ECHANTILLONNAGE SOUS CONTRAINTES DE MOTIFS STRUCTURES

Lamine Diop

UNIVERSITÉ GASTON BERGER

ÉCOLE DOCTORALE DES SCIENCES ET DES TECHNOLOGIES

Laboratoire d'Analyse Numérique et Informatique (LANI)

THÈSE présentée par :

Lamine DIOP

soutenue le : 16 juillet 2020

pour obtenir le grade de : Docteur de l'université Gaston Berger de Saint-Louis

Spécialité : Informatique

ECHANTILLONNAGE SOUS CONTRAINTES DE MOTIFS STRUCTURES

JURY COMPOSÉ DE :

Ousmane THIARÉ	Professeur titulaire, Université Gaston Berger de Saint-Louis (Sénégal)	Président
Bruno CRÉMILLEUX	Professeur des universités, Université de Caen Normandie (France)	Rapporteur
Idrissa SARR	Professeur assimilé, Université Cheikh Anta Diop de Dakar (Sénégal)	Rapporteur
Céline ROBARDET	Professeur des universités, INSA Lyon (France)	Rapporteur
Arnaud SOULET	Maître de conférences Université de Tours (France)	Co-encadrant
Cheikh Talibouya DIOP	Professeur titulaire, Université Gaston Berger de Saint-Louis (Sénégal)	Directeur
Arnaud GIACOMETTI	Professeur des universités, Université de Tours (France)	Directeur

Dédicaces

À mes parents qui ont fait de moi ce que je suis.
Je vous aime de tout mon coeur, longue vie à vous.
Vous vous êtes sacrifiés pour ma réussite, c'est sûr.
Sachez que je ne vous décevrez point, je vous le jure.

À ma famille

À mon épouse, la linguère, Mame Bousso Dieng, un vrai miroir de l'amour.
Une dame de fer qui s'est toujours distinguée de par sa dignité.
Ton sourire est devenu mon soleil
Ta tendresse une grande merveille.

À notre fleur de vie, Serigne Fallou Diop. Je te souhaite une longue vie durant laquelle tu seras un modèle pour tes frères et soeurs. Donne du respect à tous tes aînés quels que soient leurs statuts sociaux. Il faut toujours remercier Le Bon Dieu quelle que soit ta situation.
Papa t'Aime.

À mon guide religieux Serigne Fallou Mbacké, deuxième khalif de Cheikh Ahmadou Bamba.
Si le prénom de mon fils aîné n'a pas surpris certains de mes amis et proches, c'est parce que vous êtes sans doute l'homme qui m'a toujours accompagné.

À mes défunts grands parents Maguette Ka et Cheikh Diop, que la terre leur soit légère.

À mes amis

"Travaillez comme si vous ne devez jamais mourir, et priez comme si vous devez mourir demain."

Remerciements

Tout d'abord, je rends grâce à Dieu de m'avoir donné la force et le courage de terminer cette thèse en beauté.

Je suis profondément reconnaissant à mes encadrants Arnaud Giacometti, Arnaud Soulet, Cheikh Talibouya Diop et Dominique Li. Ce manuscrit est sans doute le résultat des trois années de discussion et de conseils autour du sujet de cette thèse. Sans vous, ce document n'aurait pas eu lieu. Je remercie Arnaud Giacometti et Cheikh Talibouya Diop d'avoir dirigé mes trois années de thèse. Je les remercie des efforts qu'ils ont fournis en termes d'encadrement actif et d'encouragements dans mes moments de doute.

Je m'en veux de ne pas avoir trouvé des mots assez forts pour exprimer mon sentiment de reconnaissance et de profonde gratitude à Arnaud Giacometti et Béatrice Markhoff de l'université de Tours. Les efforts qu'ils ont faits en terme d'accueil, de recherche de bourse de thèse, de recherche de logement, de guide durant mes séjours à Blois, pour ne citer que ceux là, resteront à jamais gravés dans ma mémoire. Aux remerciements de ces deux, j'ajoute Arnaud Soulet et Brunehilde Lecostey pour les multiples invitations chez eux durant les moments de fête de famille. Cheikh Talibouya Diop, je n'ai pas oublié les "ndogou" du mois de ramadan. Des mathématiques et des données, nous en avons construit une famille. C'est ça le génie de la recherche scientifique. Ce n'est pas tout, vous m'avez appris l'éthique et la déontologie de la recherche scientifique, mais aussi, le savoir vivre, le partage et la confiance en soi.

Mention spéciale à Arnaud Giacometti qui n'a ménagé aucun effort pour l'aboutissement de cette thèse. De l'initiation à la recherche, en master 2 SIAD à Blois, jusqu'à la thèse, votre confiance en ma modeste personne me va droit au coeur. Vous êtes sans doute l'homme qui m'a le plus marqué durant ces trois dernières années.

Ensuite, je tiens à remercier Ousmane Thiaré, Professeur à l'université Gaston Berger de Saint-Louis, pour m'avoir fait l'honneur de présider le jury de cette thèse malgré les circonstances et son rôle de Recteur d'université. J'exprime toute ma gratitude à Bruno Crémilleux, Professeur à l'université de Caen, Idrissa Sarr, Professeur à l'université Cheikh Anta Diop de Dakar, et Céline Robardet, Professeur à l'INSA Lyon, qui ont accepté de consacrer une partie de leur temps pour juger ce travail. Vos remarques et suggestions sont d'une importance capitale. Je les remercie vivement d'avoir accepté de faire partie du jury de mon mémoire malgré les circonstances de la Covid-19.

J'adresse également mes remerciements à tous ceux qui, de près comme de loin, d'une manière ou d'une autre, ont participé à l'aboutissement de ce travail. Je remercie notamment Ibrahima Fall, professeur à l'EPT de Dakar, Cheikh Ba et Maïssa Mbaye, professeurs à l'UGB de Saint-Louis, tout le corps professoral de l'UFR SAT de l'université Gaston Berger de Saint-Louis, et l'équipe BDTLN de l'Antenne Universitaire de Blois de m'avoir accueilli dans leurs locaux à bras ouverts. Les discussions entretenues avec les doctorants du CEA-MITIC, Moussa Ba, Mané Seck, Al Hassim Diallo, Sophie Sylla et Pape Birane Seye, ont été fructueuses.

Je voudrais aussi remercier le CEA-MITIC (Centre d'Excellence Africain en Mathématiques, Informatique et TIC) et l'Université de Tours de m'avoir accordé deux fois de suite une bourse de mobilité de recherche de cinq mois à Blois. Sans ces mobilités, le travail serait beaucoup plus difficile. Un grand merci au directeur du centre de calcul de l'UGB Pr Amsata Ndiaye et tous les techniciens de la DCM d'avoir permis à ceux qui ne pouvaient pas être présents de suivre la soutenance partout dans le monde.

Enfin, je ne saurais terminer sans remercier mon entourage pour leur soutien et leur encouragement indéfectibles. Je pense à ma famille et surtout, à mes parents et ma femme, et à ma belle famille. A mes frères et soeurs qui sont toujours à mes services : Youssou, Fatma, Arame, Ablaye, Maman et Marème. A mes amis de Keur Awa, de N'Dindy, de Mbacké, de Diourbel, de Saint-Louis, de Thiès, de Dakar et de Blois (France) qui ont toujours été présents à mes côtés, aux habitants du G4 Family, sans oublier personne. A ces personnes spéciales dont l'inspecteur Cheikh Bakhoum, Papa Mass Fall, Sophie Sylla, Fama Dieng et tous ceux qui étaient présents le jour de ma soutenance. Mention spéciale à Moussa BA qui est plus qu'un ami.

Résumé

La littérature de la découverte de motifs a longtemps lutté avec deux problèmes majeurs. Premièrement, il n'est pas possible d'utiliser directement les motifs pertinents si le seuil d'intérêt minimal est petit car ils sont bien trop nombreux. A l'opposé, si le seuil d'intérêt minimal est trop grand, certaines instances seront peu ou pas décrites. Deuxièmement, l'ensemble complet des motifs ayant satisfait la contrainte de seuil d'intérêt minimal peut contenir de nombreuses redondances.

L'échantillonnage en sortie est une méthode non exhaustive pour la découverte instantanée de motifs intéressants qui assure une bonne interactivité tout en offrant de solides garanties statistiques en raison de sa nature aléatoire. Curieusement, une telle approche étudiée pour différents types de motifs, y compris les itemsets et les sous-graphes, n'a pas encore été appliquée aux motifs séquentiels et aux bases de données distribuées. Dans cette thèse, nous proposons de nombreuses méthodes dédiées à l'échantillonnage en sortie de motifs séquentiels, l'échantillonnage en sortie de motifs dans des bases de données distribuées et l'échantillonnage en sortie de motifs basé sur les tries. En plus de répondre à ces tâches complexes, l'originalité de nos approches est d'introduire une classe de mesures d'intérêt reposant sur la norme des motifs, nommée classe de mesures d'intérêt fondées sur la norme. En particulier, cette classe permet d'ajouter des contraintes sur la norme des motifs échantillonnés pour contrôler leur longueur et éviter l'écueil de la "longue traîne" où les motifs les plus rares inondent l'utilisateur.

Dans ce cadre, nous proposons en premier lieu deux algorithmes nommés NUSSampling pour les bases de données séquentielles et DDSampling pour les bases de données distribuées. Basés sur des procédures aléatoires en deux étapes intégrant cette classe de mesures, ils tirent au hasard des motifs proportionnellement à la fréquence pondérée par une utilité fondée sur la norme. En second lieu, nous proposons TPSampling, un algorithme d'échantillonnage en sortie de motifs ensemblistes basé sur la structure du trie. Moins consommateur en mémoire, il tire aussi aléatoirement des motifs en fonction de leur fréquence pondérée par une utilité fondée sur la norme. Nous montrons que toutes nos méthodes effectuent un échantillonnage exact selon la mesure sous-jacente.

Au niveau des applications, nous nous concentrons sur l'intérêt des contraintes de norme et de décroissance exponentielle qui aident à tirer des motifs généraux de la tête de la longue traine. Nous illustrons également comment profiter de ces motifs échantillonnés pour construire des classificateurs dédiés aux séquences et aux itemsets. Cette approche de classification rivalise avec les propositions de l'état de l'art montrant l'intérêt de l'échantillonnage en sortie de motifs avec une mesure d'intérêt fondée sur la norme. Par ailleurs, nous illustrons également l'intérêt des motifs échantillonnés sur les données distribuées du Web sémantique pour détecter des entités aberrantes dans DBpedia et Wikidata.

Mots clés : Fouille de données, découverte de motifs, Echantillonnage en sortie de motifs

Abstract

The pattern discovery literature has long struggled with two major problems. First, it is not possible to use the relevant patterns directly if the minimum interest threshold is small because there are far too many. Conversely, if the minimum interest threshold is too large, certain instances will be described little or not at all. Second, the full set of patterns that met the minimum interest threshold constraint may contain many redundancies.

Output sampling is a non-exhaustive method for the instant discovery of relevant patterns which ensures good interactivity while providing strong statistical guarantees due to its random nature. Curiously, such an approach studied for different types of patterns, including itemsets and subgraphs, has not yet been applied to sequential patterns and distributed databases. In this thesis, we propose numerous methods dedicated to sequential pattern sampling, pattern sampling in distributed databases and finally trie-based pattern sampling. In addition to answering these complex tasks, the originality of our approaches is to introduce a class of interestingness measures relying on the norm of the pattern, named norm-based interestingness measures. In particular, it enables to add constraints on the norm of sampled patterns to control the length of the drawn patterns and to avoid the pitfall of the "long tail" where the rarest patterns flood the user.

In this context, we first propose two algorithms called NUSSampling for sequential databases and DDSampling for distributed databases. Based on two-step random procedures incorporating this class of interestingness measures, they randomly draw patterns proportionally to the frequency weighted by a utility based on the norm. Second, we propose TPSampling, a sampling algorithm for itemsets based on the trie structure. Less consumer in memory, it also randomly draws patterns based on frequency weighted by a utility based on the norm. We show that all of our methods perform an exact sampling according to the underlying measure.

At the application level, we focus on the interest of norm constraints and exponential decay that help to draw general patterns from the head of the long tail. We also illustrate how to benefit from these sampled patterns to build classifiers dedicated to sequences and itemsets. This classification approach rivals with state-of-the-art proposals showing the interest of sequential pattern sampling with norm-based utility. In addition, we also illustrate the usefulness of the sampled patterns on the distributed data of the Semantic Web for detecting outlier entities in DBpedia and Wikidata.

Keywords : Data mining, pattern discovery, Output space pattern sampling

Table des matières

Liste des tableaux

Table des figures

Introduction générale

Introduction

Contexte et motivations

La découverte de connaissances dans les bases de données (knowledge discovery in databases) est une activité de recherche qui permet de trouver de nouvelles tendances, corrélations et modèles à partir d'une très grande masse de données. Etant donné une base de données et un prédicat de sélection booléen, les méthodes exhaustives d'extraction de motifs [Agrawal *et al.*, 1993, Agrawal et Srikant, 1995, Srikant et Agrawal, 1996, Inokuchi *et al.*, 2000, Kuramochi et Karypis, 2001, Burdick *et al.*, 2001, Hamrouni, 2012] visent à retourner l'ensemble des motifs de la base de données qui satisfont le prédicat. La sélection de ces motifs consiste à parcourir le langage de motifs de la base de données en question et est sanctionnée souvent par un coût de calcul très élevé. De ce fait, de nombreuses méthodes ont été proposées pour optimiser le coût du parcours du langage [Aggarwal *et al.*, 2014].

Cependant, dans tous les cas, une énumération complète de l'ensemble des motifs d'un langage qui satisfont une contrainte tel qu'un seuil de support minimal reste une tâche très fastidieuse. En effet, quel que soit le seuil d'intérêt choisi par l'utilisateur, il est difficile de bien contrôler la sortie. Si le seuil est trop grand, alors la sortie risque d'être vide, par contre s'il est trop petit, alors le nombre de motifs retournés devient très élevé et impossible à analyser par un expert. Les autres méthodes [Han *et al.*, 2002, Soulet *et al.*, 2011] qui contrôlent un peu plus la sortie sont du type optimisation et bien souvent, elles se concentrent sur une seule partie de l'espace de recherche conduisant à peu de diversité dans les motifs extraits.

Face à cette situation, des méthodes d'échantillonnage en sortie de motifs ont été récemment proposées [Al Hasan et Zaki, 2009, Boley *et al.*, 2011, Dzyuba *et al.*, 2017]. Ces méthodes se distinguent des méthodes d'échantillonnage de motifs en entrée. L'échantillonnage en entrée [Toivonen *et al.*, 1996] consiste à régénérer depuis un échantillon de données $\mathcal{D}'$ de la base de données $\mathcal{D}$, tous les motifs qui auraient été extraits depuis le jeu de données complet. Souvent utilisé par les statisticiens, l'échantillonnage en entrée est une approche qui consiste à sélectionner un sous-ensemble de la base de données $\mathcal{D}$ qui est souvent très grande et excède la taille disponible en mémoire, afin de former une base de données plus petite $\mathcal{D}'$. Ainsi, la tâche d'extraction de motifs est effectuée plus

rapidement sur $\mathcal{D}'$, mais la qualité des résultats reste discutable. En effet, un motif φ peut être non pertinent dans la petite base de données $\mathcal{D}'$ et pertinent dans la grande base de données $\mathcal{D}$. Formellement, la contrainte $supp(\varphi, \mathcal{D}') \geq minsup$ peut être fausse, alors que $supp(\varphi, \mathcal{D}) \geq minsup$ est vraie. Inversement, la contrainte $supp(\varphi, \mathcal{D}') \geq minsup$ peut être vraie, alors que $supp(\varphi, \mathcal{D}) \geq minsup$ est fausse.

Pour remédier à ce problème, l'échantillonnage en sortie vise à tirer un échantillon de motifs directement à partir du langage des motifs $\mathcal{L} = \{\varphi_1, \varphi_2, \cdots\}$ de la base de données $\mathcal{D}$ avec une probabilité proportionnelle à une mesure d'intérêt choisie par l'utilisateur. En considérant la base de données de 4 instances $\mathcal{D} = \{\gamma_1, \cdots, \gamma_4\}$, la figure 1 montre l'objectif principal visé par les méthodes d'échantillonnage en sortie dans le cas où la mesure d'intérêt choisie est la fréquence.

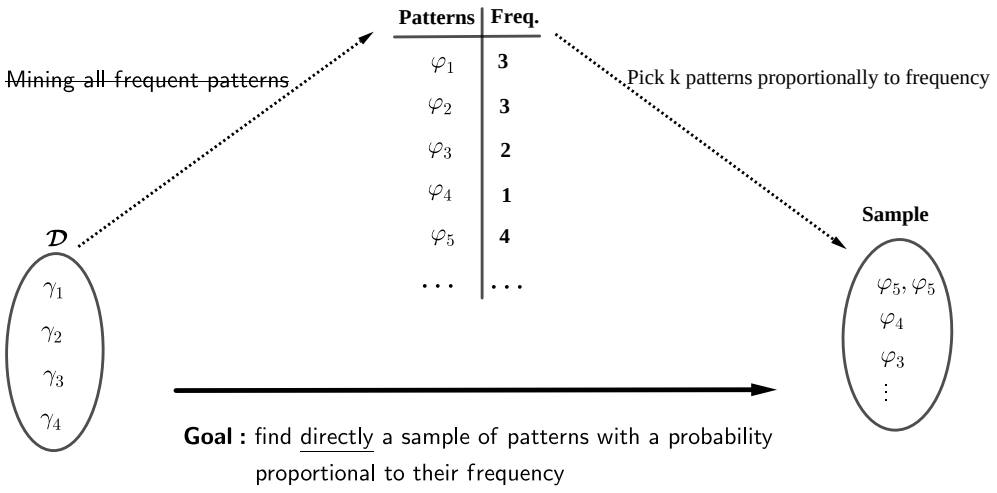


FIGURE 1 – L'objectif de l'échantillonnage en sortie de motifs

Soulignons que l'objectif de l'échantillonnage en sortie de motifs n'est pas d'énumérer tous les motifs fréquents de la base de données avec une méthode de fouille exhaustive puis d'en tirer un échantillon. Avec l'échantillonnage en sortie, la "population" n'est pas matérialisée contrairement à l'échantillonnage en entrée (cette "population" représente les instances de la base de données $\mathcal{D}$ dans le cas de l'échantillonnage en entrée et le langage de motifs $\mathcal{L}$ dans le cas de l'échantillonnage en sortie). Donc, sans matérialiser l'ensemble des motifs de la base de données, si la fréquence du motif φ_i est deux fois plus grande que celle du motif φ_j dans la base de données $\mathcal{D}$, alors une méthode d'échantillonnage en sortie vise à tirer φ_i avec une probabilité deux fois plus élevée que φ_j. Dans l'exemple de la figure 1, une méthode d'échantillonnage en sortie tirera le motif φ_5 avec une probabilité 2 fois plus élevée que le motif φ_3 car dans la base de données $\mathcal{D}$, nous avons $freq(\varphi_5, \mathcal{D}) = 2 \times freq(\varphi_3, \mathcal{D})$.

Pour faciliter le développement d'outils de fouille de données centrés sur l'utilisateur, un couplage étroit entre l'utilisateur et le système est essentiel. Il est clair que les méthodes traditionnelles d'extraction de motifs [Han *et al.*, 2007, Aggarwal *et al.*, 2014], qui parcourent tous les motifs de l'espace de recherche, ne sont pas bien adaptées au dévelop-

pement de tels systèmes. En effet, le temps de calcul nécessaire pour explorer tout l'espace de recherche peut être très coûteux, ce qui ne permet pas une interaction satisfaisante entre les outils de fouille de données et les analystes. Idéalement, un utilisateur devrait être autorisé à poser une requête et à la raffiner à tout moment. Le système devrait immédiatement renvoyer une réponse, ce qui est impossible si les temps de réponse des outils d'exploration de données sont trop longs. Dans ce contexte, l'échantillonnage en sortie de motifs [Al Hasan et Zaki, 2009, Boley *et al.*, 2011, Dzyuba *et al.*, 2017] est une piste de recherche très prometteuse. En effet, ces nouveaux algorithmes d'extraction de motifs permettent un accès rapide à tous les motifs et garantissent une très bonne diversité.

Les algorithmes d'échantillonnage en sortie de motifs ont déjà fait la preuve de leur efficacité et de leur utilité dans deux situations distinctes en proposant des algorithmes de détection de données aberrantes [Giacometti et Soulet, 2016] ou en développant un algorithme d'échantillonnage en sortie de motif pour caractériser les données non étiquetées [Giacometti et Soulet, 2017].

Problématiques

Avant les travaux présentés dans cette thèse, les méthodes d'échantillonnage en plusieurs étapes [Boley *et al.*, 2011, Boley *et al.*, 2012, Moens et Boley, 2014, Giacometti et Soulet, 2018] sont les seules méthodes qui font un tirage exact et elles forment la classe la plus rapide parmi les classes de méthodes d'échantillonnage en sortie de motifs. Malgré toutes leurs performances, les méthodes d'échantillonnage en plusieurs étapes rencontrent des problèmes plus ou moins courants avec les données complexes que nous allons souligner dans cette thèse.

Longue traîne : les méthodes d'échantillonnage en plusieurs étapes ne fournissent pas des motifs de bonne qualité sur les bases de données atteintes de la malédiction de la longue traîne. Dans de telles bases de données, l'ensemble de motifs le plus abondant est constitué des motifs longs et peu fréquents, seuls des motifs très peu fréquents sont retournés par ces méthodes. En particulier, une approche en deux étapes comparable à la méthode de [Boley *et al.*, 2011] n'est pas efficace pour échantillonner des motifs séquentiels intéressants dans le cas où les longueurs des séquences sont aussi grandes que variées. Dans ce cas, les motifs tirés sont en effet très longs, et donc, peu fréquents. L'écueil de la longue traîne est aussi très fréquent dans les données du Web Sémantique. Dans le cas du Web, le nombre de propriétés décrivant une entité peut être égal à 100 fois le nombre de propriétés décrivant une autre entité dans une même classe. Ainsi, il devient primordial de développer des méthodes qui se servent de contraintes pénalisant les motifs longs afin de retourner des motifs de très bonne qualité. La question que l'on pose finalement est : "*Comment formaliser une classe de mesures qui permettent d'échantillonner des motifs à partir d'une base de données atteinte de la malédiction de la longue traîne et qui soit utilisable sur les données structurées telles que les séquences ?*".

Bases de données distribuées : Au vu de son succès souligné dans la littérature [van Leeuwen, 2014, Giacometti et Soulet, 2016, Giacometti et Soulet, 2017], l'échan-

tillonnage en sortie ne devrait pas être confiné aux bases de données locales alors que les données à analyser peuvent souvent être distribuées dans plusieurs endroits [Cheung *et al.*, 1996, Otey *et al.*, 2003, Jin et Agrawal, 2006, Kum *et al.*, 2006]. En se limitant aux données transactionnelles, les méthodes classiques d'extraction de motifs dans des bases de données distribuées requièrent souvent un coût de communication très élevé. En outre, elles ont souvent besoin d'exécuter des routines au sein de chaque fragment. A titre d'exemple, les triplestores du Web constituent une base de données distribuées car une entité peut être décrite par plusieurs triplestores. Et contrairement aux bases de données distribuées classiques, il n'est pas possible avec les triplestores d'exécuter des routines d'extraction de motifs au sein de chaque fragment. Dans cette situation, une méthode d'échantillonnage en sortie de motifs qui ne rapatrie que les items des motifs échantillonnés pourrait réduire considérablement le coût de communication. Finalement, la question qu'on se pose est la suivante : "*Comment faire un tirage exact d'un motif sans centraliser l'ensemble des transactions ?*".

Problème de taille en mémoire : Dans la littérature, les méthodes d'échantillonnage en sortie [Al Hasan et Zaki, 2009, Boley *et al.*, 2011, Dzyuba *et al.*, 2017] ont besoin de stocker entièrement la base de données en mémoire. Dans le cas des bases de données transactionnelles, les instances sont stockées séparément ce qui fait que la mémoire occupée peut être très importante. De ce fait, avec les gros jeux de données qui ne tiennent pas en mémoire, cela constitue un véritable problème. Dans la littérature, le trie [Knuth, 1997] est souvent utilisé pour représenter des données ensemblistes sous une forme compressée afin de réduire sa taille. Dans ce cas, le problème qui se pose est : "*Comment échantillonner des motifs ensemblistes à partir d'un trie tout en étant rapide et générique dans le sens de la prise en compte d'une large classe de mesures d'intérêt ?*".

Voilà, entre autres, trois problèmes/questions phares que nous allons traiter dans cette thèse.

Contributions

Pour résoudre le problème de la longue traîne, nous avons introduit une classe de mesures d'intérêt fondées sur la norme nommée $\mathcal{M}$. Cette classe permet notamment de contrôler la taille des motifs échantillonnés afin de tirer des motifs intéressants selon le choix de l'utilisateur. L'efficacité de cette classe de mesures d'intérêt est que dans une instance, les motifs de même norme ont la même utilité. Cela nous permet de prendre en compte un très grand nombre de mesures d'intérêt sans que la méthode proposée soit coûteuse en temps.

En particulier, nous avons revisité la méthode en deux étapes de [Boley *et al.*, 2011] dans des bases de données transactionnelles atteintes de la malédiction de la longue traîne en proposant un algorithme générique prenant en compte toute mesure d'intérêt fondée sur la norme [Diop *et al.*, 2019c, Diop *et al.*, 2019a]. Nous avons ainsi revisité dans [Diop *et al.*, 2020a] la méthode de calcul des FPOF [He *et al.*, 2005] en introduisant une contrainte de norme maximale afin d'éviter l'impact du phénomène de la longue traîne sur la détection de données aberrantes à l'aide de la mesure

FPOF. Ensuite, nous avons proposé CSSampling (Constrained Subsequence Sampling) [Diop *et al.*, 2018a, Diop *et al.*, 2018b] pour échantillonner des motifs séquentiels suivant une probabilité proportionnelle à la fréquence. C'est la toute première méthode d'échantillonnage en sortie dédiée aux bases de données séquentielles. Pour étendre l'algorithme CSSampling à toute mesure d'intérêt fondée sur la norme, nous avons proposé NUSSampling (Norm-based Utility Subsequence Sampling) [Diop *et al.*, 2019d].

Une autre contribution que nous avons développée consiste en une méthode d'échantillonnage de motifs dans une base de données distribuée. En effet, la plupart des méthodes de recherche de motifs fréquents dans des bases de données distribuées ont un problème pour contrôler le coût de communication. En outre, à notre connaissance, toutes ces méthodes se limitent au partitionnement horizontal. Du coup, l'objectif de l'échantillonnage de motifs sur des bases de données distribuées est de proposer une approche non seulement parcimonieuse en coût de communication, mais aussi qui s'applique sur d'autres types de partitionnements tels que le partitionnement vertical ou hybride. Pour ce faire, nous avons proposé DDSampling (Distributed Database Sampling) [Diop *et al.*, 2019b, Diop *et al.*, 2020b], un algorithme générique dans le sens où il s'applique sur tout type de partitionnement d'une base de données distribuée et toute mesure d'intérêt fondée sur la norme de la classe $\mathcal{M}$.

Pour échantillonner des motifs ensemblistes à partir d'une base de données transactionnelles compressée sous la forme d'un trie [Knuth, 1997], nous avons finalement proposé un algorithme générique nommé TPSampling (Trie-based Pattern Sampling), qui tire des motifs proportionnellement à une mesure d'intérêt fondée sur la norme choisie dans $\mathcal{M}$ par l'utilisateur. L'une des principales forces de TPSampling est que, contrairement aux méthodes en deux étapes qui ont toujours besoin de pondérer les transactions de la base de données à chaque fois que l'on change de mesure d'intérêt, TPSampling utilise le même trie pondéré quelle que soit la mesure d'intérêt choisie.

Organisation du mémoire

La première partie du mémoire présente quelques définitions utiles pour la suite avant de passer à un état de l'art sur l'échantillonnage en sortie de motifs. Plus précisément, le chapitre 1 présente un bref état de l'art sur la fouille exhaustive de motifs intéressants tout en introduisant les notions de bases telles que celles de *langage*, *motif*, *base de données* et *contrainte*. Il se termine par une discussion sur les limites des méthodes de fouille exhaustive de motifs intéressants. Le chapitre 2 définit formellement le problème d'échantillonnage en sortie de motifs en les classant suivant *des critères dits en entrée* (sur les langages, mesures d'intérêt et contraintes prises en compte) et *des critères dits en sortie de la méthode* (sur l'exactitude du tirage et la complexité de la méthode). Le chapitre 3 montre les limites des méthodes d'échantillonnage en sortie de motifs de l'état de l'art en détaillant les problèmes soulignés ci-dessus.

