithmes CCSpan et CM-Clasp de la littérature, sur plusieurs jeux de données de la littérature, aux caractéristiques variées. Nous avons utilisé certains grands jeux de données afin d'évaluer la capacité de C3Ro à extraire un ensemble très conséquent de motifs (**2_PS2**). Nous présentons également une évaluation de l'apport des nouvelles contraintes que nous avons introduites comme solution au problème de robustesse contre le bruit de la fouille de motifs séquentiels clos contigus.
Dans ce chapitre, nous allons tout d'abord introduire les définitions nécessaires à la compréhension de notre travail. Ensuite, nous détaillerons la problématique liée au bruit dans la fouille de motifs clos contigus et les deux contraintes que nous introduisons pour y répondre. Dans la partie suivante, chaque étape de l'algorithme C3Ro sera explicitée et illustrée. Nous finirons ce chapitre par la présentation de l'évaluation de la performance de C3Ro et l'apport des nouvelles contraintes.

3.1 Le problème du bruit dans la fouille de motifs séquentiels clos contigus

3.1.1 Préliminaires

Nous introduisons et redéfinissons ici plusieurs définitions de façon à pouvoir les exploiter dans notre contexte de la fouille des motifs séquentiels clos contigus :

Definition 3.1.1. Soit M et M' deux motifs séquentiels $\langle E_1, E_2, ..., E_n\rangle, \langle E'_1, E'_2, ..., E'_m\rangle$. M est un **sous-motif contigu** de M' si et seulement si il existe des entiers consécutifs $j_1, j_2, ..., j_n$ tels que : (1) $1 \leq j_1 < j_2 < ... < j_n \leq m$; et (2) $E_1 \subseteq E'_{j_1}, E_2 \subseteq E'_{j_2}, ..., E_n \subseteq E'_{j_n}$ et $M \sqsubseteq_c M'$. M' est appelé un **super-motif contigu** de M.

Definition 3.1.2. Soit BDS une base de séquences et min_sup un support minimum. M est un **motif séquentiel fréquent clos contigu** s'il n'existe aucun motif M' tel que : $M \sqsubset_c M'$, et $sup_A^{BDS}(M) = sup_A^{BDS}(M')$.

Soit BDS une base de séquences, S une séquence $\langle E'_1, E'_2, ..., E'_m \rangle$ de BDS et M un motif séquentiel $\langle E_1, E_2, ..., E_n \rangle$ de BDS tel que $M \sqsubseteq_c S$. Il existe donc des entiers consécutifs $j_1, j_2, ..., j_n$ tels que : (1) $1 \leq j_1 < j_2 < ... < j_n \leq m$; et (2) $E_1 \subseteq E'_{j_1}, E_2 \subseteq E'_{j_2}, ..., E_n \subseteq E'_{j_n}$.

Definition 3.1.3. L'ensemble des items projetés contigus de M dans S est l'ensemble des i-extensions e_i de chaque occurrence de M dans S telles que $E_n \diamond_i e_i \subseteq E'_{j_n}$ et si $m > n$ combiné à l'ensemble des s-extensions e_s de chaque occurrence de M dans S telles que $e_s \in E'_{j_{n+1}}$.

Definition 3.1.4. L'ensemble des items projetés contigus de M dans S est appelé **la séquence projetée contiguë de M dans S**.

Definition 3.1.5. M est appelé **le préfixe contigu**.

Definition 3.1.6. L'ensemble des séquences projetées contiguës du préfixe contigu M dans BDS est appelé **la base projetée contiguë de M dans BDS**, notée $M\_{}_cBDS$.

Nous illustrons sur la figure 3.1 la différence entre la séquence projetée contiguë (définition 3.2.2) et non contiguë (voir définition 2.2.10) de notre base BDS (tableau 2.4) avec le motif b comme préfixe. Les items barrés en rouge correspondent aux items de la séquence qui ne sont pas

sid	séquence
#1	$\langle$~~$\{e, b\}$~~, $\{c\}$, ~~$\{f, d\}$~~, ~~$\{d\}$~~, ~~$\{a\}$~~$\rangle$
#2	$\langle$~~$\{e\}$~~, ~~$\{e\}$~~, ~~$\{b\}$~~, $\{e, b, a, f\}\rangle$
#3	$\langle$~~$\{e\}$~~, ~~$\{b\}$~~, $\{f, d\}$, ~~$\{a\}$~~$\rangle$
#4	$\langle$~~$\{b\}$~~, $\{f, d\}\rangle$

FIGURE 3.1 – Différence entre séquence projetée contiguë et non contiguë

conservés dans la base projetée non contiguë. Dans cet exemple, les différences entre les bases projetées contiguës et non contiguës sont les items barrés en orange.

3.1.2 Le bruit dans le contexte de la fouille de motifs séquentiels clos contigus

Lors de notre revue de la littérature, nous avons soulevé le problème lié à la fouille de motifs séquentiels clos contigus dans des données bruitées. Comme mentionné dans le chapitre précédent, le bruit dans les données représente une partie des données qui n'apporte aucune information supplémentaire. En pratique, le bruit dans les données correspond à trois types de perturbations des motifs : la perte, la substitution ou l'addition d'items dans les motifs. Le bruit est donc une perturbation aléatoire, peu reproductible et peu fréquente qu'on ne retrouve donc généralement pas dans les motifs fréquents. Comme le bruit diminue le support des motifs extraits, il peut les faire disparaître de l'ensemble final des motifs extraits. La figure 3.2 nous montre que cette perturbation des motifs due au bruit entraîne presque systématiquement une perte de l'information contenue dans l'ensemble des motifs. Par exemple, dans un contexte applicatif de fouille de texte, le bruit peut correspondre à l'oubli de mots (perte), aux fautes d'orthographe (substitution) ou à l'utilisation de mots inappropriés (addition). La phrase "Va, je ne te hais point" [6] peut ainsi devenir "Va, je ne te point" ou "Va, je ne te hias point" ou encore "Va, je ne te hais vraiment

6. Pierre CORNEILLE, *Le Cid*, 1637, Acte III Scène 4

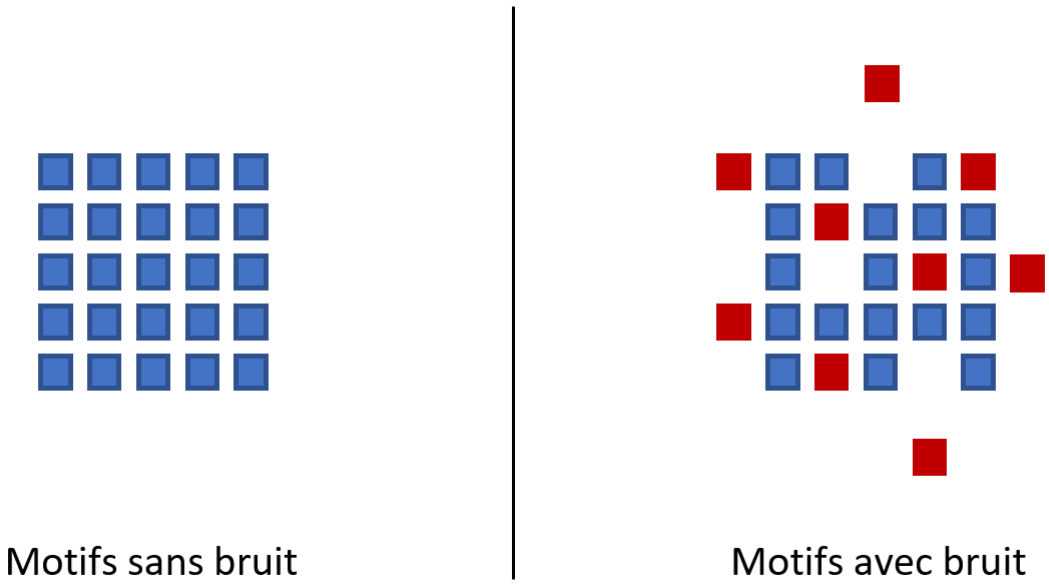


FIGURE 3.2 – Différence de l'information avec et sans bruit

point" en cas de saisie imparfaite d'un utilisateur. Il s'agit cependant du même motif "Va, je ne te hais point", deux fois sans bruit et trois fois avec une forme de bruit différente. Nous mettons en place une fouille de motifs séquentiels clos contigus où chacun de ces motifs représente une séquence dans la base CID avec un support minimum de 2 (voir tableau 3.1). L'objectif de cette fouille est d'extraire le motif $\langle Va, je, ne, te, hais, point\rangle$ qui représente l'information que nous cherchons. Finalement, lors de cette fouille, le motif $\langle Va, je, ne, te, hais, point\rangle$ n'a un support que de 2 dans cette base de cinq séquences. D'une part à cause de la segmentation du motif : $\langle Va, je, ne, te, hais, point\rangle \rightarrow \langle Va, je, ne, te, hais\rangle, \langle Va, je, ne\rangle, \langle point\rangle$. D'autre part à cause de la diminution du support de l'item $hais$: 3, où nous avons perdu l'information dans deux séquences à cause de la substitution et de la perte d'item. Nous venons de montrer que le bruit

sid	séquence	motifs extraits
#1	Va je ne te hais point	$\langle point\rangle : 5$, $\langle Va, je, ne, te\rangle : 5$,
#2	Va je ne te point	$\langle Va, je, ne, te, hais\rangle : 3$,
#3	Va je ne te hias point	$\langle Va, je, ne, te, hais, point\rangle : 2$
#4	Va je ne te hais vraiment point	
#5	Va je ne te hais point	

TABLE 3.1 – Illustration du bruit dans la base CID

dans le contexte de la fouille de motifs séquentiels clos contigus se caractérise par une perturbation des items dans les motifs (perte, substitution, addition) qui entraîne des segmentations dans l'intitulé des motifs et des diminutions de leurs supports. L'ensemble des motifs séquentiels clos contigus extraits perd alors définitivement une partie de l'information.

3.1.3 La contrainte de robustesse

La perte, la substitution et l'addition d'items dans les motifs sont la source du problème de la fouille de motifs séquentiels clos contigus dans les données bruitées. Nous proposons l'utilisation de wildcards [Li and Wang, 2008, Wu et al., 2013, Xie et al., 2017] afin de pallier l'addition d'items dans les motifs [Abboud et al., 2017]. Nous redéfinissons plusieurs concepts et introduisons le support *wildcard* afin de permettre l'utilisation de *wildcards* dans notre contexte de fouille. Pour le reste de cette thèse, nous posons $k \in \mathbb{N}$ le nombre de wildcards.

Definition 3.1.7. Une wildcard est une exception d'un item à la contrainte de contiguïté.

Definition 3.1.8. Soit M et M' deux motifs séquentiels $\langle E_1, E_2, ..., E_n\rangle$,$\langle E'_1, E'_2, ..., E'_m\rangle$ et k le nombre de *wildcards* utilisables. M est un **k-sous-motif** de M' si et seulement si il existe des entiers $j_1, j_2, ..., j_n$ tels que : (1) $1 \leq j_1 < j_2 < ... < j_n \leq m$;(2) $E_1 \subseteq E'_{j_1}, E_2 \subseteq E'_{j_2}, ..., E_n \subseteq E'_{j_n}$;(3) $j_n - j_1 - n \leq k$ et $M \sqsubseteq_k M'$. M' est appelé un **k-super-motif** de M. On dit que M' k-contient M.

Lors de la fouille de motifs clos contigus avec k wildcards, nous parlerons de fouille de motifs clos **k-contigus**.

Definition 3.1.9. Le **k-support** ou **k-support absolu** d'un motif M dans une base de séquences BDS est défini par le nombre de séquences k-contenant M (noté $sup^{BDS}_{AW_k}(M)$). Le **k-support relatif** de M dans BDS est la proportion de séquences de BDS contenant M avec k *wildcards* (noté $sup^{BDS}_{RW_k}(M)$). Le **k-support universel** de M dans BDS est le nombre d'occurrences de M avec k *wildcards* dans BDS (noté $sup^{BDS}_{UW_k}(M)$). Le **k-support wildcard** d'un motif M dans une base de séquences BDS est défini par le nombre de séquences contenant M en utilisant 1 ou au plus k wildcards(noté $sup^{BDS}_{W_k}(M)$)

La différence entre le k-support et le k-support *wildcard* d'un motif permet d'obtenir le nombre de séquences où apparaît le motif en respectant strictement la contrainte de contiguïté. Nous présentons maintenant dans le tableau 3.2 l'impact de l'utilisation d'une *wildcard* dans la fouille de motifs séquentiels clos 1-contigus ($k = 1$) sur la base précédente (tableau 3.1) en précisant le 1-support et le 1-support *wildcard* de chaque motif. Les éléments impactés par l'utilisation d'une *wildcard* sont en rouge. L'objectif de cette nouvelle fouille est toujours d'extraire le motif $\langle Va, je, ne, te, hais, point\rangle$ qui représente l'information.

sid	séquence	motifs extraits (1-support wildcard)
#1	Va je ne te hais point	$\langle point\rangle$: 5 (0), $\langle Va, je, ne, te\rangle$: 5 (0),
#2	Va je ne te point	$\langle Va, je, ne, te, point\rangle$: 4 (3),
#3	Va je ne te hias point	$\langle Va, je, ne, te, hais, point\rangle$: 3 (1)
#4	Va je ne te hais vraiment point	
#5	Va je ne te hais point	

TABLE 3.2 – Impact de l'utilisation d'une *wildcard* dans la fouille de la base CID

Nous constatons que l'utilisation d'une *wildcard* permet d'extraire un nouveau motif : $\langle Va, je, ne, te, point\rangle$. Ce motif ne contient pas l'information que nous recherchons, cela s'explique par le fait que ce motif est fouillé trois fois sur quatre grâce à l'utilisation d'une wildcard. Dans le contexte d'une fouille de motifs séquentiels clos contigus, le motif nouveau motif trouvé $\langle Va, je, ne, te, point\rangle$ n'est donc pas pertinent. Comme attendu, la *wildcard* permet de trouver une occurrence de plus du motif $\langle Va, je, ne, te, hais, point\rangle$ dans la séquence #4 et ainsi de

pallier la présence de l'item additionnel *vraiment* due au bruit. Cette augmentation de 1 du 1-support permet au motif $\langle Va, je, ne, te, hais, point\rangle$ d'avoir le même 1-support que le motif $\langle Va, je, ne, te, hais\rangle$, et comme $\langle Va, je, ne, te, hais\rangle \sqsubseteq_1 \langle Va, je, ne, te, hais, point\rangle$, il n'est plus clos et disparaît donc de l'ensemble des motifs extraits.
En conclusion, l'utilisation d'une *wildcard* permet bien de pallier l'apparition d'items additionnels et donc d'extraire de l'information jusque-là inaccessible à cause du bruit pendant une fouille de motifs séquentiels clos contigus. En contrepartie, l'utilisation de *wildcards* fait apparaître des motifs ne respectant pas ou peu la contrainte de contiguïté. Ces motifs ne sont donc pas pertinents pour l'extraction d'information dans un contexte de fouille de motifs séquentiels clos contigus. **Le problème est donc de pouvoir utiliser des *wildcards* afin de pallier l'addition d'items tout en contrôlant la proportion de motifs ne respectant pas la contrainte de contiguïté.** Afin de répondre à ce problème, nous proposons d'introduire une nouvelle contrainte : **la contrainte de robustesse**. Cette contrainte est basée sur un paramètre ajustable (**le ratio de robustesse**) pour limiter la proportion des motifs utilisant des *wildcards* dans les motifs de l'ensemble final des motifs fréquents extraits sans se passer de l'utilisation de *wildcards* pour pallier la perturbation des items additionnels lors de la fouille sur des données bruitées.

Definition 3.1.10. Soit un **ratio de robustesse** $\delta \in [0; 1]$, BDS une base de séquences et MC_k l'ensemble des motifs séquentiels k-contigus de BDS, l'ensemble MP_δ **des motifs séquentiels k-contigus δ-robustes** est défini comme suit : $MP_\delta = \{M | M \in MC_k \text{ tel que } \frac{sup_{W_k}^{BDS}(M)}{sup_{AW_k}^{BDS}(M)} \leq \delta\}$

Le ratio de robustesse fixe la proportion maximale d'occurrences utilisant une *wildcard* de chaque motif, il garantit donc la proportion minimale d'occurrences contiguës de chaque motif. Cette proportion est ajustable par l'utilisateur en fixant la valeur de ce ratio en fonction du bruit dans ses données. Cette contrainte permet ainsi de pallier l'addition d'items due au bruit tout en limitant l'introduction de motifs non pertinents dans le contexte d'une fouille de motifs contigus extraits grâce à l'utilisation de wildcards. En réduisant l'ensemble des motifs extraits tout en conservant l'information recherchée par l'utilisateur, cette contrainte améliore l'appréhendabilité des résultats de la fouille de motifs séquentiels clos k-contigus dans un contexte de données bruitées. L'utilisation de la contrainte de robustesse nécessite l'utilisation de *wildcards* et d'un ratio de robustesse δ, nous parlerons donc de la contrainte de δ-robustesse.
Dans notre exemple en cours (tableau 3.1), pour les deux motifs fouillés à l'aide de wildcards, nous avons :

$$\frac{sup_{W_k}^{CID}(\langle Va, je, ne, te, point\rangle)}{sup_{AW_k}^{CID}(\langle Va, je, ne, te, point\rangle)} = \frac{3}{4} = 0,75 \text{ et } \frac{sup_{W_k}^{CID}(\langle Va, je, ne, te, hais, point\rangle)}{sup_{AW_k}^{CID}(\langle Va, je, ne, te, hais, point\rangle)} = \frac{1}{3} \simeq 0,33.$$

Ainsi, la fouille de motifs séquentiels clos 1-contigus δ-robustes avec $\delta \in [0; 0,74]$ permet d'écarter le motif $\langle Va, je, ne, te, point\rangle$ de l'ensemble des motifs extraits et donc d'améliorer l'appréhendabilité de l'ensemble pour l'utilisateur, car ce motif n'est pas pertinent. Pour $\delta \in [0,34; 0,74]$, l'ensemble des motifs extraits devient :
$\langle point\rangle : 5, \langle Va, je, ne, te\rangle : 5, \langle Va, je, ne, te, hais, point\rangle : 3$.

3.1.4 La contrainte de clôture allégée

La contrainte de δ-robustesse que nous venons d'introduire permet de pallier l'addition d'items due au bruit lors de la fouille de motifs séquentiels clos k-contigus δ-robustes. Cependant,

les deux autres perturbations que sont la perte et la substitution d'items demeurent un problème lors de la fouille de motifs séquentiels clos contigus dans un contexte de bruit. En effet, l'ensemble des motifs extraits est encore constitué de 3 motifs alors que la base CID ne contient finalement que l'information du motif $\langle Va, je, ne, te, hais, point\rangle$ dans un contexte de bruit. Cet ensemble contient donc de l'information redondante, ce qui conduit à une perte d'appréhendabilité pour l'utilisateur. La contrainte de clôture permet d'écarter les motifs ayant un super-motif de même support. Le problème posé par la perte et la substitution d'items est la diminution du support du plus grand motif contenant l'information, les sous-motifs de ce motif n'ont alors pas le même support et ils ne peuvent pas être écartés par la contrainte de clôture. La perte et la substitution d'items empêchent donc la contrainte de clôture d'écarter l'intégralité des motifs dont l'information est contenue dans d'autres motifs. **Le problème est donc de réduire l'ensemble des motifs extraits en identifiant les motifs porteurs d'information redondante afin de pallier la perte et la substitution d'items lors de la fouille de motifs séquentiels clos contigus dans un contexte de bruit.** Afin de remédier à ce problème, nous proposons d'alléger la contrainte de clôture pour la rendre plus efficace lors de fouille de motifs séquentiels clos contigus dans des données bruitées. Notre contrainte de clôture allégée ne requiert pas que le super-motif ait le même support que ses sous-motifs pour les écarter. Cette contrainte permet de fixer jusqu'à quelle proportion du support d'un super-motif ses sous-motifs sont écartés. Nous nous appuyons sur la définition 2.3.3 pour introduire la définition suivante :

Definition 3.1.11. Soit un **ratio d'allègement** $\lambda \in [0;1]$, BDS une base de séquences et MC l'ensemble des motifs clos de BDS, l'ensemble MC_λ **des motifs λ-clos** est défini comme suit :
$MC_\lambda = \{M | M \in MC \wedge \nexists M' \in MC$ tel que $M \sqsubset M' \wedge \frac{sup^{BDS}_{AW_k}(M')}{sup^{BDS}_{AW_k}(M)} \geq \lambda\}$

Definition 3.1.12. Soit $\lambda \in [0;1]$, BDS une base de séquences et MC l'ensemble des motifs clos de BDS, M un motif séquentiel clos et M' un motif λ-clos :
$M \sqsubset M' \wedge \frac{sup^{BDS}_{AW_k}(M')}{sup^{BDS}_{AW_k}(M)} \geq \lambda \wedge \nexists M'' \in MC$ tel que $M'' \sqsubset M \wedge \frac{sup^{BDS}_{AW_k}(M')}{sup^{BDS}_{AW_k}(M'')} \geq \lambda \Rightarrow M$ est appelé **le motif séquentiel clos** de M'.

La contrainte de clôture allégée est un compromis entre la perte d'information liée au support des motifs et la réduction du nombre de motifs extraits, dont le ratio d'allègement est la variable paramétrable par l'utilisateur pour adapter la contrainte à ses besoins. Pour comprendre la flexibilité de cette nouvelle contrainte, nous ajoutons que dans la configuration où $\lambda = 0$ elle devient une contrainte de maximisation et $\lambda = 1$ elle devient une contrainte de clôture. Finalement, l'ajustabilité du ratio λ entre 0 et 1 offre une grande flexibilité et permet d'extraire un grand panel de motifs séquentiels allant des motifs séquentiels maximaux aux motifs séquentiels clos. L'ensemble des motifs séquentiels λ-clos est un sous-ensemble de l'ensemble des motifs séquentiels clos. En conséquence, l'ensemble complet des motifs séquentiels λ-clos k-contigus doit être extrait à partir de la fouille des motifs séquentiels clos k-contigus.

Nous présentons maintenant dans le tableau 3.3 l'impact de l'utilisation de la contrainte de clôture allégée sur l'ensemble des motifs séquentiels clos contigus extraits sur la base précédente (tableau 3.1) en fonction de la valeur du ratio d'allègement. L'utilisation de cette nouvelle contrainte permet effectivement de réduire le nombre de motifs extraits. La configuration $\lambda = 1$ est en fait une fouille de motifs séquentiels clos contigus comme dans le tableau 3.1. Dans la configuration où $\lambda = 0,6$, le motif $\langle Va, je, ne, te, hais\rangle$ disparaît de l'ensemble des motifs extraits, car :

λ	motifs extraits
1	$\langle point\rangle : 5, \langle Va, je, ne, te\rangle : 5, \langle Va, je, ne, te, hais\rangle : 3, \langle Va, je, ne, te, hais, point\rangle : 2$
0,6	$\langle point\rangle : 5, \langle Va, je, ne, te\rangle : 5, \langle Va, je, ne, te, hais, point\rangle : 2$
0,4	$\langle Va, je, ne, te, hais, point\rangle : 2$

TABLE 3.3 – Impact de l'utilisation de la contrainte de clôture allégée

$\langle Va, je, ne, te, hais\rangle \sqsubset_c \langle Va, je, ne, te, hais, point\rangle$ et $\frac{sup^{CID}_{AW_k}(\langle Va, je, ne, te, hais, point\rangle)}{sup^{CID}_{AW_k}(\langle Va, je, ne, te, hais\rangle)} \simeq 0,66$.
Or 0,66 $\geq \lambda = 0,6$, donc $\langle Va, je, ne, te, hais\rangle$ n'est pas un motif séquentiel 0,6-clos contigu. Dans la configuration où $\lambda = 0,4$, nous avons :
$\frac{sup^{CID}_{AW_k}(\langle Va, je, ne, te, hais, point\rangle)}{sup^{CID}_{AW_k}(\langle point\rangle)} = \frac{sup^{CID}_{AW_k}(\langle Va, je, ne, te, hais, point\rangle)}{sup^{CID}_{AW_k}(\langle Va, je, ne, te\rangle)} = 0,4.$
Comme 0,4 $\geq \lambda = 0,4$, le seul motif séquentiel 0,4-clos contigu est $\langle Va, je, ne, te, hais, point\rangle$. Même si le motif extrait est le motif contenant l'information recherchée,nous avons considéré que le bruit entraîne des perturbations aléatoires, donc peu reproductibles et non fréquentes, dans les items des motifs. Une configuration avec un ratio d'allègement aussi bas que $\lambda = 0,4$ autorise une grosse perte de l'information sur le support. Dans notre contexte, un ratio aussi bas permet d'obtenir le motif recherché, mais avec seulement un support de 2, nous ne pouvons donc pas affirmer que l'information de ce motif représente effectivement toute l'information recherchée dans la base.

3.1.5 Notations

Pour simplifier la compréhension et la lecture des libellés des différents types de motifs séquentiels fréquents abordés dans cette thèse, nous utilisons la notation du tableau 3.4 pour le reste de cette thèse (les types de motifs venant de la littérature sont grisés).

abréviations	type de motifs
MG	motifs séquentiels généraux
Cl	motifs séquentiels clos
MM	motifs séquentiels maximaux
Co	motifs séquentiels contigus
$ClCo$	motifs séquentiels clos contigus
$Cl_\lambda Co_k$	motifs séquentiels λ-clos k-contigus
$ClCo_k R_\delta$	motifs séquentiels clos k-contigus δ-robuste
$Cl_\lambda Co_k R_\delta$	motifs séquentiels λ-clos k-contigus δ-robuste
$Cl_\lambda Co_k R_\delta C_\zeta$	$Cl_\lambda Co_k R_\delta$ satisfaisant un ensemble ζ de contraintes

TABLE 3.4 – Notations

3.1.6 Présentation du problème

Nous présentons maintenant plusieurs de ces ensembles de motifs séquentiels extraits de la base de séquences CID (voir tableau 3.1) avec $min\_sup = 2$, $k = 1$ wildcard, $\delta = 0,6$ et $\lambda = 0,6$. La liste complète des motifs est présentée dans le tableau 3.5. L'ensemble des motifs séquentiels contigus fréquents Co est composé de 11 motifs et passe à 4 motifs dans l'ensemble des motifs séquentiels clos contigus $ClCo$. Ces deux ensembles sont extraits en utilisant les contraintes de

contiguïté et de clôture de la littérature (grisés dans le tableau 3.5). Comme vu précédemment, l'utilisation d'une *wildcards* permet l'augmentation du 1-support de plusieurs motifs :

- $\langle Va, je, ne, te, point\rangle$, en passant de 1 à 4. En conséquence, $\langle Va, je, ne, te, point\rangle$ devient un motif $ClCo_1$
- $\langle Va, je, ne, te, hais, point\rangle$, en passant de 2 à 3. En conséquence, $\langle Va, je, ne, te, hais, point\rangle$ a maintenant le même support que le motif $\langle Va, je, ne, te, hais\rangle$, ce dernier n'est donc pas un motif clos et il n'est pas dans les motifs $ClCo_1$.