La seconde partie du mémoire donne nos principales contributions dans cette thèse pour l'échantillonnage en sortie de motifs. Le chapitre 4 formalise d'abord notre cadre d'échantillonnage en sortie de motifs basé sur la classe de mesures d'intérêt fondées sur la norme $\mathcal{M}$. Ensuite il présente un algorithme générique d'échantillonnage en sortie de

motifs en deux étapes et se poursuit par une instanciation de l'algorithme générique aux bases de données transactionnelles. En utilisant la classe de mesures d'intérêt fondées sur la norme $\mathcal{M}$, le chapitre 5 présente l'algorithme générique NUSSampling dédié aux séquences pour l'échantillonnage en sortie en deux étapes. Sa partie expérimentale montre la rapidité de la méthode et compare les performances des classifieurs construits à l'aide de NUSSampling et ceux de l'état de l'art sur différents jeux de données. Toujours avec la classe de mesures $\mathcal{M}$, le chapitre 6 présente DDSampling, notre algorithme d'échantillonnage en sortie de motifs en deux étapes dans les bases de données distribuées. Sa partie expérimentale montre la capacité de DDSampling pour la détection de données aberrantes dans les triplestores du Web en évaluant la qualité de quelques classes du Web des données à l'aide des motifs échantillonnés avec DDSampling. Le chapitre 7 montre comment échantillonner efficacement des motifs à partir d'une base de données compressée sous la forme d'un trie. Pour terminer, une analyse théorique de TPSampling est réalisée avant de mener quelques études expérimentales relatives au coût de stockage et à la rapidité de la méthode. Enfin, le dernier chapitre du mémoire conclut nos travaux présentés dans cette thèse. Il discute d'une manière générale des forces et faiblesses des méthodes proposées avant de dégager des perspectives.

Première partie

Fouille de motifs : État de l'art

Introduction

Une méthode de fouille de données, quel que soit le langage sur lequel elle est appliquée, qu'elle soit optimale ou à base de contrainte, vise à satisfaire l'utilisateur suivant quatre grands axes : (i) la rapidité d'extraction des motifs, (ii) la prise en compte d'un nombre important de mesures d'intérêt qui offrent des motifs de bonne qualité mais n'affectant pas la rapidité de la méthode, (iii) la capacité à fournir un ensemble de motifs divers décrivant au mieux les instances de la base de données, (iv) la possibilité de contrôler la taille de la sortie afin que l'analyste ne soit pas débordé par les motifs retournés et puisse apprendre de ces motifs de la connaissance que cachent les données. L'état de l'art de cette thèse est à cheval entre *la fouille exhaustive de motifs à base de contrainte* et *l'échantillonnage en sortie de motifs*. Il permet ainsi de montrer les limites des méthodes de fouille exhaustives qui ont été surmontées par les premières méthodes proposées dans le domaine de l'échantillonnage en sortie de motifs.

Dans le cadre des études de [Mannila et Toivonen, 1997] les méthodes de fouille proposées sont dites exhaustives et visent à énumérer tous les motifs intéressants au vu de la contrainte choisie par l'utilisateur. Autrement dit, elles ont pour objectif de calculer une théorie de motifs d'un langage à partir d'une base de données suivant une contrainte choisie par l'utilisateur. Les méthodes de fouille exhaustives peuvent prendre en compte différents types de contraintes et offrent des motifs d'une très grande diversité. Cependant, le coût exorbitant du calcul de la théorie de certaines bases de données ainsi que le choix difficile de certains types de contraintes telles que les contraintes de fréquence minimale, d'aire minimale, etc, constituent de véritables obstacles pour la rapidité et le contrôle en sortie des motifs. C'est ainsi que les méthodes de fouille optimale telles que les top-k ont été proposées. Le principal problème avec ces méthodes est un défaut de diversité des motifs retournés. Pour résoudre l'ensemble de ces problèmes, les méthodes d'échantillonnage en sortie de motifs ont été proposées.

Les méthodes d'extraction de motifs par échantillonnage en sortie de la littérature [Al Hasan et Zaki, 2009, Boley *et al.*, 2011, Dzyuba *et al.*, 2017] visent principalement à répondre aux besoins des utilisateurs par rapport à la rapidité de l'extraction, au contrôle de la taille des sorties obtenues, à l'obtention d'un ensemble de motifs ayant une bonne diversité et finalement, à la prise en compte d'une large classe de mesures d'intérêt. Notons que ce dernier besoin joue fortement sur la rapidité d'une méthode d'échantillonnage en sortie. Autrement dit, plus la mesure d'intérêt est complexe, plus la rapidité diminue. Cette

rapidité est évaluée suivant le temps mis par une méthode pour tirer un motif directement à partir de la base de données et suivant une mesure d'intérêt choisie par l'utilisateur. De ce fait, nous avons vu que la méthode en deux étapes de [Boley *et al.*, 2011] est la plus rapide pour échantillonner des milliers de motifs en quelques millisecondes. Cependant, elle souffre d'une malédiction appelée la *malédiction de la longue traîne*, les motifs longs et peu fréquents étant favorisés par rapport aux motifs courts et très fréquents car ils sont parfois beaucoup plus nombreux.

Dans ce contexte, le chapitre 1 introduit la notion de fouille de motifs intéressants dans le cadre des études de [Mannila et Toivonen, 1997]. Il se termine par une présentation des limites des méthodes de fouille exhaustives de motifs intéressants.

Ensuite, le chapitre 2 présente un état de l'art sur l'échantillonnage en sortie de motifs. Après avoir formalisé le problème de l'échantillonnage en sortie de motifs, il donne une description détaillée de chacune des trois classes de méthodes existantes. Enfin, nous présentons quelques cas d'application avant de faire une synthèse sur l'ensemble des travaux du domaine de l'échantillonnage en sortie de motifs.

Enfin, le chapitre 3 présente les limites des méthodes d'échantillonnage en sortie et propose des défis à relever pour étendre ce paradigme.

Chapitre 1

Fouille exhaustive de motifs intéressants

Sommaire

D'après une étude faite en 2014 par [Giacometti *et al.*, 2014], 50% des publications faites dans le domaine de la fouille de données ont ciblé la découverte de motifs fréquents durant ces dernières décennies. La solution de cette problématique consiste à parcourir l'espace de recherche de tous les motifs présents dans la base de données afin de ne récupérer que ceux qui satisfont les contraintes posées par l'utilisateur. Dans ce chapitre, nous allons introduire la problématique de l'extraction de motifs sous contraintes à partir de bases de données structurées. Étant donnés que les types de motifs (itemsets, séquences et graphes) que nous avons traités ainsi que les contraintes sont divers et variés, nous avons fait le choix d'utiliser le cadre générique des études de [Mannila et Toivonen, 1997]. Heikki Mannila et Hannu Toivonen ont ainsi proposé un algorithme générique résolvant plusieurs tâches de l'extraction de motifs (séquentiels, ensemblistes,...) sous contraintes.

1.1 La fouille de motifs sous contraintes

Considérons une base de données $\mathcal{D}$, un langage $\mathcal{L}$ pour l'expression de propriétés dans les données et un prédicat de sélection booléen q permettant de dire si oui ou non, un

élément du langage doit être considéré comme pertinent sur la base de données $\mathcal{D}$. Une tâche d'extraction peut alors être formalisée comme le calcul de la théorie de $\mathcal{D}$ pour le langage $\mathcal{L}$ et le prédicat q, c'est-à-dire l'ensemble défini par :

$$\mathcal{T}h(\mathcal{L}, \mathcal{D}, q) = \{\varphi \in \mathcal{L} : q(\varphi, \mathcal{D}) \text{ est } vrai\}.$$

Cette formalisation fait apparaître un lien fort entre le langage et la base de données au travers des éléments du langage appelés *motifs*.

1.1.1 Langage, motif et base de données

Nous allons d'abord définir explicitement ces trois notions en faisant apparaître les relations qui existent entres elles.

Définition 1. *Un langage $\mathcal{L}$ est un ensemble d'éléments partiellement ordonné suivant une relation de spécialisation $\preceq$. Un motif φ est un élément du langage.*

Il existe plusieurs types de langages de motifs, mais nous allons nous limiter aux langages de motifs ensemblistes, séquentiels et de graphes. Un tel choix s'explique du fait que les transactions et les séquences correspondent aux types de données que nous avons traités dans cette thèse, et le langage de graphes est la première structure à laquelle l'échantillonnage en sortie a été appliqué.

Définition 2. *Une base de données $\mathcal{D}$ est un multi-ensemble d'éléments de $\mathcal{L}$. Une instance γ est un élément d'une base de données.*

Notons qu'il existe des cas tels que l'extraction d'itemsets fréquents dans une séquence [Cule *et al.*, 2009] où les instances de la base de données ne sont pas des éléments du langage $\mathcal{L}$. Dans ce cas, une relation de couverture doit être introduite pour faire le lien entre les instances de la base de données et les éléments du langage.

Introduit dès le début des années 90 par [Agrawal *et al.*, 1993], les motifs ensemblistes sont issus de l'étude du « panier du consommateur ». La découverte de motifs ensemblistes fréquents est une spécialisation très populaire de la découverte des motifs fréquents. Ce problème a été la base d'un nombre important d'approches de la fouille de données comme la découverte de règles d'association, la sélection d'attributs, etc.

Soit $\mathcal{I} = \{e_1, ..., e_n\}$ un ensemble fini de littéraux nommés items. Un itemset ou motif ensembliste φ est un sous-ensemble de $\mathcal{I}$. La taille de l'itemset φ, notée $|\varphi|$, est sa cardinalité. L'ensemble de tous les itemsets définis sur $\mathcal{I}$, dénoté par $\mathcal{L}_\mathcal{I} = 2^\mathcal{I}$, s'appelle le langage des motifs ensemblistes. Une base de données transactionnelles $\mathcal{T}$ définie sur $\mathcal{I}$ est un multi-ensemble d'itemsets de $\mathcal{L}_\mathcal{I}$. $|\mathcal{T}|$ est le nombre de transactions qu'elle contient. Dans la suite, sauf indication contraire, une transaction d'identifiant i est notée par t_i.

Exemple 1. *Le tableau 1.1 est un exemple de base de données contenant 4 transactions construites à partir de l'ensemble des items $\mathcal{I} = \{A, \cdots, F\}$. Les sous-ensembles AB, DE sont deux motifs ensemblistes de cette base de données car ils apparaissent respectivement dans les transactions d'identifiants 1, 4 et dans 1, 2, 3.*

TABLE 1.1 – Exemple d'une base de données transactionnelles $\mathcal{T}$

tid	Transaction
1	A B D E F
2	A D E
3	B D E
4	A B C D F

Muni d'une relation de spécialisation, un langage a une forme particulière suivant la structure des données. Dans la suite de cette section, nous allons décrire la structuration de l'espace des motifs des langages sur lesquels portent nos principales contributions.

Nous disposons d'un langage de motifs $\mathcal{L}$, muni d'une relation de spécialisation $\preceq$ qui est un ordre partiel où $\varphi_1 \preceq \varphi_2$ signifie que φ_1 est plus général que φ_2 (de manière duale, que φ_2 est plus spécifique que φ_1). Par exemple, pour le langage des motifs ensemblistes $\mathcal{L}_\mathcal{I}$, la relation d'inclusion est une relation de spécialisation. Dans ce cas, un itemset φ_1 est plus général qu'un itemset φ_2 si et seulement si $\varphi_1 \subseteq \varphi_2$. Cela signifie que dans ce cas, la relation de spécialisation $\preceq$ est la relation d'inclusion $\subseteq$. φ_2 est une spécialisation immédiate de φ_1, dénoté par $\varphi_1 \dot{\prec} \varphi_2$, (inversement, φ_1 est une généralisation immédiate de φ_2) s'il n'existe pas de motif φ' tel que $\varphi_1 \prec \varphi' \prec \varphi_2$. Si un motif φ ne possède pas de généralisation immédiate, sa norme est définie comme nulle, et on note : $||\varphi|| = 0$. Sinon, la norme d'un motif est définie comme étant la plus petite norme de ses généralisations immédiates plus 1 : $||\varphi|| = \arg\min_{\varphi' \dot{\prec} \varphi} ||\varphi'|| + 1$ s'il existe un motif $\varphi' \prec \varphi$. Par exemple, pour les itemsets, l'ensemble vide ne dispose pas de généralisation et sa norme est 0. Plus généralement, la norme d'un itemset φ correspond à sa taille : $||\varphi|| = |\varphi|$, si $\varphi \in \mathcal{L}_\mathcal{I}$. La figure 1.1 donne une vision sur la structuration des langages de motifs transactionnels et séquentiels (présentés ci-après).

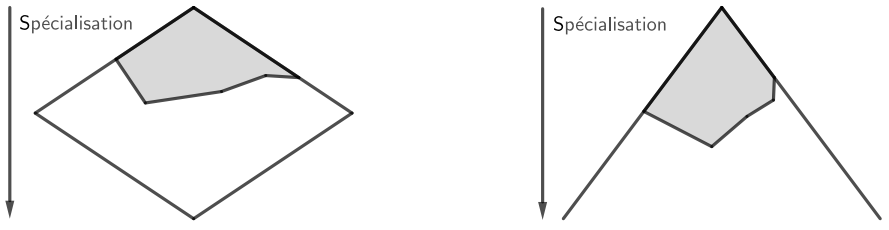


FIGURE 1.1 – Espaces de recherche des motifs ensemblistes et séquentiels [Soulet, 2006].

Sur la figure 1.1, les motifs les plus généraux (resp. spécifiques) sont situés en haut (resp. en bas). L'espace de recherche des motifs ensemblistes constitue un treillis dont la forme en losange traduit la répartition des motifs en fonction de leur niveau de spécialisation. La répartition des motifs séquentiels constitue un triangle ouvert car le langage est infini. Les formes grisées schématisent les motifs plus généraux qu'au moins une instance de la

TABLE 1.2 – Calcul de nombre de généralisations suivant la norme

tid	Φ	$\Phi_{\leq 0}$	$\Phi_{\leq 3}$	$\Phi_{[1,3]}$
1	32	1	26	25
2	8	1	8	7
3	8	1	8	7
4	8	1	26	25

base de données.

Nous introduisons maintenant quelques notations pour désigner l'ensemble des généralisations d'une instance ou l'ensemble de ses généralisations d'une norme donnée.

Définition 3. *L'ensemble des généralisations d'une instance γ d'une base de données $\mathcal{D}$ est défini et noté par $\phi(\gamma) = \{\varphi \in \mathcal{L} : \varphi \preceq \gamma\}$. Sa cardinalité est notée $\Phi(\gamma) = |\phi(\gamma)|$. L'ensemble des généralisations de norme ℓ de l'instance γ est noté $\phi_\ell(\gamma)$. Sa cardinalité est notée $\Phi_\ell(\gamma) = |\phi_\ell(\gamma)|$.*

Dans la suite, nous notons par $\Phi_{\leq \ell}(\gamma)$ la cardinalité de l'ensemble des motifs de norme inférieure ou égale à ℓ de l'instance γ, $\Phi_{\leq \ell}(\gamma) = |\phi_{\leq \ell}(\gamma)|$. Enfin, le nombre de généralisations de l'instance γ de normes appartenant à l'intervalle $[\ell'..\ell]$ est noté par $\Phi_{[\ell'..\ell]}(\gamma) = \Phi_{\leq \ell}(\gamma) - \Phi_{<\ell'}(\gamma)$.

Exemple 2. *En considérant la base de données $\mathcal{T}$ présenté précédemment, nous avons calculé le nombre de généralisations de chaque instance au tableau 1.2.*

Avec la transaction d'identifiant 2, on a $\phi(ADE) = \{\emptyset, A, D, E, AD, AE, DE, ADE\}$. Alors $\Phi(ADE) = |\phi(ADE)| = 8$ et $\Phi_{\leq 0}(ADE) = |\{\emptyset\}| = 1$, $\Phi_{\leq 3}(ADE) = |\phi(ADE)| = 8$. Le nombre de généralisations de l'instance ADE ayant une norme comprise entre 1 et 3 est $\Phi_{[1..3]}(ADE) = \Phi_{\leq 3}(ADE) - \Phi_{<1}(ADE) = 8 - 1 = 7$.

1.1.2 Exemples de langages de motifs structurés

Nous allons poursuivre cette section en donnant d'autres exemples de langage que celui des itemsets, à savoir des exemples de langages séquentiels et de graphes. Nous considérons dans le reste de cette partie les bases des données liées ou bases de données du Web sémantique pour donner des exemples. Cela nous permettra aussi de mener des expérimentations avec des données réelles dans la partie contribution de la thèse. Dans le domaine du Web sémantique, les entités sont décrites par des propriétés stockées dans un ou plusieurs triplestores. En prenant la classe *Person* par exemple, certaines propriétés décrivant dans le Web l'entité associée à la personne dénommée "*Cheikh Amadou Bamba*" (*<http ://dbpedia.org/resource/Amadou_Bamba>*) sont stockées à la fois dans *DBpedia* et *Wikidata*. Si nous nous limitons à *DBpedia*, chaque ensemble de propriétés décrivant une même personne forme un motif ensembliste ou un itemset. La figure 1.2 représente un extrait de données liées issu du Web sémantique ou LOD (Linked Open Data).

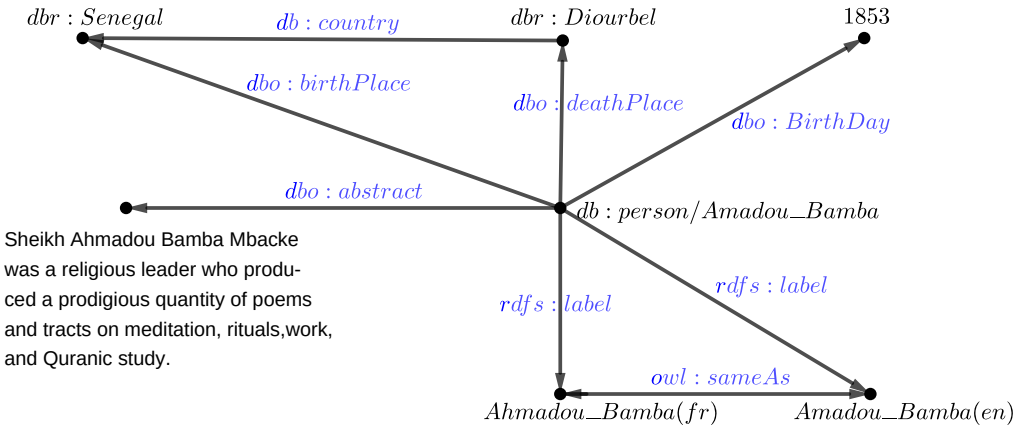


FIGURE 1.2 – Extrait de graphe de données liées du Web

Motif séquentiel Maintenant, si on se restreint seulement aux poètes de la classe *Person* de *DBpedia*, on peut voir que le grand marabout "*Cheikh Amadou Bamba*" fut aussi un grand poète. Si on considère l'ensemble des mots clés des poèmes qu'il a écrit par an, on obtient une séquence d'ensemble de mots clés. Ainsi, la liste des ensembles de mots clés ordonnés suivant les dates de publication constitue un motif séquentiel.

L'extraction de motifs séquentiels a été introduite il y a deux décennies par [Agrawal et Srikant, 1995] et son utilité a été largement démontrée pour différentes tâches d'extraction et domaines d'application telles que l'exploration de l'utilisation du web, la fouille de texte, la bioinformatique, la détection de fraude, etc.

Définition 4 (Séquence)**.** *Une* séquence $s = \langle X_1 \dots X_n \rangle$ *définie sur un ensemble fini de littéraux* $\mathcal{I}$ *est une liste ordonnée d'itemsets* $X_i \subseteq \mathcal{I}$ $(1 \leq i \leq n,\ n \in \mathbb{N})$*.* n *est la* taille *de la séquence* s *dénotée par* $|s|$*. Une base de séquences, notée par* $\mathcal{S}$*, est un multi-ensemble de séquences.*

Définition 5 (Sous-séquence)**.** *Une séquence* $s' = \langle X'_1 \dots X'_{n'} \rangle$ *est une sous-séquence d'une séquence* $s = \langle X_1 \dots X_n \rangle$*, dénoté par* $s' \sqsubseteq s$*, s'il existe une séquence d'indices* $1 \leq i_1 < i_2 < \dots < i_{n'} \leq n$ *telle que pour tout* $j \in [1..n']$*, on a* $X'_j \subseteq X_{i_j}$*. On dit aussi que la sous-séquence* s' *est plus générale que la sous-séquence* s*. Dans le cas des bases de données séquentielles, la relation de spécialisation est la relation de subsomption* $\sqsubseteq$*. La* norme *de la séquence* s*, dénotée par* $\|s\|$*, est la somme des cardinalités de tous ses itemsets, i.e.* $\|s\| = \sum_{i=1}^{n} |X_i|$*. Le langage de motifs séquentiels de la base de données* $\mathcal{S}$ *noté par* $\mathcal{L}_\mathcal{S}$ *est l'ensemble des sous-séquences des séquences de* $\mathcal{S}$*.*

Exemple 3. *Nous utilisons la base de données séquentielles* $\mathcal{S}$ *présentée au tableau 1.3.*

Cette base de données contient 4 *séquences* s_1*,* s_2*,* s_3 *et* s_4 *définies sur l'ensemble des items* $\mathcal{I} = \{a, b, c, d\}$*. Par exemple, la taille de* $s_1 = \langle (ab)c \rangle$ *est égale à* 2*, i.e.* $|s_1| = 2$*, alors que sa norme est égale à* 3*, i.e.* $\|s_1\| = 2 + 1 = 3$*. Finalement, les sous-séquences de la séquence* s_1 *sont :* $\langle\rangle, \langle a \rangle, \langle b \rangle, \langle c \rangle, \langle (ab) \rangle, \langle ac \rangle, \langle bc \rangle, \langle (ab)c \rangle$*. Si on se restreint au motif de norme* $\ell = 2$*, alors on a* $\phi_\ell(s_1) = \{\langle (ab) \rangle, \langle ac \rangle, \langle (bc) \rangle\}$*, ce qui nous donne* $\Phi_\ell(s_1) =$

TABLE 1.3 – Une base de données séquentielles $\mathcal{S}$

Sid	Séquence
s_1	$\langle(ab)c\rangle$
s_2	$\langle(ab)c(ac)\rangle$
s_3	$\langle c(ac)\rangle$
s_4	$\langle(ab)(cd)\rangle$

$|\phi_\ell(s_1)| = 2$. *Le nombre de généralisations de* s_1 *de normes comprises entre 1 et 2 est égal à* $\Phi_{[1..2]}(s_1) = \Phi_{\leq 2}(s_1) - \Phi_{<1}(s_1) = 7 - 1 = 6$.

Motifs de graphes Les motifs de graphes permettent de décrire les corrélations entres les entités des bases de données graphes telles que les entités du Web sémantique. L'extraction de sous-graphes intéressants à partir d'une base de données de graphes est un domaine de recherche très étudié dans le domaine de la fouille de données [Cook et Holder, 1994, Inokuchi *et al.*, 2000, Xifeng Yan et Jiawei Han, 2002]. Majoritairement, les travaux proposés consistent à identifier des sous-graphes fréquents dans des ensembles de données de graphes en utilisant des algorithmes très sophistiqués minimisant le temps de calcul. En effet, les graphes font partie de la famille des données structurées les plus complexes. Par exemple, toutes les données du Web au format RDF (Resource Description Framework) sont stockées dans le LOD (Linked Open Data) sous forme de graphes (voir extrait à la figure 1.2).

Notons que la complexité structurelle des graphes est fortement liée au fait qu'ils encodent deux informations au minimum : l'information représentée par les sommets, ainsi que celle représentée par les arêtes.

Définition 6 (Graphe)**.** *Un graphe* G *est un couple d'ensembles finis de sommets (vertices en anglais) noté par* $V = \{v_1, v_2, \cdots, v_n\}$*, et d'arcs (edges en anglais) noté par* $E = \{e_1, e_2, \cdots, e_m\}$ *tels que* $E \subseteq V \times V$*, avec* $m, n \in \mathbb{N}^*$*. La taille du graphe, notée par* $|G|$ *est la cardinalité de l'ensemble de ses arcs,* $|G| = |E|$*. Une base de données graphes* $\mathcal{G}$ *est un multi-ensemble de graphes.*

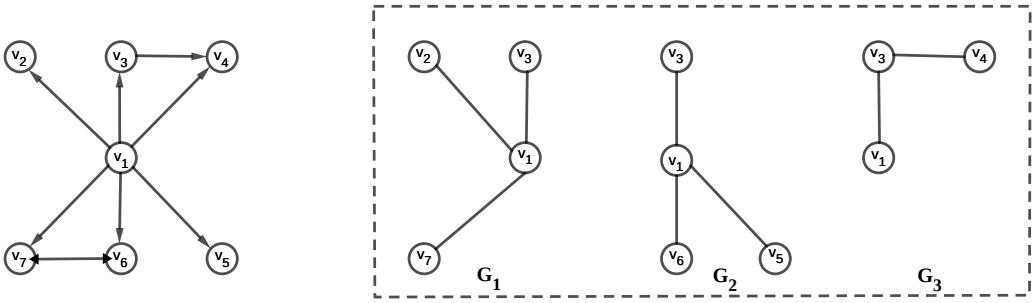
Un graphe $G = (V, E)$ est dit non-orienté si pour tout arc $(x, y) \in E$, on a aussi $(y, x) \in E$. Dans ce cas, les arcs de E sont appelés arrêtes. Sinon, un graphe est dit orienté.

Dans cette thèse, nous allons nous limiter aux graphes simples non étiquetés, mais notons qu'il existe d'autres types de graphes comme les *graphes valués*, les *pseudo-graphes* et les *multi-graphes*.

Exemple 4. *Le graphe de la figure 1.2 que nous avons reporté, sans les étiquettes, à la figure 1.3-(a) est un graphe orienté de taille 8 construit à partir de l'ensemble des sommets* $V = \{v_1, \cdots, v_7\}$ *et de l'ensemble des arcs définis sur* $V \times V$ *par* $E = \{\langle v_1, v_2\rangle, \langle v_1, v_3\rangle, \langle v_1, v_4\rangle, \langle v_1, v_5\rangle, \langle v_1, v_6\rangle, \langle v_1, v_7\rangle, \langle v_3, v_4\rangle, \langle v_6, v_7\rangle, \langle v_7, v_6\rangle\}$.

La figure 1.3-(b) est une base de données de 3 graphes non-orientés identifiés par G_1*,* G_2 *et* G_3*. Le premier graphe* G_1 *est construit à partir de 4 sommets* $\{v_1, v_2, v_3, v_7\}$*, le*

second graphe G_2 à partir de 4 sommets $\{v_1, v_3, v_5, v_6\}$ et le troisième graphe G_3 à partir de 3 sommets $\{v_1, v_3, v_4\}$.



(a) Graphe orienté

(b) Base de données de graphes non orientés $\mathcal{G} = \{G_1, G_2, G_3\}$

FIGURE 1.3 – Quelques exemples de graphes

Définition 7 (Sous-graphe)**.** *$G' = (V', E')$ est un sous-graphe de $G = (V, E)$ si $V' \subseteq V$ et $E' \subseteq E$. Nous disons aussi que G' est plus général que G. Ce qui veut dire aussi que la relation de spécialisation $\preceq$ est remplacée par la relation d'inclusion $\subseteq$. Le langage de motifs de graphes de la base de données $\mathcal{G}$ noté par $\mathcal{L}_\mathcal{G}$ est l'ensemble des sous-graphes de $\mathcal{G}$.*

Exemple 5. *Nous remarquons que seul le sous-graphe $v_1 - v_3$ a une fréquence supérieure à 1 dans $\mathcal{G}$, $freq(v_1 - v_3, \mathcal{G}) = 3$.*

Pour finir, notons que le langage des motifs ensemblistes correspond exactement à tous les sous-ensembles de $\mathcal{I}$ i.e., $\mathcal{L}_\mathcal{I} = 2^\mathcal{I}$, et donc, que c'est un ensemble fini. Mais un langage de motifs peut être infini dans certains cas comme pour les séquences. En effet, pour un ensemble d'items spécifiés $\mathcal{I}$, le langage des motifs séquentiels $\mathcal{L}_\mathcal{S}$, représente l'ensemble de toutes les listes possibles et ordonnées d'éléments de $\mathcal{L}_\mathcal{I}$. Autrement dit, il n'est pas possible d'énumérer l'ensemble des motifs séquentiels possibles d'un langage. Cependant, pour une base de données transactionnelles $\mathcal{T}$, il est bien possible d'énumérer l'ensemble des motifs qui y sont présents, du moins si on a assez de mémoire et de patience. En effet, pour une base de données construite sur 500 d'items, le nombre de motifs possible est de 2^{500}, nombre supérieur au nombre approximatif d'atomes dans l'univers, donc un espace de recherche pléthorique. Pour éviter le problème d'explosion du langage de motifs, il devient primordial de poser des contraintes pour ne sélectionner que les motifs qui les satisfont.

1.1.3 Les contraintes

Une contrainte évalue si un motif φ est intéressant ou non. Elle est aussi appelée prédicat ou requête, et peut être définie comme suit :

Définition 8 (Contrainte)**.** *Une contrainte q est un prédicat booléen tel que pour tout motif φ d'un langage $\mathcal{L}$ et une base de données $\mathcal{D}$ on a : $q(\varphi, \mathcal{D}) \in \{vrai, faux\}$.*

Cette définition met en exergue le lien fort que la contrainte établit entre le langage et la base de données. Notons que la définition 8 n'exige aucune propriété sur la contrainte. Cependant, dans cette thèse, notre contribution s'appuie sur les contraintes de types syntaxiques.

Contraintes syntaxiques : Une contrainte q est dite syntaxique si pour tout motif φ et pour toutes bases de données $\mathcal{D}$ et $\mathcal{D}'$, on a $q(\varphi, \mathcal{D}) = q(\varphi, \mathcal{D}')$. Les contraintes syntaxiques dont nous allons le plus parler dans ce document sont les contraintes de tailles ou de normes maximales ou minimales. Elles sont souvent utilisées pour contrôler les tailles ou les normes des motifs extraits. Dans [Fournier-Viger *et al.*, 2013a], les auteurs ont permis de poser des contraintes de norme maximale sur les motifs séquentiels extraits pour retrouver les top-k motifs. On sait que les motifs les plus courts sont souvent les plus fréquents. Ainsi, étant donné une base de données et une contrainte de norme maximale, il suffit de fouiller dans l'espace des motifs de normes maximales pas trop élevées pour avoir des motifs représentatifs de la base de données. On peut alors noter qu'en présence de contraintes syntaxiques, la taille de l'espace des motifs est plus réduite.

Les langages de motifs de la figure 1.4 sont des restrictions de ceux de la figure 1.1. Le triangle en gris foncé représente les motifs qui sont en accord avec la contrainte syntaxique posée par l'utilisateur et présents dans la base de données.

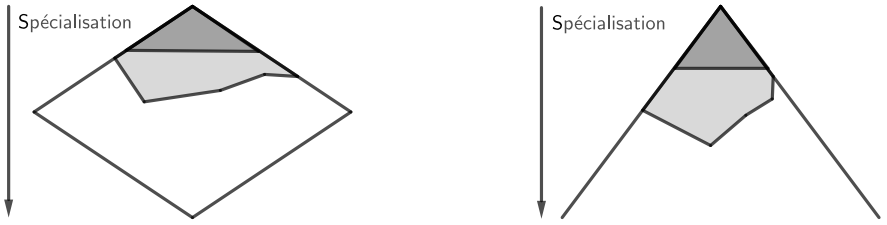


FIGURE 1.4 – Espaces de recherche des motifs ensemblistes et séquentiels sous contrainte syntaxique.

Cependant dans la fouille de motifs intéressants, ces contraintes syntaxiques sont souvent associées à des mesures d'intérêt permettant d'évaluer le degré de pertinence des motifs de l'espace de recherche.

Contraintes basées sur les mesures d'intérêt : Les mesures d'intérêt permettent à l'utilisateur d'évaluer la pertinence d'un motif du langage par rapport à la base de données. Les plus populaires des mesures d'intérêt dans la littérature sont probablement le *support relatif* et le *support absolu* que nous appelons respectivement le *support* et la *fréquence*. Le support (respectivement la fréquence) d'un motif φ dans une base de données $\mathcal{D}$ est noté(e) par $supp(\varphi, \mathcal{D})$ (respectivement $freq(\varphi, \mathcal{D})$).

Définition 9 (Fréquence et Support)**.** *Soit $\mathcal{D}$ une base de données et φ un motif défini*

dans $\mathcal{D}$. La fréquence du motif φ dans $\mathcal{D}$ est donnée par la formule suivante :

$$freq(\varphi, \mathcal{D}) = |\{\gamma \in \mathcal{D} : \varphi \preceq \gamma\}|.$$

Le support du motif φ est défini comme suit :

$$supp(\varphi, \mathcal{D}) = \frac{freq(\varphi, \mathcal{D})}{|\mathcal{D}|}.$$

Il est aussi très courant d'associer à un motif un intérêt en fonction de sa norme afin de favoriser les motifs de très grandes normes dans la base de données comme c'est le cas de l'aire.

Définition 10 (Mesures d'aire)**.** *Soit $\mathcal{D}$ une base de données, φ un motif défini dans $\mathcal{D}$. La mesure d'aire d'un motif φ dans la base de données $\mathcal{D}$ est le produit entre sa norme et sa fréquence dans $\mathcal{D}$:*

$$area(\varphi, \mathcal{D}) = freq(\varphi, \mathcal{D}) \times ||\varphi||.$$

Exemple 6. *En considérant l'exemple du tableau 1.1, on a $freq(DE, \mathcal{T}) = 3$ et aussi $supp(DE, \mathcal{T}) = \frac{3}{4}$ car le motif DE apparaît dans les transactions d'identifiants 1, 2 et 3. Si on considère comme mesure d'intérêt l'aire du motif, alors $area(DE, \mathcal{T}) = 6$ car nous avons deux items D et E.*

Dans le tableau 1.3, la fréquence du motif séquentiel $\varphi = \langle cc \rangle$ est égale à la cardinalité de l'ensemble de ses spécialisations $\{s_2, s_3\}$. Donc $freq(\langle cc \rangle, \mathcal{S}) = 2$ et $supp(\langle cc \rangle, \mathcal{S}) = \frac{2}{4}$. Notons que $area(\langle cc \rangle, \mathcal{S}) = freq(\langle cc \rangle, \mathcal{S}) \times ||\langle cc \rangle|| = 2 \times (1 + 1) = 4$.

Nous parlerons à plusieurs reprises d'autres mesures d'intérêt plus complexes telles que la *fréquence au carré pondérée* et la *discriminativité*. Ces mesures que nous allons définir dans les lignes suivantes, seront évoquées en guise d'exemples.

Définition 11 (Fréquence au carré et Discriminativité)**.** *Etant donné une base de données $\mathcal{D}$, la fréquence au carré d'un motif φ de $\mathcal{D}$ est définie par :*

$$sqfreq(\varphi, \mathcal{D}) = freq(\varphi, \mathcal{D})^2.$$

De même, étant donné une base de données positive $\mathcal{D}^+$ et une base de données négative $\mathcal{D}^-$, la discriminativité d'un motif φ est définie par :

$$disc(\varphi, \mathcal{D}) = freq(\varphi, \mathcal{D}^+) \times (|\mathcal{D}^-| - freq(\varphi, \mathcal{D}^-)).$$

La pertinence d'un motif suivant une contrainte basée sur une mesure d'intérêt est généralement définie par rapport à un seuil donné. De telles contraintes sont souvent appelées des *contraintes de seuil minimal*.

Définition 12 (Contrainte de seuil minimal)**.** *Soient $\mathcal{D}$ une base de données, m une mesure d'intérêt et minseuil un seuil minimal. Un motif φ du langage $\mathcal{L}$ de la base de donnée $\mathcal{D}$ est dit intéressant si sa mesure d'intérêt dans la base de données $m(\varphi, \mathcal{D})$ est au moins égale au seuil minimal minseuil. La contrainte minseuil $\leq m(\varphi, \mathcal{D})$ est appelée une contrainte de seuil minimal.*

En prenant le support comme mesure d'intérêt, un motif intéressant est un motif (itemset, sous-séquence, sous-graphe, etc) qui apparaît dans une base de données $\mathcal{D}$ avec un support supérieur ou égal à un seuil de support minimal donné *minsup*. Formellement, φ est intéressant par rapport à *minsup* si : $supp(\varphi, \mathcal{D}) \geq minsup$. Dans ce cas, pour un motif φ du langage d'une base de données $\mathcal{D}$, la contrainte $supp(\varphi, \mathcal{D}) \geq minsup$ est soit *vraie* soit *fausse*.

Exemple 7. *Considérons que* $minsup = \frac{3}{4}$. *Alors le motif* DE *est intéressant dans la base de données transactionnelles* $\mathcal{T}$ *du tableau 1.1 car son support est égal à* $\frac{3}{4}$. *Alors qu'avec la base de données de séquences* $\mathcal{S}$ *du tableau 1.3, le motif* $\langle cc \rangle$ *n'est pas intéressant car son support est plus petit que le support minimal,* $\frac{1}{2} < \frac{3}{4}$. *Ainsi, nous disons que la contrainte* $supp(DE, \mathcal{T}) \geq \frac{3}{4}$ *est* vraie *alors que la contrainte* $supp(\langle cc \rangle, \mathcal{S}) \geq \frac{3}{4}$ *est* fausse.

1.2 Calcul d'une théorie de motifs intéressants

Les types de motifs extraits dépendent du domaine d'application, donc de la base de données à exploiter, mais aussi à la fois du langage (itemsets, séquences,$\cdots$) et des contraintes posées par l'utilisateur. L'extraction de motifs à partir d'une base de données $\mathcal{D}$ n'est rien d'autre que la sélection des motifs d'un langage $\mathcal{L}$ jugés intéressants par rapport à une contrainte q. Plus formellement, il s'agit de déterminer la théorie correspondante. Cependant, la construction d'une théorie est souvent très difficile car elle est fortement liée à la contrainte posée sur les motifs et la taille du langage. L'objectif d'un algorithme d'extraction de motifs est alors de trouver l'ensemble des motifs potentiellement intéressants vis-à-vis de l'utilisateur tout en minimisant le nombre de motifs parcourus dans l'espace de recherche. Dans [Mannila et Toivonen, 1997], les auteurs ont proposé un algorithme générique d'extraction exhaustive de motifs structurés qui se base sur le calcul d'une théorie à l'Apriori [Agrawal et Srikant, 1994]. Il exploite le principe d'anti-monotonie afin de minimiser le coût de parcours de l'espace de recherche. Notamment, c'est la propriété sans laquelle la généralisation dans le cadre de Mannila et Toivenen est impossible, car cela reviendrait à visiter chaque motif du langage.

Définition 13 (Monotonie ou anti-monotonie [Agrawal et Srikant, 1994])**.** *Une contrainte* q *est monotone (respectivement anti-monotone) suivant la relation de spécialisation* $\preceq$ *si et seulement si pour tout motif satisfaisant* q*, ses spécialisations (respectivement généralisations) satisfont également la contrainte* q.

La contrainte basée sur le support ou la fréquence est anti-monotone. En effet, si un motif φ est intéressant par rapport à un seuil donné, alors toute généralisation de φ est aussi intéressante par rapport à ce seuil. D'autre part, si φ n'est pas intéressant par rapport à ce seuil, alors toute spécialisation de φ est aussi non intéressante. Formellement, si la contrainte $supp(\varphi, \mathcal{D}) \geq minsup$ est *vraie* (respectivement *fausse*) alors pour toute généralisation (respectivement spécialisation) φ' de φ, la contrainte $supp(\varphi', \mathcal{D}) \geq minsup$ est *vraie* (respectivement *fausse*).

1.2.1 Algorithme générique d'extraction de motifs intéressants

Le premier algorithme d'extraction de motifs intéressants APRIORI a été présenté par Agrawal et Srikant dans [Agrawal et Srikant, 1994] pour les bases de données transactionnelles en se basant sur l'algorithme AIS [Agrawal *et al.*, 1993]. Par rapport à ces premières approches, la mesure d'intérêt considérée est la fréquence ou le support des motifs du langage. Par la suite, plusieurs algorithmes se sont basés sur la méthode d'APRIORI parmi lesquels nous pouvons citer GSP par [Srikant et Agrawal, 1996] pour la découverte de sous-séquences fréquentes à partir d'une base de données séquentielles et AGM et FSG respectivement par [Inokuchi *et al.*, 2000] et [Kuramochi et Karypis, 2001] pour trouver des sous-graphes fréquents dans des bases de graphes. D'une manière générale, le principe de l'algorithme APRIORI est d'abord de générer tous les motifs d'un même niveau. Ensuite, on élague tous les motifs n'ayant pas un support au moins égal au seuil de support minimal *minsup* (les motifs non fréquents).

Plus généralement, l'algorithme 1 de [Mannila et Toivonen, 1997] prend en entrée une base de données $\mathcal{D}$, un langage de motifs $\mathcal{L}$ muni d'une relation de spécialisation $\preceq$ et d'une contrainte anti-monotone q. Ensuite il s'exécute par niveau. A la ligne 1, il commence par récupérer l'ensemble des motifs les plus généraux du langage $\mathcal{L}$. Ainsi, les motifs les plus généraux qui sont en accord avec la contrainte q constituent les motifs du premier niveau (ligne 6). Dans la suite, on génère les candidats du niveau i+1 en considérant toutes les spécialisations immédiates des motifs du niveau i. Parmi ces spécialisations, seules celles dont toutes les généralisations ont été sélectionnées aux niveaux inférieurs sont gardées, les autres étant élaguées (ligne 7). Le processus est ainsi répété jusqu'à ce qu'il n'y ait plus la possibilité de générer de nouveaux candidats.

Algorithm 1 Algorithme générique de [Mannila et Toivonen, 1997]

1: **Input :** Une base de données $\mathcal{D}$, un langage $\mathcal{L}$, et une relation de spécialisation $\preceq$ par rapport à une contrainte anti-monotone q
2: **Output :** la théorie $\mathcal{T}h(\mathcal{L}, \mathcal{D}, q)$
3: $C_1 \leftarrow \{\varphi \in \mathcal{L} \mid \nexists\ \varphi' \prec \varphi\}$
4: $i \leftarrow 1$
5: **while** $C_i \neq \emptyset$ **do**
6: $\quad \mathbb{T}_i \leftarrow \{\varphi \in C_i \mid q(\varphi, \mathcal{D}) \text{ est vrai}\}$ ▷ Évaluation des candidats satisfaisant q
7: $\quad C_{i+1} \leftarrow \{\varphi \in \mathcal{L} \mid (\forall\ \varphi' \prec \varphi)(\varphi' \in \bigcup_{j \leq i} \mathbb{T}_j)\} \setminus \bigcup_{j \leq i} C_j$ ▷ Génération des candidats
8: $\quad i \leftarrow i + 1$
9: **return** $\bigcup_{j<i} \mathbb{T}_j$ ▷ L'ensemble des motifs de la théorie

La difficulté de l'approche de l'algorithme 1 est d'une part liée à la nature de la structure des éléments de la base de données en question. En effet, l'évaluation des candidats satisfaisant la contrainte q à la ligne 6 de l'algorithme 1 est très coûteuse avec les données complexes. D'autre part, la génération des candidats peut être très coûteuse, en particulier en espace, les candidats pouvant être très nombreux.

1.2.2 Apriori : instance de [Mannila et Toivonen, 1997]

Apriori [Agrawal et Srikant, 1994] est une instance de l'algorithme 1. Il prend en entrée une base de données transactionnelles $\mathcal{T}$, ainsi qu'un seuil minimum de support *minsup*. Dans ce cas, la contrainte q est vérifiée si la fréquence du motif est supérieure ou égale à *minsup*. En exécution, il vise la récupération de tous les itemsets fréquents par rapport à *minsup* dans $\mathcal{T}$, à l'aide d'une recherche par niveau. Pour se faire, à la ligne 6 de l'algorithme 1, on commence par rechercher les 1-itemsets fréquents, ceux dont les supports sont supérieurs au seuil *minsup*, $\mathcal{T}h(\mathcal{L}_\mathcal{I}, \mathcal{T}, freq(\varphi, \mathcal{T}) \geq minsup \wedge ||\varphi|| = 1)$, en faisant une passe dans la base de données pour calculer les fréquences par comptage. Les 1-itemsets fréquents sont utilisés, par union deux-à-deux, pour générer les 2-itemsets candidats (ligne 7), puis filtrer par rapport au seuil minimal *minsup* (ligne 6) pour calculer $\mathcal{T}h(\mathcal{L}_\mathcal{I}, \mathcal{T}, freq(\varphi, \mathcal{T}) \geq minsup \wedge ||\varphi|| = 2)$. A partir du niveau 2, après chaque phase d'élagage, la génération des motifs du niveau $k > 2$ se fait comme suit : l'union de deux k-itemsets est candidat à l'étape $k+1$ si et seulement si leur intersection contient un seul item, et donne donc un $(k+1)$-itemset. Enfin, on répète le processus jusqu'à ce que la génération de nouveaux candidats soit impossible. On peut noter que l'anti-monotonie de la contrainte de seuil minimal est exploitée pour ne traiter qu'une partie de l'ensemble des itemsets candidats.

Notons qu'il existe d'autres méthodes de fouille de motifs intéressants comme Fsg [Kuramochi et Karypis, 2001] et Gsp [Srikant et Agrawal, 1996], mais toutes utilisent ce même principe : i) définition d'une classe de contraintes ou mesures vérifiant une propriété, (ii) proposition d'un algorithme d'élagage utilisant cette propriété.

1.3 Discussion

Les algorithmes d'extraction exhaustive de motifs intéressants sont des méthodes très importantes pour la découverte de connaissances dans une base de données. Leurs utilités ont été largement démontrées dans beaucoup d'applications de la vie réelle. Cependant, elles présentent d'importantes limites mêmes si d'autres propositions ont été faites pour les contourner.

1.3.1 Limites des méthodes de fouilles de motifs avec contrainte

Si l'approche d'origine d'Apriori est efficace pour traiter des données faiblement corrélées telles que les données transactionnelles, les données les plus corrélées et les plus complexes telles que les séquences et les graphes entraînent une chute terrible de ses performances. Depuis lors, de nombreuses études ont été consacrées aux améliorations ou extensions d'Apriori, telles que la technique de partitionnement [Savasere *et al.*, 1995], la méthode d'échantillonnage en entrée [Toivonen *et al.*, 1996], l'extraction incrémentielle [Cheung *et al.*, 1996], les techniques de calcul parallèle et distribuée [Zaki *et al.*, 1997b], etc. Mais dans tous les cas, nous notons que le principal problème de la méthode d'Apriori et de ses extensions est que (i) le nombre de passes pour calculer les supports des motifs candidats reste très élevé et (ii) la génération des motifs candidats n'est pas toujours faisable.

D'une manière plus générale, l'efficacité des méthodes de fouille exhaustives de motifs intéressants se heurte à deux problèmes :

1. Le premier problème est lié au choix de la valeur du seuil minimal. Malheureusement, il n'est pas possible d'utiliser directement les motifs fréquents si le seuil de support minimal *minsup* est petit car ils sont bien trop nombreux. A l'opposé, si *minsup* est trop grand, certaines instances seront peu ou pas décrites.
2. L'ensemble complet des motifs ayant satisfait la contrainte peut contenir de nombreuses redondances. Autrement dit, il peut être très difficile pour un expert d'évaluer et d'explorer l'ensemble des motifs extraits si la base est très grande.

La persistance de ces problèmes sur les données complexes en terme de rapidité et de faisabilité, a conduit à la proposition d'autres techniques de fouille de motifs intéressants.

1.3.2 Méthodes d'extraction de motifs optimaux

D'abord, pour résoudre le problème du nombre de motifs en sortie, plusieurs méthodes [Burdick *et al.*, 2001, Calders *et al.*, 2006, Hamrouni, 2012] ont été développées en passant par une représentation condensée des motifs intéressants. Malheureusement les techniques de représentation condensée peuvent rester très coûteuses en temps de calcul et ne résolvent pas entièrement ce problème.

Suite à cette persistance, un grand nombre de propositions sont destinées à se concentrer sur les meilleurs motifs en fonction d'un ordre de préférence spécifié par l'utilisateur [Soulet, 2017]. Cette relation de préférence est une relation binaire R (partielle ou totale), où $\varphi R \varphi'$ signifie que φ est préféré à φ' ou inversement que φ' est dominé par φ. Par exemple, on peut considérer que le motif φ est préféré à φ' si la fréquence de φ est plus élevée que celle de φ'. Dans ce cas, nous avons : $\varphi R_{freq} \varphi' \Leftrightarrow freq(\varphi, \mathcal{D}) \geq freq(\varphi', \mathcal{D})$. Dans ce cadre, le nouveau problème de base peut être formulé comme suit :

Soient $\mathcal{L}$ un langage de motifs, $\mathcal{D}$ une base de données et R une relation de préférence. L'objectif de la fouille de motifs à base de préférences utilisateurs est de retrouver les k motifs du langage $\mathcal{L}$ les plus intéressants dans la base de données $\mathcal{D}$ suivant la relation R :

$$\mathcal{B}est_k(\mathcal{L}, \mathcal{D}, R) = \{\varphi \in \mathcal{L} : (\nexists\, \varphi_1, \cdots, \varphi_k \in \mathcal{L} \setminus \varphi)(\varphi_1 R \varphi, \cdots, \varphi_k R \varphi)\}.$$

Ainsi, les méthodes à base de préférences utilisateurs extraient uniquement les motifs les plus intéressants, tels que les *top-k patterns* [Han *et al.*, 2002] et *Skypatterns* (ou *Skyline pattern*) [Soulet *et al.*, 2011]. L'avantage majeur de ces méthodes est qu'elles ne nécessitent pas de fixer des seuils d'intérêt tels qu'un seuil de support minimal. Cependant, les meilleurs motifs pour ces préférences sont parfois trop évidents pour l'utilisateur, on sait que pour la fréquence, l'ensemble vide est toujours au top car son support est toujours égal à 1. Pour surmonter cet obstacle, il faut alors combiner plusieurs contraintes, par exemple en utilisant des contraintes de tailles minimales. Une des meilleures solutions à ce problème est l'application des algorithmes *Skyline patterns*, qui en plus de ne pas considérer une contrainte de support lors de la découverte des motifs, permettent d'intégrer plusieurs mesures d'intérêt. Etant donné n mesures d'intérêt $m_1, \cdots, m_n$, les motifs *Skyline* sont les motifs les plus préférés suivant la relation $\varphi R_{m_1, \cdots, m_n} \varphi' \Leftrightarrow \varphi R_{m_1} \varphi' \wedge \cdots \wedge \varphi R_{m_n} \varphi'$. Par

exemple, un utilisateur peut préférer les motifs fréquents avec une grande aire. Dans ce cas, un motif φ domine un motif φ' noté par $\varphi R \varphi'$ si on a $freq(\varphi, \mathcal{D}) \geq freq(\varphi', \mathcal{D})$ et $area(\varphi, \mathcal{D}) \geq area(\varphi', \mathcal{D})$.

Exemple 8. *La figure 1.5 est une représentation des meilleurs motifs de la base de données $\mathcal{T}$ du tableau 1.1 suivant différentes préférences utilisateurs, combinant une ou plusieurs mesures d'intérêt.*

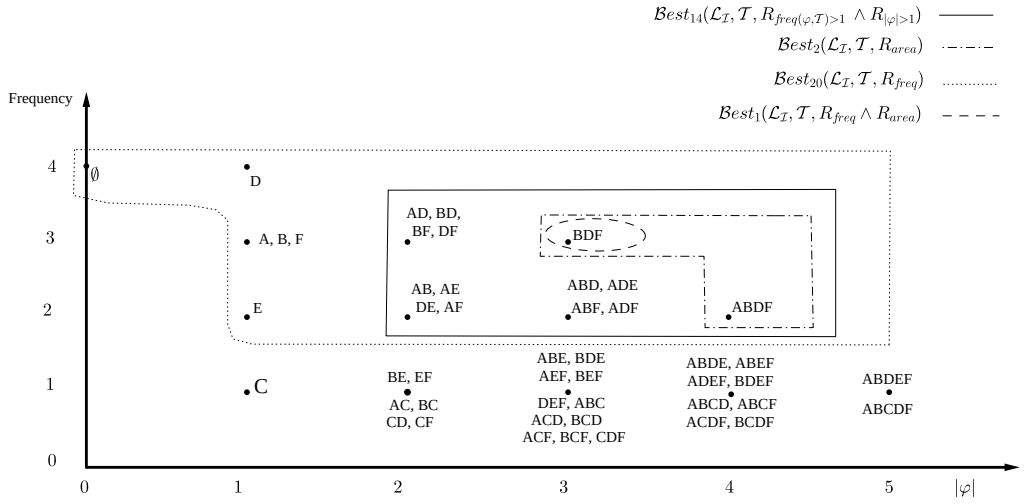


FIGURE 1.5 – Représentation des meilleurs motifs suivant les préférences utilisateurs

Dans la zone délimitée avec une ligne en pointillé, nous avons les top-20 meilleurs motifs suivant leurs fréquences dans la base de données $\mathcal{T}$, $\mathcal{B}est_{20}(\mathcal{L}, \mathcal{T}, R_{freq})$. Parmi ces top-20, nous avons dans la zone délimitée avec une ligne solide, les 14 motifs qui ont une fréquence supérieure à 1 et une taille plus grande que 1, $\mathcal{B}est_{14}(\mathcal{L}_\mathcal{I}, \mathcal{T}, R_{freq(\varphi,\mathcal{T})} \wedge R_{|\varphi|>1})$. Les deux meilleurs motifs ayant une grande aire noté $\mathcal{B}est_2(\mathcal{L}_\mathcal{I}, \mathcal{T}, R_{area})$ sont dans la zone délimitée avec une ligne en points et traits discontinus $\{BDF, ABDF\}$. Enfin, le motif ayant la plus grande aire parmi les top-20 $\mathcal{B}est_1(\mathcal{L}_\mathcal{I}, \mathcal{T}, R_{freq,area})$ est le motif BDF, $area(BDF, \mathcal{D}) = 3 \times 3 = 9$.

1.3.3 Synthèse des limites des méthodes de fouille de motifs intéressants

La figure 1.6 montre les points forts et les points faibles des méthodes de fouille de motifs intéressants suivant quatre axes. Parmi ces axes, nous avons la *qualité des motifs* et la *complexité des mesures d'intérêt* qui permettent d'évaluer la pertinence des motifs retournés. Disons que la qualité d'un motif est liée à la mesure d'intérêt utilisée. Plus la mesure d'intérêt est complexe en évaluation, plus les motifs retournés ont une bonne qualité [Crémilleux *et al.*, 2019]. Nous avons aussi la *faible quantité de motifs* qui permet de contrôler l'explosion de la quantité de motifs, la *rapidité* pour évaluer le temps durant lequel la méthode trouve les motifs recherchés, et enfin, la *diversité* pour évaluer la proportion de motifs distincts dans l'ensemble des motifs extraits [Giacometti et Soulet, 2018].