Ainsi, l'ensemble de motifs $ClCo_1$ est constitué de 4 motifs. Le motif $\langle Va, je, ne, te, point\rangle$ étant extrait trois fois sur quatre grâce à l'utilisation d'une wildcard, nous avons $\frac{3}{4} = 0,75 > \delta = 0,6$. Ce motif ne respecte pas la contrainte de 0,6-robustesse, car il n'est contigu que dans 25% des séquences dans lesquels il apparaît. Or cette contrainte en demande au moins 60%. L'ensemble de motifs $ClCo_1R_{0,6}$ est donc constitué de 3 motifs. Sachant que $\langle point\rangle \sqsubset_1 \langle Va, je, ne, te, hais, point\rangle$ et $\frac{sup^{CID}_{AW_k}(\langle Va, je, ne, te, hais, point\rangle)}{sup^{CID}_{AW_k}(\langle point\rangle)} = 0,6$. De même, $\langle Va, je, ne, te\rangle \sqsubset_1 \langle Va, je, ne, te, hais, point\rangle$ et $\frac{sup^{CID}_{AW_k}(\langle Va, je, ne, te, hais, point\rangle)}{sup^{CID}_{AW_k}(\langle Va, je, ne, te\rangle)} = 0,6$. La contrainte de clôture allégée de ratio d'allègement $\lambda = 0,6$ permet d'écarter les motifs $\langle point\rangle$ et $\langle Va, je, ne, te\rangle$.

Finalement, l'ensemble de motifs $Cl_{0,6}Co_1R_{0,6}$ est réduit au motif $\langle Va, je, ne, te, hais, point\rangle$ avec un 1-support de 3.

Parmi les 5 séquences de la base CID, 2 contiennent le motif $\langle Va, je, ne, te, hais, point\rangle$ sans bruit et 3 le contiennent avec des perturbations d'items de type perte, substitution ou addition. L'utilisation de la contrainte de robustesse permet de pallier l'addition d'items en trouvant une occurrence de plus du motif $\langle Va, je, ne, te, hais, point\rangle$ et d'écarter le motif non pertinent apparu à cause de l'utilisation d'une wildcard. L'utilisation de la contrainte allégée de clôture permet d'écarter les motifs apparus à cause de la perte et de la substitution d'items dans la base CID.

type de motifs	motifs
Co	$\langle hais\rangle : 5, \langle je\rangle : 5, \langle ne\rangle : 5, \langle point\rangle : 5, \langle te\rangle : 5, \langle Va\rangle : 5,$ $\langle Va, je\rangle : 5, \langle Va, je, ne\rangle : 5, \langle Va, je, ne, te\rangle : 5,$ $\langle Va, je, ne, te, hais\rangle : 3, \langle Va, je, ne, te, hais, point\rangle : 2$
$ClCo$	$\langle point\rangle : 5, \langle Va, je, ne, te\rangle : 5, \langle Va, je, ne, te, hais\rangle : 3,$ $\langle Va, je, ne, te, hais, point\rangle : 2$
$ClCo_1$	$\langle point\rangle : 5, \langle Va, je, ne, te\rangle : 5, \langle Va, je, ne, te, point\rangle : 4,$ $\langle Va, je, ne, te, hais, point\rangle : 3$
$ClCo_1R_{0,6}$	$\langle point\rangle : 5, \langle Va, je, ne, te\rangle : 5, \langle Va, je, ne, te, hais, point\rangle : 3$
$Cl_{0,6}Co_1R_{0,6}$	$\langle Va, je, ne, te, hais, point\rangle : 3$

TABLE 3.5 – Type de motifs extraits dans la base CID

En conclusion, l'utilisation de nos deux nouvelles contraintes permet d'améliorer l'appréhendabilité de l'ensemble de motifs $ClCo$ dans un contexte de bruit. D'une part, en palliant l'addition d'items dans les motifs tout en préservant au maximum la contrainte de contiguïté. D'autre part, en écartant les motifs apparus à cause de la perte et de la substitution d'items dont l'information est contenue dans d'autres motifs de l'ensemble des motifs extraits.

Dans le cadre de notre problématique de thèse qui est d'améliorer l'efficience de la fouille de motifs et l'appréhendabilité de ses résultats, nous voulons proposer un algorithme de fouille de motifs intégrant nativement les contraintes de clôture, de contiguïté, de robustesse, de clôture allégée et offrant la possibilité d'intégrer un ensemble de contraintes préfixes-monotones. Finalement, notre algorithme doit répondre au problème suivant :

Énoncé du problème :
Soit BDS une base de séquences, min_sup un support minimum , k un nombre de wildcards, δ un ratio de robustesse, λ un ratio d'allègement et ζ un ensemble de contraintes préfixes-monotones. Comment extraire les motifs $Cl_\lambda Co_k R_\delta C_\zeta$, c'est-à-dire l'ensemble des motifs séquentiels λ-clos k-contigus δ-robustes fréquents satisfaisant les contraintes de ζ dans BDS ?

3.2 L'algorithme C3Ro

Dans cette section, nous introduisons notre **algorithme C3Ro** basé sur une représentation en motifs croissants, dédié à la fouille des motifs de type $Cl_\lambda Co_k R_\delta C_\zeta$. L'objectif de cet algorithme est de répondre aux trois problématiques posées lors de notre revue de la littérature (**2.2_PS1, 2_PS1, 2_PS2**). Afin de réduire l'ensemble des motifs séquentiels fréquents extraits sans perdre l'information recherchée par l'utilisateur (**2.2_PS1**), C3Ro intègre nativement les contraintes de clôture et de contiguïté (**2.3_E1, 2.4_E1**). Afin de limiter au maximum la consommation de mémoire (**2.2_PS1**), C3Ro se base sur une représentation en motifs croissants qui apparaît être la représentation la plus efficiente pour la fouille de motifs (**2.2_E2**). De plus, cette représentation permet d'intégrer un ensemble de contraintes préfixes-monotones en fonction de l'objectif de la fouille. L'utilisation de ce type de contraintes permet d'améliorer significativement l'efficience de la fouille (**2.3_E2**) et l'appréhendabilité de ses résultats. Afin de permettre une fouille de motifs séquentiels clos contigus dans des données bruitées (**2_PS1**), C3Ro intègre l'utilisation de *wildcards* et les deux contraintes que nous venons d'introduire et : la contrainte de robustesse et la contrainte de clôture allégée. L'intégration des *wildcards* et de ces deux contraintes permet également à C3Ro d'être très générique, puisqu'en fonction du nombre de *wildcards* et du ratio d'allègement C3Ro a la possibilité d'extraire des motifs séquentiels généraux ou contigus, clos ou maximaux.
L'algorithme C3Ro a été conçu pour offrir à l'utilisateur la fouille de motifs la plus efficiente avec les résultats les plus appréhendables possible. Afin de remplir au mieux cette fonction, C3Ro est un algorithme générique permettant à l'utilisateur de fixer un nombre important de paramètres pour fouiller les motifs séquentiels correspondant au mieux à ses besoins :

— **Un nombre k correspondant au nombre maximum de *wildcards* autorisées par motif.**
Quand $k = 0$, C3Ro extrait les motifs séquentiels contigus et les motifs séquentiels généraux lorsque $k = \infty$. Les *wildcards* sont un moyen de pallier l'addition d'items introduits par le bruit dans les données. Dans un contexte de fouille de motifs séquentiels contigus, même si les données sont bruitées, le nombre de *wildcards* n'a pas vocation à être trop grand, car il s'agit d'une exception à la contrainte de contiguïté. C'est pourquoi même si l'utilisation de *wildcard* augmente le nombre de motifs, la contrainte de contiguïté reste très efficace pour réduire le nombre de motifs extraits.
— **Un ratio de robustesse** δ, permettant d'extraire les motifs séquentiels k-contigus δ-robustes. Ce ratio permet de limiter l'impact de l'utilisation des *wildcards* dans l'extraction d'un motif. La valeur du ratio de robustesse garantit une proportion de contiguïté

sid	séquence
#1	$\langle\{a\}\{b\}\{c\}\{bd\}\rangle$
#2	$\langle\{a\}\{b\}\{c\}\rangle$
#3	$\langle\{ab\}\{d\}\{b\}\{c\}\{d\}\rangle$

TABLE 3.6 – Base de séquences BDS

pour chaque motif. Cette garantie est essentielle lors de la fouille de motifs séquentiels k-contigus dans un contexte de bruit pour conserver la cohérence de la contrainte de contiguïté.

— **Un ratio d'allègement** λ, permettant de maîtriser l'allègement de la contrainte de clôture sur les motifs extraits. Cet allègement est nécessaire lors de la fouille de motifs dans un contexte de bruit, car il permet de pallier la perte et la substitution d'items au prix d'une perte d'information potentielle sur le support des motifs. La valeur du ratio d'allègement est le moyen de contrôler cette perte, allant des motifs séquentiels clos ($\lambda = 1$) aux motifs séquentiels maximaux ($\lambda = 0$).

— **Un ensemble de contraintes préfixes-monotones** ζ.

C3Ro est un algorithme de recherche en profondeur, il suit donc le même processus que celui introduit dans la revue de la littérature.
Premièrement, C3Ro utilise une représentation en motifs croissants pour énumérer les motifs séquentiels k-contigus δ-robustes ($Co_kR_\delta C_\zeta$) fréquents satisfaisant l'ensemble des contraintes préfixes-monotones ζ. Deuxièmement, C3Ro utilise une stratégie de vérification à la volée de la clôture allégée des motifs, basée sur la stratégie de vérification de la clôture des motifs de l'algorithme BIDE [Wang and Han, 2004]. Cette stratégie a l'avantage de ne pas conserver des motifs candidats en mémoire et donc d'être plus efficiente. Plusieurs stratégies d'élagage de l'espace de recherche sont appliquées lors de ces deux étapes.
Nous introduisons la base BDS (tableau 3.6) qui sera utilisée tout au long de cette section pour illustrer le processus de l'algorithme C3Ro.

3.2.1 Énumération des motifs $Co_kR_\delta C_\zeta$

Nous introduisons et redéfinissons ici plusieurs concepts de façon à pouvoir les exploiter dans notre contexte de la fouille des motifs $Co_kR_\delta C_\zeta$:

Soit BDS une base de séquences, S une séquence $\langle E'_1, E'_2, ..., E'_m\rangle$ de BDS et M un motif séquentiel k-contigu $\langle E_1, E_2, ..., E_n\rangle$ de BDS tel que $M \sqsubseteq_k S$. Il existe donc des entiers $j_1, j_2, ..., j_n$ tels que : (1) $1 \leq j_1 < j_2 < ... < j_n \leq m$;(2) $E_1 \subseteq E'_{j_1}, E_2 \subseteq E'_{j_2}, ..., E_n \subseteq E'_{j_n}$;(3) $j_n - j_1 - n \leq k$.

Definition 3.2.1. Le nombre $k - j_n - j_1 - n$ représente **le nombre de *wildcards* restantes** pour étendre le k-préfixe M dans la séquence S.

Definition 3.2.2. L'ensemble des k-items projetés de M dans S est l'ensemble des i-extensions e_i telles que $E_n \diamond_i e_i \subseteq E'_{j_n}$ et l'ensemble des s-extensions e_s telles que $e_s \in E'_{j_{n+1}} \cup E'_{j_{n+2}} ... \cup E'_{min(j_{n+(k-j_n-j_1-n)+1}, m)}$.

Definition 3.2.3. L'ensemble des k-items projetés de M dans S est appelé **la k-séquence projetée de M dans S**.

Definition 3.2.4. M est appelé **le k-préfixe**.

Definition 3.2.5. L'ensemble des k-séquences projetées du préfixe contigu M dans BDS est appelé **la k-base projetée de M dans BDS**, notée $M\_{}_kBDS$.

Par exemple, dans la base BDS, la 0-base projetée du préfixe $\langle\{a\}\rangle$ est : $\{\langle\{b\}\rangle, \langle\{b\}\rangle, \langle\{\_b\}, \{d\}\rangle\}$. Si une *wildcard* est utilisable et utilisée, le symbole "*" accompagne l'item correspondant. Dans notre configuration précédente, l'utilisation d'une *wildcard* permet de découvrir les items suivants : $\{\{c^*\}, \{c^*\}, \{b^*\}\}$. Finalement, la 1-base projetée du préfixe $\langle\{a\}\rangle$ dans BDS est : $\{\langle\{b\}, \{c^*\}\rangle, \langle\{b\}, \{c^*\}\rangle, \langle\{\_b\}, \{d\}, \{b^*\}\rangle\}$. Les définitions que nous venons d'introduire permettent l'énumération des motifs Co_k. Afin de pouvoir énumérer les motifs Co_kR_δ, nous devons pouvoir identifier les k-items projetés pouvant étendre un k-préfixe δ-robuste pour obtenir un motif séquentiel k-contigu δ-robuste.

Definition 3.2.6. Soit M un k-préfixe et e un item tel que $e \in M\_{}_kBDS$. **Le support *wildcard* de e dans $M\_{}_kBDS$** est le nombre de séquences contenant e^* suivant une occurrence de M n'ayant pas utilisée de wildcards, noté $sup_W^{M_{-k}BDS}(e)$.

Nous illustrons le support *wildcard* dans une base projetée de la base BDS, avec le 2-préfixe $\langle\{a\}, \{b\}\rangle$ et l'item $\{d\}$:
Dans la séquence #1, $\langle\{a\}, \{b\}\rangle$ est extrait sans utiliser de *wildcard* mais une est utilisée pour trouver $\{d\}$. Cette *wildcard* augmente le support de 1. Dans la séquence #3, $\langle\{a\}, \{b\}\rangle$ a déjà utilisé une *wildcard* et en utilise une seconde pour trouver $\{d\}$. Par conséquent, cette *wildcard* n'est pas comptabilisée pour le support *wildcard* de $\{d\}$ dans la 2-base projetée de $\langle\{a\}, \{b\}\rangle$ car la *wildcard* de cette occurrence était déjà comptabilisée. Au final, $sup_W^{\langle\{a\},\{b\}\rangle_{-2}BDS}(\{d\}) = 1$.

Le support *wildcard* d'un item dans une k-base projetée nous permet d'introduire le théorème 3.2.1 pour identifier les items permettant d'énumérer des motifs Co_kR_δ :

Theorem 3.2.1. *Soit BDS une base de séquences, δ un ratio de robustesse, M un motif Co_kR_δ et e un item tel que $e \in M\_{}_kBDS$. $\langle M \diamond e\rangle$ est un motif Co_kR_δ si :*

$$\frac{sup_{W_k}^{BDS}(M) + sup_W^{M_{-k}BDS}(e)}{sup_{AW_k}^{M_{-k}BDS}(e)} \leq \delta \tag{3.1}$$

Démonstration. Soit M un motif Co_kR_δ et e un item tel que $e \in M\_{}_kBDS$ vérifiant l'inégalité 3.1. En s'appuyant sur la définition 3.2.6, nous avons $sup_{W_k}^{BDS}(M) + sup_W^{M_{-k}BDS}(e) = sup_{W_k}^{BDS}(\langle M \diamond e\rangle)$. Comme $sup_{AW_k}^{M_{-k}BDS}(e) = sup_{AW_k}^{BDS}(\langle M \diamond e\rangle)$:
$\frac{sup_{W_k}^{BDS}(M) + sup_W^{M_{-k}BDS}(e)}{sup_{AW_k}^{M_{-k}BDS}(e)} \leq \delta \Leftrightarrow \frac{sup_{W_k}^{BDS}(\langle M \diamond e\rangle)}{sup_{AW_k}^{BDS}(\langle M \diamond e\rangle)} \leq \delta$ alors $\langle M \diamond e\rangle$ est un motif Co_kR_δ.

□

L'utilisation de ce théorème permet l'énumération des motifs Co_kR_δ. En reprenant l'exemple précédent, dans la base BDS avec le motif 2-contigu 0,5-robuste $\langle\{a\}, \{b\}\rangle$. La 2-base projetée du 2-préfixe 0,5-robuste $\langle\{a\}, \{b\}\rangle$ est : $\{\langle\{c, b^*, d^*\}, \{c\}, \{c, d^*\}\rangle\}$. L'item c n'utilise jamais de wildcard, le motif $\langle\{a\}, \{b\}, \{c\}\rangle$ est donc lui aussi 2-contigu 0,5-robuste. L'item b utilise une *wildcard* alors que cette occurrence du préfixe n'en avait pas utilisé, nous avons donc :
$\frac{sup_{W_k}^{BDS}(\langle\{a\}, \{b\}\rangle) + sup_W^{\langle\{a\},\{b\}\rangle_{-k}BDS}(b)}{sup_{AW_k}^{\langle\{a\},\{b\}\rangle_{-k}BDS}(b)} = \frac{1+1}{1} = 2 > 0,5.$

Le motif $\langle\{a\},\{b\},\{c\}\rangle$ n'est donc pas $Co_2R_{0,5}$. L'item d utilise deux *wildcards* dont une après une occurrence de $\langle\{a\},\{b\}\rangle$ n'en utilisant pas.
Nous avons donc $\frac{sup_{W_k}^{BDS}(\langle\{a\},\{b\}\rangle) + sup_{W}^{\langle\{a\},\{b\}\rangle_{-k}BDS}(d)}{sup_{AW_k}^{\langle\{a\},\{b\}\rangle_{-k}BDS}(d)} = \frac{1+1}{2} = 1 > 0,5$. Le motif $\langle\{a\},\{b\},\{d\}\rangle$ n'est également pas $Co_2R_{0,5}$. Finalement, seul l'item c peut étendre le motif $\langle\{a\},\{b\}\rangle$ en un motif $Co_2R_{0,5}$.

La définition et le théorème que nous venons d'introduire permettent l'énumération des motifs Co_kR_δ. Nous présentons maintenant comment intégrer l'ensemble des contraintes préfixes-monotones ζ lors de cette énumération.
Grâce à la représentation en motifs croissants, nous pouvons utiliser les règles d'extension d'un motif séquentiel M avec une contrainte préfixe-monotone C comme introduit dans [Pei et al., 2007] :

1. Quand C est préfixe anti-monotonique,
   $C(M) = Faux \Rightarrow \nexists M' | M' = M \diamond Q \wedge C(M') = Vrai$ où Q est un motif séquentiel.
2. Quand $C(M)$ est préfixe monotonique,
   $C(M) = Vrai \wedge M' = M \diamond Q \Rightarrow C(M') = Vrai$ où Q est un motif séquentiel.

En pratique, voici les deux règles que nous introduisons pour l'énumération des motifs $Co_kR_\delta C_\zeta$ en se basant sur les deux propriétés précédentes :

1. **Si C est préfixe anti-monotonique et $C(M) = Faux$ alors on peut stopper l'extension du préfixe M.**
2. **Si C est préfixe monotonique et $C(M) = Vrai$ alors il est inutile de tester si les motifs ayant pour préfixe M satisfont la contrainte C.**

Nous pouvons maintenant présenter le processus complet de l'énumération des motifs $Co_kR_\delta C_\zeta$ dans C3Ro. Lors d'une fouille de motifs séquentiels fréquents, l'énumération des motifs recherchés peut être assimilée à la construction d'un arbre où chaque nœud représente un item et chaque branche un motif [Ayres et al., 2002]. Le premier nœud de l'arbre, étiqueté $\emptyset$, est le niveau le plus haut de l'arbre (niveau 0). Le deuxième niveau (niveau 1) est composé des items fréquents de la base, le troisième niveau (niveau 2) est composé des motifs séquentiels fréquents de taille 2. Finalement, l'arbre possède $n+1$ niveaux où n est la taille du plus grand motif séquentiel fréquent. La construction de l'arbre dépend du type de recherche, du type de représentation et des contraintes de l'algorithme utilisé.
Dans le cas de l'algorithme C3Ro et de la représentation en motifs croissants, l'énumération des motifs se fait en étendant récursivement chaque préfixe avec certains items de sa k-base projetée. Un préfixe correspond au motif constitué à partir du chemin entre le niveau 1 et le nœud que l'on souhaite étendre.
Lors de l'énumération des motifs $Co_kR_\delta C_\zeta$, **la première étape est l'identification des items fréquents satisfaisant toutes les contraintes préfixes-monotones de ζ**. Chaque item est ensuite évalué en fonction du type (préfixe anti-monotonique ou préfixe monotonique) de chaque contrainte de ζ et des règles que nous avons introduites précédemment pour déterminer si cet item doit devenir un préfixe à étendre ou non. À la fin de cette étape, nous avons un ensemble d'items validés comme étant des motifs $Co_kR_\delta C_\zeta$, ces items représentent le niveau 1 de l'arbre. Nous savons également quels items dans cet ensemble sont des préfixes à étendre pour obtenir l'ensemble complet des motifs $Co_kR_\delta C_\zeta$.
La deuxième étape est la recherche d'extensions pour chaque préfixe. Cette étape

consiste à examiner les bases projetées k-contiguës de chaque préfixe pour évaluer le k-support et le k-support *wildcard* de chaque item. Si un item :

— est fréquent ;
— a un k-support *wildcard* vérifiant le théorème 3.2.1 ;
— forme avec son préfixe un motif vérifiant chaque contrainte préfixe-monotone de ζ ;

alors cet item étend son préfixe pour donner un nouveau motif faisant partie de l'ensemble de motifs $Co_k R_\delta C_\zeta$.
Chaque nouveau motif est à nouveau évalué en fonction du type de chaque contrainte de ζ et des règles pour déterminer si c'est un nouveau préfixe qui doit être étendu. La deuxième étape est récursivement répétée avec les nouveaux préfixes jusqu'au moment où il n'y a plus d'items répondant aux conditions pour étendre un préfixe. À la fin de l'énumération, l'ensemble de motifs $Co_k R_\delta C_\zeta$ est complet.

Nous illustrons l'énumération des $Co_k R_\delta$ de C3Ro sur la base de séquences BDS avec un support minimum $min\_sup = 2$, $k = 1$ *wildcard* et un ratio de robustesse $\delta = 0,6$. La figure 3.3 montre l'arbre formé à partir de l'énumération des motifs $Co_1 R_{0,6}$ de la base BDS.
Tous les items de la base sont fréquents, C3Ro commence donc par étendre le 1-préfixe $P = \langle\{a\}\rangle$ (exploitation de l'ordre alphabétique). Sa 1-base projetée est : $\{\langle\{b\},\{c^*\}\rangle, \langle\{b\},\{c^*\}\rangle, \langle\{\_b\},\{d\},\{b^*\}\rangle\}$. Dans cette 1-base projetée, $\{b\}$ a un 1-support absolu de 3 et un 1-support *wildcard* de 1 tandis que $\{c\}$ a un 1-support absolu de 2 et un 1-support *wildcard* de 2. Ces deux items ont un 1-support supérieur à $min\_sup = 2$, ils sont donc fréquents.
Nous avons $\dfrac{sup_{W_1}^{BDS}(P) + sup_{W^{-1}}^{P^{-1}BDS}(b)}{sup_{AW_k^{-1}}^{P^{-1}BDS}(b)} = \dfrac{0+1}{3} \simeq 0,33$ et $0,33 \geq \delta = 0,6$.
De même $\dfrac{sup_{W_1}^{BDS}(P) + sup_{W^{-1}}^{P^{-1}BDS}(c)}{sup_{AW_k^{-1}}^{P^{-1}BDS}(c)} = \dfrac{0+2}{2} = 1$ or $1 > \delta = 0,6$.
L'item $\{c\}$ ne peut donc pas être conservé, car il ne permet pas de construire des motifs 0,6-robustes selon le théorème 3.2.1. Le 1-préfixe $P = \langle\{a\}\rangle$ peut donc uniquement être étendu par l'item $\{b\}$. Nous considérons dorénavant le 1-préfixe $P = \langle\{a\},\{b\}\rangle$, sa 1-base projetée est : $\{\langle\{c\},\{bd^*\}\rangle, \langle\{c\}\rangle, \langle\{c\},\{d^*\}\rangle\}$. Les items $\{c\}$ et $\{d\}$ ont un 1-support de 3 et 2 respectivement, ils peuvent donc potentiellement étendre P car ils sont fréquents. L'item $\{d\}$ a un 1-support *wildcard* de 2, il ne peut donc pas être conservé en accord avec le théorème 3.2.1. Le 1-préfixe $P = \langle\{a\},\{b\}\rangle$ peut être étendu avec l'item $\{c\}$, nous posons maintenant $P = \langle\{a\},\{b\},\{c\}\rangle$. La 1-base projetée de P est : $\{\langle\{bd\}\rangle, \langle\{d\}\rangle\}$. L'item $\{d\}$ de 1-support 2, peut donc étendre le 1-préfixe P. Le 1-préfixe $P = \langle\{a\},\{b\},\{c\},\{d\}\rangle$ n'a pas de 1-base projetée. Le motif $\langle\{a\},\{b\},\{c\},\{d\}\rangle$ est donc le dernier motif $Co_1 R_{0,6}$ extrait par extension de l'item a (voir la figure 3.3). La même méthode peut être appliquée avec $P = \langle\{b\}\rangle$, $P = \langle\{c\}\rangle$ et $P = \langle\{d\}\rangle$ pour trouver l'ensemble de 10 motifs qui forme les $C_{0,6}^2 W_1 SP$ de la base BDS : $\{\langle\{a\}\rangle : 3, \langle\{a\}\{b\}\rangle : 3, \langle\{a\}\{b\}\{c\}\rangle : 3, \langle\{a\}\{b\}\{c\}\{d\}\rangle : 2, \langle\{b\}\rangle : 3, \langle\{b\}\{c\}\rangle : 3, \langle\{b\}\{c\}\{d\}\rangle : 2, \langle\{c\}\rangle : 3, \langle\{c\}\{d\}\rangle : 2, \langle\{d\}\rangle : 2\}$.