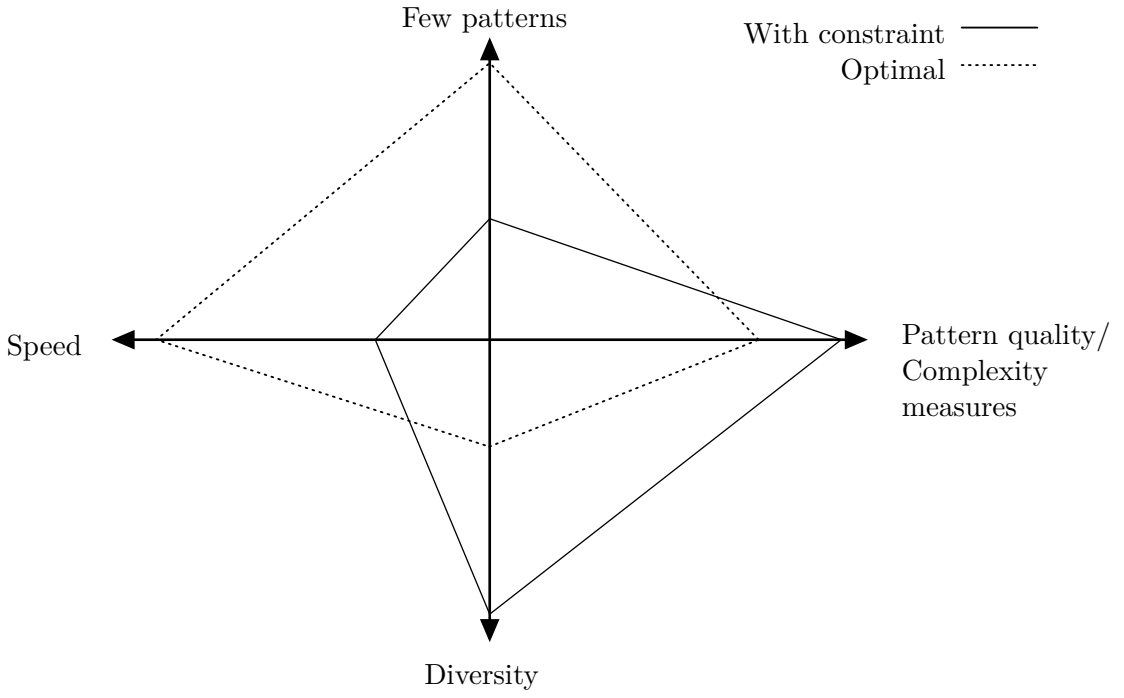


FIGURE 1.6 – Caractéristiques des méthodes de fouille de motifs

D'une manière générale, nous avons remarqué que les méthodes de fouille exhaustives avec contrainte de motifs intéressants retournent des motifs de bonne qualité qui sont en accord avec les mesures d'intérêt. En outre, elles assurent une excellente diversité de l'ensemble des motifs intéressants qu'elles retournent en sortie. Cependant, elles retournent trop de motifs quand le seuil d'intérêt minimal est trop petit. Enfin, les méthodes de fouille exhaustives avec contrainte de motifs intéressants sont trop lentes pour autoriser un processus interactif car leur rapidité dépend de la taille du langage et de la complexité des structures de motifs. En général, il est constaté que les méthodes de fouille de motifs optimaux retournent des motifs de qualité presque aussi bonne que les méthodes de fouille exhaustives de motifs intéressants. En outre, elles sont nettement meilleures pour contrôler la quantité des motifs retournés car l'utilisateur peut spécifier le nombre de motifs qu'il souhaite avoir en sortie. De ce fait, elles sont plus rapides, mais par contre, elles n'offrent généralement pas une bonne diversité entre les motifs retournés. Pour pallier ce problème, de nouvelles méthodes de fouille de données dites méthodes d'échantillonnage en sortie ont été récemment proposées.

Chapitre 2

Echantillonnage en sortie de motifs

Sommaire

Les méthodes d'échantillonnage en sortie de motifs sont probablement les méthodes les plus efficaces que le domaine de la fouille de motifs n'a jamais connues jusqu'ici. Elles trouvent leurs richesses dans des concepts purement mathématiques que sont les probabilités.

L'échantillonnage en sortie [Al Hasan et Zaki, 2009] consiste à générer un échantillon de motifs parmi les motifs qui auraient été extraits depuis le jeu de données complet. L'échantillonnage en sortie de motifs est une méthode non-exhaustive d'extraction de motifs pertinents qui assure une bonne interaction tout en offrant des garanties statistiques fortes grâce à sa nature aléatoire.

Dans ce chapitre, nous allons présenter les méthodes existantes dans le domaine de l'échantillonnage en sortie de motifs et ensuite souligner quelques cas d'utilisation des motifs échantillonnés. En particulier, nous montrerons comment les motifs échantillonnés peuvent être utilisés pour la construction de variables pour la classification [Boley *et al.*, 2011], la détection d'anomalies dans des bases de données transactionnelles [Giacometti et Soulet, 2016] et enfin, pour la découverte interactive centrée sur l'utilisateur [van Leeuwen, 2014, Dzyuba *et al.*, 2014, Giacometti et Soulet, 2017].

2.1 Formalisation du problème

Dans cette section, nous allons commencer par formuler le problème de l'échantillonnage en sortie de motifs. Ensuite, nous introduisons des critères de comparaison des méthodes d'échantillonnage en sortie.

2.1.1 Définition du problème de l'échantillonnage en sortie de motifs

L'échantillonnage en sortie de motifs est une méthode de fouille de données qui garantit une bonne interaction avec l'espace des motifs et une grande diversité entre les motifs extraits. Il vise à accéder à l'espace des motifs $\mathcal{L}$ par une procédure d'échantillonnage efficace simulant une distribution $\pi : \mathcal{L} \to [0; 1]$ qui est définie par rapport à une certaine mesure d'intérêt $m : \pi(\cdot) = m(\cdot)/Z$ où Z une constante de normalisation définie par $Z = \sum_{\varphi \in \mathcal{L}} m(\varphi, \mathcal{D})$. L'échantillonnage en sortie de k motifs du langage $\mathcal{L}$ selon une distribution proportionnelle à la mesure d'intérêt m dans la base de données $\mathcal{D}$ peut être formulé par l'opérateur suivant :

$$\mathcal{S}amp\ell e_k(\mathcal{L}, \mathcal{D}, m) = \bigcup_{i=1}^{k} \{\varphi_i \sim m(\mathcal{L}, \mathcal{D})\}.$$

où $\varphi \sim m(\mathcal{L}, \mathcal{D})$ veut dire que le motif φ est tiré avec une probabilité proportionnelle à la mesure d'intérêt m. Formellement, $\varphi \sim m(\mathcal{L}, \mathcal{D}) \Leftrightarrow \pi(\varphi) = m(\varphi, \mathcal{D})/Z$.

Par définition, si dans le langage $\mathcal{L}$, il y a au moins deux motifs de probabilités non nulles, alors l'opérateur $\mathcal{S}ample_k$, avec $k > 0$, est non déterministe. Autrement dit, deux tirages avec la même mesure d'intérêt dans la même base de données peuvent ne pas retourner les mêmes k motifs. Notons aussi que tirer $k = k_1 + k_2$ motifs suivant une mesure d'intérêt m dans le langage $\mathcal{L}$ d'une base de données $\mathcal{D}$ avec $\mathcal{S}amp\ell e_k(\mathcal{L}, \mathcal{D}, m)$, revient à tirer k_1 motifs dans $\mathcal{L}$ suivant m avec $\mathcal{S}amp\ell e_{k_1}(\mathcal{L}, \mathcal{D}, m)$, puis k_2 motifs dans $\mathcal{L}$ suivant m avec $\mathcal{S}amp\ell e_{k_2}(\mathcal{L}, \mathcal{D}, m)$. Cependant, cela ne veut pas dire que $\mathcal{S}amp\ell e_k(\mathcal{L}, \mathcal{D}, m)$ et $\mathcal{S}amp\ell e_{k_1}(\mathcal{L}, \mathcal{D}, m) \cup \mathcal{S}amp\ell e_{k_2}(\mathcal{L}, \mathcal{D}, m)$ retournent le même échantillon de motifs.

L'objectif principal des méthodes d'échantillonnage en sortie est d'avoir un échantillon de motifs représentatif de l'ensemble des motifs qui peuvent être extraits de la base de données. Si par exemple, un motif φ_1 a un intérêt deux fois plus élevé que celui d'un motif φ_2 selon le choix de l'utilisateur, alors φ_1 a deux fois plus de chance d'être dans l'échantillon que le motif φ_2. Soit $m(\cdot) = \mathit{freq}(\cdot)$ par exemple, si φ_1 et φ_2 sont deux motifs de l'espace de recherche $\mathcal{L}$ et de fréquences respectives $\mathit{freq}(\varphi_1, \mathcal{D})$ et $\mathit{freq}(\varphi_2, \mathcal{D})$ telles que $\mathit{freq}(\varphi_1, \mathcal{D}) = 2 \times \mathit{freq}(\varphi_2, \mathcal{D})$, alors la probabilité de tirage de φ_1 suivant la fréquence est deux fois plus élevée que celle de φ_2.

Une méthode naïve d'échantillonnage en sortie de motifs est d'énumérer en premier lieu l'ensemble de tous les motifs intéressants à l'aide des algorithmes d'extraction exhaustive de motifs tels que Apriori [Agrawal et Srikant, 1994], FP-growth [Han *et al.*, 2000], etc. En second lieu, on construit un échantillon de motifs à partir de l'espace des motifs extraits tel que chaque motif soit tiré avec une probabilité proportionnelle à son intérêt dans la base de données. Ainsi, nous pouvons aussi écrire que calculer $\mathcal{S}amp\ell e_k(\mathcal{L}, \mathcal{D}, m)$ revient à calculer $\mathcal{S}amp\ell e_k(\mathcal{T}h(\mathcal{L}, \mathcal{D}, \mathit{freq}(\varphi, \mathcal{D}) \geq 1), \mathcal{D}, m)$.

Exemple 9. *En considérant la base de données transactionnelles du tableau 1.1 que nous avons reportée au tableau 2.1, énumérons l'ensemble des motifs du langage ayant une fréquence strictement supérieure à 0,* $\mathcal{T}h(\mathcal{L}_\mathcal{I}, \mathcal{T}, freq(\varphi, \mathcal{T}) \geq 1)$*. Maintenant, l'objectif est de construire un échantillon de motifs de la théorie précédemment définie.*

TABLE 2.1 – Exemple d'une base de données transactionnelles $\mathcal{T}$

tid	Transaction
1	A B D E F
2	A D E
3	B D E
4	A B C D F

Nous notons que tous les motifs du langage $\mathcal{L}_\mathcal{I}$ *plus généraux qu'au moins une instance de la base de données* $\mathcal{T}$ *ont une probabilité non nulle. La constante de normalisation est* $Z = 80$*, c'est la somme des fréquences des motifs de la théorie* $\mathcal{T}h(\mathcal{L}_\mathcal{I}, \mathcal{T}, freq(\varphi, \mathcal{T}) \geq 1)$*. Dans ce cas, la probabilité de tirer le motif* D *de fréquence 4 est égale à la probabilité de tirer l'ensemble vide* $\pi(D) = \pi(\emptyset) = \frac{4}{80}$*. Considérons deux échantillons de* 3 *motifs tirés à partir de* $\mathcal{L}_\mathcal{I}$ *suivant la fréquence à deux dates* θ_1 *et* θ_2 *respectivement* $\mathcal{S}ampl\mathcal{e}_3(\mathcal{L}_\mathcal{I}, \mathcal{T}, freq)_{\theta_1} = \{D, AD, BF\}$ *et* $\mathcal{S}ampl\mathcal{e}_3(\mathcal{L}_\mathcal{I}, \mathcal{T}, freq)_{\theta_2} = \{D, AD, BDF\}$*. On peut noter que ces deux tirages ont la même probabilité de se réaliser* $\pi(D) \times \pi(AD) \times \pi(BF) = \pi(D) \times \pi(AD) \times \pi(BDF) = \frac{4}{80} \times \frac{3}{80} \times \frac{2}{80}$*.*

Malheureusement, le coût d'une telle méthode naïve est potentiellement très élevé, surtout avec les grands jeux de données, à cause du coût de l'extraction exhaustive de tous les motifs intéressants. En effet, comme nous l'avons souligné à la section 1.3.3, la complexité des algorithmes d'extraction exhaustive de motifs intéressants est exponentielle sur les gros jeux de données ayant une structure complexe (séquences et graphes).

Dès lors, l'idée générale des méthodes développées dans le domaine de l'échantillonnage en sortie est de tirer les motifs directement à partir de la base de données comme le montre la figure 2.1 sur les données ensemblistes.

D'après la figure 2.1, nous avons dans l'échantillon, le motif D deux fois plus que le motif AB, car dans la base de données, la fréquence de D est égale à deux fois la fréquence de AB. Notons que même les motifs ayant des fréquences très faibles sont probables d'être tirés. Néanmoins, leurs probabilités d'être dans l'échantillon sont aussi très faibles. Par exemple, le motif ABC a une fréquence égale à 1, donc très faible, cependant sa probabilité d'être dans l'échantillon n'est pas nulle car $\pi(\mathcal{S}ampl\mathcal{e}_1(\mathcal{L}_\mathcal{I}, \mathcal{T}, freq) = ABC) = \frac{1}{80}$.

Pour tirer directement un échantillon de motifs, sans avoir à réaliser préalablement une extraction exhaustive, différentes méthodes ont été proposées. Dans la littérature, les méthodes d'échantillonnage en sortie aussi riches que variées, se distinguent suivant plusieurs critères que nous présentons dans la section qui suit.

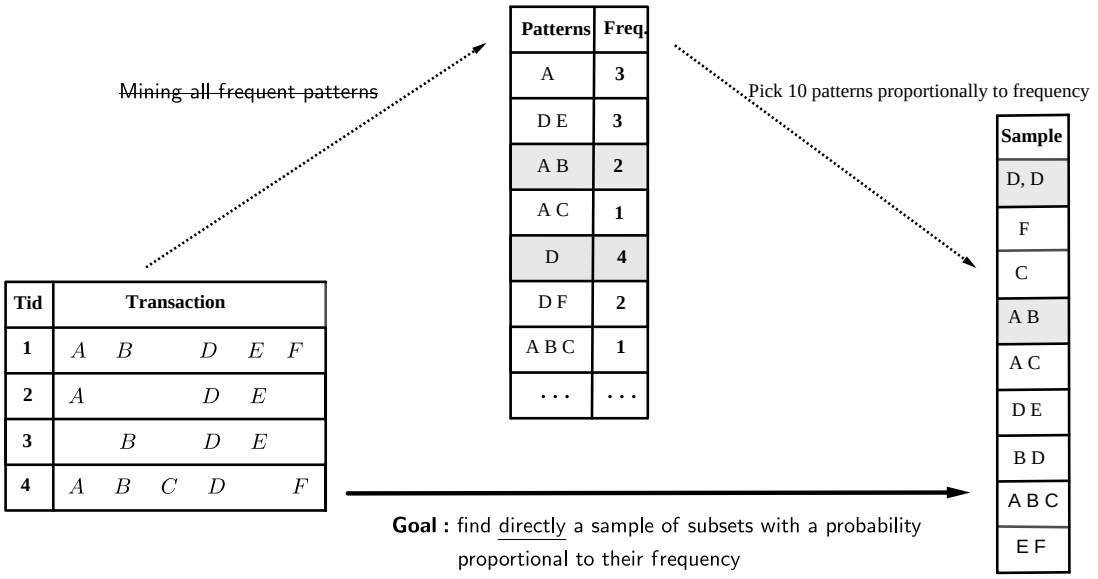


FIGURE 2.1 – Le but de l'échantillonnage en sortie de motifs

2.1.2 Critères de comparaison des méthodes d'échantillonnage en sortie

Pour comparer les différentes classes de méthodes déjà existantes, nous utiliserons différents critères de comparaison. Ces critères sont organisés en deux grandes classes, à savoir les critères en entrée qui portent sur les paramètres en entrée des méthodes (par exemple, les langages qu'elles peuvent prendre en compte en entrée) et les critères en sortie qui portent sur le calcul de la sortie (son exactitude et sa complexité). Plus précisément, nous étudierons chaque classe de méthodes en utilisant les critères suivants :

Les critères en entrée Parmi les critères d'entrée sur lesquels se porte notre évaluation, nous avons :

— **Langages :** Pour chaque méthode présentée, nous préciserons les langages de motifs pour lesquels elle peut s'appliquer, en considérant les différents langages présentés à la section 1.1 du chapitre précédent.

— **Mesures d'intérêt :** Pour chaque méthode, nous préciserons les mesures ou classes de mesures d'intérêt qu'elle peut prendre en compte. On prendra ici en compte les mesures d'intérêt déjà présentées à la section 1.1.3 (support, aire, etc), mais on sera également amené à présenter des mesures plus spécifiques.

— **Contraintes :** Enfin, pour chaque méthode, nous préciserons les contraintes qui peuvent être prises en compte. En plus des contraintes présentées à la section 1.1.3, nous présenterons d'autres contraintes spécifiques aux méthodes.

Les critères en sortie Ces critères permettent d'évaluer la correction et la rapidité de l'algorithme d'échantillonnage en sortie de motifs. Nous avons les critères suivantes :

— **Exactitude du tirage :** On note que certaines méthodes tirent un motif avec une probabilité exactement égale à son intérêt dans la base de données divisé par la somme totale des intérêts des motifs du langage, alors que d'autres méthodes garantissent l'exactitude du tirage avec un certain taux d'erreur. On a aussi les méthodes pour lesquelles la probabilité de tirage d'un motif converge vers la probabilité exacte au bout d'un certain nombre d'itérations, alors que d'autres méthodes n'offrent aucune garantie sur l'exactitude du tirage.

— **Complexité de la méthode :** Ce critère évalue la rapidité de la méthode d'échantillonnage. Selon les paramètres d'entrée, le langage de motifs considéré et les caractéristiques de la base de données en entrée, il donne une idée du temps de prétraitement (s'il existe) et du temps de tirage d'un motif ou d'un échantillon de motifs.

2.2 Les classes de méthodes d'échantillonnage en sortie

Dans la littérature, trois principales classes de méthodes ont été proposées pour l'échantillonnage en sortie de motifs. La première classe de méthodes est basée sur les algorithmes à marche aléatoire, et a initialement été utilisée pour la découverte de sous-graphes dans des bases de graphes [Al Hasan et Zaki, 2009]. Dans [Boley *et al.*, 2011], les auteurs ont introduit une deuxième classe de méthodes basée sur une procédure aléatoire en deux étapes. Initialement, cette classe de méthodes a été utilisée pour la découverte d'itemsets intéressants dans des bases de données transactionnelles. Enfin, plus récemment, des travaux [Dzyuba *et al.*, 2017] ont introduit une troisième classe de méthodes basée sur le formalisme SAT (SATisfiable) et l'utilisation de solveurs (pour la découverte d'itemsets dans des bases de données transactionnelles). Malgré leur diversité en terme d'exactitude du tirage, contraintes utilisées, structures de données sur lesquelles elles sont appliquées, etc, toutes ces méthodes visent à résoudre le même problème : échantillonner un ensemble de motifs directement à partir d'une base de données.

Cette section décrit les trois principales classes de méthodes d'échantillonnage en sortie que nous avons précédemment énumérées.

2.2.1 Échantillonnage par marche aléatoire

La première classe de méthodes utilise des marches aléatoires dans l'espace de recherche pour échantillonner des motifs intéressants. Dans cette classe, la plupart des méthodes [Al Hasan et Zaki, 2009, Bhuiyan *et al.*, 2012, Boley *et al.*, 2010, Li et Zaki, 2012] sont basées sur les méthodes de Chaîne de Markov par Monte Carlo (MCMC). Une chaîne d'états finis $v_0, v_1, \cdots, v_n$ est dite chaîne de Markov si la probabilité d'atteindre un état v_i ne dépend que de l'état v_{i-1} et pas des états précédents. Ainsi les algorithmes de fouille de motifs en bénéficient surtout quand la mémoire physique disponible n'est pas si importante. L'idée est de construire une chaîne de Markov simulant une distribution d'une loi de probabilité. [Al Hasan et Zaki, 2009] ont proposé la première méthode de cette classe pour échantillonner des sous-graphes fréquents dans une grande base de données de graphes, et puis [Bhuiyan *et al.*, 2012] ont proposé une nouvelle méthode pour la découverte interactive.

La méthode de [Bhuiyan *et al.*, 2012] utilise comme mesure d'intérêt la fréquence et la méthode de [Al Hasan et Zaki, 2009], en plus de la fréquence, elle peut faire un tirage uniforme. Mais nous avons noté que théoriquement elles peuvent prendre en compte d'autres mesures d'intérêt. Une année plus tard, [Boley *et al.*, 2010] ont développé un algorithme basé sur les MCMC pour échantillonner des motifs ensemblistes en prenant en compte toute mesure d'intérêt strictement positive.

Dans la suite, nous allons présenter un algorithme générique d'échantillonnage en sortie de motifs à l'aide de l'approche à base de MCMC. Ensuite, nous évaluons la méthode en fonction des critères présentés à la section 2.1.

Approche : L'algorithme 2 est une généralisation de celui de [Al Hasan et Zaki, 2009] basé directement sur l'algorithme de Metropolis-Hastings. Il prend en entrée une base de données $\mathcal{D}$, une mesure d'intérêt m, une contrainte q et un nombre d'itérations j puis retourne un motif tiré avec une probabilité proportionnelle à son intérêt et vérifiant la contrainte dans la base de données $\mathcal{D}$. Nous rappelons que l'algorithme de [Al Hasan et Zaki, 2009] est générique dans le sens où il peut prendre en compte plusieurs mesures d'intérêt telles que la fréquence, l'aire, les mesures discriminatives, etc., et aussi il peut être appliqué sur d'autres structures de données telles que les itemsets et les séquences. Initialement, on tire aléatoirement un motif φ respectant la contrainte q (ligne 3). Pour faire une marche aléatoire sur le langage des motifs respectant la contrainte q, à chaque itération, il faut calculer pour chaque voisin φ' du motif φ respectant la contrainte q son degré $d_{\varphi'}$, qui n'est rien d'autre que la somme de la cardinalité de ses spécialisations et généralisations immédiates respectant la contrainte q (lignes 4 et 8), ainsi que sa mesure d'intérêt. Ensuite, on tire aléatoirement un voisin φ' de φ (ligne 7). Enfin, on accepte φ' si la probabilité d'acceptation $\min\{\frac{m(\varphi',\mathcal{D})\times d_{\varphi}}{m(\varphi,\mathcal{D})\times d_{\varphi'}}, 1\}$ est supérieure à la valeur tirée uniformément entre 0 et 1 (lignes 9 et 10) puis on décrémente le nombre d'itérations j (ligne 11) le cas échéant. Le processus est ainsi répété jusqu'à ce que le nombre d'itérations j soit atteint puis on retourne le dernier motif visité (ligne 13).

Exemple 10. *Dans cet exemple, nous allons expliquer l'échantillonnage en sortie de motifs par marche aléatoire dans le cadre de [Al Hasan et Zaki, 2009]. A l'aide de l'algorithme* Metropolis-Hastings*, une méthode MCMC, [Al Hasan et Zaki, 2009] parviennent à générer des échantillons de motifs de graphes où chaque motif est tiré avec une probabilité approximativement proportionnelle à sa fréquence dans la base. Dans le cas des graphes, l'algorithme 2 s'applique sur un graphe d'ordre partiel (POG : Partial Order Graph) défini suivant la relation d'inclusion $\subseteq$ définie au chapitre 1.*

Graphe d'ordre partiel (POG) : *L'ensemble de tous les sous-graphes fréquents muni de la relation d'inclusion entre graphes constitue un ensemble partiellement ordonné appelé Graphe d'Ordre Partiel (POG). C'est le langage des motifs (ordonné partiellement par sa relation de spécialisation). Chaque noeud d'un POG correspond à un motif de graphe fréquent distinct. Plus précisément, chaque noeud d'un POG correspond à une classe de sous-graphes isomorphes, sachant que lui est associé un représentant canonique de la classe (définie dans [Xifeng Yan et Jiawei Han, 2002] par le code DFS). Chaque arc dans le POG représente qu'un sous-graphe est une spécialisation directe (ou généralisation directe) d'un autre sous-graphe. L'élément inférieur dans le POG est le graphe vide (qui est fréquent*

Algorithm 2 Algorithme d'échantillonnage en sortie de motifs par MCMC

Input : Une base de données $\mathcal{D}$, une mesure d'intérêt m, une contrainte q et le nombre d'itérations j
Output : un motif intéressant $\varphi \in \mathcal{L}$
$\varphi \sim \mathcal{T}h(\mathcal{L}, \mathcal{D}, q)$ ▷ On tire au hasard un motif φ respectant la contrainte q
$d_\varphi \leftarrow |\{\varphi' \in \mathcal{T}h(\mathcal{L}, \mathcal{D}, q) : \varphi' \dot{\prec} \varphi \vee \varphi \dot{\prec} \varphi'\}|$ ▷ Calcul du degré de φ
repeat
    $\mathcal{N}_q(\varphi) = \{\varphi' \in \mathcal{T}h(\mathcal{L}, \mathcal{D}, q) : \varphi' \dot{\prec} \varphi \vee \varphi \dot{\prec} \varphi'\}$ ▷ L'ensemble des voisins de φ suivant la contrainte q
    $\varphi' \sim \mathcal{N}_q(\varphi)$ ▷ Tirage au hasard d'un voisin de φ sous la contrainte q
    $d_{\varphi'} \leftarrow |\{\varphi'' \in \mathcal{T}h(\mathcal{L}, \mathcal{D}, q) : \varphi'' \dot{\prec} \varphi' \vee \varphi' \dot{\prec} \varphi''\}|$ ▷ Calcul du degré de φ'
    if $unif(0,1) \leq \min\{\frac{m(\varphi', \mathcal{D}) \times d_\varphi}{m(\varphi, \mathcal{D}) \times d_{\varphi'}}, 1\}$ **then** ▷ On vérifie si φ' est accepté
        $\varphi \leftarrow \varphi'$ ▷ On passe au voisin φ' s'il est accepté
        $j \leftarrow j - 1$ ▷ On décrémente le nombre d'itérations
until $j = 0$
return φ ▷ Le motif φ est retourné en sortie

par défaut). Les algorithmes d'énumération de tous les sous-graphes fréquents traversent généralement le POG en profondeur d'abord ou en largeur en premier, en commençant par le bas. De ce fait, un graphe peut être construit de différentes manières (selon l'ordre dans lequel les arcs sont ajoutés). A partir du sous-graphe vide, il existe plusieurs chemins conduisant à un noeud dans le POG. Ainsi, différents noeuds du POG ont des degrés différents. La figure 2.2 montre le POG de la base de données de graphes de la figure 1.3-b.

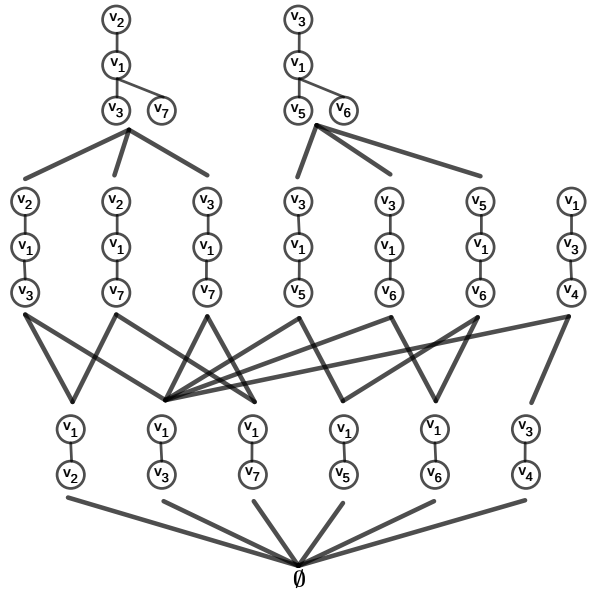


FIGURE 2.2 – Graphe d'ordre partiel des sous-graphes fréquents de $\mathcal{T}h(\mathcal{L}, \mathcal{D}, q)$

Dans notre exemple, on sait que tous les motifs non vides dans le POG ont une fréquence égale à 1 sauf le motif $v_1 - v_3$ qui a une fréquence de 3 dans $\mathcal{G}$. Considérons la fréquence comme mesure d'intérêt. Donc tous les motifs du langage $\mathcal{L}_\mathcal{G}$ sont fréquents. Si à la ligne 3, on a $\varphi = v_1 - v_3 - v_4$, qui est le motif le plus à droite de la figure 2.2, alors φ a 2 voisins plus génériques que lui et 0 voisin plus spécifique. Si à la ligne 7 on choisit le motif $\varphi' = v_1 - v_3$ ayant pour degré 6, alors sa probabilité d'acceptation à la ligne 9 est égale à $\min(\frac{3\times2}{1\times6}, 1) = 1$. Donc on accepte φ' et on a $\varphi = v_1 - v_3$.

Nous allons maintenant analyser l'approche proposée suivant les critères en entrée et en sortie introduits à la section 2.1.2.

Les critères en entrée

— **Langages :** L'algorithme d'échantillonnage en sortie par MCMC a été initialement appliqué sur des bases de données de graphes [Al Hasan et Zaki, 2009]. Nous notons qu'il peut être appliqué sur d'autres types de langages tels que les itemsets [Boley *et al.*, 2010] et les séquences. [Moens et Goethals, 2013, Boley *et al.*, 2010] appliquent la même approche sur des données ensemblistes et treillis de concepts (ce qui est original car les motifs sont des concepts et pas seulement des itemsets).

— **Mesures d'intérêt :** L'algorithme de [Al Hasan et Zaki, 2009] peut prendre en compte toute mesure d'intérêt strictement positive. D'une façon intéressante, ils l'ont utilisé à la fois pour l'échantillonnage uniforme de motifs fréquents, pour l'échantillonnage en sortie de motifs proportionnellement à leurs fréquences et enfin pour l'échantillonnage en sortie de motifs discriminants. [Moens et Goethals, 2013] utilisent la fréquence comme mesure d'intérêt et [Boley *et al.*, 2010] peuvent prendre en compte toute mesure d'intérêt strictement positive.

— **Contraintes :** Tout type de contraintes n'ayant pas un coût très élevé peut être utilisé avec l'algorithme de [Al Hasan et Zaki, 2009]. Nous notons aussi que les méthodes [Moens et Goethals, 2013, Boley *et al.*, 2010] se basent sur les contraintes pour restreindre l'espace de recherche et renvoient uniquement les ensembles d'itemsets maximaux et fermés, respectivement.

Les critères en sortie

— **Exactitude du tirage :** La méthode de tirage de l'algorithme 2 n'est pas exacte. Cependant, on sait que quand le nombre d'itérations j tend vers l'infini, alors la probabilité de tirage tend vers la probabilité de tirage exact. Ainsi en pratique, dans [Al Hasan et Zaki, 2009], les auteurs suggèrent de laisser tourner le processus suffisamment longtemps pour bien approcher un tirage exact. Face à un problème de convergence trop lente, des méthodes heuristiques [Moens et Goethals, 2013, Bendimerad *et al.*, 2016] ont été proposées pour accélérer la convergence et favoriser les motifs les plus pertinents (selon une mesure d'intérêt donnée). Cependant, comme ils utilisent des méthodes heuristiques pour simuler une marche aléatoire et explorer le langage de motifs, ils n'offrent aucune garantie sur la qualité de l'échantillon généré.

— **Complexité de la méthode :** Nous avons noté que la complexité du calcul des supports et des degrés avant l'acceptation du prochain état n'est pas négligeable surtout avec les très grandes bases de données. Pour le calcul des supports, [Al Hasan et Zaki, 2009] utilisent la méthode de [Ullmann, 1976] pour le calcul d'isomophisme de sous-graphes. Cette dernière a une complexité en $O(|V|^3)$, avec V l'ensemble des sommets de la base de données de graphes. Or, dans [Al Hasan et Zaki, 2009], on calcule les supports de tous les voisins d'un noeud. Notons par $d_{\max}$ le degré maximal des noeuds. De ce fait, pour j itérations, l'algorithme 2 s'exécute en $O(j \times |V|^3 \times d_{\max})$. Cette complexité montre que les performances des méthodes MCMC dépendent fortement de la taille du langage de motifs. En outre, la convergence du processus peut être très lente en nécessitant un grand nombre d'itérations. Ce qui fait qu'en pratique, il a été montré que les méthodes heuristiques [Moens et Goethals, 2013, Bendimerad *et al.*, 2016] sont plus efficaces en terme de temps de calcul.

2.2.2 Échantillonnage par le formalisme SAT

La seconde classe de méthodes d'échantillonnage en sortie est basée sur le formalisme logique SAT, et a été mise en oeuvre pour l'échantillonnage en sortie de motifs ensemblistes [Dzyuba *et al.*, 2017]. Dans cet article, les auteurs ont à la fois présenté un algorithme générique appelé GFlexic et un algorithme spécialisé appelé EFlexic. L'algorithme GFlexic est un algorithme générique dans le sens où il peut considérer un vaste ensemble de contraintes pour définir ce que sont les motifs intéressants, alors que EFlexic, qui utilise une version étendue de Eclat [Zaki *et al.*, 1997a], est limité à l'usage de la fréquence comme mesure d'intérêt.

Approche : Pour cette classe de méthodes, l'idée de base est d'utiliser la programmation par contraintes et des solveurs logiques pour effectuer de la fouille de motifs. Dans le cas des motifs ensemblistes, il a été initialement montré dans [Khiari *et al.*, 2010, Guns *et al.*, 2011] comment transformer une large classe de contraintes q en des formules logiques $\mathcal{F}$ telles que toute valuation θ permettant de satisfaire $\mathcal{F}$ corresponde à un motif intéressant selon q. Plus précisément, si on note $\mathcal{F}(q)$ la formule logique associée à la contrainte q et $\varphi(\theta)$ le motif associé à une valuation θ de la formule $\mathcal{F}(q)$, on a : $\mathcal{T}h(\mathcal{L}, \mathcal{T}, q) = \{\varphi(\theta) : \theta \in SolveSAT(\mathcal{F}(q))\}$ où $SolveSAT$ est un solveur logique permettant de trouver toutes les valuations permettant de satisfaire $\mathcal{F}$. Une telle approche permet une fouille exhaustive de l'ensemble des motifs vérifiant la contrainte q.

Pour effectuer un échantillonnage en sortie de motifs intéressants, [Dzyuba *et al.*, 2017] ont montré comment utiliser cette approche et des résultats récents obtenus en Intelligence Artificielle [Chakraborty *et al.*, 2014]. Ces travaux montrent comment échantillonner un ensemble de valuations satisfaisant une formule logique $\mathcal{F}$, i.e. les valuations appartenant à $SolveSAT(\mathcal{F})$, selon une distribution de probabilité $\pi(\theta) = \frac{w(\theta)}{\sum_{\theta \in SolveSAT(\mathcal{F})} w(\theta)}$ où $w(\theta)$ est un poids entre 0 et 1 associé à toute valuation possible θ d'une formule $\mathcal{F}$. Dans son application à la fouille de motifs, si θ est une valuation permettant de satisfaire une formule $\mathcal{F}(q)$ et $\varphi(\theta)$ le motif associé à la valuation θ, alors $w(\theta)$ peut être la valeur d'une mesure

d'intérêt $m(\varphi(\theta), \mathcal{T})$, par exemple la fréquence de $\varphi(\theta)$ dans $\mathcal{T}$.

Nous ne détaillerons pas dans cette thèse les algorithmes permettant d'effectuer un échantillonnage de l'espace des valuations satisfaisant une formule logique, car ces algorithmes sont particulièrement complexes, mais l'idée de base est de partitionner l'espace des valuations satisfaisant une formule $\mathcal{F}(q)$ en ajoutant récursivement à q des contraintes XOR. Ces contraintes effectuent un OU exclusif entre un sous-ensemble (tiré aléatoirement) des variables binaires représentant les motifs ensemblistes recherchés. En ajoutant une contrainte XOR à q, on découpe l'espace des valuations satisfaisant $\mathcal{F}(q)$ en deux sous-ensembles ou cellules, i.e. les valuations qui satisfont ou pas la contraintes XOR ajoutée. Plus généralement, en ajoutant k contrainte XOR à q, on découpe l'espace des valuations satisfaisant $\mathcal{F}(q)$ en 2^k sous-ensembles ou cellules. Ce qui est montré dans [Chakraborty *et al.*, 2014], et est appliqué dans [Dzyuba *et al.*, 2017] à l'échantillonnage en sortie de motifs, c'est comment découper aléatoirement et intelligemment l'espace des valuations satisfaisant $\mathcal{F}(q)$ pour finalement arriver à faire un tirage aléatoire des valuations θ satisfaisant $\mathcal{F}(q)$ selon un poids $w(\theta)$ défini par une mesure d'intérêt quelconque.

Nous allons maintenant analyser la classe des méthodes fondée sur SAT en nous basant sur les critères en entrée et en sortie introduits précédemment :

Les critères en entrée

— **Langages :** [Dzyuba *et al.*, 2017] ont montré comment appliquer l'approche proposée au cas des données ensemblistes. Dans l'absolu, la mise en oeuvre de la même approche à des données plus complexes telles que les séquences et les graphes est envisageable. Mais elle est probablement complexe, et nécessiterait d'adapter les solveurs logiques existants pour obtenir des implémentations efficaces.

— **Mesures d'intérêt :** L'algorithme GFlexic de [Dzyuba *et al.*, 2017] peut prendre en compte toute mesure d'intérêt strictement positive, alors que l'algorithme EFlexic ne peut utiliser que la fréquence. Par contre, comme EFlexic utilise un solveur logique basé sur Eclat développé spécifiquement, il est beaucoup plus efficace.

— **Contraintes :** Toute contrainte peut être utilisée avec l'algorithme GFlexic de [Dzyuba *et al.*, 2017]. Ainsi, des contraintes peuvent être utilisées pour contrôler la norme des motifs échantillonnés et éviter le phénomène de la longue traîne que nous présenterons au chapitre 3. Par contre, l'algorithme EFlexic peut uniquement utiliser une contrainte de fréquence minimale.

Les critères en sortie

— **Exactitude du tirage :** Dans [Dzyuba *et al.*, 2017], la probabilité de tirage d'un motif n'est pas exacte, mais approximativement exacte. Plus précisément, leur méthode d'échantillonnage en sortie de motifs garantit l'exactitude du tirage avec un taux d'erreur $\epsilon(\kappa)$ fonction d'un paramètre $\kappa \in]0, 1[$. Etant donné une mesure d'intérêt m, elle tire un motif φ du langage $\mathcal{L}$ suivant une distribution π telle que :

$$\frac{m(\varphi, \mathcal{D})}{Z} \times \frac{1}{(1+\epsilon(\kappa))} \leq \pi(\mathcal{S}amp\ell e_1(\mathcal{L}, \mathcal{D}, m) = \varphi) \leq \frac{m(\varphi, \mathcal{D})}{Z} \times (1+\epsilon(\kappa))$$

où $\pi(\mathcal{S}amp\ell e_1(\mathcal{L}, \mathcal{D}, m) = \varphi)$ est la probabilité que φ soit le motif échantillonné, Z est une constante de normalisation et $\epsilon(\kappa) = (1+\kappa)(2.36+0.51/(1+\kappa)^2) - 1$. Ainsi, il est important de noter que le taux d'erreur $\epsilon(\kappa)$ est nécessairement supérieur à 1.87. La garantie apportée théoriquement à l'exactitude du tirage n'est donc pas très bonne. Toutefois, il est noté expérimentalement que la qualité du tirage est en général significativement supérieure à ce qui est prédit par la théorie.

— **Complexité de la méthode :** Sur un plan théorique, la complexité de la méthode peut être évaluée en nombre d'appels à un solveur logique. Grace aux travaux de [Chakraborty *et al.*, 2014], il est montré que ce nombre d'appels est notamment linéaire en fonction de $1/\epsilon(\kappa)$ et $\hat{r}$ où $\hat{r}$ est une borne supérieure du ratio entre les mesures d'intérêt maximale m_{max} et minimale m_{min} des motifs satisfaisant la contrainte q posée, i.e. $m_{max} = max_{\varphi \in \mathcal{T}h(\mathcal{L},\mathcal{T},q)} m(\varphi, \mathcal{T})$ et $m_{min} = min_{\varphi \in \mathcal{T}h(\mathcal{L},\mathcal{T},q)} m(\varphi, \mathcal{T})$. Sur un plan pratique, s'il a été montré que EFlexic pouvait être utilisé sur de grosses bases de données transactionnelles (plus de 300,000 transactions), GFlexic n'a été testé que sur de petites bases de données transactionnelles (au plus 4,000 transactions et 300 items), ce qui s'explique par la très grande généricité de la méthdode.

2.2.3 Échantillonnage à plusieurs étapes

Les méthodes d'échantillonnage à plusieurs étapes forment la dernière classe du domaine de l'échantillonnage en sortie. Dans cette section, nous allons présenter, en premier lieu, un algorithme générique basé sur un tirage en deux étapes tel que : (1) la première étape permet de tirer une instance d'une base de données proportionnellement à la somme des intérêts des motifs au sein d'elle, (2) la seconde étape est destinée à tirer un motif de l'instance précédemment tirée proportionnellement à son intérêt. En second lieu, nous présentons l'algorithme d'échantillonnage en sortie de motifs en deux étapes de [Boley *et al.*, 2011], une instanciation de l'algorithme générique sur les données ensemblistes.

Approche : Les méthodes d'échantillonnage en sortie de motifs à plusieurs étapes ne s'appliquent pas à toutes les mesures d'intérêt. Pour qu'une mesure d'intérêt puisse être prise en compte par une méthode à plusieurs étapes, il faut qu'elle s'écrive sous la forme d'une somme d'utilités de motifs. Autrement dit, pour une mesure d'intérêt donnée, une fonction d'utilité associée doit permettre d'évaluer l'utilité de chaque motif pour chaque instance.

Définition 14 (Fonctions d'utilité et mesure d'intérêt)**.** *Une fonction d'utilité notée par u et définie de $\mathcal{L} \times \mathcal{D}$ dans $\mathbb{R}^+$, permet d'assigner un poids à chaque motif d'une instance de la base de données (de manière à ce que les motifs les plus préférés par l'utilisateur reçoivent les poids les plus élevés).*

Soient u la fonction d'utilité associée à la mesure d'intérêt m, $\mathcal{D}$ une base de données et $\mathcal{L}$ son langage de motifs. Pour tout motif φ de $\mathcal{L}$, nous avons :

$$m(\varphi, \mathcal{D}) = \sum_{\gamma \in \mathcal{D} \wedge \varphi \preceq \gamma} u(\varphi, \gamma).$$

Si on considère que la mesure d'intérêt choisie est la fréquence, alors pour toute instance γ de la base de données $\mathcal{D}$, on a : $u_{freq}(\varphi, \gamma) = 1$ si $\varphi \preceq \gamma$ et 0 sinon. Dans ce cas, la mesure d'intérêt d'un motif $\varphi \in \mathcal{L}$ dans la base de données $\mathcal{D}$ est égale à la fréquence du motif dans $\mathcal{D}$ et s'écrit formellement comme suit : $freq(\varphi, \mathcal{D}) = \sum_{\gamma \in \mathcal{D} \wedge \varphi \preceq \gamma} u_{freq}(\varphi, \gamma)$. Si maintenant on considère que l'aire est la mesure d'intérêt choisie par l'utilisateur, alors pour toute instance γ de la base de données $\mathcal{D}$, on a : $u_{area}(\varphi, \gamma) = ||\varphi||$ si $\varphi \preceq \gamma$ et 0 sinon. Et dans ce cas, nous avons : $area(\varphi, \mathcal{D}) = \sum_{\gamma \in \mathcal{D} \wedge \varphi \preceq \gamma} u_{area}(\varphi, \gamma) = ||\varphi|| \times freq(\varphi, \mathcal{D})$.

Exemple 11. *Considérons la base de données transactionnelles de la figure 2.3. On sait que le motif AD apparait dans les transactions t_1, t_2 et t_4. Alors d'une part nous avons $u_{freq}(AD, t_1) = 1$, $u_{freq}(AD, t_2) = 1$, $u_{freq}(AD, t_3) = 0$ et $u_{freq}(AD, t_4) = 1$. Donc, $freq(AD, \mathcal{T}) = 3$ et $area(AD, \mathcal{T}) = 2 \times 3 = 6$.*

L'algorithme 3 reçoit en entrée une base de données $\mathcal{D}$ et une mesure d'intérêt choisie par l'utilisateur, et retourne un motif φ tiré suivant une probabilité proportionnelle à son intérêt dans la base de données. Pour ce faire, on commence par une phase de prétraitement en pondérant chaque instance γ de $\mathcal{D}$ avec la somme des utilités des motifs qu'elle contient (ligne 3). Ensuite, à la première étape de la phase de tirage (ligne 4), on tire une instance γ suivant une probabilité proportionnelle à son poids. Enfin, à la deuxième étape de la phase de tirage, un motif φ tiré proportionnellement à son utilité parmi l'ensemble des généralisations possibles de γ est retourné (lignes 5 et 6).

Algorithm 3 Echantillonnage en deux étapes

1: **Input :** Une base de données $\mathcal{D}$ et une mesure d'intérêt m telle qu'il existe une fonction d'utilité u vérifiant $m(\varphi, \mathcal{D}) = \sum_{\gamma \in \mathcal{D} \wedge \varphi \preceq \gamma} u(\varphi, \mathcal{D})$ pour tout $\varphi \in \mathcal{L}$
2: **Output :** Un motif tiré aléatoirement $\varphi \sim m(\mathcal{L}, \mathcal{D}) : \pi(\varphi) = \frac{m(\varphi, \mathcal{D})}{Z}$
Phase de prétraitement
3: Soient les poids ω_m définis par $\omega_m(\gamma) = \sum_{\varphi \preceq \gamma} u(\varphi, \gamma)$ pour toute $\gamma \in \mathcal{D}$
Phase de tirage
Etape 1 : Tirage d'une instance proportionnellement à son poids
4: $\gamma \sim \omega_m(\mathcal{D})$ ▷ Tirage d'une instance γ proportionnellement à $\omega_m(\gamma)$
Etape 2 : Tirage d'un motif proportionnellement à son intérêt dans γ
5: $\varphi \sim u(\{\varphi : \varphi \preceq \gamma\})$ ▷ Tirage de φ proportionnellement à $u(\varphi, \gamma)$
6: **return** φ

Défis : L'algorithme 3, tel que nous l'avons présenté, peut prendre tout type de mesure d'intérêt définie comme une somme d'utilités et tout type de langage. Cependant, la phase de pondération et la phase de tirage d'un motif restent très complexes pour certains types de données ainsi que certaines mesures d'intérêt. Ainsi, on a besoin de relever deux défis pour l'appliquer efficacement sur une base de données. Le premier défi à résoudre pour appliquer l'algorithme 3 est de calculer efficacement le poids d'une instance. En effet, la méthode de calcul à la ligne 3 qui consiste à faire la somme des utilités de toutes les généralisations d'une instance est exponentielle en fonction de la taille du langage en temps

de calcul. Le second défi est le tirage d'un motif proportionnellement à son poids dans la base de données à la ligne 4. D'une manière générale, l'idée principale est de consacrer la première étape au tirage d'une instance proportionnellement à son poids dans la base de données (la somme des utilités de ses motifs), et les étapes suivantes au tirage d'un motif proportionnellement à son utilité dans l'instance précédemment tirée.

L'exemple 12 montre comment l'algorithme 3 est instancié dans [Boley *et al.*, 2011] pour échantillonner des motifs ensemblistes suivant la fréquence.

Exemple 12. *La figure 2.3 donne un exemple de tirage d'un motif avec les deux étapes de [Boley* et al.*, 2011]. La mesure d'intérêt choisie est la fréquence. Donc, chaque transaction t est pondérée avec le nombre de ses sous-ensembles qui est égal à $\omega_{freq}(t) = 2^{|t|}$ car $u(\varphi, t) = 1$ pour tout motif φ d'une transaction t. Par exemple, nous avons supposé que la transaction d'identifiant 4 a été tirée à la première étape, et à la deuxième étape, l'itemset $\varphi = BD$ est retourné parmi l'ensemble des motifs de la transaction t_4. Lors de la deuxième étape, le tirage d'un motif peut se faire en parcourant les items de la transaction avec une fonction "Pile/Face"(si "Pile" alors l'item est ajouté dans le motif à retourner).*

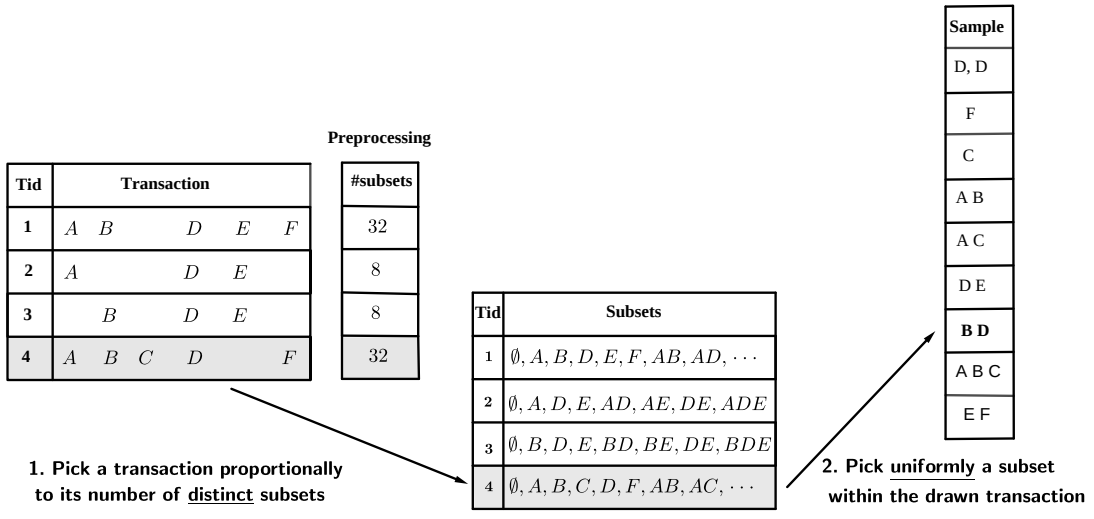


FIGURE 2.3 – Echantillonnage en deux étapes de motifs ensemblistes

Nous allons maintenant analyser la classe de méthodes en deux étapes suivant les critères en entrée et en sortie précédemment présentés :

Les critères en entrée

— **Langages :** [Boley *et al.*, 2011] ont introduit une méthode d'échantillonnage en deux étapes pour échantillonner des motifs sur une base de données transactionnelles. Ensuite [Boley *et al.*, 2012] étendent leurs premiers travaux pour trouver des motifs ensemblistes discriminatifs afin de décrire plusieurs bases de données transactionnelles. Enfin plus récemment, [Giacometti et Soulet, 2018] ont étendu la classe de méthodes d'échantillonnage en plusieurs étapes aux données numériques en ajoutant

une troisième étape. En effet, pour prendre en compte le fait que les données numériques sont souvent représentées dans un espace à plusieurs dimensions, après avoir tiré un point γ proportionnellement à son poids, ils ont divisé la deuxième étape du tirage en deux sous-étapes : (i) on tire aléatoirement un ensemble de dimensions, noté $\mathbb{D}$ par exemple, (ii) un motif φ dans le voisinage de $\gamma(\mathbb{D})$ est uniformément retourné.

— **Mesures d'intérêt :** Dans [Boley *et al.*, 2011], les auteurs proposent des algorithmes d'échantillonnage à l'aide de plusieurs mesures d'intérêt telles que la fréquence, l'aire, le carré de la fréquence, et les mesures discriminatives pour échantillonner des motifs ensemblistes. D'une façon plus générale, [Boley *et al.*, 2011, Boley *et al.*, 2012] montrent comment tirer des motifs ensemblistes en considérant un très grand ensemble de mesures d'intérêt. Plus précisément, étant donné un motif φ et une base de données transactionnelles $\mathcal{D}$ partitionnée en K sous-bases de données $\{\mathcal{D}_1, \cdots, \mathcal{D}_K\}$, ils considèrent l'ensemble de toutes les mesures d'intérêt de la forme $f(\varphi, \mathcal{D}) = m_\star(\varphi) \times \prod_{i=1}^{K} q_i(\varphi, \mathcal{D}_i)$ où $m_\star(\varphi) = \bigstar_{e\in\varphi} b(e)$ avec $\bigstar \in \{\prod, \sum\}$, q_i est la fréquence positive ou négative de φ dans une partie spécifique $\mathcal{D}_i$ de l'ensemble de données d'entrée $\mathcal{D}$ et b renvoie l'utilité d'un item (par exemple le prix, la quantité, etc). Afin de traiter rapidement des processus d'analyse de données interactifs instantanés, [Moens et Boley, 2014] proposent une mesure à base de modèle exceptionnel. C'est une extension des travaux de [Boley *et al.*, 2011] pour l'échantillonnage de motifs contrôlés en introduisant une fonctionnalité de pondération supplémentaire. Cela permet de donner plus d'importance à certains enregistrements de la base de données transactionnelles. [Giacometti et Soulet, 2018] tirent des motifs numériques suivant une probabilité proportionnelle à leurs densités (la densité d'un motif est définie par le nombre de points dans son voisinage, i.e. à une distance inférieure à un rayon r, divisé par le volume de son voisinage).

— **Contraintes :** Avant les travaux réalisés dans cette thèse, aucune méthode d'échantillonnage à plusieurs étapes ne permettait d'utiliser des contraintes.

Les critères en sortie

— **Exactitude du tirage :** La propriété 1 montre que la méthode de tirage de l'algorithme 3 est exacte.

Propriété 1 (Exactitude du tirage)**.** *Soient $\mathcal{D}$ une base de données et m une mesure d'intérêt vérifiant la définition 14, l'algorithme 3 tire un motif φ de $\mathcal{L}$ proportionnellement à $m(\varphi, \mathcal{D})$.*

Preuve. Soient Z la constante définie par $Z = \sum_{\varphi\in\mathcal{L}} m(\varphi, \mathcal{D})$, et $\pi(\varphi)$ la probabilité de tirer le motif φ à l'aide de l'algorithme 3. D'après les lignes 3 et 4, la probabilité de tirer le motif φ est égale à $\sum_{\gamma\in\mathcal{D}\wedge\varphi\preceq\gamma} \pi(\gamma) \times \pi(\varphi/\gamma)$. C'est la probabilité de tirer une instance γ et de tirer le motif φ dans γ. Or $\pi(\gamma) = \sum_{\varphi\preceq\gamma} u(\varphi, \gamma)/Z$ et $\pi(\varphi/\gamma) = u(\varphi, \gamma)/\sum_{\varphi\preceq\gamma} u(\varphi, \gamma)$ d'après la ligne 5, alors on a $\pi(\varphi) = \sum_{\gamma\in\mathcal{D}\wedge\varphi\preceq\gamma} \frac{\sum_{\varphi'\preceq\gamma} u(\varphi',\gamma)}{Z} \times \frac{u(\varphi,\gamma)}{\sum_{\varphi'\preceq\gamma} u(\varphi',\gamma)} = \sum_{\gamma\in\mathcal{D}\wedge\varphi\preceq\gamma} u(\varphi, \gamma)/Z = m(\varphi, \mathcal{D})/Z$. D'où le résultat. □

Notez que dans [Boley *et al.*, 2012], les auteurs utilisent CFTP (Coupling From The Past) pour simuler un tirage exact lors de la première étape sans pour autant calculer la constante de normalisation qui est la somme totale des intérêts des motifs du langage.

— **Complexité de la méthode :** La complexité de l'algorithme 3 est fortement liée aux mesures d'intérêt utilisées et à la nature du langage. Dans le cas des bases de données transactionnelles et des mesures d'intérêt telles que l'aire et la fréquence, la complexité du prétraitement est linéaire par rapport à la taille de la base de données. Par exemple, dans [Boley *et al.*, 2011], leur algorithme d'échantillonnage de motifs suivant la fréquence a une complexité en $O(|\mathcal{D}|)$ pour le prétraitement et une complexité de tirage d'un motif en $O(\log(|\mathcal{D}|)+|\mathcal{I}|)$. Ainsi, le calcul d'un échantillon de k motifs avec $\mathcal{S}amp\ell e_k(\mathcal{L},\mathcal{T},m)$ et $m \in \{freq, supp\}$ nécessite une complexité en $O(k \times (\log(\mathcal{D}) + |\mathcal{I}|))$. Une telle complexité montre la rapidité des méthodes d'échantillonnage suivant la fréquence et l'aire sur des données transactionnelles.

Par contre, la complexité de la première étape peut être très couteuse pour des mesures d'intérêt plus sophistiquées telle que la discriminativité ou la puissance $n^{ième}$ de la fréquence (avec $n \geq 3$). C'est pour cette raison qu'a été proposée dans [Boley *et al.*, 2012] l'utilisation de la méthode de tirage CFPT qui est par exemple en $O(|\mathcal{T}|^n)$ pour le prétraitement et en $O(\log^n(|\mathcal{T}|))$ pour le tirage d'un motif dans le cas de la puissance $n^{ième}$ de la fréquence.

2.3 Utilisation de l'échantillonnage en sortie de motifs

Dans la littérature, il existe plusieurs domaines d'application de l'échantillonnage en sortie de motifs. Parmi ces domaines d'application, nous avons la construction de variables pour la classification [Boley *et al.*, 2011], la découverte interactive centrée sur l'utilisateur [van Leeuwen, 2014, Dzyuba *et al.*, 2014, Giacometti et Soulet, 2017] et la détection d'anomalies dans des bases de données transactionnelles [Giacometti et Soulet, 2016].

2.3.1 Construction de variables pour la classification

Notons que dans le cas de la classification, on prend en entrée des bases de données labellisées où chaque instance γ est étiquetée par une classe $\mathcal{C}(\gamma)$. La construction de variables pour la classification à l'aide d'une méthode d'échantillonnage en sortie se fait en deux phases dans [Boley *et al.*, 2011]. La première phase consiste à tirer un échantillon $\mathbb{S}_k = \{\varphi_1, \cdots, \varphi_k\}$ de k motifs suivant une distribution proportionnelle à une mesure d'intérêt m choisie par l'utilisateur avec $\mathcal{S}amp\ell e_k(\mathcal{L},\mathcal{D},m)$. A partir des motifs dans $\mathbb{S}_k$, on transforme la base de données en construisant un ensemble de variables binaires $\mathbb{V} = \{v_1, \ldots, v_k\}$ où chaque variable v_j est définie pour toute instance γ_i par $v_j[i] = 1$ si $\varphi_j \preceq \gamma_i$. On a en outre une variable contenant les classes des instances définie par $Class[i] = \mathcal{C}(\gamma_i)$ pour toute instance γ_i de la base de données. Quel que soit son type (transactionnel, séquentiel, ...), tout jeu de données étiqueté peut ainsi être transformé en une matrice Mat définie avec une ligne par instance γ_i et une colonne par variable de l'ensemble $\{v_1, \cdots, v_k, Class\}$.