Nous pouvons voir sur la figure 3.3 que l'ensemble de motifs Co est constitué de 7 motifs. L'utilisation d'une *wildcard* permet de découvrir 6 motifs dont le motif $\langle\{a\}\{b\}\{c\}\{d\}\rangle$ qui est le plus grand motif de l'ensemble. Cet ensemble de motifs Co_1 est donc constitué de 13 items. La contrainte de 0,6-robustesse permet d'écarter 3 motifs dont aucune des occurrences n'est contiguë. Finalement, l'utilisation d'une *wildcard* et de la contrainte de 0,6-robustesse permettent de trouver 3 motifs supplémentaires par rapport à l'ensemble de motifs Co.

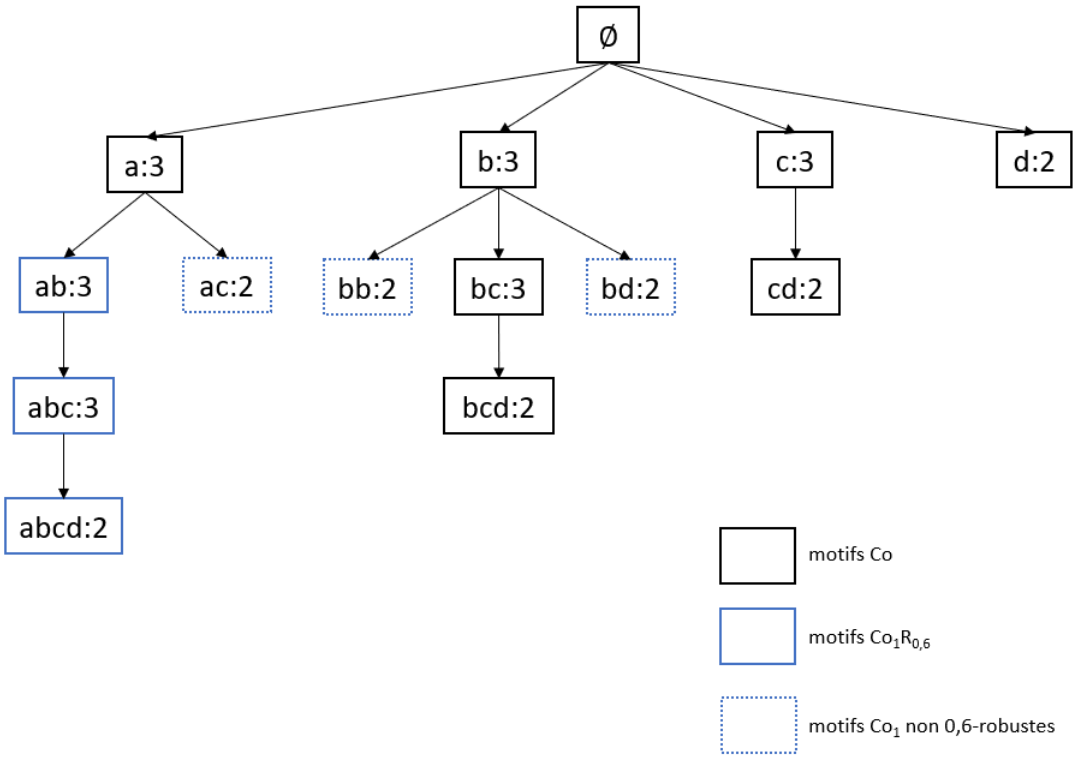


FIGURE 3.3 – Arbre de la fouille des motifs $ClCo_1R_{0,6}$

3.2.2 Vérification des motifs λ-clos

Dans la section précédente, nous avons montré comment énumérer les motifs s$Co_kR_\delta C_\zeta$. Nous allons maintenant présenter la stratégie de vérification de la clôture allégée des motifs précédemment énumérés de façon à obtenir les motifs $Cl_\lambda Co_kR_\delta C_\zeta$. Nous avons basé notre stratégie sur la même philosophie que la stratégie BI-Directionnelle utilisée dans l'algorithme BIDE. C'est-à-dire une vérification de la contrainte de clôture allégée immédiate, sans génération de candidats, afin de minimiser la consommation de mémoire et donc d'améliorer l'efficience de l'algorithme.
Étant donné que la définition 3.1.11 d'un motif λ-clos repose sur la définition 2.3.3 d'un motif clos, nous devons d'abord identifier les motifs clos avant de pouvoir ensuite identifier les motifs λ-clos.
Nous introduisons et redéfinissons ici plusieurs concepts et théorèmes de l'algorithme
BIDE [Wang and Han, 2004] de façon à pouvoir les exploiter dans notre contexte de la fouille des motifs $Cl_\lambda Co_kR_\delta C_\zeta$:

Definition 3.2.7. Soit MKC l'ensemble des motifs séquentiels k-contigus, l'ensemble MCC **des motifs séquentiels clos k-contigus** est défini comme suit : $MCC = \{M | M \in MF \wedge \nexists M' \in MF$ tel que $M \sqsubset M' \wedge sup(M) = sup(M')\}$

En accord avec cette définition, si un motif séquentiel k-contigu $M = \langle E_1, ..., E_n \rangle$ n'est pas clos, alors il existe au moins un item e qui peut étendre le motif M pour obtenir un motif séquentiel clos k-contigu M' vérifiant $sup_{AW_k}(M) = sup_{AW_k}(M')$. Le motif M peut donc être étendu de trois façons pour obtenir le motif M' :

1. $M' = \langle E_1, ..., E_n \diamond e \rangle$
2. $M' = \langle e \diamond E_1, ..., E_n \rangle$
3. $\exists i, 1 < i < n,\ M' = \langle E_1, ..., E_i \diamond e, ..., E_n \rangle$

Dans le premier cas, l'item e apparaît après l'itemset E_n, on dit que e est une **extension-avant** [7] du motif M. Dans le deuxième et troisième cas, l'item e se trouve avant l'itemset E_n, e est alors une **extension-arrière**[8] du motif. Le théorème suivant est donc trivial :

Theorem 3.2.2. *Un motif séquentiel k-contigu est clos si et seulement si il n'a aucune extension-avant et aucune extension-arrière.*

La stratégie de vérification de la clôture d'un motif séquentiel de l'algorithme BIDE repose sur la détection d'extension-avant et d'extension-arrière. Nous utilisons donc la même stratégie dans le contexte des motifs $Cl_\lambda Co_k R_\delta C_\zeta$.

Similairement, si un motif séquentiel clos k-contigu $M = \langle E_1, ..., E_n \rangle$ n'est pas λ-clos (définition 3.1.11), alors il existe au moins un item e qui peut étendre le motif M pour obtenir un motif λ-clos M' vérifiant : $\frac{sup_{AW_k}(M')}{sup_{AW_k}(M_{clos})} \geq \lambda$ (où M_{clos} est le motif clos de M' et $M_{clos} \sqsubseteq_k M$, voir définition 3.1.12).
Le motif M peut être étendu pour obtenir le motif M' des trois mêmes façons que pour les motifs clos. Dans le premier cas, l'item e apparaît après l'itemset E_n, e est alors une **λ-extension-avant** du motif M. Dans le deuxième et troisième cas, l'item e se trouve avant l'itemset E_n, e est alors une **λ-extension-arrière** du motif. Le théorème suivant est donc lui aussi trivial :

Theorem 3.2.3. *Un motif séquentiel clos k-contigu est λ-clos si et seulement si il n'a aucune λ-extension-avant et aucune λ-extension-arrière.*

Nous venons de montrer que grâce aux quatre types d'extension nous pouvons vérifier si un motif est clos et ensuite s'il est λ-clos.
Nous proposons de suivre la démarche suivante pour identifier les motifs λ-clos lors de l'énumération des motifs $Co_k R_\delta C_\zeta$. Nous évaluons chaque motif $Co_k R_\delta C_\zeta$ venant d'être validé pour d'abord déterminer si ce motif est clos. Ensuite, nous vérifions s'il est λ-clos, s'il est λ-clos nous le conservons, s'il ne l'est pas c'est qu'il existe un super-motif qui l'est. Nous conservons donc le support du motif clos pour trouver le motif λ-clos lors d'une prochaine extension du préfixe en cours.
Nous introduisons maintenant les lemmes pour identifier les quatre types d'extension et évaluer la clôture et la clôture allégée d'un motif. Nous commençons par les deux lemmes pour vérifier les extensions-avant et les λ-extensions-avant :

Lemma 3.2.4. *Soit BDS une base de séquences et M un motif $Co_k R_\delta C_\zeta$. L'ensemble complet de ses extensions-avants est l'ensemble des items e de sa k-base projetée qui vérifie : $sup_{AW_k}^{M_{-k}BDS}(e) = sup_{AW_k}^{BDS}(M)$.*

Démonstration. Pour chaque item e de la k-base projetée de M, il existe un super-motif M' tel que $M' = \langle M \diamond e \rangle$. Si $sup_{AW_k}^{M_{-k}BDS}(e) = sup_{AW_k}^{BDS}(M)$ alors $sup_{AW_k}^{BDS}(M') = sup_{AW_k}^{BDS}(M)$. En conséquence, M n'est pas clos et e est une extension-avant de M. □

7. Ces extensions sont à ne pas confondre avec les i-extensions et les s-extensions.

Lemma 3.2.5. *Soit BDS une base de séquences et M un motif $ClCo_kR_\delta C_\zeta$. L'ensemble complet de ses λ-extensions-avants est l'ensemble des items e de sa k-base projetée qui vérifie : $\frac{sup_{AW_k}^{M_{-k}BDS}(e)}{sup_{AW_k}^{BDS}(M)} \geq \lambda$.*

Démonstration. Pour chaque item e de la k-base projetée de M, il existe un super-motif M' tel que $M' = \langle M \diamond e \rangle$. Si $\frac{sup_{AW_k}^{M_{-k}BDS}(e)}{sup_{AW_k}^{BDS}(M)} \geq \lambda$ alors $\frac{sup_{AW_k}^{BDS}(M')}{sup_{AW_k}^{BDS}(M)} \geq \lambda$. En conséquence, M n'est pas λ-clos et e est une λ-extension-avant de M. □

Ces lemmes nous montrent que l'évaluation du k-support des items de la k-base projetée d'un motif nous permet de déterminer à la fois si ce motif a une extension-avant et également une λ-extension-avant.

Nous présentons maintenant comment identifier les extensions-arrières et les λ-extensions-arrières. Ces extensions rassemblent deux façons d'étendre le motif, soit l'item apparaît entre le premier et le dernier itemset du motif, soit l'item apparaît avant le premier itemset du motif. Nous introduisons un théorème permettant un élagage des extensions entre le premier et le dernier itemset du motif, pendant la phase d'énumération. Ce théorème s'inspire de l'élagage Backscan de l'algorithme BIDE en l'adaptant aux motifs $Co_kR_\delta C_\zeta$.

Theorem 3.2.6. *Soit BDS une base de séquences, M un motif $Co_kR_\delta C_\zeta$ et e_1 un item de la k-base projetée de M. Si un item e_2 apparaît toujours après e_1 dans chaque k-séquence projetée de M, alors e_2 peut être retiré de la base projetée.*

Démonstration. Soit BDS une base de séquences, M un motif $Co_kR_\delta C_\zeta$ et e_1, e_2 deux items de la k-base projetée de M. Si e_2 apparaît toujours derrière e_1 dans chaque k-séquence projetée de M, alors il existe M' un motif $Co_kR_\delta C_\zeta$ tel que $M' = \langle M \diamond e_1 \diamond e_2 \rangle$ avec $sup_{UW_k}^{BDS}(M') = sup_{UW_k}^{BDS}(\langle M \diamond e_2 \rangle)$. Comme chaque occurrence de $\langle M \diamond e_2 \rangle$ est inclue dans une occurrence de M' alors toutes ses extensions le sont aussi. Par conséquent, il est inutile d'étendre M avec e_2 et e_2 peut être retiré de la k-base projetée de M. □

Grâce à ce théorème, les futures extensions entre le premier et le dernier itemset du motif sont élaguées durant l'énumération, nous pouvons donc concentrer uniquement l'identification des extension-arrières et λ-extensions-arrières lorsque l'item apparaît avant le premier itemset du motif.

Soit BDS une base de séquences, S une séquence $\langle E'_1, E'_2, ..., E'_m \rangle$ de BDS et M un motif séquentiel k-contigu $\langle E_1, E_2, ..., E_n \rangle$ de BDS tel que $M \sqsubseteq_k S$. Il existe donc des entiers $j_1, j_2, ..., j_n$ tels que : (1) $1 \leq j_1 < j_2 < ... < j_n \leq m$;(2) $E_1 \subseteq E'_{j_1}, E_2 \subseteq E'_{j_2}, ..., E_n \subseteq E'_{j_n}$;(3) $j_n - j_1 - n \leq k$. Le nombre $k - (j_n - j_1 - n)$ représente le nombre de *wildcards* disponibles pour le motif M.

Definition 3.2.8. La k-séquence arrière de M dans S est l'ensemble des i-extensions e_i telles que $e_i \diamond_i E_1 \subseteq E'_{j_1}$ et l'ensemble des s-extensions e_s telles que $e_s \in E'_{j_1-1} \cup E'_{j_1-2} ... \cup E'_{max(j_1-(k-(j_n-j_1-n))-1,1)}$, noté $M_{\_AR_k}S$.

Definition 3.2.9. L'ensemble des k-séquences arrières de M dans BDS est appelé **la k-base arrière de M dans BDS**, notée $M_{\_AR_k}BDS$.

Nous introduisons les deux lemmes pour identifier les extensions-arrières et les λ-extensions-arrières :

Lemma 3.2.7. *Soit BDS une base de séquences et M un motif $Co_kR_\delta C_\zeta$. L'ensemble complet des extensions-arrières de M est l'ensemble d'items e de la k-base arrière de M dans BDS qui vérifient : $sup_{AW_k}^{M_{-AR}BDS}(e) = sup_{AW_k}(M)$.*

Démonstration. Pour chaque item e de la k-base arrière de M, il existe un super-motif M' tel que $M' = \langle e \diamond M \rangle$. Si $sup_{AW_k}^{M_{-AR}BDS}(e) = sup_{AW_k}(M)$ alors $sup_{AW_k}^{BDS}(M') = sup_{AW_k}^{BDS}(M)$. En conséquence, M n'est pas clos et e est une extension-arrière de M. □

Lemma 3.2.8. *Soit BDS une base de séquences et M un motif $ClCo_kR_\delta C_\zeta$. L'ensemble complet des λ-extensions-arrières de M est l'ensemble d'items e de la k-base arrière de M dans BDS qui vérifient : $\dfrac{sup_{AW_k}^{M_{-AR}BDS}(e)}{sup_{AW_k}^{BDS}(M)} \geq \lambda$.*

Démonstration. Pour chaque item e de la k-base arrière de M, il existe un super-motif M' tel que $M' = \langle e \diamond M \rangle$. Si $\dfrac{sup_{AW_k}^{M_{-AR}BDS}(e)}{sup_{AW_k}^{BDS}(M)} \geq \lambda$ alors $\dfrac{sup_{AW_k}^{BDS}(M')}{sup_{AW_k}^{BDS}(M)} \geq \lambda$. En conséquence, M n'est pas λ-clos et e est une λ-extensions-arrière de M. □

Grâce au théorème 3.2.6 et aux lemmes 3.2.7 et 3.2.8, nous avons montré que les extensions-arrières et λ-extensions-arrières sont identifiables dans la k-base arrière de chaque motif.
Partant de ces théorèmes, notre stratégie de vérification de la clôture et de la clôture allégée est donc basée sur l'évaluation du k-support des items dans les k-bases arrières et les k-bases projetées de chaque motif.

3.2.3 Illustration de la fouille de $Cl_\lambda Co_k R_\delta$ avec C3Ro

Nous avons introduit les définitions et théorèmes nécessaires à l'énumération des motifs $Co_kR_\delta C_\zeta$ puis à l'évaluation de la clôture et clôture allégée de ces motifs $Co_kR_\delta C_\zeta$, nous allons donc maintenant illustrer le processus de l'algorithme C3Ro sur la base de séquences BDS (tableau 3.6) pour extraire les motifs $Cl_{0,65}Co_1R_{0,6}$ avec $min\_sup = 2$. L'arbre construit à l'issue de cette fouille est visible sur la figure 3.4.
Les items $a : 3$, $b : 3$,$c : 3$ et $d : 2$ sont fréquents. Selon l'ordre alphabétique, le 1-préfixe $P = \langle\{a\}\rangle$ est le premier motif à étendre, sa 1-base projetée est : $\{\langle\{b\},\{c^*\}\rangle, \langle\{b\},\{c^*\}\rangle,$
$\langle\{\_b\},\{d\},\{b^*\}\rangle\}$. Les items b et c sont fréquents dans la 1-base projetée de $\langle\{a\}\rangle$. Nous remarquons que l'item c est toujours précédé par l'item b dans la 1-base projetée de $\langle\{a\}\rangle$. En accord avec le théorème 3.2.6, l'item c peut donc être retiré de la 1-base projetée. Nous remarquons également que, $\dfrac{sup_{W_k}^{BDS}(\langle\{a\}\rangle) + sup_{W}^{M_{-k}BDS}(c)}{sup_{AW_k}^{M_{-1}BDS}(c)} = \dfrac{0+2}{2} = 1 > 0,6$, selon le théorème 3.2.1 l'item c peut également être retiré de la 1-base projetée de $P = \langle\{a\}\rangle$. Le théorème 3.2.6 permet un élagage en identifiant si un item précède toujours un autre tandis que le théorème 3.2.1 évalue le k-support et k-support *wildcard* de chaque item et élague les items ne pouvant pas former de motifs δ-robustes. Sachant que l'item b a un 1-support de 3, tout comme le 1-préfixe $P = \langle\{a\}\rangle$, d'après le lemme 3.2.4 ce 1-préfixe n'est ni clos ni 0,65-clos.

Passons au 1-préfixe $P = \langle\{a\},\{b\}\rangle$, dont la 1-base projetée est : $\{\langle\{c\},\{bd^*\}\rangle, \langle\{c\}\rangle, \langle\{c\},\{d^*\}\rangle\}$. Les items $\{c\},\{d\}$ sont fréquents. L'item d est toujours précédé par l'item c dans la 1-base projetée, il est donc inutile de conserver d dans la 1-base projetée. L'item c a un 1-support de 3, tout comme P, le motif $P = \langle\{a\},\{b\}\rangle$ n'est donc ni clos ni 0,65-clos.

Considérons maintenant le préfixe $P = \langle\{a\},\{b\},\{c\}\rangle$, sa 1-base projetée est : $\{\langle\{bd\}\rangle, \langle\{d\}\rangle\}$. L'item d est fréquent, il peut donc étendre le préfixe. Il n'y a pas d'items de même 1-support que le 1-préfixe dans la 1-base projetée et donc pas d'extension-avants. Comme la 1-base arrière du 1-préfixe est vide car il n'a aucun item avant a, P n'a donc pas d'extensions-arrière non plus. D'après le théorème 3.2.2, le motif $\langle\{a\},\{b\},\{c\}\rangle$ est donc clos. D'après le lemme 3.2.5, d est une 0,65-extension-avant de $\langle\{a\},\{b\},\{c\}\rangle$ car le 1-support de d dans la 1-base projetée vérifie : $\frac{sup_{AW_k}^{\langle\{a\},\{b\},\{c\}\rangle_{-AR_1}BDS}(d)}{sup_{AW_k}^{BDS}(\langle\{a\},\{b\},\{c\}\rangle)} = \frac{2}{3} = 0,66 \geq \lambda = 0,65$. Par conséquent, le motif $\langle\{a\},\{b\},\{c\}\rangle$ n'est pas 0,65-clos.

Le 1-préfixe $P = \langle\{a\},\{b\},\{c\},\{d\}\rangle$ est maintenant considéré. Ce 1-préfixe n'a ni 1-base arrière ni 1-base projetée, car il n'y a jamais d'items avant et après P, il n'a donc aucune extension. Conformément au théorème 3.1.11, le motif $\langle\{a\},\{b\},\{c\},\{d\}\rangle$ est donc le seul motif 0,65-clos ayant comme premier item a.
La même méthode peut être appliquée avec les 1-préfixes $P = \langle\{b\}\rangle$, $P = \langle\{c\}\rangle$ et $P = \langle\{d\}\rangle$. Au final, un seul motif forme l'ensemble des motifs $Cl_{0,65}Co_1R_{0,6}$ de la base BDS : $\{\langle\{a\},\{b\},\{c\},\{d\}\rangle\}$ parmi les 13 motifs Co. Nous avons ici une illustration de la capacité de réduction des contraintes de robustesse et de clôture allégée de l'ensemble des motifs extraits. Le tableau 3.7 est un récapitulatif des différents motifs séquentiels fréquents avec $min\_sup = 2$, que nous avons présenté tout au long de cette section.

type de motifs	motifs
Co	$\{a\}:3$, $\{a\}\{b\}:2$, $\{a\}\{b\}\{c\}:2$, $\{b\}:3$, $\{b\}\{c\}:3$, $\{b\}\{c\}\{d\}:2$, $\{c\}:3$, $\{c\}\{d\}:2$, $\{d\}:2$
$Co_1R_{0,6}$	$\{a\}:3$, $\{a\}\{b\}:3$, $\{a\}\{b\}\{c\}:3$, $\{a\}\{b\}\{c\}\{d\}:3$, $\{b\}:3$, $\{b\}\{c\}:3$, $\{b\}\{c\}\{d\}:2$, $\{c\}:3$,$\{c\}\{d\}:2$, $\{d\}:2$
$ClCo$	$\{a\}\{b\}\{c\}:2$, $\{b\}\{c\}:3$, $\{b\}\{c\}\{d\}:2$
$ClCo_1$	$\{a\}\{b\}\{c\}:3$, $\{a\}\{b\}\{c\}\{d\}:2$, $\{b\}\{b\}:2$
$ClCo_1R_{0,6}$	$\{a\}\{b\}\{c\}:3$, $\{a\}\{b\}\{c\}\{d\}:2$
$Cl_{0,65}Co_1R_{0,6}$	$\{a\}\{b\}\{c\}\{d\}:2$

TABLE 3.7 – Différents types de motifs séquentiels fréquents dans BDS

3.2.4 L'algorithme C3Ro

Dans cette section, nous présentons l'algorithme C3Ro qui procède à la fois à l'énumération des motifs $Co_kR_\delta C_\zeta$, et à la vérification de la clôture et de la clôture allégée tout en utilisant les stratégies d'élagage que nous proposons pour extraire l'ensemble de motifs $Cl_\lambda Co_kR_\delta C_\zeta$.

Les paramètres d'entrée de l'algorithme C3Ro sont une base de séquences BDS, un support minimum min_sup, un ensemble de contraintes préfixes-monotones ζ, un ratio de robustesse δ, un ratio d'allègement λ et un nombre maximum de *wildcards* k. C3Ro débute par une évalua-

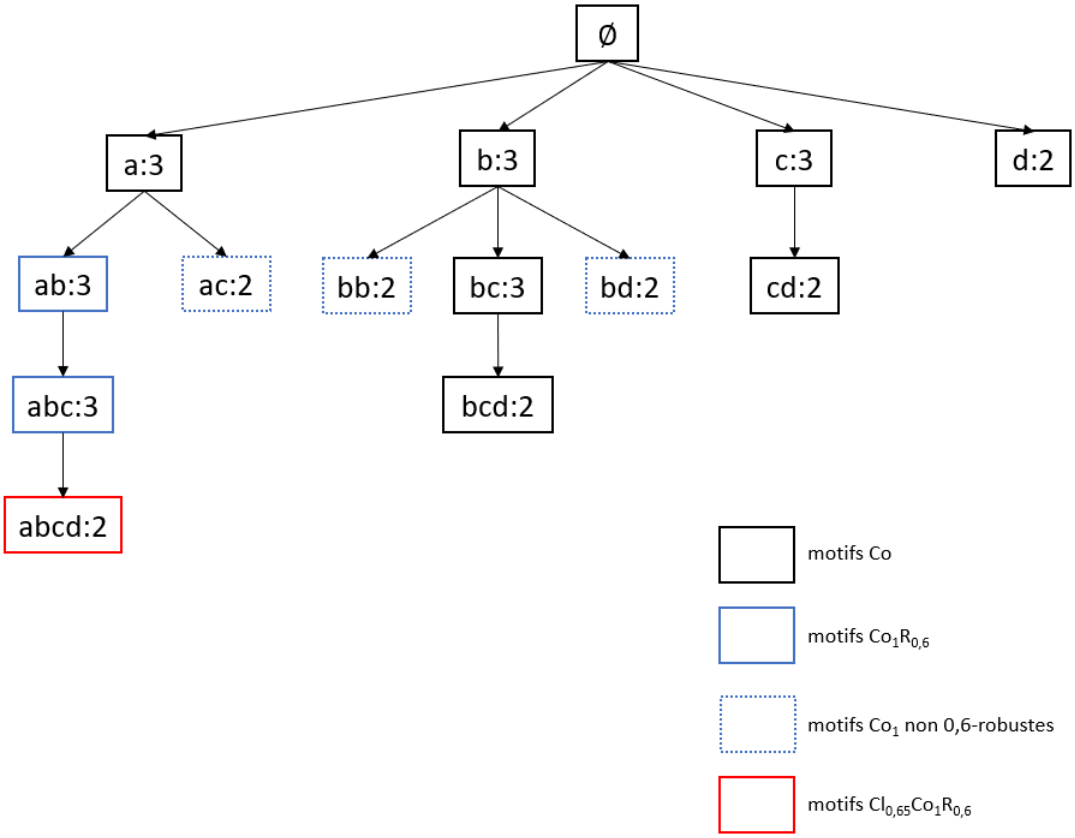


FIGURE 3.4 – Arbre de la fouille des motifs $Cl_{0,65}Co_1R_{0,6}$

tion du k-support de chaque item pour extraire $F1$, l'ensemble des items fréquents satisfaisant l'ensemble de contraintes ζ selon les règles introduites dans [Pei et al., 2007] via la fonction **Enumeration_items** (ligne 2). La k-base projetée de chaque item dans $F1$ est créée avec la fonction **Base_projetee** (ligne 4). Le sous-algorithme **P_extension** est ensuite appelé avec chacun des items fréquents satisfaisant l'ensemble des contraintes préfixes-monotones ζ (ligne 5), son but est le parcours récursif des k-bases projetées de chaque k-préfixe pour extraire l'ensemble complet des motifs $Cl_\lambda Co_k R_\delta C_\zeta$. Dans le sous-algorithme **P_extension**, la fonction **Enumeration** permet l'énumération des motifs $Co_k R_\delta C_\zeta$ dans la k-base projetée du k-préfixe P (ligne 1). La fonction **Extension_arriere** permet d'obtenir l'ensemble EAR des extensions-arrière d'un motif conformément au lemme 3.2.7(ligne 2). L'ensemble EA est l'ensemble des extension-avants du k-préfixe P, il est obtenu en accord avec le lemme 3.2.4 (ligne 3). Si les ensembles EAR et EA sont vides, alors le k-préfixe P est clos et son k-support doit donc être conservé pour la vérification de la contrainte de clôture allégée (lignes 4,5,6). Dans le cas où le k-préfixe P est clos, les λ-extensions-arrière et λ-extensions-avant sont recherchées conformément aux lemmes 3.2.8 et 3.2.5 (lignes 9,10). Si le k-préfixe n'a aucune extension et qu'il satisfait l'ensemble des contraintes ζ (ligne 11), alors il est ajouté à l'ensemble F des motifs $Cl_\lambda Co_k R_\delta C_\zeta$ (ligne 12). Le sous-algorithme **P_extension** est appelé récursivement (ligne 16) avec pour nouveau k-préfixe : le motif P étendu avec chaque item dans l'ensemble FI, crée pendant la phase d'énumération (ligne 1). Lorsque tous les k-préfixes ont été étendus, l'ensemble F est l'ensemble complet des motifs $Cl_\lambda Co_k R_\delta C_\zeta$.

Algorithm 1 – C3Ro(BDS, min_sup, ζ, δ, λ, k)

Données : une base de séquence BDS, un support minimum min_sup, un ensemble de contraintes préfixes-monotones ζ, un ratio de robustesse δ, un ratio d'allègement λ et un nombre maximum de *wildcards* k.
Résultat : F, l'ensemble des $Cl_\lambda Co_k R_\delta C_\zeta$

1: $F = \{\emptyset\}$
2: $F1$= **Enumeration_items**(BDS, min_sup, ζ);
3: **pour chaque** $f1$ dans $F1$ faire
4: $\quad f1\_BDS =$ **Base_projetee**$(BDS, f1, k)$;
5: $\quad$ **P_extension**$(f1\_BDS, f1$,min_sup$, \zeta, \delta, \lambda, k, clos_sup = -1, F)$;
6: **retourner** F ;

L'ajustement du ratio d'allègement λ et du nombre maximum de *wildcards* k, avec un ratio de robustesse $\delta = 1$, permet à l'algorithme C3Ro d'extraire un grand nombre de types de motifs (voir tableau 3.8).

k \ λ	absent	0	1
0	Co	$MMCo$	$ClCo$
∞	MG	MM	Cl

TABLE 3.8 – Types de motifs extraits par C3Ro avec $\delta = 1$

L'algorithme C3Ro est l'une des contributions de ma thèse, nous avons publié l'algorithme

Algorithm 2 – P_extension(P_BDS, P, min_sup, ζ, δ, λ, k, $clos\_sup$, F)

Données : la base projetée P_BDS du préfixe P, le préfixe P, un support minimum min_sup, un ensemble de contraintes préfixes-monotones ζ, un ratio de robustesse δ, un ratio d'allègement λ et un nombre maximum de *wildcards* k, le support du précédent motif clos $clos\_sup$ et F l'ensemble précédent des $Cl_\lambda Co_k R_\delta C_\zeta$.
Résultat : F, l'ensemble actuel des $Cl_\lambda Co_k R_\delta C_\zeta$

1: FI= **Enumeration**$(P\_BDS, min\_sup, \zeta, \delta, k)$;
2: EAR = **Extension_arriere**(P, k)
3: $EA = \left|\left\{e \in FI \mid sup^{\overline{P}^{BDS}}_{A\overline{W}_k}(e) = sup^{BDS}(P)\right\}\right|$;
4: **si** $(\{EAR \cup EA\} == \{\emptyset\})$
5: **si** $clos\_sup == -1$
6: $clos\_sup = sup^{BDS}_{AW_k}(P)$;
7: $\lambda EAR = \lambda$-**Extension_arriere**(P, k)
8: $\lambda EA = \left|\left\{e \in FI \mid sup^{\overline{P}^{BDS}}_{A\overline{W}_k}(e) \geq clos\_sup * \lambda\right\}\right|$;
9: **si** $(\{\lambda EAR \cup \lambda EA\} == \{\emptyset\})$
10: $clos\_sup = -1$;
11: **si** $Check\_C(P, \zeta)$
12: $F = F \cup \{P\}$;
13: **pour chaque** i *dans* FI faire
14: $P_i = \langle P \diamond i \rangle$;
15: P_i_BDS = **Base_projetee**(P_BDS, P_i, k) ;
16: **P_extension**$(P_i\_BDS, P_i, min\_sup, \zeta, \delta, \lambda, k, clos\_sup, F)$;

CCPM [Abboud et al., 2017] qui est une version antérieure de C3Ro n'incluant pas les contraintes de robustesse et de clôture allégée.

L'un des objectifs de cette thèse est de proposer un algorithme permettant d'améliorer l'efficience de la fouille de motifs et l'appréhendabilité de ses résultats pour la rendre accessible au plus grand nombre d'utilisateurs. Notre revue de la littérature nous a permis d'identifier deux points critiques pour accomplir cet objectif. D'une part, la taille de l'ensemble des motifs extraits est souvent trop importante, ce qui entraîne à la fois une surconsommation de mémoire et une incapacité de l'utilisateur à appréhender cet ensemble. D'autre part, les algorithmes de fouille de motifs ne sont pas toujours efficients dans la gestion de la mémoire, ce qui empire le problème de la consommation de mémoire.
Nous avons proposé l'algorithme C3Ro qui dispose de nombreux atouts pour pallier ces problèmes. Dans le but de réduire au maximum l'ensemble des motifs extraits, C3Ro utilise :

— les contraintes de k-contiguïté, de clôture allégée et de robustesse
— les contraintes préfixes-monotones selon les besoins de l'utilisateur

Afin de proposer une fouille efficiente en consommation mémoire, C3Ro utilise une représentation en motifs croissants.
Ces trois caractéristiques devraient permettre à C3Ro de proposer une fouille de motif plus efficiente avec des résultats plus appréhendables que les algorithmes vus dans notre revue de la littérature. L'évaluation des performances de la section suivante permettra la confirmation de cette hypothèse.

3.3 Évaluation des contributions scientifiques

Dans cette section, nous menons à bien une évaluation des différentes contributions scientifiques de cette thèse :

— La contrainte de robustesse
— La contrainte de clôture allégée
— L'algorithme générique C3Ro

Lors de notre revue de la littérature sur la fouille de motifs, nous avons mis en évidence (2.4_E1) : la fouille de motifs clos contigus semble être la fouille la plus efficiente, en termes de temps d'exécution et de consommation mémoire, pour des données non bruitées. Dans cette section, nous utilisons l'abréviation du type de motifs extraits (voir tableau 3.4) pour désigner la configuration de l'algorithme C3Ro utilisée.

Nous allons donc dans un premier temps évaluer la performance de l'algorithme C3Ro lors de la fouille de motifs clos k-contigus, c'est-à-dire $Cl_\lambda Co_k R_\delta C_\zeta$ avec $\lambda = 1$, $\delta = 1$ et $\zeta = \emptyset$, que nous simplifions donc par $ClCo_k$. Nous soulignons qu'il s'agit uniquement d'une configuration de C3Ro, le changement des paramètres λ et δ n'a donc pas d'impact sur les performances de C3Ro. Cette évaluation de la performance porte à la fois sur le temps d'exécution et la consommation de mémoire de l'algorithme. Afin de contextualiser les performances de C3Ro, il est impératif de les comparer aux performances d'autres algorithmes du domaine. Nous avons choisi de comparer C3Ro à CCSpan et CM-Clasp. Nous avons choisi CCSpan [Zhang et al., 2015], car il est lui aussi un algorithme de fouille de motifs $ClCo_k$ et que les auteurs ont montré que ses performances étaient meilleures que plusieurs algorithmes de fouille de motifs Cl, comme Clospan et BIDE. Nous avons également choisi CM-Clasp [Fournier-Viger et al., 2014a], car il est l'un des algorithmes de fouille de motifs Cl les plus performants (mais sans la contrainte de contiguïté).
Toujours dans le cadre de l'évaluation de la performance de C3Ro, nous évaluons d'abord l'impact de l'utilisation des *wildcards* sur le temps d'exécution et sur la consommation en mémoire de C3Ro. Nous évaluons ensuite le passage à l'échelle de C3Ro en fonction du support, du nombre de *wildcards* et des caractéristiques de la base de séquences fouillée.

Dans un second temps, nous évaluons l'apport des deux nouveaux types de motifs que nous avons introduit : les motifs $Co_k R_\delta$ et les motifs Cl_λ. Pour cette évaluation, nous avons choisi de comparer les motifs séquentiels clos k-contigus satisfaisant un ensemble de contraintes préfixes-monotones ($ClCo_k R_\delta C_\zeta$) avec les motifs séquentiels λ-clos k-contigus δ-robustes satisfaisant un ensemble de contraintes préfixes-monotones ζ ($Cl_\lambda Co_k R_\delta C_\zeta$) avec un ratio de robustesse et/ou avec un ratio d'allègement, afin d'évaluer leur impact sur l'ensemble final de motifs extraits. Comme notre contexte applicatif est le secteur de l'emploi, nous conduirons cette évaluation dans le cadre d'une fouille de motifs opérée sur un jeu de données constitué d'offres d'emploi en ligne.

Nous présentons tout d'abord l'environnement et les jeux de données utilisés lors de nos expérimentations pour effectuer notre évaluation.

3.3.1 Environnement et jeux de données

Nos expérimentations ont été menées sur un ordinateur utilisant Windows 10 avec un processeur i7-7700HQ 2.8GHz et 16GB de mémoire vive. L'expérimentation pour l'évaluation de

la performance de C3Ro est menée sur trois jeux de données (voir le tableau 3.9) de référence dans la littérature de la fouille de motifs séquentiels [Yan et al., 2003, Wang and Han, 2004, Fournier-Viger et al., 2014a, Zhang et al., 2015, Abboud et al., 2017]. $BMS1$ *Gazelle*, *Kosarak* et une version réduite de *Kosarak* : *LKosarak*. $BMS1$ *Gazelle* et *Kosarak* sont disponibles sur le site SPMF [8].
Le jeu de données $BMS1$ *Gazelle* contient les séquences de clics et d'achats venant du site marchand Gazelle qui est un revendeur de produits pour les jambes. Cette base de séquences a été initialement utilisée lors de la compétition KDD-CUP 2000 et est toujours exploitée dans de nombreux travaux. Le second jeu de données *Kosarak* est un grand jeu de données puisqu'il contient presque 1 million de séquences. *Kosarak* est constitué de clics venant d'un portail d'information hongrois. La version réduite, *LKosarak*, contient un peu moins d'un tiers des séquences de *Kosarak* avec des caractéristiques similaires. Ces trois jeux de données sont creux, mais *LKosarak* et *Kosarak* le sont bien plus que $BMS1$ *Gazelle*, comme le montrent les écarts-types de leur longueur de séquences (voir tableau 3.9).
L'expérimentation pour l'évaluation de l'apport des deux nouveaux types de motifs est menée sur le jeu de données $Jobs60K$ [Abboud et al., 2017]. Ce jeu de données est composé de 60 000 offres d'emploi collectées sur différents sites web entre janvier 2016 et juin 2016. $Jobs60K$ est le parfait exemple du type de données trouvé sur le web : bruitées et creuses. En effet, comme sur beaucoup de sites web, les données sont saisies par les utilisateurs dans les formulaires de soumissions d'offres d'emploi. Dans ces conditions, les erreurs de saisies sont donc inévitables ce qui engendre du bruit dans les données.

jeux de données	#seq.	#items	longueur moy.	longueur max.	écart-type
$BMS1$ *Gazelle*	59 601	499	6,0	535	9,7
LKosarak	305 281	32 138	8,1	2 498	23,6
Kosarak	990 002	41 270	8,1	2 498	23,7
$Jobs60K$	60 000	92 394	77,0	1 667	56,8

TABLE 3.9 – Caractéristiques des jeux de données

3.3.2 Évaluation des performances de C3Ro

C3Ro vs CM-Clasp vs CCSpan

L'évaluation des performances commence par la comparaison du temps d'exécution des trois algorithmes sur les trois jeux de données (Figures 3.5, 3.6, 3.7), puis de leur consommation de mémoire sur les trois mêmes jeux de données (voir tableau 3.10).

Le temps d'exécution de CCSpan sur *LKosarak* et *Kosarak* est trop important pour être indiqué sur nos graphiques. C'est pourquoi nous commençons par comparer le temps d'exécution de $ClCo_0$ et CCSpan (ils fouillent tous les deux les motifs séquentiels clos contigus) sur $BMS1$ *Gazelle* en faisant varier le support relatif minimum (Figure 3.5). Par la suite, nous comparons $ClCo_k$, avec k variant de 0 à ∞, et CM-Clasp (qui fouille les motifs séquentiels clos sans contrainte de contiguïté) en faisant toujours varier le support relatif minimum (Figure 3.5, 3.6, 3.7).

L'exécution de $ClCo_0$ sur $BMS1$ *Gazelle* est toujours plus rapide que celle de CCSpan, quel que soit le support minimum. Plus le support est bas, plus cette différence est importante,

8. http ://www.philippe-fournier-viger.com/spmf/index.php

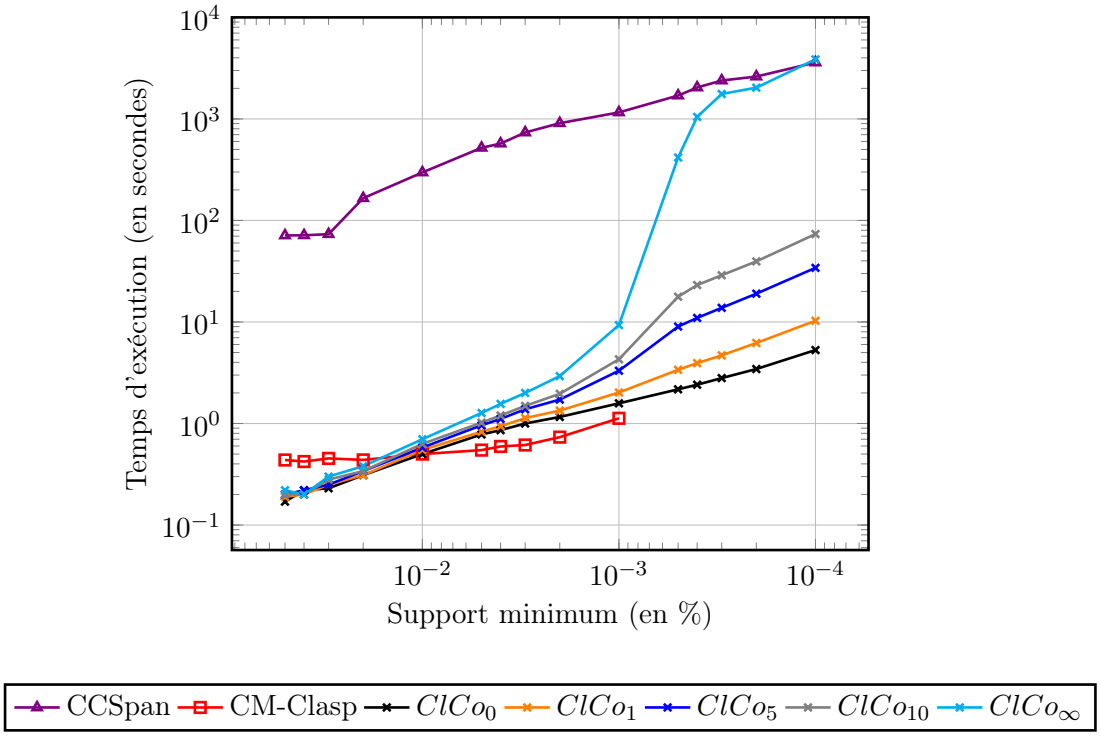


FIGURE 3.5 – Temps d'exécution sur *BMS1 Gazelle*

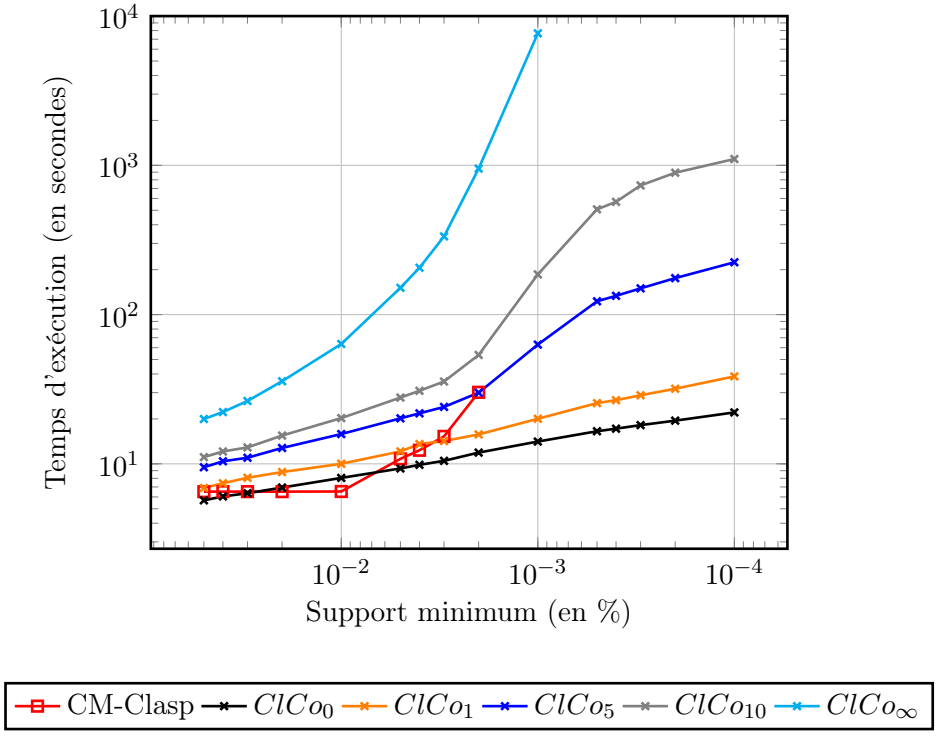


FIGURE 3.6 – Temps d'exécution sur *LKosarak*

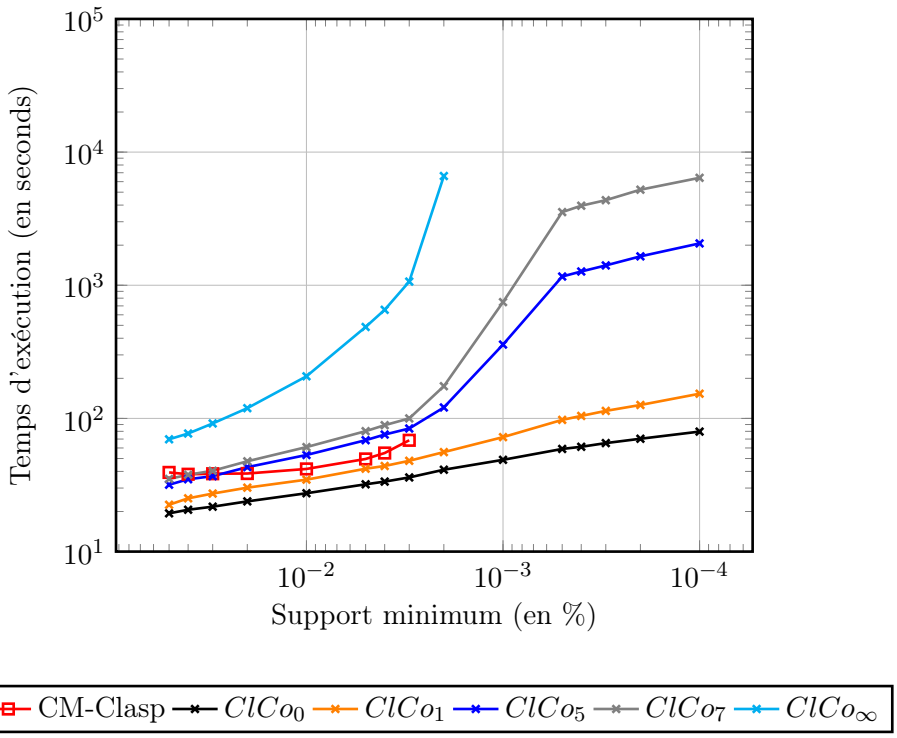


FIGURE 3.7 – Temps d'exécution sur *Kosarak*

jusqu'à 700 fois. De plus, la diminution du support minimum impacte deux fois moins le temps d'exécution de $ClCo_0$ que celui de CCSpan.
Nous pouvons conclure que concernant la fouille de motifs séquentiels clos contigus, $ClCo_0$ est bien plus rapide et bien moins impacté par la diminution du support minimum que CCSpan sur le jeu de données *BMS1 Gazelle*. Plus généralement, C3Ro est également toujours plus rapide que CCSpan sur le jeu de données *BMS1 Gazelle*, quel que soit le nombre de *wildcards* utilisé.

Nous comparons maintenant le temps d'exécution de $ClCo_k$ et CM-Clasp sur la fouille de motifs non contigus. Nous observons que $ClCo_k$ est légèrement plus rapide que CM-Clasp lorsque le nombre de *wildcards* est limité avec un support minimum élevé. Leurs temps d'exécution sont similaires lorsque le nombre de *wildcards* est de 5 et le support minimum est 10^{-3}. Ce résultat confirme la rapidité d'exécution des algorithmes de fouille de motifs séquentiels clos basés sur la représentation verticale, adoptée par CM-Clasp. Cependant, cette représentation a également l'inconvénient de consommer énormément de mémoire avec l'augmentation du nombre de motifs fréquents fouillés. En effet, CM-Clasp ne peut être exécuté avec un support minimum inférieur à 10^{-3} dû à une explosion de la consommation de mémoire, et ce sur les trois jeux de données. Nous pouvons donc conclure que, tant que le nombre de *wildcards* est limité, C3Ro s'exécute aussi rapidement que les meilleurs algorithmes de fouille de motifs séquentiels clos contigus, comme CM-Clasp, tout en étant capable de s'exécuter sur de grands jeux de données comme *Kosarak* avec un support minimum très bas.

Nous passons à l'analyse de la consommation en mémoire des trois algorithmes, détaillée dans le tableau 3.10. Nous avons choisi trois supports minimum représentatifs de ceux utilisés sur les figures 3.5, 3.6 et 3.7. Le premier constat est que $ClCo_k$ ne consomme jamais plus de 1 300MB de mémoire quels que soient le jeu de données, le nombre de *wildcards* et le support minimum.

jeux de données	$min\_sup(\%)$	$\|FI\|$	CCSpan	CM-Clasp	$ClCo_0$	$ClCo_1$	$ClCo_\infty$
$BMS1\ Gazelle$	0,05	4	250	4 200	<50	<50	<50
	0,005	150	250	4 500	200	200	200
	0,0005	374	300	-	200	200	250
$LKosarak$	0,05	10	-	5,000	500	500	500
	0,005	158	-	6 700	500	500	500
	0,0005	2303	-	-	550	550	550
$Kosarak$	0,05	10	-	10 600	1 200	1 200	1 200
	0,005	156	-	10 600	1 200	1 200	1 200
	0,0005	2297	-	-	1 300	1 300	1 300

TABLE 3.10 – Consommation mémoire (MB)

Lors de leurs exécutions sur $BMS1\ Gazelle$, CCSpan et $ClCo_k$ ont une consommation très basse : jamais plus de 300MB de mémoire. Au contraire, CM-Clasp consomme toujours plus de 4 200MB de mémoire, soit 15 fois plus que CCSpan, quand exécuté sur $BMS1\ Gazelle$ et cette consommation explose dès que le support minimum est inférieur à 5.10^{-3}, rendant impossible l'exécution de l'algorithme. Nous observons un comportement similaire sur les jeux de données $LKosarak$ et $Kosarak$, où CM-Clasp a une consommation de mémoire encore plus élevée, respectivement 6 700MB et 10 600MB, et une explosion lorsque le support minimum est inférieur à 2.10^{-2}. $ClCo_k$ a, quant à lui, une consommation de mémoire stable, autour de 500MB et 1200MB lors de l'exécution sur $LKosarak$ et $Kosarak$ quel que soit le support minimum. En moyenne, $ClCo_k$ consomme quinze fois moins de mémoire que CM-Clasp.
Pour résumer, CCSpan et $ClCo_k$ ont une consommation de mémoire faible et indépendante du support minimum pendant l'exécution sur le jeu de données $BMS1\ Gazelle$. De plus, la consommation de mémoire ne limite jamais l'exécution de $ClCo_k$ sur les trois jeux de données, contrairement à CM-Clasp. La mémoire consommée par $ClCo_k$ est également toujours significativement inférieure à celle consommée par CM-Clasp.