Après cette transformation, on peut appliquer un algorithme de classification classique tel que le SVM. On peut remarquer que ceci est une méthode très simple et rapide pour construire des variables de classification.

Exemple 13. *Considérons le jeu de données transactionnelles $\mathcal{T}$ telles que les transactions sont étiquetées comme suit : $\mathcal{C}(t_1) = c_1$, $\mathcal{C}(t_2) = c_1$, $\mathcal{C}(t_3) = c_2$ et $\mathcal{C}(t_4) = c_2$. La figure 2.4 montre comment construire un classifieur à partir d'un échantillon de motifs $Sampl\grave{e}_3(\mathcal{L}_\mathcal{I}, \mathcal{T}, freq)$. Les motifs échantillonnés $\{ADE, DE, BD\}$ constituent les attributs pour la classification. On sait que $ADE \subseteq t_1$ et $ADE \not\subseteq t_3$, d'où nous avons $Mat[1][1] = 1$ et $Mat[3][1] = 0$. Dans cet exemple, nous avons $v_1 = ADE$, $v_2 = DE$ et $v_3 = BD$.*

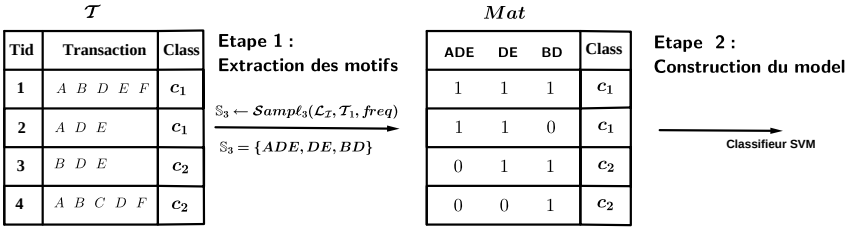


FIGURE 2.4 – Méthodologie de construction de classifieur à partir d'un échantillon de motifs

2.3.2 Détection d'anomalies

Une anomalie ou une donnée aberrante (outlier en anglais) est une instance qui s'écarte suffisamment des autres instances pour être suspectée d'avoir été générée par un mécanisme différent [Hawkins, 1980]. La détection d'anomalies est une tâche importante et nécessaire dans de nombreuses applications réelles telles que la détection de fraude par carte de crédit, la découverte d'activités criminelles dans le commerce électronique, le marketing et la segmentation de la clientèle. Dans [He *et al.*, 2005], les auteurs ont l'intuition que les itemsets fréquents découverts par les algorithmes de recherche de règles d'association [Agrawal *et al.*, 1993] reflètent les motifs communs du jeu de données, alors que les motifs peu fréquents contenus dans quelques instances de la base de données peuvent être utilisés pour décrire des valeurs aberrantes. Autrement dit, une anomalie ne contient que peu de motifs et ces motifs ne sont pas très fréquents dans la base de données. Nous distinguons dans la suite deux types de méthodes de détection d'anomalies qui se basent sur le calcul exact/approché des motifs fréquents.

Méthode exacte : La définition 15 est une généralisation de l'intuition de [He *et al.*, 2005] dans le sens où la formule 2.1 prend en plus en compte une contrainte syntaxique q :

Définition 15 (FPOF)**.** *Soient γ une instance d'une base de données $\mathcal{D}$ et q une contrainte syntaxique. Le FPOF (Frequent Pattern Outlier Factor) exact sous une contrainte syntaxique q d'une instance γ dans la base de données $\mathcal{D}$ est exprimé de la manière suivante :*

$$fpof(\gamma, \mathcal{D}, q) = \frac{score(\gamma, \mathcal{D}, q)}{\max_{\gamma' \in \mathcal{D}} score(\gamma', \mathcal{D}, q)} \tag{2.1}$$

où $score(\gamma, \mathcal{D}, q) = \sum_{\varphi \in \mathcal{T}h(\mathcal{L}, \mathcal{D}, q) \wedge \varphi \preceq \gamma} freq(\varphi, \mathcal{D})$. *Le FPOF est compris entre 0 et 1.*

Exemple 14. *Considérons le jeu de données transactionnelles du tableau 2.2 et la contrainte syntaxique* $q_{>0}$ *définie pour tout motif* φ *et base de données* $\mathcal{D}$ *par* $q_{>0}(\varphi, \mathcal{D}) = true$ *ssi* $\|\varphi\| > 0$. *Calculons les fpof des transactions dans cette base de données.*

TABLE 2.2 – *fpof* des transactions de la base de données transactionnelles $\mathcal{T}_2$

Tid	**Transactions**	*fpof*
t_1	C	0.12
t_2	A B D E	0.96
t_3	A B C E	1.00
t_4	C D	0.25

Considérons la base de données $\mathcal{T}_2$. *L'ensemble des motifs de la transaction* t_2 *de norme strictement positive est* $2^{t_2} \setminus \emptyset = \{A, B, D, E, AB, AD, AE, BD, BE, DE, ABD, ABE, ADE, BDE, ABDE\}$. *Or les fréquences de ces motifs dans* $\mathcal{T}_2$ *sont respectivement* $freq(A, \mathcal{T}_2) = 2$, $freq(B, \mathcal{T}_2) = 2$, $freq(D, \mathcal{T}_2) = 2$, $freq(E, \mathcal{T}_2) = 2$, $freq(AB, \mathcal{T}_2) = 2$, $freq(AD, \mathcal{T}_2) = 1$, $freq(AE, \mathcal{T}_2) = 2$, $freq(BD, \mathcal{T}_2) = 1$, $freq(BE, \mathcal{T}_2) = 2$, $freq(DE, \mathcal{T}_2) = 1$, $freq(ABD, \mathcal{T}_2) = 1$, $freq(ABE, \mathcal{T}_2) = 2$, $freq(ADE, \mathcal{T}_2) = 1$, $freq(BDE, \mathcal{T}_2) = 1$ *et* $freq(ABDE, \mathcal{T}_2) = 1$. *Alors son score est* $score(t_2, \mathcal{T}_2, q_{>0}) = \sum_{\emptyset \not\subseteq \varphi \subseteq t_2} freq(\varphi, \mathcal{T}_2) = 23$. *En appliquant ce même principe sur toutes les transactions de* $\mathcal{T}_2$, *on a les scores suivants :* $score(t_1, \mathcal{T}_2, q_{>0}) = 3$, $score(t_3, \mathcal{T}_2, q_{>0}) = 24$ *et* $score(t_4, \mathcal{T}_2, q_{>0}) = 6$. *Après normalisation avec le maximum des scores on obtient la dernière colonne du tableau de la figure 2.2. Dans cet exemple, la transaction* t_1 *a le plus faible score de fpof suivie de la transaction* t_4. *Ainsi, nous disons que les top-2 outliers de la base de données transactionnelles* $\mathcal{T}_2$ *sont* t_1 *et* t_4.

La mesure FPOF [He *et al.*, 2005] est considérée comme une approche de base permettant de calculer le score d'aberration d'une instance en utilisant la fréquence des motifs qu'elle contient. Etant donné que les auteurs utilisent les algorithmes classiques d'extraction de motifs fréquents, alors le temps de calcul des FPOF devient très coûteux car la taille de la sortie est très grande. Dès lors, plusieurs variantes de l'approche FPOF ont été proposées : WCFPOF [Ren *et al.*, 2009], FPCOF [Tang *et al.*, 2009], LFPOF [Zhang *et al.*, 2010], MFPOF [Lin *et al.*, 2010] et FPI [Kuchar et Svatek, 2017]. Cependant, tous ces algorithmes utilisent des méthodes exhaustives d'extraction de motifs intéressants telle que APRIORI [Agrawal *et al.*, 1993] ou FP-GROWTH [Han *et al.*, 2000] en ajoutant des contraintes pour réduire la taille de la sortie. D'une façon plus détaillée, [Ren *et al.*, 2009] utilisent uniquement des motifs fréquents fermés (motifs n'ayant pas de spécialisation avec la même fréquence). [Tang *et al.*, 2009] mesurent à quel point les motifs existants sont contradictoires, où un ensemble de motifs moins contradictoires signifie que l'instance est plus probablement une instance normale. [Zhang *et al.*, 2010] ne trouvant pas intéressant de considérer un motif et ses généralisations dans une même instance, ils

utilisent seulement le motif fréquent le plus long de chaque instance pour le calcul du score alors que [Lin *et al.*, 2010] utilisent des motifs fréquents maximaux (motifs n'ayant pas de spécialisation fréquent). Enfin, [Kuchar et Svatek, 2017] utilisent une contrainte de taille maximale pour pénaliser les motifs longs.

Méthode Approchée : L'extraction de tous les motifs fréquents et leurs applications sont généralement intensives en calculs. Il existe récemment des approches plus performantes qui se basent sur l'échantillonnage en sortie [Giacometti et Soulet, 2016], et permettent de calculer rapidement la valeur approchée du FPOF d'une transaction de la base de données. Dans [Giacometti et Soulet, 2016], les auteurs ont aussi proposé une méthode appelée FPOF k-échantillonné pour la détection d'anomalies à l'aide des algorithmes d'échantillonnage en sortie. L'idée est de calculer une valeur approchée du FPOF à l'aide d'un échantillon de motifs tirés proportionnellement à leurs fréquences dans la base de données. La définition 16 est une généralisation de celle de [Giacometti et Soulet, 2016].

Définition 16 (FPOF k-échantillonné)**.** *Soient un entier $k > 0$ et une contrainte syntaxique q. Le FPOF k-échantillonné d'une instance γ d'une base de données $\mathcal{D}$ sous une contrainte syntaxique q est défini comme suit :*

$$\widetilde{fpof}_k(\gamma, \mathcal{D}, q) = \frac{\sum_{\varphi \preceq \gamma} count(\varphi, \mathbb{S}_k)}{\max_{\gamma' \in \mathcal{D}} \sum_{\varphi \preceq \gamma'} count(\varphi, \mathbb{S}_k)} \tag{2.2}$$

où $\mathbb{S}_k$ est un échantillon de k motifs satisfaisant la contrainte q et tirés suivant une probabilité proportionnelle à la fréquence : $\mathbb{S}_k = \mathcal{S}ample_k(\mathcal{T}h(\mathcal{L}, \mathcal{D}, q), \mathcal{D}, freq)$, et $count(\varphi, \mathbb{S}_k)$ est le nombre de fois que le motif φ apparaît dans l'échantillon $\mathbb{S}_k$.

Exemple 15. *En considérant la base de données transactionnelles $\mathcal{T}_2$, nous allons calculer un échantillon de 10 motifs ayant une norme non nulle que l'on note formellement par : $\mathcal{S}ample_{10}(\mathcal{T}h(\mathcal{L}_{\mathcal{I}}, \mathcal{T}, ||\varphi|| > 0), \mathcal{T}, freq)$. Supposons que l'on obtient l'échantillon $\mathbb{S}_{10} = \{A,$ AB, AB, C, C, AB, AC, D, BE, $AE\}$. Calculons le score de t_2 à l'aide de la formule 2.2 de la définition 16. Les motifs de l'échantillon qui sont inclus dans t_2 sont A, AB, D, BE, et AE. Nous avons dans ce cas $count(A, \mathbb{S}_{10}) = 1$, $count(AB, \mathbb{S}_{10}) = 3$, $count(D, \mathbb{S}_{10}) = 1$, $count(BE, \mathbb{S}_{10}) = 1$ et $count(AE, \mathbb{S}_{10}) = 1$. Alors $score(t_2, \mathbb{S}_{10}) = 1 + 3 + 1 + 1 + 1 = 7$.*

TABLE 2.3 – $\widetilde{fpof}$ des transactions de la base de données transactionnelles $\mathcal{T}_2$

Tid	**Transactions**	*fpof*	$\widetilde{fpof}$
t_1	C	0.12	0.22
t_2	A B D E	0.96	0.77
t_3	A B C E	1.00	1.00
t_4	C D	0.25	0.33

En faisant de même pour les autres transactions, nous avons $score(t_1, \mathbb{S}_{10}) = 2$, $score(t_3, \mathbb{S}_{10}) = 9$ et $score(t_4, \mathbb{S}_{10}) = 3$. En normalisant avec le maximum des scores nous avons les statistiques de la dernière colonne du tableau 2.3. On note aussi que la transaction t_1 a le plus

faible score de $\widetilde{fpof}$ suivie de la transaction t_4. Ainsi, nous retrouvons les mêmes top-2 outliers que la méthode exacte.

Propriété 2 (Convergence)**.** *Soit $\mathcal{D}$ une base de données et k un entier strictement positif et q une contrainte syntaxique. Le $\widetilde{fpof}_k$ k-échantillonné tend vers le fpof exact quand k tend vers l'infini pour toute instance γ de la base de données $\mathcal{D}$:*

$$\lim_{k \longrightarrow \infty} \widetilde{fpof}_k(\gamma, \mathcal{D}, q) = fpof(\gamma, \mathcal{D}, q).$$

Preuve. $\mathbb{S}_k = freq(\mathcal{T}h(\mathcal{L}, freq, q), \mathcal{D})$ signifie qu'il existe une constante $\alpha > 0$ telle que pour tout motif $\varphi \in \mathcal{T}h(\mathcal{L}, freq, q)$, on a :
$\lim_{k \longrightarrow \infty} count(\varphi, \mathbb{S}_k) = \alpha \times freq(\varphi, \mathcal{D})$. Ce qui veut dire que pour chaque instance $\gamma \in \mathcal{D}$, on a : $\lim_{k \longrightarrow \infty} \sum_{\varphi \preceq \gamma} count(\varphi, \mathbb{S}_k) = \alpha \sum_{\varphi \preceq \gamma} freq(\varphi, \mathcal{D}) = \alpha \times \max_{\gamma' \in \mathcal{D}} \sum_{\varphi \preceq \gamma} count(\varphi, \mathbb{S}_k)$.
En injectant ce résultat dans la définition 16, on obtient le résultat suivant :
$\lim_{k \longrightarrow \infty} \widetilde{fpof}_k(\gamma, \mathcal{D}, q) = \frac{\alpha \times \sum_{\varphi \preceq \gamma} count(\varphi, \mathbb{S}_k)}{\alpha \times \max_{\gamma' \in \mathcal{D}} \sum_{\varphi \preceq \gamma'} count(\varphi, \mathbb{S}_k)}$.

Donc $\lim_{k \longrightarrow \infty} \widetilde{fpof}_k(\gamma, \mathcal{D}, q) = \frac{\sum_{\varphi \preceq \gamma} count(\varphi, \mathbb{S}_k)}{\max_{\gamma' \in \mathcal{D}} \sum_{\varphi \preceq \gamma'} count(\varphi, \mathbb{S}_k)}$. D'où le résultat. □

2.3.3 Découverte interactive

La découverte interactive est un processus qui nécessite une courte boucle avec une interaction rapide entre le système et l'utilisateur [van Leeuwen, 2014, Dzyuba *et al.*, 2014, Bhuiyan et Hasan, 2016, Giacometti et Soulet, 2017]. Elle se fait en 3 grandes phases qui forment le circuit "*Mine-Interact-Learn*" [van Leeuwen, 2014] comme le montre la figure 2.5.

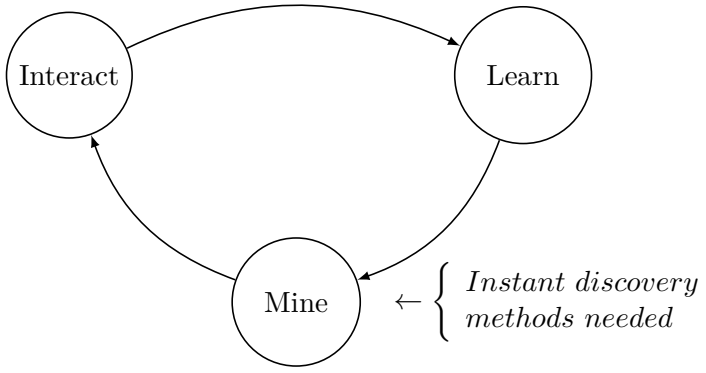


FIGURE 2.5 – Processus de la découverte interactive

— **Extraction :** cette phase permet au système de proposer à l'utilisateur des motifs extraits suivant la mesure d'intérêt qu'il a choisie. Son principal défi est de pouvoir prendre en compte aux itérations suivantes, non seulement la mesure d'intérêt choisie, mais aussi des préférences utilisateurs exprimées lors de la phase d'interaction.

— **Interaction :** c'est la phase où l'utilisateur interagit avec le système via les motifs extraits. C'est ici que des retours implicites tels que des clics sur les motifs ou des retours explicites formulés par l'utilisateur (en indiquant explicitement que ce motif l'intéresse ou pas) sont injectés dans le système pour les prochaines itérations.

— **Apprentissage :** l'objectif de cette phase est de construire un modèle des préférences de l'utilisateur en prenant en compte les retours implicites ou explicites qu'il a fait lors des phases d'interaction. Ce modèle de préférence sera utilisé par le système lors de la prochaine phase d'extraction pour extraire des motifs plus pertinents pour l'utilisateur. Ce qui va alors différencier une méthode d'une autre, c'est quel type de modèle de préférences est construit/appris, et ensuite comment ce modèle de préférence est utilisé (conjointement avec la mesure) dans la phase d'extraction.

L'exploration instantanée impose une contrainte sur le délai de réponse de quelques secondes pour extraire les motifs. Des méthodes d'extraction exhaustive ne permettent pas d'obtenir des motifs intéressants dans un délai de réponse aussi court. Les méthodes optimales se concentrent sur la recherche des meilleurs motifs par rapport à une relation de préférence. Par conséquent, elles se concentrent souvent sur la même partie du langage de motifs $\mathcal{L}$ qui est souvent sous-optimale et qui contient des motifs légèrement différents. Cela entraîne un défaut de diversité comme nous l'avons souligné dans la section 1.3. En effet, la diversité, un point crucial pour les méthodes interactives, est très importante pour présenter à l'utilisateur un ensemble de motifs variés à chaque étape afin d'améliorer sa vision des données et d'aider le système pour apprendre ses préférences.

Heureusement, l'échantillonnage en sortie de motifs est une solution pertinente pour avoir un bon système interactif à l'aide de ses 3 propriétés *Rapidité-Diversité-Flexibilité*. Rappelons que les défis résolus par l'échantillonnage en sortie sont : d'abord échantillonner un motif suivant une distribution proportionnelle à une mesure d'intérêt choisie par l'utilisateur en temps très court (*extraction* avec *rapidité*), proposer des motifs divers afin de faciliter l'apprentissage par le système d'un modèle de préférence de l'utilisateur (*interaction* pour une meilleure *diversité*), et enfin d'intégrer dans le processus d'extraction le modèle de préférences de l'utilisateur modélisées par des mesures d'intérêt et des contraintes (*apprentissage* avec une *flexibilité*).

2.4 Synthèse sur les méthodes d'échantillonnage en sortie

Les méthodes d'échantillonnage en sortie, contrairement aux méthodes de fouille exhaustives, sont rapides (si les mesures d'intérêt ne sont pas complexes et la base de données n'est pas trop volumineuse), garantissent une bonne diversité des motifs échantillonnés, et une bonne flexibilité dans le sens où elles peuvent prendre en compte plusieurs types de mesures d'intérêt. D'une façon intéressante, elles permettent une bonne interaction entre l'utilisateur et le système dans les processus de découverte instantanée de motifs centrée sur l'utilisateur.

Dans la littérature, nous notons que les méthodes à plusieurs étapes ("multi-steps") [Boley *et al.*, 2012, Boley *et al.*, 2012, Moens et Boley, 2014, Giacometti et Soulet, 2018] garantissent l'exactitude du tirage de motifs ; nous les appelons les *méthodes exactes*.

TABLE 2.4 – Typologie des méthodes d'échantillonnage en sortie de motifs

Type	**Guarantees**	**Referencee**	**Langage**	**Measure**	**Constraint**
Random walk : MCMC	convergence	[Al Hasan et Zaki, 2009, Bhuiyan *et al.*, 2012]	graphs	frequency	no
		[Boley *et al.*, 2010]	itemsets	any strictly positive measure	closed
		[Li et Zaki, 2012]	itemsets	frequency	minimal
Random walk : Heuristics	no	[Moens et Goethals, 2013]	itemsets	frequency	maximal
		[Bendimerad *et al.*, 2016]	graphs	weighted relative accuracy	valid
SAT	bounds	[Dzyuba *et al.*, 2017, Gueguen *et al.*, 2019]	itemsets	any strictly positive measure	any constraint
Multi-steps	exact	[Boley *et al.*, 2011]	itemsets	frequency, area, discriminative, power of frequency	no
		[Boley *et al.*, 2012]	itemsets	frequency, area, discriminative, power of frequency, rare	no
		[Moens et Boley, 2014]	itemsets	exceptional model	no
		[Giacometti et Soulet, 2018]	numerical	density	no
–	–	–	sequence	–	–

Par contre, les méthodes qui se basent sur des marches aléatoires à l'aides de chaînes de Markov par Monte-Carlo [Al Hasan et Zaki, 2009, Bhuiyan *et al.*, 2012, Boley *et al.*, 2010, Li et Zaki, 2012] assurent la convergence de la distribution vers la distribution exacte au bout d'un certain nombre d'itérations, ce sont des *méthodes dites convergentes*. Les méthodes basées sur formalisme le SAT [Dzyuba *et al.*, 2017, Gueguen *et al.*, 2019] garantissent l'exactitude avec un certain taux d'erreur, ce sont les *méthodes dites approchées*, alors que les méthodes heuristiques [Moens et Goethals, 2013, Bendimerad *et al.*, 2016] n'offrent aucune garantie sur l'exactitude du tirage. L'inexactitude du tirage des méthodes approchées est due au fait que les méthodes approchées préfèrent approximer la constante de normalisation, qui est la somme totale des poids des motifs du langage, car elle est souvent difficile à calculer.

On peut aussi noter que la plupart des méthodes d'échantillonnage en sortie ont attaqué les bases de données transactionnelles [Boley *et al.*, 2010, Li et Zaki, 2012, Moens et Goethals, 2013, Dzyuba *et al.*, 2017, Boley *et al.*, 2011, Boley *et al.*, 2012, Moens et Boley, 2014]. [Al Hasan et Zaki, 2009, Bendimerad *et al.*, 2016] ont proposé des méthodes d' échantillonnage en sortie de motifs de graphes et [Giacometti et Soulet, 2018] une méthode d' échantillonnage en sortie de motifs à partir des données numériques. Par contre, aucune méthode d'échantillonnage en sortie n'a encore été proposée pour l'échantillonnage en sortie de motifs séquentiels.

La figure 2.6 montre une répartition des méthodes d'échantillonnage en sortie de l'état de l'art suivant les axes de la figure 1.6. Nous notons que les méthodes à base du framework SAT et des MCMC offrent potentiellement une grande qualité des motifs en sortie car elles ne sont pas limitées à l'usage de certaines mesures d'intérêt, mais génériques et peuvent être utilisées avec des mesures d'intérêt complexes. C'est d'ailleurs, cette complexité des mesures d'intérêt qui fait que les méthodes SAT et MCMC peuvent être très lentes. A contrario, les méthodes en deux étapes (Two-Step) sont très rapides mais elles ne prennent en compte que peu de mesures d'intérêt. Nous avons aussi noté le fait qu'elles ne prennent pas de contraintes. A cause de ces problèmes, les méthodes en Two-Step de l'état de l'art ne renvoient pas des motifs ayant une qualité aussi bonne que celles de SAT et de MCMC. De ce fait, nous avons noté d'importantes limites des méthodes d'échantillonnage en sortie de motifs de la littérature qui pourtant devraient être surmontées.

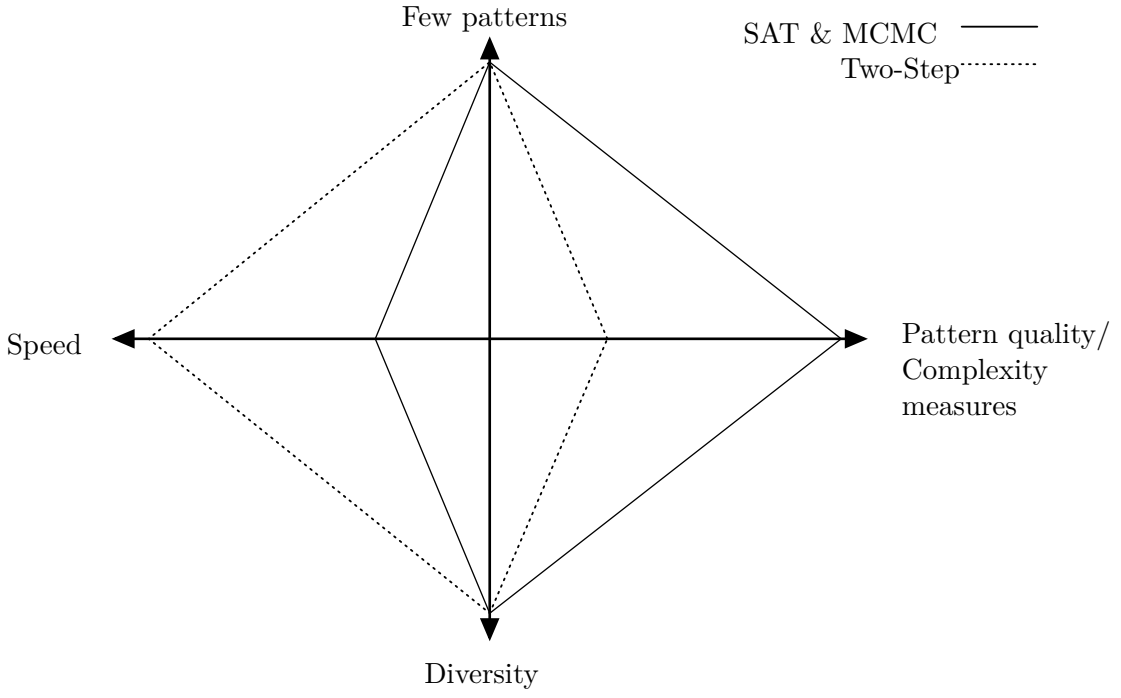


FIGURE 2.6 – Caractéristiques des méthodes d'échantillonnage en sortie

Chapitre 3

Limites des méthodes existantes

Sommaire

Les méthodes d'échantillonnage en sortie de l'état de l'art se heurtent à 3 problèmes majeurs. Le premier est le problème de la longue traîne, le second est lié à la représentation des données en mémoire et le dernier problème est lorsque la base de données est distribuée. Dans ce chapitre, nous allons commencer par présenter à la section 3.1 les bases de données du Web qui montrent les principales limites des méthodes existantes, et qui font partie des jeux de données que nous allons aussi utiliser dans cette thèse lors de la partie expérimentale de nos contributions au chapitre 6. Ensuite, la section 3.2 explique le phénomène de la longue traîne sur des jeux de données centralisés et en particulier son impact sur la qualité des motifs échantillonnés pour le calcul des FPOF (définis

à la section 2.3.2) lors de la détection d'anomalie. A la section 3.3, nous soulignons l'intérêt d'une représentation compacte des données en mémoire quand la base de données devient très volumineuse et aussi les limites des méthodes existantes pour échantillonner des motifs à partir de ces structures de données compactes. Enfin, la section 3.4 présente les limites des méthodes de fouille exhaustives de motifs et des méthodes existantes d'échantillonnage en sortie lorsque les données sont distribuées.

3.1 Présentation des données du Web

Comme nous l'avons précédemment souligné au chapitre 1, les données du Web nous permettent de construire des jeux de données réels tels que ceux des données ensemblistes afin de valider nos approches. Rappelons que ces données sont distribuées sur plusieurs triplestores tels que DBpedia [1][Auer *et al.*, 2007], Yago [2][Suchanek *et al.*, 2007], GeoNames [3][Ahlers, 2013], Wikidata [4][Vrandecic et Krötzsch, 2014], etc. Ceci est dû au fait que certaines entités sont décrites par plusieurs triplestores. Par exemple, l'entité "Cheikh Amadou Bamba" de la classe *Person* est décrite dans DBpedia et Wikidata.

3.1.1 Préliminaires

Pour les bases de données distribuées, cette thèse se limite au cas des données ensemblistes. Ainsi, nous allons définir les notions essentielles pour faire de la fouille de motifs dans des bases de données transactionnelles distribuées.