Nous rappelons que même si $ClCo_k$ peut fouiller des motifs séquentiels généraux, nous l'avons conçu pour extraire des motifs séquentiels contigus et donc le nombre de *wildcards* n'est pas supposé être grand.
En conclusion, $ClCo_k$, avec un nombre raisonnable de wildcards, a un temps d'exécution comparable à CM-Clasp, avec la capacité supplémentaire de pouvoir fouiller de grands jeux de données avec un support minimum bien plus faible. Dans le cadre spécifique de la fouille de motifs séquentiels clos contigus, $ClCo_0$ est jusqu'à 700 fois plus rapide que CCSpan avec une consommation de mémoire similaire. Enfin, $ClCo_k$ a une consommation mémoire basse et stable, au contraire de CM-Clasp qui est incapable de fouiller de grands jeux de données avec un support minimum faible dû à la consommation de mémoire importante de la représentation verticale.

Impact du nombre de wildcards

Nous venons de montrer que le nombre de *wildcards* n'avait pas d'impact sur la consommation mémoire de l'algorithme $ClCo_k$. Nous étudions maintenant l'impact du nombre de *wildcards* sur le temps d'exécution de $ClCo_k$, sur les trois jeux de données : $BMS1\ Gazelle$, $LKosarak$ et $Kosarak$. Nous étudions également l'impact du nombre de *wildcards* sur la taille de l'ensemble des motifs extraits.

Premièrement, nous nous intéressons à la configuration où le support minimum est le plus élevé. Sur le jeu de données $BMS1$ *Gazelle*, l'ajout de chaque *wildcard* augmente le temps d'exécution de 2% en moyenne. L'impact de chaque *wildcard* sur l'augmentation du temps d'exécution diminue de 9% à moins de 1% en passant de la première à la dixième wildcard. Sur les jeux de données $LKosarak$ et $Kosarak$, l'augmentation due à l'ajout de *wildcard* est plus grande, en moyenne 7% et 11% respectivement.
Concernant le nombre de motifs extraits associés à cette configuration, l'ensemble final n'est que très peu impacté par le nombre de wildcards. Cet impact mineur est certainement dû à la valeur du support minimum qui est très élevée. Par ailleurs, le faible impact sur le nombre de motifs supplémentaires extraits justifie certainement le faible impact sur le temps d'exécution.

Deuxièmement, nous nous focalisons sur la configuration où le support minimum est le plus faible. En termes d'impact sur le temps d'exécution, une conclusion similaire à celle de la configuration précédente peut être tirée. La seule différence vient de la valeur de l'impact qui est plus grande quel que soit le rang de la wildcard : environ 30% sur $BMS1$ *Gazelle* et 100% sur $Kosarak$. en termes de nombre de motifs extraits, contrairement à la configuration précédente, chaque *wildcard* permet d'extraire significativement plus de motifs (environ 30% de plus), sur tous les jeux de données.

Pour résumer, et comme attendu, l'utilisation de *wildcards* impacte le temps d'exécution de $ClCo_k$. Cette augmentation est de 33% par *wildcard* en moyenne sur les trois jeux de données. Plus le support est faible, plus l'impact est important. La raison qui explique cette augmentation est le grand nombre de motifs supplémentaires extraits grâce à chaque wildcard.
Ce résultat montre que le nombre de *wildcards* est un paramètre clé qui impacte à la fois le temps d'exécution et le nombre de motifs extraits qu'il est donc très pertinent de limiter surtout dans un contexte de fouille de motifs séquentiels clos.

Passage à l'échelle

Nous proposons maintenant d'évaluer la capacité de $ClCo_k$ à passer à l'échelle, en l'évaluant sur des jeux de données avec des nombres de séquences élevés et variés.

A notre connaissance, aucun algorithme de fouille de motifs séquentiels de la littérature ne peut être exécuté sur les jeux de données $LKosarak$ et $Kosarak$, en un temps raisonnable, avec un support minimum aussi bas que 10^{-4}. Le fait que $ClCo_k$, c'est-à-dire C3Ro en configuration $Cl_1Co_kR_1C_\emptyset$, puisse être exécuté sur $Kosarak$, avec un support minimum aussi faible, en moins de 80 secondes sans *wildcard* et en moins de 2 heures avec 7 wildcards, est un premier indicateur fort de sa capacité à passer à l'échelle, même s'il n'est pas suffisant.

Pour cette évaluation, nous utilisons les résultats des fouilles sur les jeux de données $LKosarak$ et $Kosarak$, qui ont des caractéristiques similaires, mais un nombre de séquences trois fois plus faible pour $LKosarak$. Lorsque le support est le plus bas, ce qui représente la configuration où le temps d'exécution est le plus important tout comme le nombre de motifs extraits, $ClCo_0$ s'exécute en 22 secondes sur $LKosarak$ et en moins de 80 secondes sur $Kosarak$. Nous constatons que $ClCo_0$ passe bien à l'échelle, car le temps d'exécution augmente linéairement avec l'augmentation du nombre de séquences (ratio de 3,6 et ratio de 3 respectivement). De plus, nous notons que $ClCo_5$ s'exécute 10 fois plus rapidement sur $LKosarak$ que sur $Kosarak$, toujours dans la configuration où le support minimum est le plus bas. Comme attendu, l'utilisation de *wild-*

cards impactant le temps d'exécution, cette utilisation impacte également légèrement le passage à l'échelle. Ces éléments montrent que l'algorithme $ClCo_k$ passe très bien à l'échelle, même si cette capacité est amoindrie par l'utilisation de wildcards.

Il est important de signaler que CCSpan, qui fouille les motifs séquentiels clos contigus, ne peut être exécuté dans un temps raisonnable (moins de 24h) sur les jeux de données $LKosarak$ et $Kosarak$. De surcroît, CM-Clasp qui est l'un des algorithmes de fouille de motifs séquentiels clos les plus performant, ne peut être exécuté sur ces trois jeux de données lorsque le support minimum est inférieur à 5.10^{-3}, alors que $ClCo_k$ est exécutable quand le support minimum est aussi faible et avec une consommation de mémoire faible. Enfin, nous rappelons que dans cette expérimentation, C3Ro a été paramétré pour fouiller les motifs $Cl_1Co_kR_1C_\emptyset$ afin de pouvoir être comparé à CCSpan et CM-Clasp. De fait, nous avons montré la capacité de C3Ro à passer à l'échelle lors de la fouille des motifs $Cl_\lambda Co_kR_\delta C_\zeta$.

Pour conclure, cette évaluation de la performance de C3Ro a mise en évidence que l'exécution de C3Ro est toujours significativement plus rapide que celle de CCSpan et quand le nombre de *wildcards* est limité, l'exécution de C3Ro est similaire à celle de CM-Clasp. De plus, cette évaluation a montré que le nombre de *wildcards* impactait le temps d'exécution de C3Ro, avec une moyenne de 33% de temps en plus pour chaque *wildcard* sur les trois jeux de données. Sur ces trois jeux de données, l'utilisation de *wildcards* n'impacte que légèrement le passage à l'échelle de C3Ro. Cette augmentation du temps d'exécution était attendue en raison du grand nombre de nouveaux motifs extraits en utilisant les *wildcards* (en moyenne 40% de motifs en plus par wildcard). Il a également été montré que l'algorithme C3Ro consomme très peu de mémoire, quel que soient le support minimum et le nombre de wildcards, alors que la consommation mémoire est le principal défaut de CM-Clasp dû à la représentation verticale. C3Ro s'est révélé être capable de passer à l'échelle avec un temps d'exécution très faible sur un grand jeu de données tel que $Kosarak$, particulièrement en comparaison de CCSpan et CM-Clasp.

Nous avons donc répondu à la problématique scientifique **2_PS2** : Est-il possible de mettre en place une fouille de motifs séquentiels dans une grande base de données contenant un ensemble des motifs séquentiels fréquents très conséquent ?

3.3.3 Évaluation de l'apport des motifs δ-robustes et λ-clos

Dans cette section, nous évaluons l'impact des motifs δ-robustes et l'impact des motifs λ-clos sur l'ensemble des motifs extraits. L'expérimentation utilisée pour cette évaluation est menée sur le jeu de données $Jobs60K$, qui contient 60 000 séquences d'une longueur moyenne de 77 items et composé à partir de 92 394 items distincts. Dans le secteur de l'emploi, les activités sont le cœur de l'offre d'emploi. Une activité est un ensemble cohérent de tâches ou séquences de travail finalisées, identifiées, organisé selon un processus logique, et orienté vers un objectif prédéfini et un résultat qui pourra être mesuré. Elle est formalisée en employant des verbes d'action [Abboud et al., 2015]. Dans le contexte de la fouille de motifs séquentiels, les activités sont des motifs contigus commençant par un verbe d'action et de taille supérieure ou égale à 3. "Élaborer un plan qualité", "Organiser le travail par soi-même", "Établir un plan de crise" sont trois exemples d'activités.

Nous fixons le support relatif minimum à $15.10^{-4}\%$ ce qui équivaut à un support absolu de 90. Le support a été fixé à cette faible valeur due à l'hétérogénéité du jeu de données, afin de garantir l'extraction d'un nombre suffisamment important de motifs pour permettre une

analyse. La contrainte d'avoir un item de type "verbe d'action au début de chaque motif" est une contrainte préfixe anti-monotonique et la contrainte de "taille supérieure ou égale à 3" est une contrainte préfixe monotonique, nous notons A cet ensemble de deux contraintes. A est donc parfaitement intégrable dans l'algorithme C3Ro. Le jeu de données $Jobs60K$ est constitué d'offres d'emploi publiées sur le marché de l'emploi en ligne, elles viennent donc de sources hétérogènes et non vérifiées. Nous pouvons considérer ce jeu de données comme bruité, ce qui justifie l'utilisation de *wildcards* pour notre fouille. Avant d'évaluer l'apport des motifs δ-robustes et λ-clos, il nous faut déterminer le nombre de *wildcards* à utiliser lors de la fouille. En effet, comme l'utilisation de *wildcards* augmente grandement le nombre de motifs extraits (comme vu dans l'expérimentation précédente), nous allons chercher la valeur qui permet d'augmenter le nombre de motifs en ayant le meilleur ratio entre les motifs pertinents et les motifs non pertinents découverts. En considérant qu'une activité a majoritairement une taille entre 3 et 6, nous étudions la taille des activités extraites par $Cl_1Co_kR_1C_A$ avec $k \in \{0, 1, 2, 3\}$ pour déterminer la valeur la plus pertinente (voir tableau 3.11). Nous constatons que la médiane augmente avec le nombre

k	0	1	2	3
médiane	4	6	11	16
mode	3	3	3	3

TABLE 3.11 – Médiane et mode des activités extraites par $Cl_1Co_kR_1C_A$ sur $Jobs60K$

de *wildcards* mais le mode reste stable. Dès la deuxième wildcard, la médiane atteint 11, les motifs sont ainsi très souvent de taille bien supérieure à 6 donc trop grand pour correspondre à des activités. Par conséquent, le nombre de *wildcards* optimal pour fouiller le jeu de données $Jobs60K$ est $k = 1$.
Nous fixons le ratio de robustesse à $\delta = 0,5$ et le ratio d'allègement à $\lambda = 0,8$. Ces valeurs ont été déterminées expérimentalement pour extraire des activités pertinentes, qui sont très peu parasitées par le bruit, tout en perdant le moins d'information possible. Ainsi, $\delta = 0,5$ signifie qu'une activité qui apparaît avec une *wildcard* plus de 50% du temps ne représente pas une activité pertinente pour nous. Alors que $\lambda = 0,8$ permet de pallier une partie du bruit au prix d'une petite partie de l'information du support des motifs (maximum 20%). Nous n'étudierons pas l'impact de la variation des valeurs de δ et λ, puisque cela ne donne de l'information que sur le jeu de données utilisé.

configuration	activité
$Cl_1Co_0R_1C_A$	352
$Cl_1Co_1R_1C_A$	698
$Cl_1Co_0R_{0,5}C_A$	429
$Cl_{0,8}Co_0R_1C_A$	601
$Cl_{0,8}Co_0R_{0,5}C_A$	384

TABLE 3.12 – $Cl_{0,8}Co_0R_{0,5}C_A$ sur $Jobs60K$

Le tableau 3.12 montre le nombre d'activités extraites par C3Ro lors de la fouille de motifs clos contigus ou non contigus (c'est-à-dire sans *wildcard* $k = 0$ et avec une *wildcard* $k = 1$), en

utilisant ou non chacune des contraintes (c'est-à-dire sans un ratio de robustesse $\delta = 1$, avec un ratio de robustesse $\delta = 0,5$, sans un ratio d'allègement $\lambda = 1$ et avec un ratio d'allègement $\lambda = 0,8$). $Cl_1Co_0R_1C_A$ extrait 352 activités, tandis que $Cl_1Co_1R_1C_A$ (qui est la même configuration avec une wildcard) en extrait 698. L'utilisation d'une *wildcard* permet donc de découvrir 346 activités supplémentaires, ce qui double le nombre d'activités extraites. Cependant, cette configuration ne fournit aucune information sur les nouvelles activités découvertes. En effet, dans cette configuration il est impossible de différentier les activités qui sont fréquentes et majoritairement contiguës, mais non trouvées par $Cl_1Co_0R_1C_A$ à cause du bruit, des activités qui sont majoritairement non contiguës et donc ne correspondant pas à notre définition d'une activité. Nous remarquons que $Cl_1Co_1R_{0,5}C_A$ extrait 429 activités, ce qui signifie que 269 activités parmi les 346 trouvées en plus par $Cl_1Co_1R_1C_A$, sont extraites plus d'une fois sur deux en utilisant une wildcard, elles sont donc majoritairement non contiguës : ces activités sont considérées non pertinentes. $Cl_1Co_1R_{0,5}C_A$ extrait 39% moins d'activités non pertinentes que $Cl_1Co_1R_1C_A$ tout en permettant d'extraire 22% plus d'activités que $Cl_1Co_0R_1C_A$.
La configuration $Cl_{0,8}Co_1R_1C_A$ extrait 601 activités. Sur ce corpus, le ratio d'allègement $\lambda = 0,8$ diminue donc le nombre d'activités extraites par $Cl_1Co_1R_1C_A$ de 97 (14%), sans retirer d'activités pertinentes. Les caractéristiques de la contrainte de clôture allégée permettent, comme attendu, "d'absorber" les sous-motifs avec plus de flexibilité que la contrainte de clôture de l'état de l'art et donc de diminuer davantage le nombre de motifs dans l'ensemble final.
Ici, lorsque C3Ro combine $\delta = 0,5$ et $\lambda = 0,8$, $Cl_{0,8}Co_1R_{0,5}C_A$ extrait 384 activités, soit 11% de moins que l'ensemble extrait par $Cl_1Co_1R_{0,5}C_A$, et ce, sans perte d'information utile, ce qui est un résultat très positif.

La comparaison entre les ensembles d'activités extraites par $Cl_{0,8}Co_1R_{0,5}C_A$ et par $Cl_1Co_0R_1C_A$ révèle que 280 activités sont commune aux deux ensembles. Ces 280 activités représentent, de fait, le sous-ensemble des activités non parasitées par le bruit. Nous allons donc concentrer notre évaluation sur les activités qui sont spécifiques à chaque ensemble pour évaluer l'impact des contraintes sur l'extraction d'activités pertinentes. Un expert du domaine a étudié ces deux ensembles d'activités spécifiques pour nous aider à déterminer la pertinence de ces activités. Parmi les 72 activités extraites uniquement par $Cl_1Co_0R_1C_A$, plus de 50% d'entre elles sont non pertinentes ou redondantes selon l'expert. Cette redondance vient du fait qu'elles sont incluses dans une autre activité avec un support très proche. Concernant les 104 activités extraites uniquement par $Cl_{0,8}Co_1R_{0,5}C_A$, seulement 10% de ces activités sont non pertinentes ou incluses dans une autre activité avec un support très proche. Cette analyse confirme que les motifs k-contigus δ-robustes permettent de réduire l'ensemble final des motifs séquentiels extraits en réduisant le nombre de motifs non pertinents extraits. Ces motifs permettent donc d'augmenter la pertinence de l'ensemble final. Les motifs séquentiels λ-clos, quant à eux, permettent bien d'obtenir un ensemble de motifs séquentiels plus compact (14%) que l'ensemble des motifs séquentiels clos. En un mot, $Cl_{0,8}Co_1R_{0,5}C_A$ permet d'extraire 20% d'activités pertinentes de plus que $Cl_1Co_0R_1C_A$. De fait, nous avons montré que $Cl_{0,8}Co_1R_{0,5}C_A$ améliore l'efficacité de la fouille des activités.

En conclusion de cette évaluation, nous avons montré que C3Ro est capable de fouiller des motifs séquentiels clos contigus dans des données bruitées, grâce à l'utilisation de *wildcards* associée aux contraintes de robustesse et de clôture allégée. De plus, ces contraintes sont flexibles grâce à l'utilisation des ratios (de robustesse et d'allègement) et du nombre de *wildcards* comme des paramètres pour permettre à l'utilisateur d'adapter la fouille à ses besoins. Finalement, C3Ro répond à la problématique scientifique **2_PS1** : Comment mener à bien une fouille de motifs

séquentiels dans des données bruitées en conservant l'apport de la contrainte de contiguïté ?

Dans ce chapitre, nous avons introduit deux nouvelles contraintes : la contrainte de robustesse et la contrainte de clôture allégée. Nous avons montré que ces deux contraintes permettaient de pallier aux perturbations d'items qui biaisent la fouille de motifs séquentiels clos contigus dans des données bruitées. Nous avons proposé un l'algorithme C3Ro basé sur une représentation en motifs croissants permettant de fouiller les motifs séquentiels clos allégés contigus robustes tout en intégrant des contraintes préfixes-monotones. Notre évaluation a montré que C3Ro est aussi rapide que les algorithmes de fouille de motifs séquentiels clos de la littérature tout en consommant significativement moins de mémoire. L'utilisation d'une représentation en motifs croissants associée à de nombreuses contraintes pour réduire la taille de l'ensemble des motifs extraits fait de C3Ro un algorithme capable de fouiller de grandes bases de données en un temps raisonnable. A notre connaissance, aucun autre algorithme de fouille de motifs de la littérature n'est capable de fouiller de grandes bases de données, essentiellement à cause de la consommation de mémoire. Je constate que la littérature de la fouille de motifs semble avoir dernièrement priorisé la rapidité des algorithmes plutôt que la consommation mémoire. Or, c'est bien cette dernière qui est limitante dès que l'ensemble des motifs à extraire devient conséquent. L'algorithme C3Ro que nous proposons est donc une vraie avancée pour faciliter et démocratiser l'utilisation de la fouille de motifs par les utilisateurs.

Chapitre 4

Contribution applicative

Cette thèse a été mené dans le cadre d'un partenariat avec l'entreprise People Compétences. Elle a par conséquent dû être menée en tenant compte de la problématique de développement de l'entreprise. J'ai donc décidé de me focaliser sur la problématique de l'exploitabilité de la fouille de données par les utilisateurs afin de proposer un algorithme capable de s'adapter à une problématique de développement de l'entreprise People Compétences. C'est dans ce contexte que nous avons proposé l'algorithme générique C3Ro, de fouille de motifs intégrant de nombreuses contraintes adaptables au contexte applicatif de l'utilisateur. Je vais maintenant montrer en quoi mes contributions, exposées dans le chapitre précédent, permettent de répondre à la problématique de développement de l'entreprise.

L'entreprise People Compétences propose du conseil et de la formation dans le domaine des ressources humaines et plus particulièrement dans le champ des compétences. Une grande partie de son activité consiste à aider à la mise en place de la démarche compétence dans les organisations demandeuses. Cette démarche repose sur l'élaboration de référentiels métiers qui répertorient l'ensemble des compétences requises dans une organisation. La construction de ces référentiels métiers est donc une étape incontournable pour mettre en place la démarche compétences. A l'heure actuelle, cette élaboration se base essentiellement sur la description des activités de l'organisation par les collaborateurs. Afin de faciliter cette étape et de rendre cette élaboration plus pertinente, nous avons convenu avec l'entreprise People Compétences de mettre en place une fouille de données visant à identifier des compétences fréquemment utilisées dans les offres d'emploi publiées par les organisations. En effet, en proposant ces compétences lors de l'élaboration des référentiels métiers, les organisations pourraient s'appuyer sur des compétences fréquemment utilisées et validées par d'autres organisations.
Nous avons choisi de fouiller le marché des offres d'emploi en ligne pour en extraire les compétences fréquentes. En effet, une offre d'emploi est supposée contenir les compétences demandées par l'organisation qui a formulé cette offre. Ici, le secteur des offres d'emploi en ligne constitue un parfait cadre d'application pour exécuter l'algorithme C3Ro. Ce secteur possède énormément de données accessibles et bruitées avec pas moins de 7,4 millions d'offres d'emploi diffusées en 2016 [9].
Dans ce chapitre, nous commençons par introduire la démarche compétences qui est le cœur notre problématique applicative. Ensuite nous présentons le corpus d'offres d'emploi que nous avons constitué. Nous exposons ensuite la démarche que nous avons suivie pour extraire les

9. http ://www.pole-emploi.org/statistiques-analyses/entreprises/offres-demploi-et-recrutement/les-offres-demploi-diffusees-p-2.html ?type=article

compétences avec notre algorithme C3Ro. Nous finissons par la présentation et l'évaluation de l'encodage des offres d'emploi extraites.

4.1 La démarche compétence

4.1.1 Vers une gestion par les compétences

Depuis les années 80 et le début de la société de l'information, l'économie mondiale a commencé à radicalement changer. Les organisations sont confrontées à la mondialisation des marchés via Internet, à l'accroissement de la concurrence et de ce fait, à une exigence accrue de la clientèle. Les organisations ont dû faire face à une explosion du volume de l'information, à un accroissement continu de la complexité des tâches et à une accélération des évolutions technologiques. Afin de rester compétitives, elles doivent s'adapter à la multiplicité des sauts technologiques, mais aussi initier les changements requis par la transition vers l'économie de la connaissance [Machlup, 1962] où le capital immatériel (l'information, les savoir-faire et les connaissances) devient le principal levier de compétitivité. "Vers la deuxième moitié des années 90, le capital humain (les compétences de la population) est devenu l'un des principaux moteurs de la croissance, contribuant à la croissance de la productivité du travail dans les pays du G7 dans une proportion allant de 15% à 90%" selon un rapport fait en 2006 par l'OCDE [10] (Organisation de coopération et de développement économique).
Les organisations et les individus font face à l'obligation de faire évoluer la gestion du travail dans l'organisation pour permettre une évaluation et un ajustement à court terme des compétences présentes chez les collaborateurs pour correspondre aux compétences requises par l'évolution permanente de l'activité de l'organisation. La gestion basée sur les compétences, ou démarche compétence, est l'outil stratégique permettant aux organisations d'agir face aux évolutions et de réussir le changement requis par ces évolutions.

4.1.2 La compétence

La compétence est une notion complexe au cœur de la démarche compétence et donc de la stratégie d'adaptation des entreprises face aux évolutions permanentes, nous allons maintenant l'étudier en détail. La notion de compétence a commencé à être utilisé pendant les années 70 [McClelland, 1973] pour montrer que les tests conventionnels d'aptitudes et d'intelligence ne permettaient pas de prédire le rendement professionnel. Pourtant, c'est bien dans les années 80 que la compétence est envisagée comme un outil de gestion du travail au sein des organisations, appelé gestion des compétences ou démarche compétence.
Nous nous intéressons donc à la littérature traitant de la "gestion des compétences". De nombreux travaux du domaine se focalisent sur la notion de compétence, pourtant il n'existe aucun consensus autour de sa définition. C'est souvent le cas des notions qui sont riches de sens et lourdes d'enjeux [Reinbold and Breillot, 1993]. Nous trouvons chez [Wittorski, 1998] que "la compétence correspond à la mobilisation dans l'action d'un certain nombre de savoirs combinés de façon spécifique en fonction du cadre de perception que se construit l'auteur de la situation". Ici la compétence est indissociable de l'action, on ne parle donc de compétence que si la personne concernée est dans la réalisation d'une action. Selon [Held and Riss, 1998], "La compétence est la capacité à réaliser les activités professionnelles attendues d'une personne dans le cadre du rôle qu'elle doit remplir, dans une organisation ou dans la société. Ou si l'on veut, l'ensemble des savoirs, savoir-faire mis en œuvre dans un contexte donné". Même s'il n'existe pas de consensus

10. http ://www.oecd.org/fr/sti/inno/36701585.pdf

sur une définition arrêtée de la compétence, la compétence est presque systématiquement définie comme une combinaison de savoirs, savoir-faire et savoir-être mobilisés dans un contexte professionnel défini pour réaliser un objectif.
Ainsi la compétence est une mobilisation de ressources afin de réaliser un objectif donné dans un contexte donné [Le Boterf, 1994].

Definition 4.1.1. Une compétence est une combinaison de connaissances, savoir-faire, expériences et comportements s'exerçant dans un contexte précis pour réaliser un objectif.

Elle se constate lors de sa mise en œuvre en situation professionnelle (activités) à partir de laquelle elle est validable. C'est donc dans le cadre d'une organisation de travail qu'il appartient de la repérer, de l'apprécier, de la valider et de la faire évoluer. La compétence n'est pas directement observable, mais elle peut être appréhendée par déduction à partir des activités réalisées.

Definition 4.1.2. Une activité est un ensemble cohérent de tâches orienté vers un objectif prédéfini et un résultat qui pourra être mesuré.

Elle est formalisée en employant des verbes d'action (contrôler, effectuer, réaliser, etc) et peut être déclinée en sous activités. Sa description doit permettre de mesurer sa complexité ainsi que les ressources nécessaires.

Finalement, les compétences et les activités permettent à la fois de définir les besoins d'une organisation par emploi, mais également de déterminer le capital humain de l'entreprise par collaborateur.

4.1.3 La démarche compétence

La démarche de compétence consiste à placer la gestion par les compétences au centre de la stratégie globale de l'organisation. Elle se déroule en trois étapes illustrées sur la figure 4.1.
La première étape du processus de la démarche compétence est la clarification des exigences de l'organisation. Pendant cette étape, les référentiels d'activités et de compétences de chaque emploi de l'organisation sont élaborés. La cohérence globale des référentiels est assurée lors de la cartographie des emplois de l'organisation. Cette cartographie consiste à catégoriser les différents emplois et à les hiérarchisés en accord avec les compétences et les niveaux de compétences requis dans les référentiels.
La deuxième étape de la démarche compétence consiste à mettre en évidence le capital humain de l'organisation en positionnant les collaborateurs dans ces référentiels. L'évaluation des compétences des collaborateurs permet d'identifier les compétences actuellement mobilisées et les compétences mobilisables dans le futur. La comparaison entre les exigences de l'organisation et les évaluations permet de constater les écarts entre l'organisation actuelle et l'organisation qui est visée.
La troisième étape est une optimisation de la gestion des ressources humaines de l'organisation reposant sur l'analyse des écarts que nous venons de mentionner. L'objectif de cette optimisation est de mettre en place des mesures permettant à l'organisation d'évoluer vers l'organisation visée. Ces mesures vont de la formation de collaborateurs pour développer certaines compétences au recrutement de nouveaux collaborateurs.
Finalement, la représentation en compétences des exigences de l'organisation confrontée à l'évaluation des compétences des collaborateurs rend possible la mise en place d'une gestion des

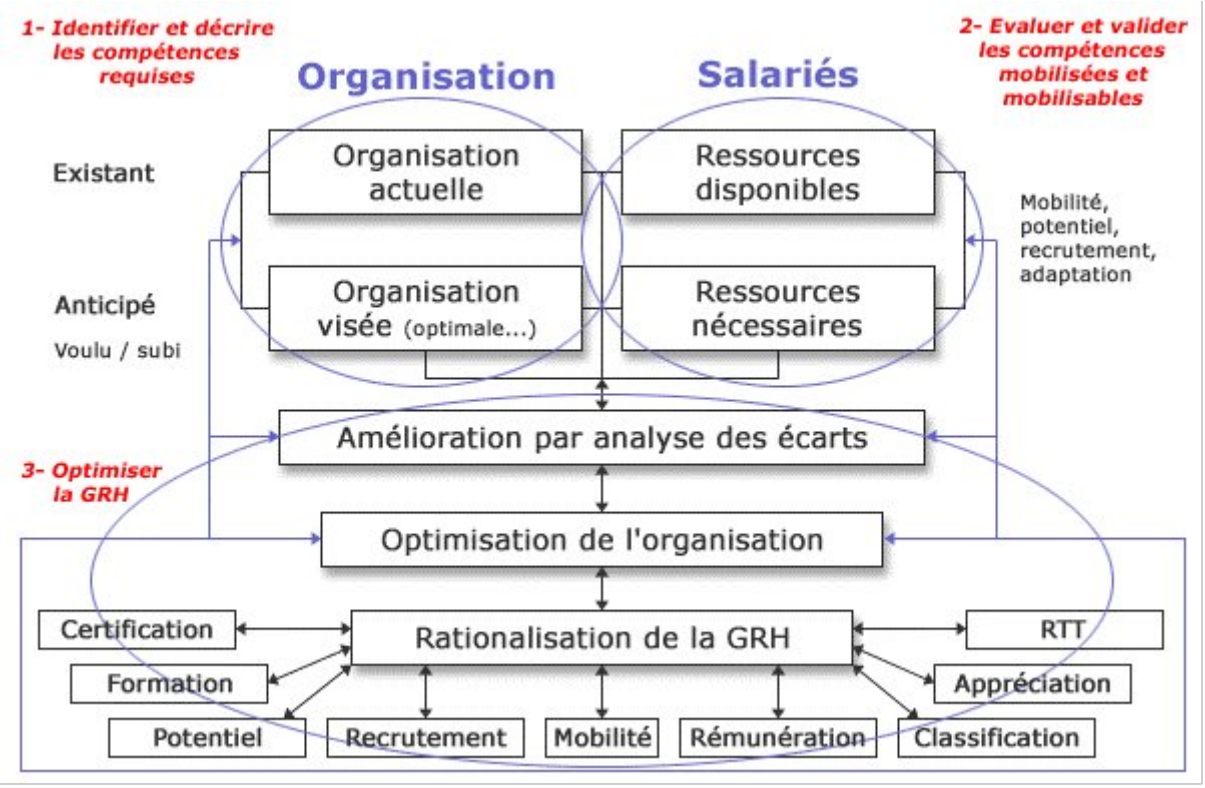


FIGURE 4.1 – La démarche compétence en 3 étapes [a]

a. Support utilisé par l'entreprise People Compétences

ressources humaines, centrée sur les compétences. C'est cette nouvelle gestion qui permet à l'organisation d'être réactive face aux évolutions et d'engager les changements pour y faire face.

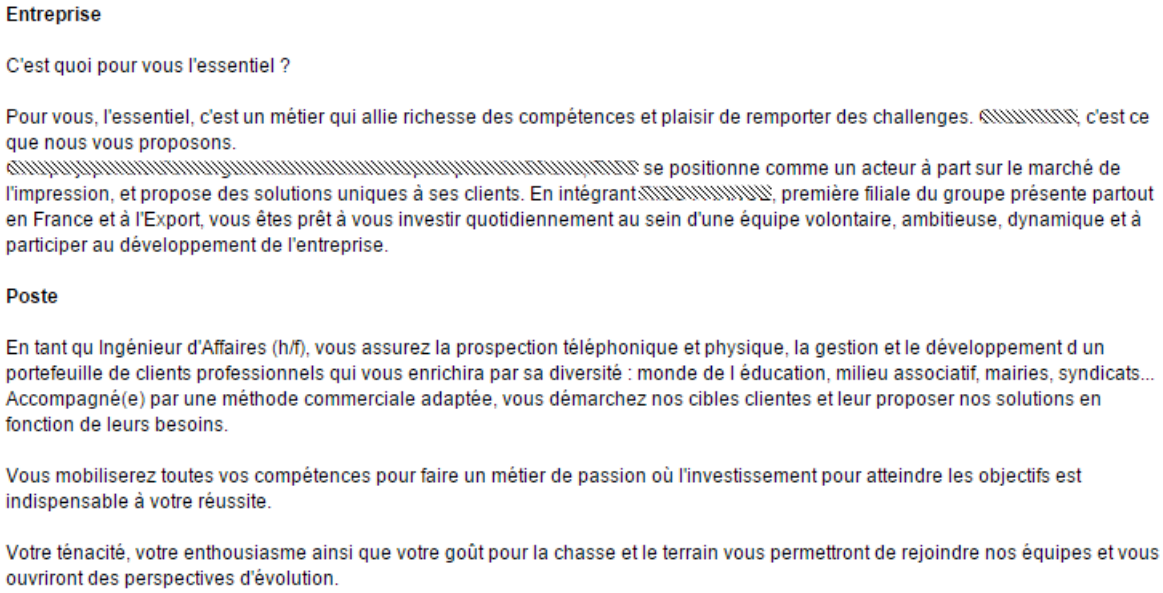
Nous venons d'expliciter les détails de la démarche compétence. Il apparaît maintenant évident que la phase d'élaboration des référentiels de compétences et d'activités de l'organisation conditionne le reste de la démarche. Ainsi, il est très judicieux d'apporter des informations préalablement validées par la communauté pour améliorer la pertinence des référentiels construits. C'est dans cette optique que s'inscrit l'application que nous présentons maintenant.

4.2 Fouille d'activités

Dans cette section nous allons justifier notre choix de nous focaliser sur la fouille d'activités plutôt que sur la fouille de compétences. Nous présentons ensuite les différentes étapes nécessaires à l'extraction d'activités à partir d'offres d'emploi.

4.2.1 Corpus d'offre d'emploi

Le but de cette application est double. Nous cherchons d'une part à évaluer la capacité de l'algorithme C3Ro à être paramétré puis implémenté dans un contexte applicatif contenant beaucoup de données, ici le marché des offres d'emploi en ligne. D'autre part, nous souhaitons extraire les compétences fréquentes contenues dans les offres d'emploi pour proposer à l'entreprise



Entreprise

C'est quoi pour vous l'essentiel ?

Pour vous, l'essentiel, c'est un métier qui allie richesse des compétences et plaisir de remporter des challenges. [redacted], c'est ce que nous vous proposons.
[redacted] se positionne comme un acteur à part sur le marché de l'impression, et propose des solutions uniques à ses clients. En intégrant [redacted], première filiale du groupe présente partout en France et à l'Export, vous êtes prêt à vous investir quotidiennement au sein d'une équipe volontaire, ambitieuse, dynamique et à participer au développement de l'entreprise.

Poste

En tant qu Ingénieur d'Affaires (h/f), vous assurez la prospection téléphonique et physique, la gestion et le développement d un portefeuille de clients professionnels qui vous enrichira par sa diversité : monde de l éducation, milieu associatif, mairies, syndicats... Accompagné(e) par une méthode commerciale adaptée, vous démarchez nos cibles clientes et leur proposer nos solutions en fonction de leurs besoins.

Vous mobiliserez toutes vos compétences pour faire un métier de passion où l'investissement pour atteindre les objectifs est indispensable à votre réussite.

Votre ténacité, votre enthousiasme ainsi que votre goût pour la chasse et le terrain vous permettront de rejoindre nos équipes et vous ouvriront des perspectives d'évolution.

FIGURE 4.2 – Exemple d'offre d'emploi du corpus

People Compétences une information supplémentaire lors de l'élaboration des référentiels dans les organisations.
Pour répondre à ces deux enjeux, nous avons constitué un corpus d'offres d'emploi à partir de plusieurs sites d'offres d'emploi français au cours des années 2016 et 2017. Ce corpus est composé de 800 000 offres d'emploi, où l'information est bruitée (saisie par l'utilisateur) et très peu structurée par une séparation en plusieurs champs utilisés ou non : intitulé du poste, date de publication, entreprise, description du poste, salaire, etc. Certains champs diffèrent en fonction du site de l'offre, mais les champs principaux, que nous venons de citer, sont communs à tous les sites. La figure 4.2 illustre une offre d'emploi du corpus. L'information qui nous intéresse est contenue dans le champ description de poste ou "poste" dans notre exemple. Dans ce corpus, les descriptions de poste contiennent en moyenne 63 mots avec une médiane de 50. La description la plus longue contient 3082 mots. Nous avons déterminé que du fait de la nature très théorique de la compétence, elle est complexe à identifier dans un texte. De plus, les compétences sont rarement renseignées dans les offres d'emploi. Par contre, les missions ou les tâches demandées par l'emploi sont évidemment toujours renseignées et elles correspondent aux activités que nous avons décrites plus haut. Au vu de ces éléments et comme les activités sont tout aussi indispensables que les compétences pour la phase d'élaboration des référentiels, nous choisissons de concentrer notre fouille sur les activités dans les offres d'emploi. Nous avons précédemment indiqué qu'une activité est formalisée avec des verbes d'action. Ainsi, dans l'offre précédente, la figure 4.3 indique les activités présentes dans la description du poste.

4.2.2 Préparation des données

Nous avons montré dans notre revue de la littérature que le processus de la fouille de données commence par la préparation des données (voir figure 2.1). Dans notre contexte, nous cherchons à appliquer C3Ro, un algorithme de fouille de motifs séquentiels sur des offres d'emploi, qui sont un contenu textuel. Nous cherchons donc à représenter les offres d'emploi sous forme de séquences tout en écartant un maximum de motifs introduisant du bruit dans les séquences.

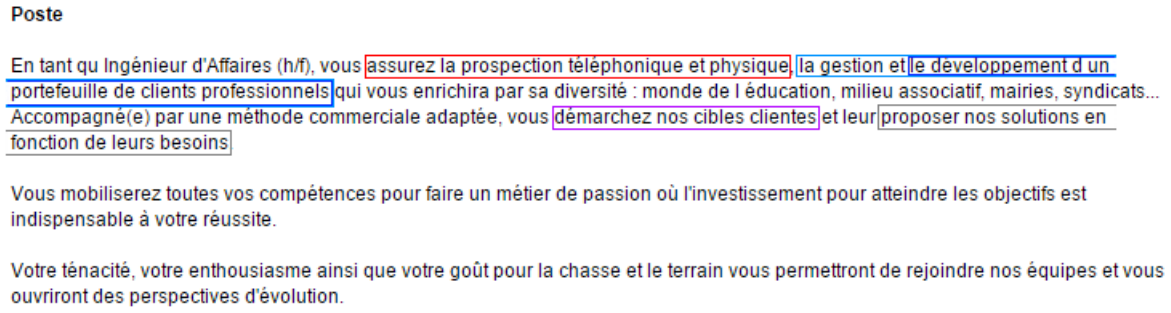


FIGURE 4.3 – Activités dans l'offre d'emploi de la figure 4.2

Les techniques de préparation de données utilisées lors de la fouille de textes font appel aux outils issus du traitement automatique des langues ou TAL.
Il existe plusieurs représentations pour la fouille de texte, la plus utilisée est certainement la représentation en sac de mots dite *bag-of-words* [Salton and McGill, 1986]. Son concept est basé sur le principe qu'un texte peut être représenté par un sac de mots ne contenant que les mots pertinents et le nombre de fois où ils apparaissent [Hotho et al., 2005]. Cette représentation a l'inconvénient de supprimer l'ordre des mots et donc de priver d'analyse sémantique. Dans notre cas, nous souhaitons implémenter C3Ro qui est un algorithme de fouille de motifs séquentiels, nous conservons donc l'ordre d'apparition des mots dans les offres d'emploi. Nous choisissons donc la représentation en n-grammes [Fürnkranz, 1998] où chaque offre d'emploi est représentée par une séquence contiguë de n mots (où n est le nombre de mots dans l'offre) qui permet de conserver l'ordre des mots. Nous présentons les étapes permettant de transformer le corpus d'offres d'emploi en une base de séquences fouillable par C3Ro.

1. Tokenisation [Hotho et al., 2005] La première chose à faire est de découper le texte en entités lexicales ou *tokens*, c'est-à-dire individualiser chaque élément du texte. Ainsi la phrase : "Il fait beau, le soleil rayonne." deviendrait :

Il	fait	beau	,	le	soleil	rayonne	.

TABLE 4.1 – Tokenisation

   Nous avons fait le choix de supprimer tous les accents pendant la tokenisation de notre corpus, afin d'écarter les problèmes fautes d'orthographes liées aux accents.

2. Retirer les mots vides ou *stopwords* [Voorhees, 1999] Cette étape permet de se débarrasser de tous les mots jugés inutiles dans le contexte de la fouille. La création d'une liste de mots vides est impérative pour obtenir des résultats satisfaisants. Par défaut elle contient tous les mots communs n'apportant aucune information comme "le", "la", "les" et également la ponctuation, mais elle peut être étoffée par d'autres mots en fonction du contexte. Si nous l'appliquons à l'exemple précédent :

fait	beau	soleil	rayonne

TABLE 4.2 – Mots vides

   Dans le cadre de la fouille d'activités, notre liste de mots vides ne contient que les mots communs.

3. Racinisation et Lemmatisation Il n'est pas rare dans la langue française, de trouver un même mot mais accordé ou conjugué différemment. Si notre représentation conserve le même mot mais sous différentes formes, il va les considérer comme des mots complètement différents et ainsi rater des informations précieuses dans l'analyse globale. En effet, si des termes de sens similaires sont détectés comme étant de sens différent, il sera impossible d'évaluer correctement la fréquence d'apparition du terme et donc impossible d'estimer son impact réel dans le corpus. Afin d'obtenir une représentation pertinente, il est donc impératif de rassembler les différentes formes d'un mot. Deux techniques permettent de faire ce rapprochement.
   — La racinisation ou *stemming* [Hotho et al., 2005] utilise la racine des mots, c'est la partie restante d'un mot lorsqu'on lui retire son suffixe, aussi appelée préfixe avec radical [Porter, 1980]. L'algorithme le plus connu, pour transformer un mot en son radical, est celui développé par Porter en 1980, il utilise une cinquantaine de règles réparties en cinq étapes qui transforment un mot en son radical. Il existe aussi celui de Paice/Husk et Carry, qui lui est propre au français.
   — La lemmatisation, elle, transforme un mot en son lemme. C'est la forme non conjuguée et non accordée d'un mot [Hull and Grefenstette, 1996]. Cette technique varie énormément selon la richesse morphologique de langue concernée. La classification des langues selon leur complexité, par rapport à la lemmatisation, place l'anglais comme la plus simple et l'hébreu comme la plus compliquée. Le français se trouve entre les deux [Namer, 2000].

mot	racine	lemme
apporte	apport	apporter

TABLE 4.3 – Racinisation et Lemmatisation

   Comment choisir entre les deux ? La racinisation est très rapide et agrège des mots assez différents. Par contre, il est très rare d'avoir des formes radicales qui ont du sens, il est donc difficile de refaire un traitement sémantique derrière. La lemmatisation s'inscrit dans un processus d'analyse syntaxique plus élaboré puisqu'elle détermine la nature des mots. Elle est plus difficile à mettre en place que la racinisation, car elle nécessite un dictionnaire associant les mots aux lemmes. De plus, elle ne permet pas l'agrégation d'autant de termes que la racinisation. Nous pouvons illustrer les deux méthodes avec notre phrase d'exemple :

fait	beau	soleil	rayon

TABLE 4.4 – Racinisation

faire	beau	soleil	rayonner

TABLE 4.5 – Lemmatisation

   Dans le cadre de l'extraction d'activités, nous devons identifier les verbes d'action, car ils constituent le premier item des activités. La lemmatisation est donc requise pour déterminer quels mots sont des verbes puis s'ils sont des verbes d'action.

Nous avons maintenant une base de séquences où chaque séquence correspond à une offre d'emploi tokenisée, nettoyée des mots vides et lemmatisée avec identification des verbes d'action.
Toujours dans le but d'extraire des activités des offres d'emploi, nous cherchons à avoir une base

avec des séquences représentatives des offres d'emploi. Nous avons donc choisi d'introduire deux filtres pour écarter les séquences jugées non pertinentes pour l'objectif que nous nous sommes fixé. Le premier filtre (F_1), écarte les séquences ne contenant pas au moins 10 mots et 2 verbes afin de ne conserver que les offres d'emplois avec suffisamment d'information, évitant ainsi de fouiller inutilement de nombreuses offres. Le filtre F_1 permet d'écarter 95328 séquences (12% des séquences de la base).
Nous avons utilisé un deuxième filtre pour pallier le biais que nous avons introduit en constituant le notre corpus à partir de plusieurs sources. En effet, les différents sites d'offres d'emploi ont l'habitude de récupérer leurs offres à partir de sources qui sont souvent les mêmes. En conséquence, notre corpus comporte de nombreuses offres identiques. Notre deuxième filtre F_2 écarte donc toutes les séquences identiques à une autre. Le filtre F_2 permet d'écarter 258 431 séquences (37% des séquences). L'importance de ce nombre confirme bien le besoin d'un tel filtre. Les résultats de cette préparation des données sont récapitulés dans le tableau 4.6.

filtre	nb. séq.	moy. motifs/séq.	médiane	min	max
-	797 264	62,79	50	0	3 082
F_1	701 936	68,74	55	10	3 082
$F_1 + F_2$	443 505	73,34	58	10	3 082

TABLE 4.6 – Préparation des données

4.2.3 Extraction des activités

La phase de préparation des données a permis de transformer notre corpus de 800k offres d'emploi en une base de séquences de près de 444k séquences, contenant les offres d'emploi tokenisées, nettoyées des mots vides et lemmatisées avec identification des verbes d'action. Nous passons maintenant à l'extraction des activités dans notre base de séquences.
Nous avons identifié un biais potentiel lié au manque de structure de l'information. En effet, il apparaît que le champ description de l'entreprise et le champ description du poste sont très rarement distingués. La plupart du temps, ils sont réunis dans le champ description du poste. Finalement, nous n'avons aucun moyen de faire la différence entre les activités propres aux entreprises et les activités propres aux postes. Sachant que la partie description de l'entreprise est toujours présente au début de la description et occupe jusqu'à la moitié de la description, nous allons comparer les résultats de deux fouilles d'activités : une fouille sur les séquences entières et une fouille sur la deuxième moitié des séquences où il n'y avait pas de descriptions de l'entreprise dans l'offre d'emploi correspondante. Nous avons donc deux bases de séquences issues de notre corpus d'offres d'emploi, BS et BMS (M pour moitié).

L'objectif de cette section est l'extraction d'activités fréquentes dans les deux bases de séquences BS et BMS en utilisant l'algorithme C3Ro. Nous devons donc fixer les paramètres min_sup, k, λ, δ et ζ pour rendre possible cette fouille et de l'optimiser. Comme nous cherchons à extraire un maximum d'activités tout évitant les activités trop rares, nous avons choisi de conserver les activités apparaissant au moins 300 fois : $min\_sup = 300$. Nous avons vu dans le chapitre précédent que les activités peuvent être assimilées à des motifs contigus commençant par un verbe d'action et de taille supérieure ou égale à 3. Nous définissons donc $A = \{$"verbe d'action au début de chaque motif","taille supérieure ou égale à 3"$\}$ et nous fixons $\zeta = A$. Tout comme dans l'expérimentation du chapitre précédent, nous fixons $k = 1$, $\lambda = 0,8$ et $\delta = 0,5$. Finalement, nous exécutons une fouille de motifs 0,6-clos 1-contigus 0,5-robustes satisfaisant l'ensemble des

contraintes de A pour extraire les activités des bases BS et BMS. Cette fouille illustre bien les possibilités offertes par la généricité de l'algorithme C3Ro puisqu'elle permet l'utilisation d'une combinaison de nombreuses contraintes pertinentes pour la fouille des activités.

Le tableau 4.7 montre les caractéristiques des activités extraites de chaque base alors que la figure 4.4 indique la proportion d'activités extraites en fonction de leur longueur. Nous constatons plusieurs différences entre les activités extraites de chaque base. En effet, alors que les séquences de la base BMS correspond à la deuxième moitié de chaque séquence de la base A, le nombre d'activités extraites lui, est trois fois moins important sur BMS que sur BS. De plus, la moyenne et la médiane de mots par activité sont également significativement inférieures sur BMS que sur BS. Nous expliquons cette différence par le biais des descriptions d'entreprises dans l'ensemble BS qui est théoriquement plus important que dans l'ensemble BMS. Plus précisément, les grandes entreprises publient un grand nombre d'offres d'emploi, or la partie description de l'entreprise est toujours la même, quel que soit l'emploi recherché. La conséquence directe est l'apparition d'activités fréquentes très longues. Cette analyse est confirmée par le tableau 4.8 qui

base	nb. act.	moy. mots/act.	médiane	max	moy. sup.	médiane sup.	$3^{\text{ème}}$ quartile sup.	max sup.
BS	1539	9,06	4	124	839,9	478	796,5	44 515
BMS	558	6,44	3	76	943,8	488,5	784,5	44 515

TABLE 4.7 – Informations sur les activités extraites de BS et BMS

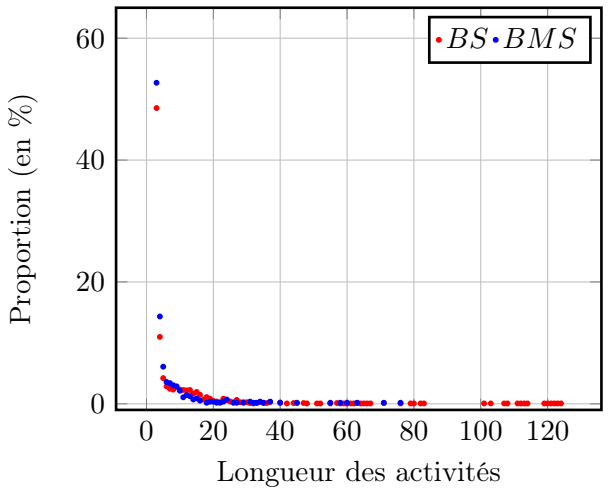


FIGURE 4.4 – Répartition des activités par longueur

montre les 10 activités extraites les plus fréquentes dans les deux bases. Nous constatons que 7 activités sur 10 sont communes aux deux bases (indiquées en bleu dans le tableau). L'intitulé, la longueur et la très haute fréquence de ces activités nous indiquent que 5 d'entre elles proviennent des descriptions d'entreprise fréquemment rencontrées dans les offres d'emploi (intitulés grisés dans le tableau). Sur cet échantillon de 10 activités, la base de séquences BMS n'a pas permis d'écarter les activités que nous considérons comme non pertinentes. Cependant, la différence de longueur moyenne que nous avons mentionnée plus haut, ajoutée à la différence du nombre de grandes activités visible sur la figure 4.4, nous montre que la base BMS semble malgré tout avoir significativement diminué le nombre d'activités issues des descriptions d'entreprise. Nous ajoutons que les activités présentées dans le tableau 4.8 ne sont pas représentatives de l'ensemble des activités extraites au vu du support moyen et du $3^{\text{ème}}$ quartile qui sont autour de 800. D'ailleurs,

BS	
activités	support absolu
etudier competences egales candidature situation	44515
etudier competences egales candidature situation handicapduree mission	34142
etudier competences egales candidature situation handicapduree mission mois	24381
garder enfant baby sitting	16757
tacher confiees garder enfant baby sitting	13422
etudier competences egales candidature situation handicapduree mission mois renouvelable	13155
etudier competences egales candidature situation handicap	10452
tacher confiees garder enfant baby sitting sortir ecole	10291
justifier experience an	9269
garder enfant lundi	7491
BMS	
etudier competences egales candidature situation	44513
etudier competences egales candidature situation handicapduree mission	34142
etudier competences egales candidature situation handicapduree mission mois	24381
etudier competences egales candidature situation handicapduree mission mois renouvelable	13155
etudier competences egales candidature situation handicap	10449
justifier experience an	8319
garder enfant lundi	7175
faire difference www springfrance com	7050
maitriser outil informatique	5163
etudier competences egales candidature situation handicapduree mission semaine	4489

TABLE 4.8 – Activités extraites les plus fréquentes de BS et BMS

le $3^{\text{ème}}$ quartile du support sur les deux bases est inférieur à la moyenne, ce qui nous indique que les activités les plus fréquentes sont significativement plus fréquentes que les autres et très peu nombreuses.
Nous remarquons que la moyenne de support des activités extraites de la base BMS est plus de 100 points supérieure à celle de la base BM alors que le $3^{\text{ème}}$ quartile du support est très légèrement supérieur sur la base BM. Nous pouvons en déduire que la suppression de la première moitié des offres d'emploi écarte majoritairement des activités avec un support faible.
Pour avoir une meilleure idée des activités extraites, nous illustrons dans le tableau 4.9, les verbes d'action les plus utilisés. Nous constatons que seulement 4 des verbes d'action les plus utilisés sur 10 sont communs aux deux bases, ce qui confirme que la suppression de la première moitié des séquences a un impact sur la nature des activités extraites. Finalement, il nous apparaît complexe d'évaluer en l'état la pertinence des activités extraites des bases BS et BMS. Nous proposons donc une analyse basée sur l'encodage des offres d'emploi de notre corpus avec les activités que nous venons d'extraire.