Définition 17 (Base de données distribuée/centralisée)**.** *Une base de données distribuée* $\mathcal{P} = \{\mathcal{T}_1, ..., \mathcal{T}_K\}$ *est un ensemble de bases de données (appelées fragments) tel que pour tout identifiant de transaction* $j \in \mathbb{N}$*, on a* $\mathcal{T}_k[j] \cap \mathcal{T}_l[j] = \emptyset$ *pour tous les fragments* $\mathcal{T}_k$ *et* $\mathcal{T}_l$ *où* $k \neq l$*.* $\mathcal{T}_k[j]$ *représente la transaction d'identifiant* j *dans le fragment* $\mathcal{T}_k$*. La base de données centralisée* $\mathcal{P}^*$ *correspondant à* $\{\mathcal{T}_1, \cdots, \mathcal{T}_K\}$ *est la fusion de toutes les transactions de chacun des fragments :* $\mathcal{P}^* = \{(j, \mathcal{T}_1[j] \cup \cdots \cup \mathcal{T}_K[j]) \in \mathbb{N} \times \mathcal{L}_\mathcal{I} : \exists k \in [1..K] \wedge \mathcal{T}_k[j] \neq \emptyset\}$.

Dans la suite, sauf indication contraire, $\mathcal{P} = \{\mathcal{T}_1, ..., \mathcal{T}_K\}$ désigne une base de données distribuée de K fragments et $\mathcal{P}^*$ correspond à sa base de données centralisée. On dit que la base de données distribuée $\mathcal{P} = \{\mathcal{T}_1, ..., \mathcal{T}_K\}$ est un partitionnement de la base de données centralisée $\mathcal{P}^*$.

Exemple 16. *Par exemple, dans la figure 3.1, on constate que la base de données distribuée* $\mathcal{P} = \{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4\}$ *est un partitionnement de la base de données centralisée* $\mathcal{P}^*$*. Dans ce cas, la transaction d'identifiant 4 est répartie sur les fragments* $\mathcal{T}_1, \mathcal{T}_2$ *et* $\mathcal{T}_4$*.*

3.1.2 Typologie des bases de données distribuées

Une base de données distribuée est une base de données dans laquelle tous les périphériques de stockage ne sont pas connectés à un processeur commun. Elle peut être stockée

1. https://dbpedia.org/sparql
2. https://linkeddata1.calcul.u-psud.fr/sparql
3. http://www.geonames.org/
4. https://query.wikidata.org/sparql

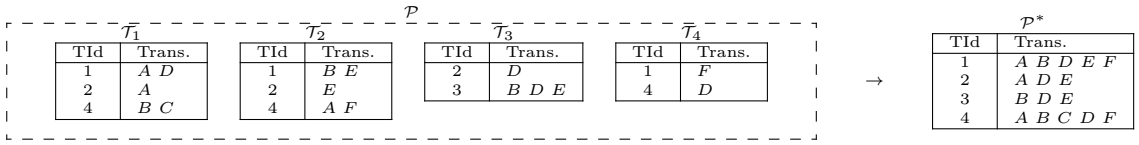


FIGURE 3.1 – Exemple de base de données distribuée $\mathcal{P} = \{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4\}$

sur plusieurs ordinateurs, situés dans le même emplacement physique ; ou elle peut être dispersée sur un réseau d'ordinateurs interconnectés. Contrairement aux systèmes parallèles, dans lesquels les processeurs sont étroitement couplés et utilisent un système de base de données unique, un système de base de données distribuée se compose de sites faiblement couplés qui ne partagent aucun composant physique. Cela peut être nécessaire lorsqu'une base de données particulière doit être accessible à différents utilisateurs dans le monde. Elle doit être gérée de telle sorte que pour les utilisateurs, tout se passe comme si une seule base de données existait.

D'une manière générale, suivant la répartition des données sur les fragments, on distingue deux types de partitionnements particuliers de $\mathcal{P}^*$. Si chaque transaction est contenue sur un seul fragment, on parle de partitionnement horizontal. Si chaque item est contenu sur un seul fragment, on parle de partitionnement vertical. Un partitionnement hybride est un partitionnement qui n'est ni horizontal, ni vertical (e.g., partitionnement de la figure 3.1). Formellement, on a :

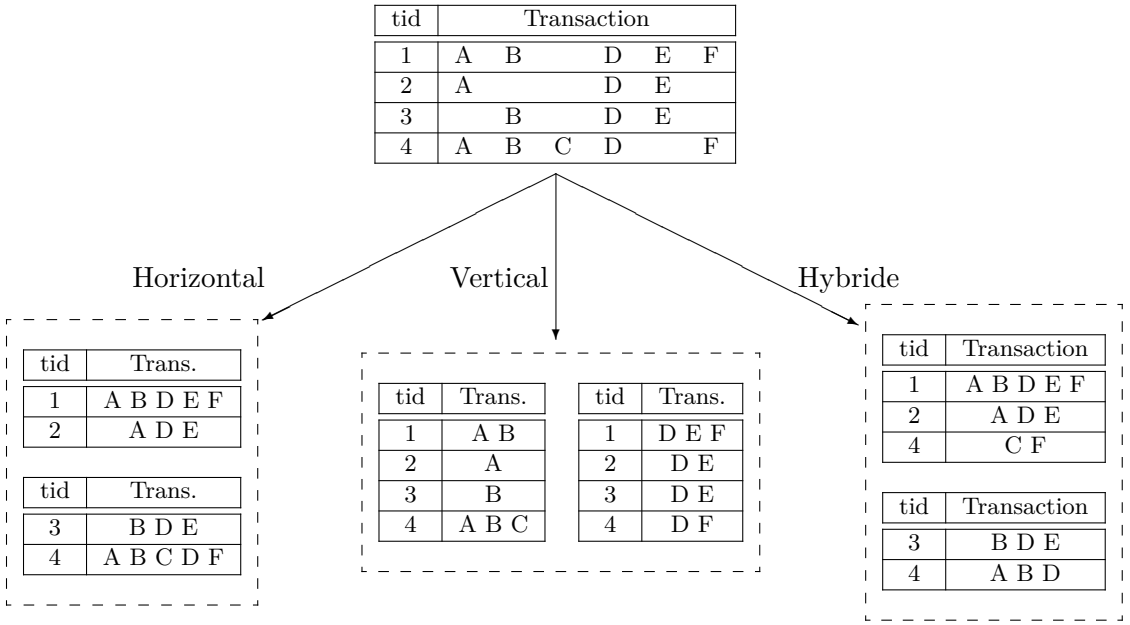


FIGURE 3.2 – Type de partitionnement d'une base de données transactionnelles

Définition 18 (Partitionnent horizontal et vertical)**.** *Soit $\mathcal{P} = \{\mathcal{T}_1, \ldots, \mathcal{T}_K\}$ un partitionnement d'une base de données $\mathcal{T}$. On dit que $\mathcal{P}$ est un partitionnement horizontal de $\mathcal{T}$ si et seulement si pour toute transaction d'identifiant j, on a $(\forall k, k')(\mathcal{T}_k[j] \neq \emptyset \wedge \mathcal{T}_{k'}[j] \neq$*

$\emptyset \Rightarrow k = k'$). *On dit que* $\mathcal{P}$ *est un partitionnement vertical de* $\mathcal{T}$ *si et seulement si pour tout item* $e \in \mathcal{I}$, *on a* $(\forall j, j', k, k')((e \in \mathcal{T}_k[j]) \wedge (e \in \mathcal{T}k'[j']) \Rightarrow k = k')$.

Un partitionnement hybride est un partitionnement qui n'est ni horizontal, ni vertical. Par exemple, dans le cas des données du Web sémantique, une entité est décrite sur plusieurs triplestores. Dans ce cas, il est pertinent d'extraire des motifs sur l'ensemble de la base de données distribuée pour rechercher des corrélations entre les propriétés de triplestores distincts. La figure 3.2 montre les différents types de partitionnement sur un jeu de données transactionnelles $\mathcal{T}$.

3.1.3 Construction des jeux de données

Parmi les jeux de données transactionnelles que nous allons utiliser dans la suite, nous avons `Person` contenant l'ensemble des entités de la classe "Person" conjointes à DBpedia et Wikidata et `Organisation`, l'ensemble des entités de la classe "Organisation" conjointes à DBpedia et Wikidata.

Pour chaque entité `e` d'une classe $C \in \{\texttt{Person}, \texttt{Organisation}\}$, nous considérons simplement qu'elle est décrite par l'ensemble des items ou propriétés `p` tels qu'un triplet RDF (Resource Description Framework) $(s, \texttt{p}, o)$ appartient à DBpedia ou Wikidata, où s et o sont respectivement le sujet et l'objet, et `p` une propriété décrivant la relation entre le sujet et l'objet. Donc chaque entité d'une classe est décrite par l'ensemble de ses propriétés conjointes à DBpedia et Wikidata. Par exemple, si une entité `e` de la classe `Person` est décrite par les propriétés $\{\texttt{p}_1, \cdots, \texttt{p}_i\}$ dans DBpedia, et $\{\texttt{p}'_1, \cdots, \texttt{p}'_j\}$ dans Wikidata, alors les items de `e` dans `Person` sont $\{\texttt{p}_1, \cdots, \texttt{p}_i, \texttt{p}'_1, \cdots, \texttt{p}'_j\}$. En procédant ainsi, nous obtenons plus de $700,000$ transactions pour la base de données `Person` et plus de $300,000$ transactions pour la base de données `Organisation`.

Dans le tableau 3.1, nous donnons le nombre total d'entités dans les classes `Person` et `Organisation`, le nombre total de propriétés décrivant ces entités dans DBpedia et Wikidata, et enfin les nombres minimum, maximum et moyen de propriétés décrivant une entité `e` d'une classe C avec $C \in \{\texttt{Person}, \texttt{Organisation}\}$.

TABLE 3.1 – Caractéristiques des bases de données issues du Web

$\mathcal{T}$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{I}\rvert_{DBpedia}$	$\lvert\mathcal{I}\rvert_{Wikidata}$	$\lvert t\rvert_{min}$	$\lvert t\rvert_{max}$	$\lvert t\rvert_{avg}$
`Person`	772,432	13,142	6,213	8	552	50.02
`Organisation`	338,402	19,022	5,504	8	328	36.22

Nous allons travailler sur $\mathcal{P}^\star$ aux sections 3.2 et 3.3 et directement sur les fragments à la section 3.4.

3.2 Malédiction de la longue traîne

En statistiques et en finance, la longue traîne d'une distribution est sa portion ayant un grand nombre d'occurrences loin de la partie centrale de la distribution [Anderson, 2004]. Dans le reste de cette section, nous allons parler du phénomène de la longue traîne sur la répartition des motifs du langage.

3.2.1 Problème de la longue traîne sur les motifs échantillonnés

Dans notre contexte, la longue traîne désigne les motifs longs et rares beaucoup plus nombreux dans la base de données que les motifs courts et fréquents (la "tête"). En conséquence, il est presque impossible d'extraire les motifs les plus généraux malgré le biais de la fréquence. De telles répartitions sont étonnamment fréquentes dans la fouille de données. Ces distributions ont généralement la forme de la figure 3.3.

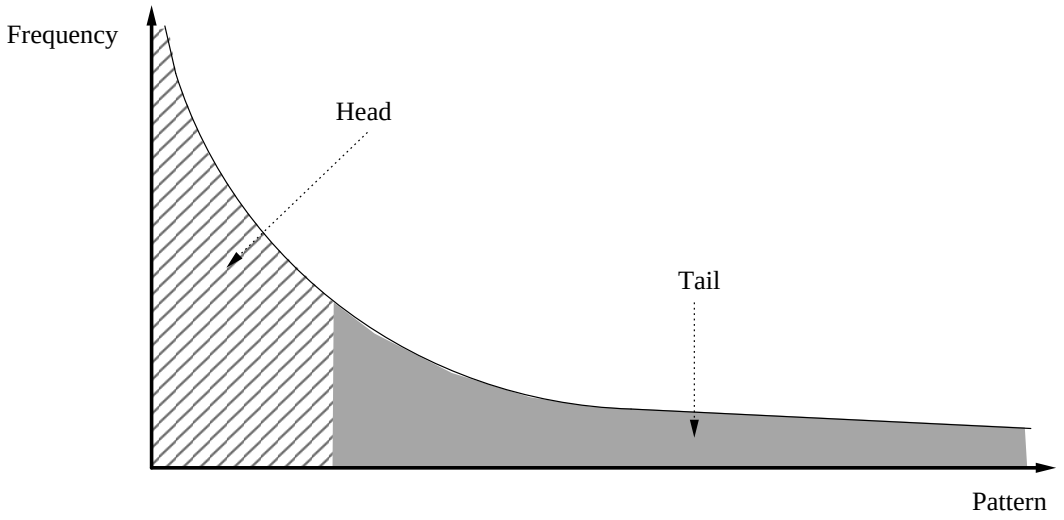


FIGURE 3.3 – Phénomène de la longue traîne sur la distribution des motifs suivant la fréquence

On peut remarquer déjà que les deux jeux de données `Person` et `Organisation` décrits au tableau 3.1 souffrent de la malédiction de la longue traîne par une simple comparaison des longueurs moyennes de leurs transactions, 50.02 pour `Person` et 36.02 pour `Organisation` avec la plus petite transaction contenant seulement 8 items pour les deux jeux de données alors que la plus longue contient 552 items pour `Person` et 328 pour `Organisation`. Dans le cas de la base de données `Person`, l'entité "Barack Obama" contient $2^{552} - 1$ motifs qui peuvent être tirés, contre $2^{17} - 1$ pour "Amar Hussain". Ainsi, une méthode naïve d'échantillonnage en sortie de motifs tire la quasi-totalité des motifs à partir de la transaction d'identifiant "Barack Obama". De ce fait, nous notons qu'il n'y a pas de motifs représentatifs dans l'échantillon ni une diversité des motifs échantillonnés alors que ces deux points sont a priori la force de l'échantillonnage en sortie.

La seule méthode qui utilise un algorithme d'échantillonnage en sortie pouvant faire face à ce problème est celle proposée dans [Dzyuba *et al.*, 2017]. Elle pose des contraintes de taille maximale pour éviter d'avoir en sortie trop de motifs peu fréquents du fait de leurs longueurs. Malheureusement, non seulement elle n'est pas exacte, mais aussi en pratique, elle ne s'applique pas sur des données plus complexes que les itemsets comme les séquences. Et pourtant, le problème de la longue traine est encore plus important avec les données séquentielles qu'avec les données transactionnelles car le biais sur les longueurs des instances est beaucoup plus important sur les séquences que les itemsets. Le chapitre 5 montre l'effet de la longue traîne sur les données séquentielles ainsi que la solution que nous avons proposée.

3.2.2 Impact de la longue traîne sur les usages

Dans cette partie, nous allons montrer les effets de la longue traîne sur le calcul des FPOF $k-$échantillonnées par la méthode de [Giacometti et Soulet, 2016] pour la détection de données aberrantes. Nous avions souligné à la section 2.3.2 que la formule du FPOF, telle qu'elle est définie dans [He *et al.*, 2005], est atteinte de la malédiction de la longue traîne car elle favorise les transactions les plus longues de la base de données, et donc les motifs longs et rares. Dans cette partie, nous allons montrer l'impact de la longue traîne sur cette mesure.

Pour ce faire, nous allons utiliser la définition 16 afin d'approximer le FPOF de chaque transaction à l'aide d'un échantillon de $k = 10^6$ motifs, $\mathcal{S}amp\ell e_k(\mathcal{L}_\mathcal{I}, \texttt{Person}, freq)$ et $\mathcal{S}amp\ell e_k(\mathcal{L}_\mathcal{I}, \texttt{Organisation}, freq)$ pour les bases de données `Person` et `Organisation` respectivement. La figure 3.4 montre la répartition des transactions des bases de données `Person` et `Organisation` suivant leurs FPOF.

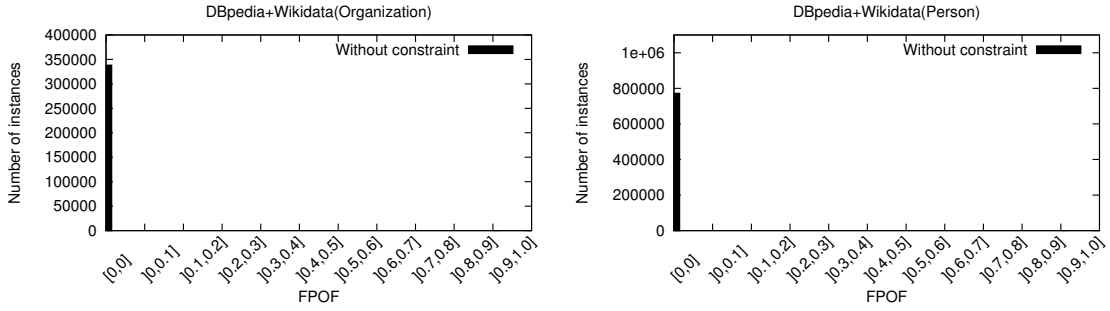


FIGURE 3.4 – Phénomène de la longue traîne sur les FPOF

Pour chacune des bases de données `Person` et `Organisation`, nous constatons que seule la plus longue transaction a un FPOF non nul. Tous les motifs tirés sont très longs et proviennent de la transaction ayant la longueur maximale, ce qui fait que leur fréquence est égale à 1. Dans une pareille situation, nous sommes incapables de nous prononcer sur les degrés d'aberration des instances des bases de données. De ce fait, nous pouvons alors dire qu'il n'est pas prometteur d'appliquer la formule du FPOF définie dans [He *et al.*, 2005] sur des bases de données séquentielles qui sont très souvent atteintes de la malédiction de

la longue traîne. Cela est dû au fait qu'il ne sera pas possible de distinguer les outliers à cause d'un nombre très important de séquences avec des FPOF nuls.

3.2.3 Idée clé pour éviter le phénomène de la longue traîne

Pour illustrer ce problème, nous allons reconsidérer le jeu de données jouet transactionnel $\mathcal{T}$ de la section 1.1 rappelé dans le tableau 3.2. On note que la transaction d'identifiant 4 est la plus longue de toutes les transactions de la base de données avec 15 items, alors que les trois autres transactions en contiennent au plus 5. Ainsi, le nombre de sous-ensembles non vides de la quatrième transaction est de $2^{15} - 1 = 32767$. Le nombre total de sous-ensembles non vides de la base est de $45 + 32767 = 32812$. De ce fait, une méthode de tirage comme celle proposée dans [Boley *et al.*, 2011] va tirer dans 99.86% des cas des motifs dans la quatrième transaction de la base. Cela aura pour conséquence que les motifs retournés seront très peu fréquents, en étant souvent des motifs uniquement contenus dans la quatrième transaction. Du coup, comme nous l'avons dit à la section 3.2.2, toutes les autres transactions auront des FPOF nuls, ce qui n'est pas concluant.

TABLE 3.2 – Base de données transactionnelles $\mathcal{T}$

tid	**Transaction**
1	A B D E F
2	A D E
3	B D E
4	A B C D F G H I J K L M N O P

Supposons maintenant que nous échantillonnons des motifs de norme inférieure ou égale à 2 avec une probabilité proportionnelle à leur fréquence. Dans ce cas, le nombre de sous-ensembles de norme comprise entre 1 et 2 de la transaction d'identifiant 4 est seulement égal à 120. Ainsi, la probabilité de tirer un sous-ensemble de la quatrième transaction sera maintenant égale à $120/(27 + 120) = 81.63\%$, ce qui conduira à retourner moins de motifs peu fréquents issus de cette transaction. La figure 3.5 montre la répartition des motifs avec et sans contrainte de taille maximale.

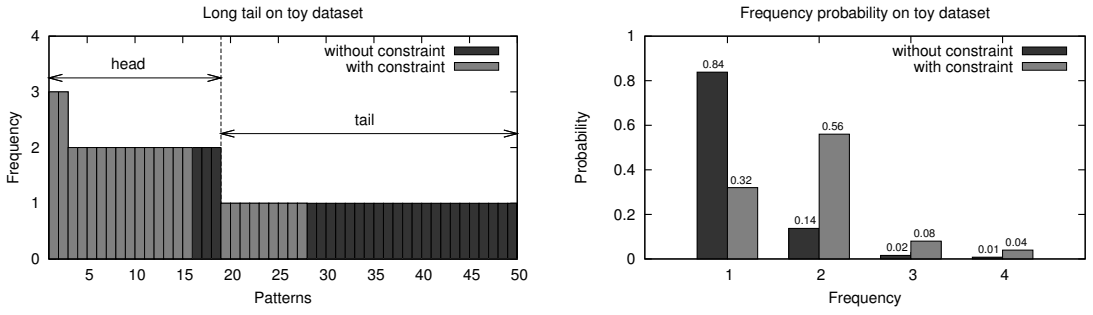


FIGURE 3.5 – Phénomène de la longue traîne sur le jeu de données transactionnelles $\mathcal{T}$

La figure 3.5 montre que 84% des motifs de la base de données $\mathcal{T}$ ont une fréquence égale à 1, ce qui veut dire aussi que les motifs fréquents sont nettement moins représentatifs dans $\mathcal{T}$. Dans ces conditions, l'histogramme de gauche montre qu'une méthode d'échantillonnage en sortie comme celle de [Boley *et al.*, 2011] retourne des motifs ayant une fréquence très faible dans 84% des cas. Par contre, avec la contrainte de norme maximale, nous notons une nette amélioration des résultats. Dans ces conditions, seulement 32% des motifs probables d'être tirés ont une fréquence égale à 1.

3.2.4 Défis pour éviter d'échantillonner des motifs de la traîne

Au vu des exemples de la section précédente, nous pensons ainsi que pour échantillonner des motifs dans une base de données atteinte de la malédiction de la longue traîne, l'une des solutions est de poser des contraintes de norme maximale pour enfin tirer des motifs fréquents (voir chapitre 4 de la contribution). Une autre solution est d'utiliser une fonction à décroissance exponentielle (voir chapitre 5 de la contribution). Dans ce cas, les motifs de norme élevée ont des utilités plus petites que les motifs de petite norme. Dans tous les cas, nous avons besoin de contrebalancer la longueur des motifs afin d'éviter de tirer trop de motifs longs. Pour résoudre ce problème de la longue traîne pour échantillonner des motifs dans une base de données $\mathcal{D}$, nous aurons besoin de relever deux défis :

— Calculer efficacement le poids des motifs du langage $\mathcal{L}$ satisfaisant une contrainte de norme maximale $||\varphi|| \leq M$, avec $\varphi \in \mathcal{L}$, c'est-à-dire la somme des mesures d'intérêt des motifs de la théorie $\mathcal{T}h(\mathcal{L}, \mathcal{D}, ||\varphi|| \leq M)$,

— Tirer un motif φ de cette théorie suivant une probabilité proportionnelle à une mesure d'intérêt m choisie par l'utilisateur : $\pi(\varphi = \mathcal{S}ampl e_1(\mathcal{T}h(\mathcal{L}, \mathcal{D}, q), \mathcal{D}, m)) = m(\varphi, \mathcal{D})/Z$ où Z est une constante de normalisation et q une contrainte de norme.

3.3 Structures de données pour échantillonner des motifs

Cette partie revisite le domaine de l'échantillonnage en sortie de motifs ensemblistes pour la mise en place d'une approche aussi efficace que celle en "deux étapes" de l'algorithme 4 mais qui présente plusieurs avantages.

Dans le domaine de l'extraction de motifs, il est aussi très important de prendre en compte la taille de la base qui peut être un obstacle lorsque les données doivent être chargées en mémoire. En effet, avec les méthodes d'échantillonnage en sortie de l'état de l'art, c'est une nécessité que la base de données entière soit stockée en mémoire. Dans certains jeux de données ayant de très grandes tailles, on peut utiliser une structure de données compacte afin d'avoir une représentation compressée de la base de données comme ça l'était pour l'extraction exhaustive de motifs intéressants [Han *et al.*, 2000]. A notre connaissance, aucune des méthodes d'échantillonnage en sortie de motifs de la littérature [Al Hasan et Zaki, 2009, Boley *et al.*, 2011, Dzyuba *et al.*, 2017] n'a encore été appliquée sur des représentations compactes des bases de données. Pourtant, presque toutes les implémentations proposées ont besoin de stocker toute la base de données en mémoire.

3.3.1 Problème et motivation

Nous pensons qu'avec les gros jeux de données transactionnelles, il est souvent important d'utiliser des structures de données compactes afin que la base de données tienne en mémoire.

On sait que dans les bases de données transactionnelles, il y a beaucoup de répétitions car plusieurs transactions peuvent contenir les mêmes informations. Ces répétitions font sens dans le domaine de la fouille de motifs car elles permettent de trouver des règles intéressantes, mais il faut bien savoir les représenter. Par exemple, dans une base de données transactionnelles, si 90% des transactions qui contiennent les items $\{e_1, e_2, e_3, e_4\}$ contiennent aussi les items $\{e_5, e_6\}$ alors autant les regrouper afin qu'elles partagent le même préfixe. Ceci réduit considérablement la taille de la base en mémoire.

Pour résoudre ce problème, nous pouvons utiliser des structures de données telles que le "FP-Tree" ou le "Trie". La différence est que, contrairement au "trie" qui ne lie un noeud qu'avec ses fils, le "FP-Tree" établit des liens entre des noeuds de branches différentes pour calculer rapidement les fréquences des motifs. Etant donné que nous ne voulons pas calculer des fréquences de motifs, nous proposons d'utiliser le "Trie" pour avoir une représentation compacte de la base de données lorsque celle-ci est en mémoire.

3.3.2 Représentation compacte avec les structures de tries

Dans [Knuth, 1997], l'auteur a proposé une structure de données appelée *trie* (on prononce "try" pour faire la différence avec "tree") permettant d'avoir une représentation compacte des chaînes de caractères.

Définition 19 (Trie)**.** *Un trie est une structure de données ayant la forme d'un arbre enraciné tel que pour tout noeud, ses descendants ont en commun le même préfixe.*

Nous rappelons que cette structure est très utilisée dans le domaine de la fouille de texte [Ferrández et Peral, 2019]. D'autres auteurs ont utilisé les tries pour la fouille de motifs fréquents à l'Apriori [Bodon et Rónyai, 2003]. A travers les travaux précédents, nous notons que le trie est beaucoup plus parcimonieux en coût de stockage dans une base de données transactionnelles que dans une base de données de textes. En effet, la profondeur de l'arbre dépend exclusivement de la taille de la plus longue chaîne. Dans le cas de la fouille de texte, les caractères peuvent se répéter plusieurs fois dans la même chaîne où ils suivent un ordre, ce qui n'est pas le cas avec les transactions. Ces dernières sont considérées comme des ensembles d'éléments, et donc, aucune transaction ne contient d'items en double.

Exemple 17. *Considérons la base de données transactionnelles du tableau 1.3 que nous avons reportée au tableau 3.3.*

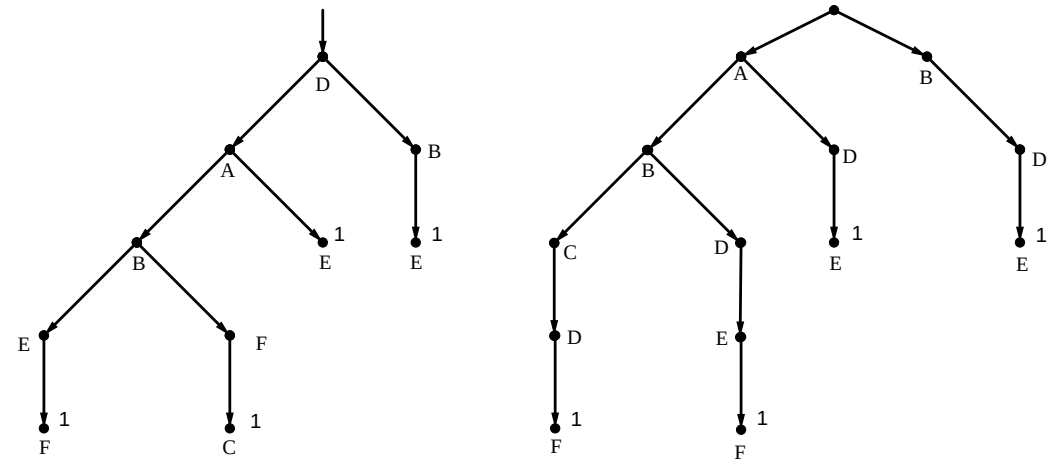
Nous allons représenter la base de données transactionnelles $\mathcal{T}$ sous la forme d'un trie. Pour ce faire, chaque noeud du trie est associé à une valeur qui est le nombre de transactions dont il est le dernier élément selon l'ordre d'insertion des items.