4.3 Évaluation de la pertinence des activités extraites

L'objectif de cette section est de déterminer l'apport des activités extraites des bases BS et BMS. Comme il semble peu pertinent d'évaluer cet apport sans prendre en compte les offres d'emploi, nous avons donc choisi de réexprimer les offres d'emploi uniquement avec les activités extraites des bases BS et BMS pour les comparer aux offres originelles. Nous choisissons d'évaluer l'apport des activités extraites en regroupant les offres en se basant sur leur contenu (activités ou description de l'offre), et en analysant le titre des offres ainsi regroupés avec les deux types de contenu. La comparaison des titres des regroupements d'offres d'emploi originelles

BS	
activités	proportion(%)
garder	17,7
assurer	8,9
faire	7,6
sortir	7,5
rechercher	6,4
gouter	5,8
mettre	5,6
tacher	5,2
postuler	5,1
proposer	5,0
BMS	
assurer	7,2
faire	7,0
postuler	5,7
mettre	5,2
justifier	5,0
garder	4,8
entreprendre	4,3
envoyer	3,9
maitriser	3,6
savoir	3,4

TABLE 4.9 – Verbes d'action les plus utilisés dans les activités extraites de BS et BMS

et exprimées uniquement avec des activités permettra alors d'identifier l'impact et l'apport des activités extraites.

Pour mettre en place cette évaluation, nous allons d'abord encoder les offres d'emploi avec les activités que nous venons d'extraire des bases BS et BMS pour obtenir deux nouvelles bases de séquences BA et BMA. Dans un second temps, nous réduisons la dimension de chacune des bases dans le même espace [Le and Mikolov, 2014] pour rendre la clusterisation possible avec nos contraintes matérielles (temps et mémoire). Nous finissons par effectuer des clusterisations sur chacune des bases réduites afin de comparer les titres des offres des clusters de chaque base entre eux pour évaluer l'impact de l'encodage et donc l'apport des activités extraites.

4.3.1 Encodage des offres d'emploi avec les activités extraites

Afin d'encoder les offres d'emploi des bases BS et BMS, nous identifions et ne conservons que les offres d'emploi comportant au moins une activité extraite. Ainsi, les activités extraites de la base BS sont présentes dans 308 745 offres d'emploi (soit 70% des offres de BS) tandis que celles de la base BMS sont présentes dans 186 399 offres (soit 42% des offres de BS). Nous créons donc deux nouvelles bases de séquences OA (Offres d'emploi avec Activités) et MOA (Moitiés d'Offres d'emploi avec Activités) où chaque séquence correspond à la liste des activités présentes dans l'offre d'emploi originelle. Lors de la constitution de chaque liste d'activités représentant une offre d'emploi, nous n'avons conservé que les activités non incluses dans une autre activité de la liste, afin d'éviter de créer trop de redondance. La base OA contient donc 308 745 séquences et la base MOA en contient 186 399. Comme nous souhaitons comparer ces bases à nos bases initiales BS et BMS, nous devons retirer les offres d'emploi qui ne sont pas présentes dans OA pour BS et dans MOA pour BMS pour ainsi former BS' et BMS'. Nous finissons par avoir

quatre nouvelles bases OA, MOA, BS', BMS' décrites dans le tableau 4.10.

base	type de séq.	nombre de séq
BS	description du poste	443 505
BS'	description du poste des offres contenant au moins une activité extraite	308 745
BMS	$2^{\text{ème}}$ moitié de description du poste	443 505
BMS'	$2^{\text{ème}}$ moitié de description du poste des offres contenant au moins une activité extraite	186 399
OA	activités extraites dans la description du poste	308 745
MOA	activités extraites dans la $2^{\text{ème}}$ moitié de description du poste	186 399

TABLE 4.10 – Descriptions des bases de séquences

4.3.2 Réduction de dimension et clusterisation des bases de séquences

Notre but est de comparer respectivement les bases BS', BMS' et les bases OA, MOA pour déterminer à la fois l'apport des activités que nous avons extraites, mais également pour valider l'intérêt de ne s'intéresser qu'à la deuxième moitié de la description des offres d'emploi. Afin de permettre la clusterisation de nos différentes bases en diminuant le coût de leur gestion en mémoire, nous avons opté pour un modèle nommé doc2vec introduit dans [Le and Mikolov, 2014]. Ce modèle est utilisé pour faire du plongement lexical (*word embedding*) en basant sur des réseaux de neurones à deux couches, entraînés pour déduire le contexte linguistique des mots. En simplifiant, doc2vec permet de transformer des documents (c'est-à-dire des ensembles de mots) et le vocabulaire (c'est-à-dire l'ensemble de mots distincts) utilisé dans ces documents en vecteurs dans un espace dont le nombre de dimensions est spécifié. Les coordonnées de chaque mot et de chaque document dans cet espace permettent notamment d'effectuer des calculs de similarité basés sur la distance entre les vecteurs (voir figure 4.5). Nous utilisons donc le modèle doc2vec pour opérer un plongement lexical de nos quatre bases en spécifiant chaque offre comme étant un document. Nous avons choisi de fixer le nombre de dimensions à 300 pour prioriser la précision plutôt que la consommation mémoire [Lau and Baldwin, 2016]. Nous obtenons donc quatre matrices $M_{BS'}$, M_{OA}, $M_{BMS'}$, M_{MOA} de 300 colonnes avec un nombre de lignes correspondant au nombre de séquences des bases BS', OA (308 745) et BMS', MOA (186 399). L'objectif est

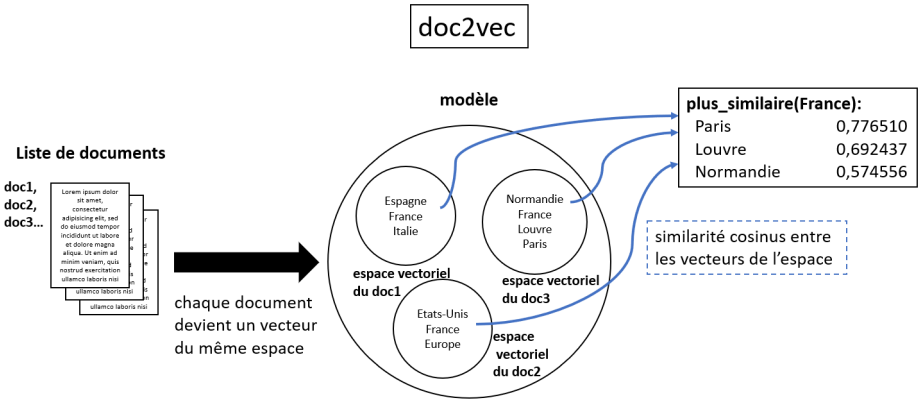


FIGURE 4.5 – Illustration du modèle doc2vec

maintenant de comparer respectivement les matrices $M_{BS'}$, $M_{BMS'}$ et M_{OA}, M_{MOA} pour déterminer l'apport de l'encodage des offres d'emploi avec les activités extraites au préalable. Nous avons choisi de procéder à une clusterisation des quatre matrices et de comparer la composition des clusters ayant le plus d'offres communes pour ensuite d'évaluer la pertinence de la présence

des offres non communes dans le même cluster. Nous utilisons une méthode de clusterisation k-means [Kanungo et al., 2002] avec un k variant de 100 à 300 pour identifier la valeur de k donnant le meilleur recouvrement entre les clusters des matrices comparées. Nous notons $C_{M_{BS'}}$, $C_{M_{BMS'}}$, $C_{M_{OA}}$, $C_{M_{MOA}}$ l'ensemble des clusters crées en se basant respectivement sur les matrices $M_{BS'}$, M_{OA}, $M_{BMS'}$, M_{MOA}. Nous introduisons une mesure d'évaluation du recouvrement 4.3.1 entre les clusters.

Definition 4.3.1. Le **score de recouvrement** entre deux clusters C et C' est défini par :

$$S_R(C, C') = \frac{|C \cap C'|}{|C| + |C'|} * 100$$

Les résultats des meilleurs scores de recouvrement entre les clusters des bases $C_{M_{BS'}}$, $C_{M_{OA}}$ et $C_{M_{BMS'}}$, $C_{M_{MOA}}$ notés $C^*_{M_{BS'}}$, $C^*_{M_{OA}}$ et $C^*_{M_{BMS'}}$, $C^*_{M_{MOA}}$ respectivement, sont récapitulés dans le tableau 4.11. Nous constatons que les résultats obtenus avec les matrices $M_{BMS'}$ et M_{MOA}

k / score	100	150	200	250	300
$S_R(C^*_{M_{BS'}}, C^*_{M_{OA}})$	8,0	4,6	8,4	7,9	7,0
$S_R(C^*_{M_{BMS'}}, C^*_{M_{MOA}})$	13,9	11,2	14,0	12,9	12,8

TABLE 4.11 – Meilleurs scores entre les clusters $C_{M_{BS'}}$, $C_{M_{OA}}$ et $C_{M_{BMS'}}$, $C_{M_{MOA}}$

sont significativement meilleurs que ceux obtenus avec les matrices $M_{BS'}$et M_{OA}. Nous pouvons donc en déduire que concentrer la fouille sur la deuxième moitié des offres d'emploi pour pallier au problème des descriptions d'entreprise, dans notre corpus, semble améliorer l'extraction des activités.
Le nombre de clusters $k = 200$ donne les meilleurs scores dans les deux configurations, nous allons donc nous concentrer sur la clusterisation des matrices $M_{BMS'}$ et M_{MOA} avec $k = 200$ pour étudier les clusters avec le meilleur score de recouvrement. Le tableau 4.12 nous montre les caractéristiques des deux clusters de $C_{M_{BMS'}}$ et $C_{M_{MOA}}$, respectivement $C^*_{M_{BMS'}}$ et $C^*_{M_{MOA}}$ ayant le plus haut score de recouvrement quand $k = 200$. Nous allons maintenant étudier en détail les

cluster	nombre d'éléments
$C^*_{M_{BS'}}$	24
$C^*_{M_{MOA}}$	69
$C^*_{M_{BS'}} \cap C^*_{M_{MOA}}$	13

TABLE 4.12 – Nombre d'éléments des clusters $C^*_{M_{BS'}}$, $C^*_{M_{MOA}}$ et $C^*_{M_{BS'}} \cap C^*_{M_{MOA}}$

offres d'emploi faisant partie des clusters $C^*_{M_{BS'}}$, $C^*_{M_{MOA}}$ et $C^*_{M_{BS'}} \cap C^*_{M_{MOA}}$. Nous commençons par effectuer une fouille de motifs séquentiels fréquents sur le titre des offres d'emploi composant ces clusters. Nous observons que le nom d'entreprise "FSI" est omniprésent et le terme "Conseil" est majoritairement présent (voir tableau 4.13) dans le titre des offres d'emploi des ensembles $C^*_{M_{BS'}} - (C^*_{M_{BS'}} \cap C^*_{M_{MOA}})$ et $C^*_{M_{BS'}} \cap C^*_{M_{MOA}}$. Dans l'ensemble $C^*_{M_{MOA}} - (C^*_{M_{BS'}} \cap C^*_{M_{MOA}})$, les termes "Commercial", "Responsable" et "Consultant" sont les plus fréquents, mais ne sont pas complètement représentatifs de l'ensemble, car leur support relatif ne dépasse pas les 18%. Nous constatons donc que les offres communes aux clusters $C^*_{M_{BMS'}}$ et $C^*_{M_{MOA}}$ portent sur des emplois de conseils de l'entreprise FSI, tout comme la grande majorité des offres du cluster $C_{M_{BMS'}}$. Cependant, en se basant sur les titres des offres d'emploi, nous ne pouvons évaluer la pertinence

$C^*_{M_{BS'}} - (C^*_{M_{BS'}} \cap C^*_{M_{MOA}})$	
motif séquentiel	support relatif
FSI	81%
Assurance	64%
Conseil	55%
FSI Assurance	55%
$C^*_{M_{MOA}} - (C^*_{M_{BS'}} \cap C^*_{M_{MOA}})$	
Commercial	18%
Responsable	11%
Consultant	9%
$C^*_{M_{BS'}} \cap C^*_{M_{MOA}}$	
FSI	100%
FSI Banque	54%
Conseil FSI	54%

TABLE 4.13 – Motifs séquentiels les plus fréquents dans titres d'offres d'emploi des clusters $C^*_{M_{BS'}}$, $C^*_{M_{MOA}}$ et $C^*_{M_{BS'}} \cap C^*_{M_{MOA}}$

des offres d'emploi non communes du cluster $C^*_{M_{MOA}}$.
Nous nous intéressons maintenant aux activités composant les offres d'emploi des clusters $C^*_{M_{BS'}}$, $C^*_{M_{MOA}}$ et $C^*_{M_{BS'}} \cap C^*_{M_{MOA}}$. Nous effectuons à nouveau une fouille de motifs séquentiels fréquents, mais cette fois sur les activités de chaque offre. Les résultats de cette fouille sont visibles dans le tableau 4.14. Les activités indiquées en bleu proviennent des offres de l'ensemble $C^*_{M_{BS'}} - (C^*_{M_{BS'}} \cap C^*_{M_{MOA}})$, celles en vert des offres de l'ensemble $C^*_{M_{MOA}} - (C^*_{M_{BS'}} \cap C^*_{M_{MOA}})$, tandis que celles en orange sont communes aux deux dans l'ensemble $C^*_{M_{BS'}} \cap C^*_{M_{MOA}}$. Parmi les activités fréquentes de l'intersection des deux clusters, trois proviennent de l'ensemble $C^*_{M_{BS'}} - (C^*_{M_{BS'}} \cap C^*_{M_{MOA}})$, cinq de l'ensemble $C^*_{M_{MOA}} - (C^*_{M_{BS'}} \cap C^*_{M_{MOA}})$ et seulement une seule est commune à ces deux ensembles. Le cluster $C^*_{M_{MOA}}$ ne semble donc pas uniquement reposer sur des activités présentes dans la description de l'entreprise FSI mais également sur des activités propres aux différents emplois à pourvoir dans ces offres. Ainsi, dans le cluster $C^*_{M_{MOA}}$, on trouve de nombreux métiers différents (voir tableau 4.13) comme "commercial", "responsable" ou "consultant" ayant la caractéristique d'avoir des activités communes comme "participer developpement client", "rediger offrir commercial" ou "piloter projet client".

Ce résultat explique la différence entre le cluster $C^*_{M_{BS'}}$ et le cluster $C^*_{M_{MOA}}$. En effet, en se basant directement sur la description de l'offre d'emploi, ou même la moitié dans notre configuration, la description des grandes entreprises est tellement spécifique et fréquente qu'elle prend le pas sur le description du poste et biaise l'information extraite des offres d'emploi lors d'une fouille de données.
Dans ce contexte de fouille, l'absence de structuration systématique de l'information contenue dans les offres d'emploi entrave l'extraction de l'information uniquement relative aux caractéristiques de l'emploi. Notre approche visant à se concentrer uniquement sur les activités dans les offres d'emploi, c'est-à-dire sur le cœur de l'emploi, permet de diminuer l'impact de ce biais et d'accéder à une partie des caractéristiques de l'emploi.
Nous analysons maintenant les caractéristiques des activités extraites, afin d'évaluer leur pertinence.

$C^*_{M_{BS'}} - (C^*_{M_{BS'}} \cap C^*_{M_{MOA}})$	
activités	support relatif
justifier experience an	91%
assurer suivi client	64%
doter bon relationel	45%
justifier experience reussir an	45%
participer developpement client	36%
identifier besoin client	36%
assurer suivi client justifier experience reussir an	27%
justifier experience professionnel an	27%
justifier experience reussir an permettre etre	18%
gérer relation client	18%
$C^*_{M_{MOA}} - (C^*_{M_{BS'}} \cap C^*_{M_{MOA}})$	
participer developpement client	46%
assurer developpement client	20%
rediger offrir commercial	20%
justifier experience an	18%
contribuer projet client	16%
piloter projet client	16%
mettre place outil	14%
analyser besoin client	14%
accompagner developpement client	14%
animer groupe travail	13%
$C^*_{M_{BS'}} \cap C^*_{M_{MOA}}$	
justifier experience an	100%
participer developpement client	100%
piloter projet client	77%
justifier experience reussir an	70%
justifier experience reussir an permettre etre	46%
rediger offrir commercial	31%
piloter projet client participer developpement client	31%
contribuer projet client	31%
gerer equipe participer developpement client	23%
justifier experience reussir an analyser donnees permettre etre	23%

TABLE 4.14 – Activités les plus fréquentes des clusters $C^*_{M_{BS'}}$, $C^*_{M_{MOA}}$ et $C^*_{M_{BS'}} \cap C^*_{M_{MOA}}$

4.3.3 Analyse des caractéristiques des activités

La fouille des bases de séquences BS et BMS a permis d'extraire respectivement 1 539 et 558 activités. Nous avons vu qu'une majorité des activités les plus fréquentes, sur ces deux bases, sont issues de la partie description de l'entreprise (voir tableau 4.8). Ces activités sont souvent non pertinentes, car elles présentent des mots trop spécifiques au contexte de l'entreprise et elles sont également trop longues pour les causes que nous avons évoquées précédemment. En revanche, certaines activités extraites sont bel et bien pertinentes, comme celles mises en évidence lors de notre clusterisation (voir tableau 4.14).
L'utilisation de l'algorithme C3Ro pour fouiller les motifs séquentiels 0,6-clos 1-contigus 0,5-robustes satisfaisant l'ensemble des contraintes de A permet donc bien d'extraire des activités sans en garantir la pertinence. En aval de la fouille, les activités extraites doivent être étudiées manuellement pour écarter celles qui ne sont pas considérées comme pertinentes. Finalement, nous pouvons fournir une liste d'activités contenant des activités pertinentes et des activités non pertinentes à écarter manuellement.

4.4 Synthèse

Ma thèse s'est déroulée dans le cadre d'un partenariat avec une entreprise du domaine de la gestion des compétences. L'un des objectifs de cette thèse était donc d'apporter une contribution applicative utilisable dans le contexte de l'entreprise. Dans cette section, nous avons présenté ce contexte et montré que la fouille d'activités dans les offres d'emploi, sur le marché de l'emploi en ligne, constituait une application pertinente. D'une part, du côté de l'entreprise, car les activités fouillées représentent une valeur ajoutée indéniable lors de l'élaboration des référentiels d'activités dans le cadre de la mise en place de la démarche compétence. D'autre part, car la fouille d'activités est une fouille dans un contexte de grands volumes de données bruitées, qui est un contexte approprié pour l'utilisation de l'algorithme C3Ro que nous avons introduit dans cette thèse.
Dans le cadre de cette fouille d'activités, nous avons constitué un corpus d'offres d'emploi contenant près de 800 000 offres. Nous avons présenté et illustré les différentes étapes de préparation des données textuelles en amont d'une fouille. Nous avons identifié un biais potentiel lors de l'extraction des activités due à l'absence de structuration de l'information dans les offres d'emploi. Cette absence de structure entraîne une impossibilité de distinguer la description de l'entreprise proposant l'emploi de la description de l'emploi proposé, ce qui entraîne l'apparition de nombreuses activités non pertinentes. Nous avons donc proposé une deuxième fouille d'activités, basée uniquement sur la deuxième moitié des offres d'emploi pour diminuer l'impact de ce biais. En nous appuyant sur ce résultat, nous pensons qu'il serait très pertinent de proposer une détection automatique de la partie de l'offre qui correspond à la description de l'entreprise pour éliminer complètement ce biais.
Nous avons ensuite proposé d'encoder les offres d'emploi en fonction des activités extraites et d'opérer une clusterisation pour comparer les clusters des offres encodées avec les clusters des offres non encodées. Cette comparaison a montré que les deux clusterisation les plus similaires étaient basées sur la deuxième moitié des offres. De plus, l'étude des clusters les plus proches a permis de montrer que l'encodage en activités permet de regrouper des emplois différents, mais avec des caractéristiques proches.
J'ai finalement montré que mes listes d'activités contiennent bien des activités pertinentes, j'ai donc transmis ces listes à l'entreprise pour lui laisser le soin d'écarter les activités jugées non pertinentes.

Chapitre 5

Conclusion et perspectives

En dépit de la démocratisation de l'accès aux données et aux méthodes de fouille de données, les utilisateurs sont presque systématiquement confrontés à des résultats inaccessibles rendant impossible l'extraction de l'information contenue dans les données. En effet, en pratique, les résultats proposés par la fouille de données sont souvent complexes et conséquents, ce qui les rend à la fois difficiles à appréhender par un utilisateur, mais également difficiles à obtenir compte tenu des contraintes de temps et de matériel.
De notre point de vue, la fouille de données est donc confrontée à deux défis pour devenir plus accessible. D'une part, proposer des résultats appréhendables par un utilisateur, c'est-à-dire compréhensibles et de taille exploitable par un humain. D'autre part, proposer une fouille efficiente capable de s'adapter aux contraintes d'un utilisateur.
Nous avons identifié la fouille de motifs comme étant une méthode de fouille de données souvent incontournable dans le processus d'extraction de l'information, puisqu'elle permet d'extraire des éléments représentatifs des données (au sens de critères définis par l'utilisateur). La fouille de motifs nous apparaît donc primordiale dans notre démarche de rendre les résultats de la fouille de données plus facilement appréhendables par les utilisateurs. Bien que cette méthode et ses résultats soient compréhensibles par un utilisateur, les résultats, quant à eux, sont souvent très conséquents et donc non appréhendables par un utilisateur, en plus d'être coûteux, voire impossible à extraire.
L'utilisation de contraintes est la direction la plus prometteuse adoptée par la littérature pour améliorer à la fois l'appréhendabilité et l'efficience de la fouille de motifs. L'utilisateur identifie les caractéristiques de l'information qu'il souhaite extraire et les spécifie sous la forme d'un ensemble de contraintes. Pourtant, lors d'une fouille de motifs satisfaisant des contraintes, la combinaison de contraintes désirée par l'utilisateur pose régulièrement problème en raison de la compatibilité des algorithmes utilisés pour permettre cette combinaison mais également parce que cette combinaison de contraintes n'est pas toujours adaptable aux caractéristiques des données fouillées tel le bruit. De plus, l'utilisation d'un ensemble de contraintes ne réduit pas toujours suffisamment l'ensemble des motifs extraits pour garantir l'efficience de la fouille et l'appréhendabilité de ses résultats. Nous avons identifié la combinaison de la contrainte de clôture et de la contrainte de contiguïté comme étant une combinaison très efficace pour améliorer l'efficience de la fouille et l'appréhendabilité de ses résultats sans dépendre du contexte applicatif de l'utilisateur. Cependant, cette combinaison de contraintes résiste peu aux perturbations occasionnées par le bruit dans les données, comme la perte, la substitution ou l'addition d'items dans les motifs ce qui peut avoir la conséquence inverse : trop réduire l'ensemble, et donc ne pas extraire certaines informations importantes.

Dans cette thèse, nous nous sommes focalisés sur la question scientifique suivante : **"Comment proposer une fouille de motifs efficiente et résistante au bruit permettant d'extraire un ensemble des motifs appréhendable par l'utilisateur ?"** Pour répondre à cette question, nous avons introduit deux nouvelles contraintes, visant à préserver l'apport des contraintes de clôture et de contiguïté, dans un contexte de données bruitées, associées à un algorithme générique :

- **La contrainte de robustesse** implique l'utilisation de *wildcards* lors de la fouille de motifs contigus. Elle est basée sur un ratio d'allègement permettant de fixer la limite de la proportion d'occurrences de chaque motif utilisant une wildcard, ce ratio garantit donc la proportion minimum d'occurrences contiguës de chaque motif. Cette contrainte permet de pallier la perturbation d'items additionnels tout en limitant l'introduction de motifs non pertinents dans le contexte d'une fouille de motifs contigus extraits grâce à l'utilisation de wildcards.
- **La contrainte de clôture allégée** est basée sur un ratio d'allègement qui permet de faire un compromis entre la contrainte de maximisation et la contrainte de clôture, c'est-à-dire un compromis entre la perte d'information liée au support des motifs et la réduction du nombre de motifs extraits. Cette contrainte est une réponse aux substitutions et pertes d'items dus au bruit.
- **L'algorithme C3Ro** intègre nativement les contraintes de clôture, de clôture allégée, de contiguïté et de robustesse tout en permettant l'intégration d'un ensemble de contraintes préfixes-monotones. C3Ro est un algorithme générique capable de fouiller des motifs maximaux, clos et clos contigus grâce à des paramètres que sont le nombre de wildcards, le ratio de robustesse, le ratio d'allègement et l'ensemble de contraintes préfixes-monotones. C3Ro propose donc une fouille de motifs capable d'être adaptée aux besoins de l'utilisateur grâce à ses nombreux paramètres.

Les contributions scientifiques de cette thèse ont été validées dans deux articles de conférences internationales ou sont en cours de validation dans un article de journal international.

L'évaluation expérimentale a permis de mettre en évidence la pertinence de l'utilisation des deux contraintes que nous avons introduites dans le cadre d'une fouille de motifs séquentiels clos contigus dans un contexte de données bruitées issues du Web. Ces contraintes nous semblent d'autant plus pertinentes qu'elles permettent à l'utilisateur d'effectuer une fouille en conservant l'appréhendabilité et l'efficience des contraintes de clôture et de contiguïté, et ce sur des données bruitées, ce qui est un contexte de fouille répandu.
De plus, l'évaluation a montré que l'algorithme C3Ro est capable d'extraire des motifs séquentiels clos contigus avec des *wildcards* de grandes bases de données en un temps raisonnable (moins d'une heure), ce que, à notre connaissance, aucun autre algorithme de fouille de motifs séquentiels de la littérature n'est capable de faire, majoritairement dû à la trop grande consommation en mémoire.

Nous avons illustré l'utilisation de l'algorithme C3Ro dans une application pour extraire les activités d'offres d'emploi sur un grand corpus. Cette application est destinée à faciliter l'élaboration des référentiels d'activités lors de la mise en place de la démarche compétences par l'entreprise People Compétences.

Le travail mené pendant cette thèse a permis de mettre en évidence plusieurs perspectives de recherche à court et moyen terme toujours avec l'objectif de rendre la fouille de données plus accessible à l'utilisateur.

À court terme :

* Nous avons proposé l'algorithme C3Ro qui est un algorithme générique permettant à l'utilisateur d'adapter la fouille de motifs séquentiels fréquents à ses besoins. Cependant, la fréquence n'est pas la seule mesure permettant d'identifier les motifs potentiellement pertinents. De nombreuses autres mesures comme la confiance, l'intérêt ou la couverture [Geng and Hamilton, 2006] permettent d'obtenir une autre représentation des données pouvant être plus pertinente pour les besoins de l'utilisateur. Toujours dans l'optique de proposer une fouille plus accessible à l'utilisateur, nous souhaiterions permettre à l'algorithme C3Ro d'effectuer des fouilles de motifs en se basant sur d'autres mesures que la fréquence pour l'extraction des motifs. Le type de mesure deviendrait simplement un autre paramètre que l'utilisateur pourrait modifier afin d'adapter au mieux la fouille à ses besoins.

À moyen terme :

* L'algorithme C3Ro nécessite que l'utilisateur fixe plusieurs paramètres, parmi lesquels k le nombre de wildcards, δ le ratio de robustesse et λ le ratio d'allègement. La valeur de ces trois paramètres dépend à la fois des besoins de l'utilisateur et des caractéristiques des données fouillées. Comme nous l'avons montré dans nos expérimentations, ces valeurs sont définies expérimentalement en observant les résultats obtenus. Afin de supprimer cette étape chronophage d'observation des résultats et d'optimiser la valeur de ces paramètres, nous souhaiterions concevoir une méthode permettant de fixer automatiquement la valeur de ces trois paramètres. Cette méthode pourrait dépendre d'une part, de la proportion de chaque type de perturbations lié au bruit dans les données fouillées (méthode d'identification à définir), car en fonction du type (addition ou perte et substitution) c'est le ratio de robustesse ou le ratio d'allègement qu'il faut faire varier. Elle pourrait dépendre d'autre part, du nombre de motifs extraits pour faire varier le nombre de *wildcards* et le ratio d'allègement.

À long terme :

* Nous avons proposé l'utilisation de *wildcards* lors de la fouille de motifs séquentiels contigus pour pallier l'addition d'items dans un contexte de données bruitées. Cependant, les *wildcards* pourraient avoir une autre utilité, plus particulièrement la position des *wildcards* dans les motifs extraits, notamment dans le domaine de la détection d'émergence. Ce domaine est de plus en plus étudié dans la littérature scientifique [Dong and Li, 1999, Feil and Fraga, 2012], car il présente de nombreux enjeux. En particulier, la capacité à déceler une émergence au plus tôt permet d'anticiper et de prendre les mesures appropriées en réponse à cette émergence. Ainsi, dans le cadre d'une fouille de motifs possédant des marqueurs temporels, l'évolution au cours du temps de la position des *wildcards* dans les motifs extraits pourrait être un indicateur permettant de détecter précocement l'émergence de nouveaux motifs. Par exemple, d'un motif fréquent $\langle a, b, c^* \rangle$ régulièrement extrait au cours du temps en utilisant une *wildcard* pour la découverte de l'item c, il serait possible d'identifier d'autres motifs comme $\langle a, b, d, c \rangle$ ou $\langle a, b, e, c \rangle$ qui ne sont pas encore fréquents, mais qui sont en train d'émerger.

Bibliographie

[Abboud et al., 2015] Abboud, Y., Boyer, A., and Brun, A. (2015). Predict the emergence : Application to competencies in job offers. In *Tools with Artificial Intelligence (ICTAI), 2015 IEEE 27th International Conference on*, pages 612–619. IEEE.

[Abboud et al., 2017] Abboud, Y., Boyer, A., and Brun, A. (2017). CCPM : A Scalable and Noise-Resistant Closed Contiguous Sequential Patterns Mining Algorithm. In *13th International Conference on Machine Learning and Data Mining MLDM 2017*, volume 89, page 15, New York, United States.

[Aggarwal, 2015a] Aggarwal, C. C. (2015a). *Data mining : the textbook*. Springer.

[Aggarwal, 2015b] Aggarwal, C. C. (2015b). Outlier analysis. In *Data mining*, pages 237–263. Springer.

[Aggarwal and Yu, 1998] Aggarwal, C. C. and Yu, P. S. (1998). Mining large itemsets for association rules. *IEEE Data Eng. Bull.*, 21(1) :23–31.

[Agrawal et al., 1993] Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *Acm sigmod record*, volume 22, pages 207–216. ACM.

[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.

[Agrawal et al., 1994] Agrawal, R., Srikant, R., et al. (1994). Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499.

[Antonie et al., 2001] Antonie, M.-L., Zaiane, O. R., and Coman, A. (2001). Application of data mining techniques for medical image classification. In *Proceedings of the Second International Conference on Multimedia Data Mining*, pages 94–101. Springer-Verlag.

[Aseervatham et al., 2006] Aseervatham, S., Osmani, A., and Viennet, E. (2006). bitspade : A lattice-based sequential pattern mining algorithm using bitmap representation. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 792–797. IEEE.

[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*, KDD '02, pages 429–435, New York, NY, USA. ACM.

[Bai and Bai, 2009] Bai, S. and Bai, S.-X. (2009). The maximal frequent pattern mining of dna sequence. In *Granular Computing, 2009, GRC'09. IEEE International Conference on*, pages 23–26. IEEE.

[Béchet et al., 2015] Béchet, N., Cellier, P., Charnois, T., and Crémilleux, B. (2015). Sequence mining under multiple constraints. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 908–914. ACM.

[Bellinger et al., 2004] Bellinger, G., Castro, D., and Mills, A. (2004). Data, information, knowledge, and wisdom.

[Bucilă et al., 2003] Bucilă, C., Gehrke, J., Kifer, D., and White, W. (2003). Dualminer : A dual-pruning algorithm for itemsets with constraints. *Data Mining and Knowledge Discovery*, 7(3) :241–272.

[Cheeseman et al., 1988] Cheeseman, P. C., Self, M., Kelly, J., Taylor, W., Freeman, D., and Stutz, J. C. (1988). Bayesian classification. In *AAAI*, volume 88, pages 607–611.

[Chen, 2008] Chen, J. (2008). Contiguous item sequential pattern mining using updown tree. *Intelligent Data Analysis*, 12(1) :25–49.

[Chen and Cook, 2007] Chen, J. and Cook, T. (2007). Mining contiguous sequential patterns from web logs. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 1177–1178, New York, NY, USA. ACM.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38.

[Dietterich, 1998] Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7) :1895–1923.

[Dong and Li, 1999] Dong, G. and Li, J. (1999). Efficient mining of emerging patterns : Discovering trends and differences. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 43–52. ACM.

[Fahed et al., 2018] Fahed, L., Brun, A., and Boyer, A. (2018). Deer : Distant and essential episode rules for early prediction. *Expert Systems with Applications*, 93 :283–298.

[Fayyad et al., 1996a] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996a). From data mining to knowledge discovery in databases. *AI magazine*, 17(3) :37.

[Fayyad et al., 1996b] Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. (1996b). *Advances in knowledge discovery and data mining*, volume 21. AAAI press Menlo Park.

[Feil and Fraga, 2012] Feil, R. and Fraga, M. F. (2012). Epigenetics and the environment : emerging patterns and implications. *Nature Reviews Genetics*, 13(2) :97.

[Fischer et al., 2006] Fischer, J., Heun, V., and Kramer, S. (2006). Optimal string mining under frequency constraints. In *PKDD*, volume 4213, pages 139–150. Springer.

[Fournier-Viger et al., 2014a] Fournier-Viger, P., Gomariz, A., Campos, M., and Thomas, R. (2014a). Fast vertical mining of sequential patterns using co-occurrence information. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 40–52. Springer.

[Fournier-Viger et al., 2014b] Fournier-Viger, P., Gomariz, A., Gueniche, T., Soltani, A., Wu, C.-W., and Tseng, V. S. (2014b). Spmf : a java open-source pattern mining library. *The Journal of Machine Learning Research*, 15(1) :3389–3393.

[Fournier-Viger et al., 2014c] Fournier-Viger, P., Gomariz, A., Šebek, M., and Hlosta, M. (2014c). Vgen : fast vertical mining of sequential generator patterns. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 476–488. Springer.

[Fournier-Viger et al., 2008] Fournier-Viger, P., Nkambou, R., and Nguifo, E. M. (2008). A knowledge discovery framework for learning task models from user interactions in intelligent tutoring systems. In *Mexican International Conference on Artificial Intelligence*, pages 765–778. Springer.

[Fournier-Viger et al., 2014d] Fournier-Viger, P., Wu, C.-W., Gomariz, A., and Tseng, V. S. (2014d). Vmsp : Efficient vertical mining of maximal sequential patterns. In *Canadian Conference on Artificial Intelligence*, pages 83–94. Springer.

[Fournier-Viger et al., 2013] Fournier-Viger, P., Wu, C.-W., and Tseng, V. S. (2013). Mining maximal sequential patterns without candidate maintenance. In *International Conference on Advanced Data Mining and Applications*, pages 169–180. Springer.

[Fürnkranz, 1998] Fürnkranz, J. (1998). A study using n-gram features for text categorization. *Austrian Research Institute for Artifical Intelligence*, 3(1998) :1–10.

[Gao et al., 2008] Gao, C., Wang, J., He, Y., and Zhou, L. (2008). Efficient mining of frequent sequence generators. In *Proceedings of the 17th international conference on World Wide Web*, pages 1051–1052. ACM.

[García-Hernández et al., 2006] García-Hernández, R. A., Martínez-Trinidad, J. F., and Carrasco-Ochoa, J. A. (2006). A new algorithm for fast discovery of maximal sequential patterns in a document collection. In *CICLing*, pages 514–523. Springer.

[Garofalakis et al., 1999] Garofalakis, M. N., Rastogi, R., and Shim, K. (1999). SPIRIT : Sequential pattern mining with regular expression constraints. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 223–234, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Geng and Hamilton, 2006] Geng, L. and Hamilton, H. J. (2006). Interestingness measures for data mining : A survey. *ACM Computing Surveys (CSUR)*, 38(3) :9.

[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.

[Hahsler and Karpienko, 2017] Hahsler, M. and Karpienko, R. (2017). Visualizing association rules in hierarchical groups. *Journal of Business Economics*, 87(3) :317–335.

[Han et al., 1992] Han, J., Cai, Y., and Cercone, N. (1992). Knowledge discovery in databases : An attribute-oriented approach. In *VLDB*, volume 92, pages 24–27.

[Han et al., 2011] Han, J., Pei, J., and Kamber, M. (2011). *Data mining : concepts and techniques*. Elsevier.

[Han et al., 2000a] Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., and Hsu, M.-C. (2000a). 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.

[Han et al., 2001] Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., and Hsu, M. (2001). Prefixspan : Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the 17th international Conference on Data Engineering*, pages 215–224.

[Han et al., 2000b] Han, J., Pei, J., and Yin, Y. (2000b). Mining frequent patterns without candidate generation. In *ACM sigmod record*, volume 29, pages 1–12. ACM.

[Han et al., 2004] Han, J., Pei, J., Yin, Y., and Mao, R. (2004). Mining frequent patterns without candidate generation : A frequent-pattern tree approach. *Data mining and knowledge discovery*, 8(1) :53–87.

[Held and Riss, 1998] Held, D. and Riss, J.-M. (1998). Le développement des compétences au service de l'organisation apprenante. Employeur Suisse, no 13.

[Hirate and Yamana, 2006] Hirate, Y. and Yamana, H. (2006). Generalized sequential pattern mining with item intervals. *JCP*, 1(3) :51–60.

[Ho et al., 2005] Ho, J., Lukov, L., and Chawla, S. (2005). Sequential pattern mining with constraints on large protein databases. In *Proceedings of the 12th International Conference on Management of Data (COMAD)*, pages 89–100.

[Hotho et al., 2005] Hotho, A., Nürnberger, A., and Paaß, G. (2005). A brief survey of text mining. volume 20, pages 19–62.

[Hull and Grefenstette, 1996] Hull, D. A. and Grefenstette, G. (1996). A detailed analysis of english stemming algorithms.

[Jain and Dubes, 1988] Jain, A. K. and Dubes, R. C. (1988). Algorithms for clustering data.

[Kanungo et al., 2002] Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, A. Y. (2002). An efficient k-means clustering algorithm : Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence*, 24(7) :881–892.

[Kaufman and Rousseeuw, 2009] Kaufman, L. and Rousseeuw, P. J. (2009). *Finding groups in data : an introduction to cluster analysis*, volume 344. John Wiley & Sons.

[Kimball and Ross, 2011] Kimball, R. and Ross, M. (2011). *The data warehouse toolkit : the complete guide to dimensional modeling*. John Wiley & Sons.

[Krempl et al., 2014] Krempl, G., Žliobaite, I., Brzeziński, D., Hüllermeier, E., Last, M., Lemaire, V., Noack, T., Shaker, A., Sievi, S., Spiliopoulou, M., et al. (2014). Open challenges for data stream mining research. *ACM SIGKDD explorations newsletter*, 16(1) :1–10.

[Lakshmanan et al., 1999] Lakshmanan, L. V., Ng, R., Han, J., and Pang, A. (1999). Optimization of constrained frequent set queries with 2-variable constraints. In *ACM SIGMOD Record*, volume 28, pages 157–168. ACM.

[Lau and Baldwin, 2016] Lau, J. H. and Baldwin, T. (2016). An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv :1607.05368*.

[Le et al., 2017] Le, B., Duong, H., Truong, T., and Fournier-Viger, P. (2017). Fclosm, fgensm : two efficient algorithms for mining frequent closed and generator sequences using the local pruning strategy. *Knowledge and Information Systems*, pages 1–37.

[Le and Mikolov, 2014] Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

[Le Boterf, 1994] Le Boterf, G. (1994). De la compétence. essai sur un attracteur étrange. page 176, Paris. Les Editions d'Organisation.

[Li and Wang, 2008] Li, C. and Wang, J. (2008). *Efficiently mining closed subsequences with gap constraints*, pages 313–322.

[Likhachev et al., 2008] Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., and Thrun, S. (2008). Anytime search in dynamic graphs. *Artificial Intelligence*, 172(14) :1613–1643.

[Lo et al., 2008] Lo, D., Khoo, S.-C., and Li, J. (2008). Mining and ranking generators of sequential patterns. In *Proceedings of the 2008 SIAM International Conference on Data Mining*, pages 553–564. SIAM.

[Lu and Weng, 2007] Lu, D. and Weng, Q. (2007). A survey of image classification methods and techniques for improving classification performance. *International journal of Remote sensing*, 28(5) :823–870.

[Lu and Li, 2004] Lu, S. and Li, C. (2004). Aprioriadjust : An efficient algorithm for discovering the maximum sequential patterns. In *Proc. Intern. Workshop knowl. Grid and grid intell.*

[Luo and Chung, 2005] Luo, C. and Chung, S. M. (2005). Efficient mining of maximal sequential patterns using multiple samples. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pages 415–426. SIAM.

[Machlup, 1962] Machlup, F. (1962). The Production and Distribution of knowledge in the United States. Princeton UP.

[MacQueen et al., 1967] MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

[Maier, 1983] Maier, D. (1983). *The theory of relational databases*, volume 11. Computer science press Rockville.

[Mannila et al., 1997] Mannila, H., Toivonen, H., and Inkeri Verkamo, A. (1997). Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3) :259–289.

[Matsui et al., 2013] Matsui, T., Uno, T., Umemori, J., and Koide, T. (2013). *A New Approach to String Pattern Mining with Approximate Match*, pages 110–125. Springer Berlin Heidelberg, Berlin, Heidelberg.

[McClelland, 1973] McClelland, D. C. (1973). Testing for competence rather than for" intelligence.". *American psychologist*, 28(1) :1.

[Namer, 2000] Namer, F. (2000). FLEMM : un analyseur flexionnel du français à base de règles. *Traitement Automatique des Langues*, 41(2) :523–547.

[Ng et al., 1998] Ng, R. T., Lakshmanan, L. V., Han, J., and Pang, A. (1998). Exploratory mining and pruning optimizations of constrained associations rules. In *ACM Sigmod Record*, volume 27, pages 13–24. ACM.

[O'Callaghan et al., 2012] O'Callaghan, D., Harrigan, M., Carthy, J., and Cunningham, P. (2012). Network analysis of recurring youtube spam campaigns. In *ICWSM*.

[Pasquier et al., 1999] Pasquier, N., Bastide, Y., Taouil, R., and Lakhal, L. (1999). Discovering frequent closed itemsets for association rules. In *Proceedings of the 7th International Conference on Database Theory*, ICDT '99, pages 398–416, London, UK, UK. Springer-Verlag.

[Pei et al., 2001] Pei, J., Han, J., Lu, H., Nishio, S., Tang, S., and Yang, D. (2001). H-mine : Hyper-structure mining of frequent patterns in large databases. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 441–448. IEEE.

[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.

[Pei et al., 2004] Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., and Hsu, M.-C. (2004). Mining sequential patterns by pattern-growth : The prefixspan approach. *IEEE Transactions on knowledge and data engineering*, 16(11) :1424–1440.

[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.

[Pham et al., 2012] Pham, T.-T., Luo, J., Hong, T.-P., and Vo, B. (2012). Msgps : a novel algorithm for mining sequential generator patterns. In *International Conference on Computational Collective Intelligence*, pages 393–401. Springer.

[Piateski and Frawley, 1991] Piateski, G. and Frawley, W. (1991). *Knowledge discovery in databases*. MIT press.

[Porter, 1980] Porter, M. (1980). An algorithm for suffix stripping. volume 14, pages 130–137.

[Quinlan, 1986] Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1) :81–106.

[Reinbold and Breillot, 1993] Reinbold, M. and Breillot, J. (1993). Gérer la compétence dans l'entreprise. Dynamiques d'entreprises. L'Harmattan.

[Rodríguez-González et al., 2017] Rodríguez-González, A. Y., Lezama, F., Iglesias-Alvarez, C. A., Martínez-Trinidad, J. F., Carrasco-Ochoa, J. A., and de Cote, E. M. (2017). Closed frequent similar pattern mining : Reducing the number of frequent similar patterns without information loss. *Expert Systems with Applications*.

[Sakurai et al., 2007] Sakurai, S., Kitahara, Y., and Orihara, R. (2007). Sequential pattern mining based on a new criteria and attribute constraints. In *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*, pages 516–521. IEEE.

[Salton and McGill, 1986] Salton, G. and McGill, M. J. (1986). Introduction to modern information retrieval. New York, NY, USA. McGraw-Hill, Inc.

[Samet, 1990] Samet, H. (1990). Applications of spatial data structures.

[Srikant and Agrawal, 1996] Srikant, R. and Agrawal, R. (1996). Mining sequential patterns : Generalizations and performance improvements. *Advances in Database Technology—EDBT'96*, pages 1–17.

[Tung et al., 1999] Tung, A. K., Lu, H., Han, J., and Feng, L. (1999). Breaking the barrier of transactions : Mining inter-transaction association rules. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 297–301. ACM.

[Uno et al., 2004] Uno, T., Kiyomi, M., and Arimura, H. (2004). Efficient mining algorithms for frequent/closed/maximal itemsets. In *PROC. IEEE ICDM WORKSHOP FREQUENT ITEMSET MINING IMPLEMENTATIONS*. Citeseer.

[Uryas' ev, 1988] Uryas' ev, S. (1988). On the anti-monotonicity of differential mappings connected with general equilibrium problem. *Optimization*, 19(5) :693–709.

[Vapnik, 1999] Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5) :988–999.

[Voorhees, 1999] Voorhees, E. (1999). Natural language processing and information retrieval. In *Information Extraction : Towards Scalable, Adaptable Systems*, pages 32–48. Springer.

[Wan, 1990] Wan, E. A. (1990). Neural network classification : A bayesian interpretation. *IEEE Transactions on Neural Networks*, 1(4) :303–305.

[Wang and Han, 2004] Wang, J. and Han, J. (2004). BIDE : Efficient mining of frequent closed sequences. In *Proceedings of the 20th International Conference on Data Engineering*, ICDE '04, pages 79–, Washington, DC, USA. IEEE Computer Society.

[Wittorski, 1998] Wittorski, R. (1998). De la fabrication des compétences. volume 135, pages 57–69. Paris : Documentation francaise.

[Wu et al., 2013] Wu, X., Zhu, X., He, Y., and Arslan, A. N. (2013). Pmbc : Pattern mining from biological sequences with wildcard constraints. *Computers in biology and medicine*, 43(5) :481–492.

[Xie et al., 2017] Xie, F., Wu, X., and Zhu, X. (2017). Efficient sequential pattern mining with wildcards for keyphrase extraction. *Knowledge-Based Systems*, 115(Supplement C) :27 – 39.

[Yan et al., 2003] Yan, X., Han, J., and Afshar, R. (2003). Clospan : Mining : Closed sequential patterns in large datasets. In *Proceedings of the 2003 SIAM international conference on data mining*, pages 166–177. SIAM.

[Yang and Gidófalvi, 2017] Yang, C. and Gidófalvi, G. (2017). Mining and visual exploration of closed contiguous sequential patterns in trajectories. *International Journal of Geographical Information Science*, pages 1–23.

[Yi et al., 2011] Yi, S., Zhao, T., Zhang, Y., Ma, S., and Che, Z. (2011). An effective algorithm for mining sequential generators. *Procedia Engineering*, 15 :3653–3657.

[Zaki, 2000a] Zaki, M. J. (2000a). Scalable algorithms for association mining. *IEEE transactions on knowledge and data engineering*, 12(3) :372–390.

[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*, CIKM '00, pages 422–429, New York, NY, USA. ACM.

[Zaki, 2001] Zaki, M. J. (2001). SPADE : An efficient algorithm for mining frequent sequences. *Machine learning*, 42(1) :31–60.

[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.

[Zhang et al., 2015] Zhang, J., Wang, Y., and Yang, D. (2015). CCSpan : Mining closed contiguous sequential patterns. *Knowledge-Based Systems*, 89 :1–13.

[Zhang et al., 2016] Zhang, J., Wang, Y., Zhang, C., and Shi, Y. (2016). Mining contiguous sequential generators in biological sequences. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 13(5) :855–867.

Résumé

L'information occupe désormais une place centrale dans notre vie quotidienne, elle est à la fois omniprésente et facile d'accès. Pourtant, l'extraction de l'information à partir des données est un processus souvent inaccessible. En effet, même si les méthodes de fouilles de données sont maintenant accessibles à tous, les résultats de ces fouilles sont souvent complexes à obtenir et à exploiter pour l'utilisateur. La fouille de motifs combinée à l'utilisation de contraintes est une direction très prometteuse de la littérature pour à la fois améliorer l'efficience de la fouille et rendre ses résultats plus appréhendables par l'utilisateur. Cependant, la combinaison de contraintes désirée par l'utilisateur est souvent problématique car, elle n'est pas toujours adaptable aux caractéristiques des données fouillées tel que le bruit. Dans cette thèse, nous proposons deux nouvelles contraintes et un algorithme pour pallier ce problème. La contrainte de robustesse permet de fouiller des données bruitées en conservant la valeur ajoutée de la contrainte de contiguïté. La contrainte de clôture allégée améliore l'appréhendabilité de la fouille de motifs tout en étant plus résistante au bruit que la contrainte de clôture classique. L'algorithme C3Ro est un algorithme générique de fouille de motifs séquentiels intégrant de nombreuses contraintes, notamment les deux nouvelles contraintes que nous avons introduites, afin de proposer à l'utilisateur la fouille la plus efficiente possible tout en réduisant au maximum la taille de l'ensemble des motifs extraits. C3Ro rivalise avec les meilleurs algorithmes de fouille de motifs de la littérature en termes de temps d'exécution tout en consommant significativement moins de mémoire. C3Ro a été expérimenté dans le cadre de l'extraction de compétences présentes dans les offres d'emploi sur le Web.

Abstract

Information now occupies a central place in our daily lives, it is both ubiquitous and easy to access. Yet extracting information from data is often an inaccessible process. Indeed, even though data mining methods are now accessible to all, the results of these mining are often complex to obtain and exploit for the user. Pattern mining combined with the use of constraints is a very promising direction of the literature to both improve the efficiency of the mining and make its results more apprehensible to the user. However, the combination of constraints desired by the user is often problematic because it does not always fit with the characteristics of the searched data such as noise. In this thesis, we propose two new constraints and an algorithm to overcome this issue. The robustness constraint allows to mine noisy data while preserving the added value of the contiguity constraint. The extended closdeness constraint improves the apprehensibility of the set of extracted patterns while being more noise-resistant than the conventional closdeness constraint. The C3Ro algorithm is a generic sequential pattern mining algorithm that integrates many constraints, including the two new constraints that we have introduced, to provide the user the most efficient mining possible while reducing the size of the set of extracted patterns. C3Ro competes with the best pattern mining algorithms in the literature in terms of execution time while consuming significantly less memory. C3Ro has been experienced in extracting competencies from web-based job postings.