La figure 3.6 montre deux représentations de la base de données $\mathcal{T}$ sous la forme d'un trie. Le trie de la figure 3.6-(a) est obtenu en insérant les items d'une transaction sui-

TABLE 3.3 – Base de données transactionnelles $\mathcal{T}$

tid	Transaction
1	A B D E F
2	A D E
3	B D E
4	A B C D F

vant l'ordre décroissant de leurs fréquences dans la base de données $\mathcal{T}$. Dans 3.6-(b), une transaction est insérée suivant l'ordre lexicographique de ses items.



(a) Ordre décroissant par rapport à la fréquence (b) Pas d'ordre

FIGURE 3.6 – Exemples de trie de la base de données $\mathcal{T}$

Nous notons que la représentation à la figure 3.6-(a) est plus parcimonieuse en coût de stockage car elle compte seulement 10 noeuds alors que la représentation (b) en compte 13. Dans la suite, sauf indication, nous optons pour la représentation 3.6-(a). Ainsi, en appliquant la représentation de la figure 3.6-(a) sur les bases de données `Person` et `Organisation` supposées centralisées, nous avons les statistiques du tableau 3.4 [5].

Les 3 dernières colonnes du tableau 3.4 nous permettent de déterminer la méthode de stockage la plus parcimonieuse entre la représentation des données sous la forme tabulaire (tableau 3.3) et sous la forme d'un trie (figure 3.6-(a)). Comme nous pouvons le voir, la taille mémoire occupée par la base de données `Person` est 44 fois plus petite que la taille occupée par cette même base de données après prétraitement sous la forme tabulaire. Avec la base `Organisation`, la taille est 4 fois plus petite.

5. Taille en mémoire calculée avec la fonction asizeof du package python asizeof http://code.activestate.com/recipes/546530-size-of-python-objects-revised/

TABLE 3.4 – Statistiques des bases de données `Person` et `Organisation` avec et sans trie

$\mathcal{T}$	Taille en mémoire (MB) Trie	Tableau	Gain avec trie
`Organisation`	213.70	954.36	×4
`Person`	69.94	3,121.90	×**44**

3.3.3 Défis de l'échantillonnage en sortie de motifs sur les tries

Comme nous l'avons souligné, le tirage d'un motif est l'une des étapes les plus importantes dans l'échantillonnage en sortie de motifs surtout dans le cas de la fouille centrée sur l'utilisateur. Après avoir construit le trie d'une base de données, l'idée est alors d'en tirer directement un motif sans avoir à construire des étapes intermédiaires (figure 3.7).

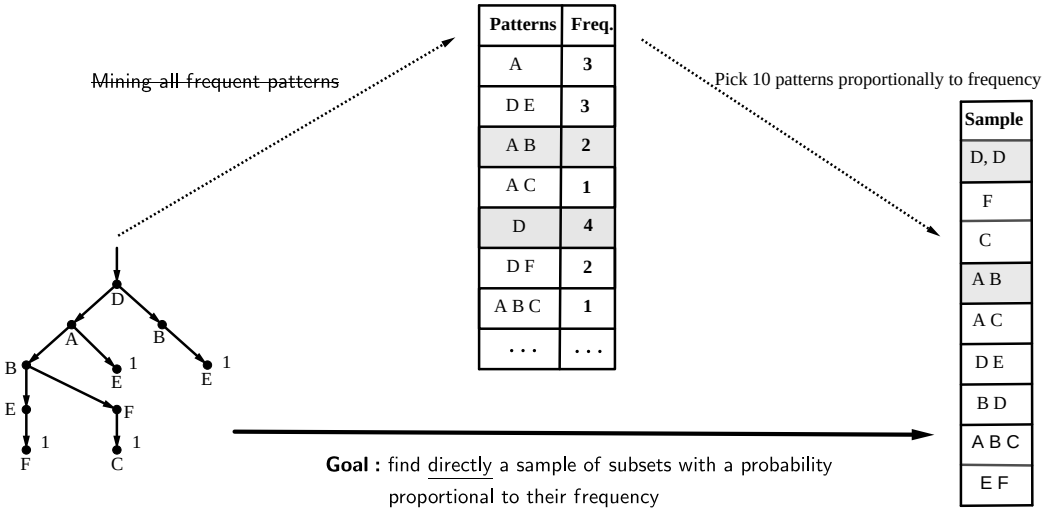


FIGURE 3.7 – L'objectif de l'échantillonnage à partir d'un trie

Pour répondre à ce problème, nous avons besoin de résoudre deux défis :

1. calculer efficacement la somme totale des poids des motifs de la base de données
2. tirer un motif directement à partir du trie suivant une probabilité exactement égale à sa mesure d'intérêt dans la base de données.

A notre connaissance, aucune méthode d'échantillonnage en sortie de l'état de l'art ne pondère les transactions d'une base de données à partir d'un trie ou d'un arbre. Ainsi, elles ne parviennent pas à faire un tirage exact d'un motif à partir du trie. Dans cette thèse, nous allons montrer comment échantillonner des motifs ensemblistes directement à partir d'un trie et proportionnellement à une mesure d'intérêt choisie par l'utilisateur.

On peut noter que le coût de la construction du trie n'est pas négligeable, cependant, il existe souvent un compromis entre le temps de prétraitement et le temps de tirage.

Autrement dit, si le nombre de motifs à échantillonner est très élevé, il est préférable de sacrifier du temps lors du prétraitement que l'on rattrapera lors du tirage des motifs. De plus, le tableau 3.4 montre que le gain en stockage mémoire obtenu en utilisant un trie plutôt qu'une représentation tabulaire peut être vraiment très important. Les algorithmes de construction du trie et de tirage d'un motif à partir du trie seront développés dans le chapitre 7 de la partie contribution.

Etant donné la taille des bases de données `Person` et `Organisation`, qui sont des bases de données distribuées dans le Web, nous pensons qu'il serait finalement plus judicieux de tirer des motifs directement à partir de la base de données distribuée. Autrement dit, il nous faut une méthode d'échantillonnage en sortie dans les bases de données distribuées sans pour autant centraliser les données.

3.4 Problème des bases de données distribuées

La méthode de centralisation des données pour construire $\mathcal{P}^\star$ à la section 3.1 entraîne une perte d'informations car les données sont très souvent mises à jour. Une autre motivation est que, de nos jours, de nombreuses applications telles que les *systèmes de positionnement global*, le *Web des données liées*, les *systèmes de contrôle du trafic aérien*, les *systèmes bancaires automatisés* etc, requièrent un stockage et une manipulation de bases de données distribuées [Özsu et Valduriez, 2011]. Dans notre contexte lié aux données du Web sémantique, une entité est décrite sur plusieurs triplestores. Dans ce cas, il est pertinent d'extraire des motifs sur l'ensemble de la base de données distribuée pour rechercher des corrélations entre les propriétés de triplestores distincts.

Dans le reste de cette partie, nous allons montrer les inconvénients de la centralisation des données dans le cadre des bases de données du Web. Ensuite, nous montrons les limites des méthodes d'extraction exhaustive de motifs ainsi que les méthodes existantes d'échantillonnage en sortie dans des bases de données distribuées. Enfin, nous terminons par présenter les défis pour échantillonner efficacement des motifs dans des bases de données distribuées.

3.4.1 Inconvénients de la centralisation des données distribuées

Une méthode naïve d'échantillonnage en sortie de motifs dans des bases de données distribuées est de centraliser toutes les données puis d'appliquer un algorithme d'échantillonnage en sortie de motifs tel que [Boley *et al.*, 2011]. Cependant, plusieurs problèmes peuvent se poser lors de la centralisation des données. Dans les paragraphes suivants, nous donnons quelques inconvénients de la centralisation d'une base de données distribuées.

Coût de la centralisation Lors de la centralisation des données, le coût de communication ainsi que celui du stockage peuvent être très importants. De plus, cette centralisation peut être impossible à cause de contraintes légales ou techniques. Par exemple dans le cas des données du Web (DBpedia, Wikidata, YAGO, GeoNames, ...), il n'y a certes pas de contraintes légales, mais, la centralisation des données de l'ensemble des triplestores est très coûteuse.

Fraîcheur Si on centralise les données distribuées à un instant θ pour découvrir des informations pertinentes, alors à un instant $\theta + 1$, pour avoir les bonnes informations, on est obligé de centraliser à nouveau les données afin qu'elles soient à jour. Cela est dû au fait qu'a priori, on ne sait pas si les données des sources distantes ont subi des modifications. Par exemple, les informations relatives à certaines entités de DBpedia peuvent être mises à jour au bout d'une semaine. Ce qui implique que ce n'est pas forcément les mêmes informations découvertes il y a de cela une semaine que nous allons de nouveau obtenir. Ce processus de mise à jour est très fréquent dans le domaine des bases de données du Web.

Ainsi, [Zhang et Zaki, 2006] soulignent l'importance d'étendre la découverte de connaissances aux bases de données distribuées.

3.4.2 Fouille de motifs dans des bases de données distribuées

Peu de travaux de la littérature se sont intéressés à la découverte de motifs dans des bases de données distribuées [Cheung *et al.*, 1996, Otey *et al.*, 2003, Jin et Agrawal, 2006, Kum *et al.*, 2006]. Ces propositions se sont focalisées sur une extraction exhaustive des motifs en fusionnant les extractions réalisées localement sur chacun des sites. Malheureusement, le volume de données à transmettre entre les différents sites exige un coût de communication bien supérieur à la centralisation des données car les motifs sont nombreux par nature et les multiples extractions génèrent de multiples doublons. De plus, le coût de calcul de ces extractions parallèles est prohibitif même si des techniques d'élagage les diminuent sensiblement en contrepartie de coûts de communication supplémentaires [Zhu et Wu, 2007, Zhu *et al.*, 2011]. La figure 3.8 donne une vision globale de l'architecture des systèmes classiques de fouille de motifs dans des bases de données distribuées.

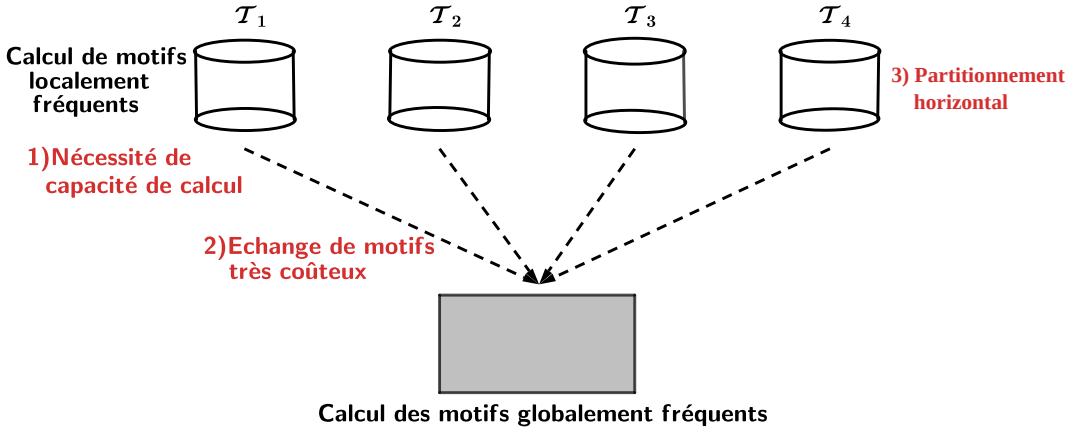


FIGURE 3.8 – Architecture des systèmes classiques de fouille de motifs dans des bases de données distribuées

$\mathcal{T}_1$, ..., $\mathcal{T}_4$ sont des fragments d'une base de données distribuée horizontalement partitionnée et répartie sur 4 sites. Au niveau de chaque site, on peut effectuer des routines de calcul

pour trouver des motifs localement fréquents. Le carré en gris est le site central où par exemple, on peut centraliser tous les motifs extraits à partir des 4 sites.

L'extraction de motifs fréquents sur une base de données distribuée est une tâche très complexe car quel que soit la fréquence exigée sur la base de données distribuée (fréquence globale), il n'est pas possible de contraindre la fréquence locale sur un fragment sans communiquer des informations entre sites. Dans ce cadre, [Cheung *et al.*, 1996] proposent le premier travail pour extraire tous les motifs globalement fréquents en identifiant les sites où les motifs sont les plus fréquents et en réduisant ainsi un peu les échanges. De manière plus drastique, [Otey *et al.*, 2003] proposent d'économiser les échanges en se limitant à la collection des motifs fréquents maximaux. Pour ne pas avoir à énumérer tous les motifs de chaque fragment et limiter les échanges, [Jin et Agrawal, 2006] imposent un seuil minimal de fréquence sur chaque fragment. A partir des différentes extractions locales, [Kum *et al.*, 2006] construisent une collection globale approchée des motifs fréquents. Un élagage centralisé proposé par [Zhu et Wu, 2007] repose sur la construction d'un arbre contenant pour chaque motif toutes ses occurrences (i.e., couples fragment/transaction), ce qui requiert encore un volume d'échanges considérable. Plus récemment, [Zhu *et al.*, 2011] parviennent à mettre en oeuvre un élagage décentralisé au sein de l'extraction de chaque fragment en échangeant des filtres de Bloom. Cette approche réduit significativement les temps de calcul mais le coût des communications amoindri demeure important. En effet, le volume de motifs extraits génère invariablement un coût de communications énorme bien supérieur à celui de la centralisation des données. En outre, toutes ces approches d'extraction de motifs fréquents se restreignent à un partitionnement horizontal des données, une même transaction ne pouvant pas être distribuée sur deux fragments distincts. Enfin, les propositions de l'état de l'art requièrent d'avoir une capacité de calcul importante sur chaque fragment ce qui n'est pas toujours possible.

3.4.3 Défis de l'échantillonnage en sortie dans des bases de données distribuées

Nous notons ainsi que l'architecture présentée à la figure 3.8 est très limitée. En plus de l'impossibilité du calcul des motifs localement fréquents, le calcul des motifs globalement fréquents requiert un coût de communication très élevé et génère trop de redondances sur les motifs extraits. En troisième lieu, le partitionnement des données n'est pas toujours horizontal, c'est le cas des données du Web où une entité peut être décrite par plusieurs triplestores. La figure 3.9 nous montre l'architecture d'un système de fouille de motifs dans le Web des données liées.

Dans cette figure, nous avons remplacé les fragments de la figure 3.8 par les triplestores, cas où une entité est décrite par plusieurs fragments. Par exemple, la figure 3.1 illustre le cas du Web Sémantique en fournissant un exemple de données RDF (Resource Description Framework) distribuées sur plusieurs triplestores $\mathcal{P} = \{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4\}$ accessible via des requêtes SPARQL. Dans ce contexte, les propriétés décrivant l'entité d'identifiant 4 (i.e., le marabout "Cheikh Amadou Bamba") sont réparties sur plusieurs fragments (i.e., DBpedia $\mathcal{T}_1$ contient les items B et C, Wikidata $\mathcal{T}_2$ les items A et F, et enfin YAGO $\mathcal{T}_4$ l'item D).

Notons finalement que si le web sémantique offre un accès à des données distribuées via une requête SPARQL il n'est pas possible d'y exécuter une routine d'extraction. Et

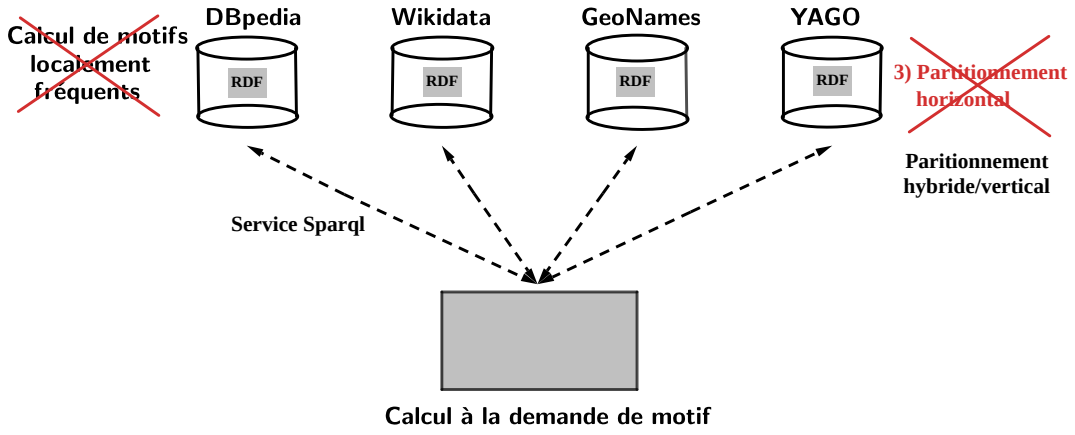


FIGURE 3.9 – Architecture d'un système de fouille de motifs dans le Web des données liées

malheureusement, SPARQL n'est pas assez expressif pour extraire directement des motifs tels que les itemsets fréquents. De ce fait, pour explorer ces bases de données, nous avons besoin de relever trois défis :

— proposer des primitives très simples et parcimonieuses en coût de communication pour interroger les bases de données quel que soit leur partitionnement

— réduire le coût de communication en bénéficiant de l'échantillonnage en sortie de motifs

— proposer un modèle d'échantillonnage qui s'applique à tout type de partitionnement d'une base de données distribuée.

3.5 Conclusion

En somme, nous avons constaté que les méthodes d'échantillonnage en sortie ont apporté une plus-value dans le domaine de la fouille de motifs. Elles sont souvent très rapides, offrent plus de possibilités pour contrôler le nombre de motifs produits et une bonne diversité entre les motifs retournés du fait de sa nature aléatoire.

Cependant, il reste encore beaucoup de pistes à explorer. Plusieurs points très importants de la fouille de motifs ne sont pas jusque là résolus par les méthodes de l'état de l'art de l'échantillonnage en sortie. En voici quatre :

1. Le premier point que nous avons remarqué est que jusque là, aucune méthode d'échantillonnage n'a encore été proposée pour les bases de données séquentielles (voir chapitre 5). Et pourtant, beaucoup d'applications de la vie réelle utilisent les données séquentielles.
2. Le second problème est que les méthodes d'échantillonnage en sortie peuvent retourner des motifs peu intéressants, du fait du problème de la longue traîne. Ce

problème sera résolu dans toutes nos contributions, en particulier par l'utilisation de contraintes sur les normes des motifs échantillonnés.

3. Toutes les méthodes d'échantillonnage en sortie que nous avons vues jusqu'ici, ont besoin que les données soient en mémoire pour que la phase de tirage des motifs soit rapide. Pourtant, aucune d'elles ne se soucie du problème de stockage qui est souvent un obstacle. Le chapitre 7 montre comment il est possible d'échantillonner efficacement des motifs avec les trie.
4. Enfin, un autre problème non résolu jusqu'ici par les méthodes d'échantillonnage en sortie de motifs de l'état de l'art concerne le cas des bases de données distribuées (voir chapitre 6). La résolution de ce cas impose l'extension des méthodes existantes pour faire de l'échantillonnage en sortie de motifs dans des bases de données distribuées.

Tous ces problèmes doivent être résolus pour mieux valoriser l'importance de l'échantillonnage en sortie car ils font partie des défis au coeur de la fouille de motifs.

Deuxième partie

Echantillonnage sous contraintes de motifs structurés

Introduction

La seconde partie de ce mémoire présente nos différentes contributions au cours de cette thèse. Nous apportons précisément une solution à chaque défi posé au chapitre 3 à partir de la page 75. Rappelons que les défis à relever dans l'algorithme 3 (voir page 62) sont la pondération d'une instance et le tirage d'un motif. En se basant sur ces deux défis, nous avons jugé intéressant de proposer une classe de mesures d'intérêt générique et applicable dans la pratique et qui se base sur les normes des motifs. Contrairement aux méthodes d'échantillonnage en sortie de motifs de la littérature où les utilités des généralisations d'une instance sont calculées séparément, la classe que nous avons proposée dans nos contributions considère que toutes les généralisations de même norme au sein d'une instance ont la même utilité. Cette classe de mesures d'intérêt fondées sur la norme notée par $\mathcal{M}$ constitue notre cadre pour l'échantillonnage en sortie de motifs sous contrainte faisant face à la malédiction de la longue traîne (voir section 3.2). De ce fait, les chapitres développés dans cette partie visent à échantillonner des motifs suivant une distribution proportionnelle à une mesure d'intérêt fondée sur la norme mais choisie librement par l'utilisateur afin d'assurer une bonne interaction avec le système.

Afin de montrer l'applicabilité de notre classe de mesures d'intérêt fondées sur la norme, nous allons commencer par la formaliser au chapitre 4, puis nous proposons un algorithme générique qui montre comment l'implémenter efficacement en se basant sur les deux étapes de l'algorithme 3. Pour illustrer simplement cet algorithme générique, nous l'instancions immédiatement sur les itemsets [Boley *et al.*, 2011] en utilisant notre classe de mesures d'intérêt fondées sur la norme. Les résultats très intéressants que nous avons obtenus avec la classe de mesures d'intérêt fondées sur la norme face à l'écueil de la longue traîne dans les données ensemblistes nous motivent à étendre notre approche à d'autres structures de données plus complexes. Dans cette lancée, le chapitre 5 présente une instanciation de l'algorithme 4 sur les bases de données séquentielles. En outre, il mène une étude théorique et pratique de la méthode montrant que l'approche proposée pour l'échantillonnage en sortie de motifs séquentiels est très efficace.

Toujours soucieux de l'immense importance de notre classe de mesures d'intérêt fondées sur la norme et de son efficacité, nous proposons au chapitre 6 une méthode d'échantillonnage en sortie de motifs à partir d'une base de données distribuée. Son efficacité se démontre par le fait que seules les normes des transactions sont stockées localement en mémoire, ce qui fait que notre approche est très parcimonieuse en coût de communication.

Seulement les items nécessaires pour la construction du motif tiré sont rapatriés lors de l'échantillonnage.

Il est aussi important de noter que dans certains cas où la base de données est stockée localement, sa taille peut être trop grande pour qu'elle soit entièrement chargée en mémoire principale. Dans ces conditions, il est préférable d'utiliser une structure de données compacte telle que les tries afin de réduire considérablement la taille de la base de données. De ce fait, nous aurons besoin de tirer des motifs à partir de la forme compacte de la base de données, et donc de pondérer les occurrences de motifs dans le trie. Avec la classe de mesures d'intérêt fondées sur la norme, on peut facilement regrouper les occurrences de même norme afin de faciliter la phase de pondération. Le chapitre 7 présente une approche efficace pour l'échantillonnage en sortie de motifs ensemblistes directement à partir du trie dans le cadre de la classe de mesures d'intérêt fondées sur la norme.

Chapitre 4

Echantillonnage de motifs fondés sur la norme

Sommaire

Dans ce chapitre, nous allons présenter une classe de mesures d'intérêt fondées sur la norme pour faire face à la malédiction de la longue traîne afin d'échantillonner efficacement des motifs représentatifs et ayant une très bonne diversité. Pour valoriser cette classe de mesures d'intérêt, nous proposons un algorithme générique qui prend en compte tout type de mesure d'intérêt fondée sur la norme afin d'échantillonner des motifs suivant une probabilité proportionnelle à leurs mesures d'intérêt. En ce qui concerne cette classe de mesures d'intérêt fondées sur la norme, nous présentons nos travaux liés aux bases de données transactionnelles à la section 4.4 de ce chapitre et au chapitre 6 (pour les bases distribuées), mais aussi aux bases de données séquentielles au chapitre 5. Le chapitre 7 montre comment traiter cette classe de mesures d'intérêt fondées sur la norme avec les tries.

Nos principales contributions dans ce chapitre sont les suivantes :

— D'abord, nous proposons une classe de mesures d'intérêt fondées sur la norme notée par $\mathcal{M}$. Cette classe de mesures d'intérêt fondées sur la norme nous permet de surmonter le phénomène de la longue traîne que nous avons présenté à la section 3.2.

— Ensuite, nous proposons un algorithme générique d'échantillonnage en sortie de motifs en deux étapes prenant en compte toute mesure d'intérêt fondée sur la norme. Notons que cet algorithme est une instance de l'algorithme 3 qui, tel que nous l'avons écrit à la section 2.2.3, n'est pas utilisable en pratique du fait de sa complexité. Dans l'algorithme que nous allons proposé dans ce chapitre, la phase de pondération ainsi que celle du tirage d'un motif se font plus efficacement.

— Enfin, nous donnons un cas d'usage de notre algorithme générique dans le cas des bases de données transactionnelles. Nous évaluons sa complexité en temps et puis nous montrons l'intérêt de la contrainte de taille maximale pour échapper à la malédiction de la longue traîne afin de construire des classifieurs de données ensemblistes.

Dans la suite de ce chapitre, la section 4.1 formalise le problème général ainsi que notre classe de mesures d'intérêt fondées sur la norme. La section 4.2 présente les défis que nous devons résoudre pour que l'algorithme 3 puisse prendre efficacement en compte notre classe de mesures d'intérêt fondées sur la norme. La section 4.3 présente l'algorithme générique et démontre l'exactitude du tirage. La section 4.4 présente un cas d'usage de l'algorithme générique sur les bases de données ensemblistes et évalue sa complexité en temps. Enfin, la section 4.5 conclue ce chapitre.

4.1 Problématique et mesures d'intérêt fondées sur la norme

A la section 3.2, nous avons montré comment la longue traîne se manifeste sur une base de données, et puis, nous avons proposé quelques défis à la section 3.2.4 pour résoudre le problème posé. Pour aborder le premier défi qui consiste à poser des contraintes de norme maximale sur les motifs retournés, nous proposons des mesures d'intérêt fondées sur la norme. Rappelons qu'à la section 1.1.3, nous avons dit que la mesure d'aire $area(\varphi, \mathcal{D}) = freq(\varphi, \mathcal{D}) \times \|\varphi\|$ est basée sur la norme des motifs. Pour cette raison, nous nous intéressons à la classe des mesures d'intérêt de la forme $freq(\varphi, \mathcal{D}) \times u(\varphi)$ où u est une utilité fondée sur la norme :

Définition 20 (Utilité fondée sur la norme)**.** *Une fonction utilité (ou simplement une utilité) u est dite fondée sur la norme s'il existe une fonction $f_u : \mathbb{N} \to \mathbb{R}$ telle que pour tout motif $\varphi \in \mathcal{L}$ et $\gamma \in \mathcal{D}$, on a $u(\varphi, \gamma) = f_u(\|\varphi\|)$.*

L'ensemble de toutes les utilités fondées sur la norme est désigné par $\mathcal{U}$.

D'après la définition 14 du chapitre 2, une mesure d'intérêt dans le cadre de l'échantillonnage en deux étapes est définie comme suit :

$$m(\varphi, \mathcal{D}) = \sum_{\gamma \in \mathcal{D} \wedge \varphi \preceq \gamma} u(\varphi, \gamma).$$

où $u(\varphi, \gamma)$ représente l'utilité du motif φ dans l'instance γ.

Lemme 1. *Soient $\mathcal{D}$ une base de données et m une mesure d'intérêt associée à une fonction d'utilité fondée sur la norme. Alors pour tout motif φ du langage de motifs de $\mathcal{D}$, on a :*

$$m(\varphi, \mathcal{D}) = f_u(\|\varphi\|) \times freq(\varphi, \mathcal{D}).$$

Preuve. On sait que $m(\varphi, \mathcal{D}) = \sum_{\gamma \in \mathcal{D} \wedge \varphi \preceq \gamma} u(\varphi, \gamma)$. Etant donné que u est fondée sur la norme, alors $u(\varphi, \gamma) = f_u(\|\varphi\|)$. D'après la définition 20, nous avons $m(\varphi, \mathcal{D}) = \sum_{\gamma \in \mathcal{D} \wedge \varphi \preceq \gamma} f_u(\|\varphi\|) = f_u(\|\varphi\|)(\sum_{\gamma \in \mathcal{D} \wedge \varphi \preceq \gamma} 1)$. Donc une mesure d'intérêt fondée sur la norme peut s'écrire comme suit : $m(\varphi, \mathcal{D}) = f_u(\|\varphi\|) \times \mathit{freq}(\varphi, \mathcal{D})$. D'où le résultat. □

L'ensemble des mesures d'intérêt fondées sur la norme est dénoté par $\mathcal{M}$. Par exemple, l'utilité $u_{area}(\varphi) = \|\varphi\|$ nous permet de considérer la mesure d'aire formellement définie par $\mathit{area}(\varphi, \mathcal{D}) = \mathit{freq}(\varphi, \mathcal{D}) \times \|\varphi\|$ et dans ce cas, on a $f_{u_{area}}(\ell) = \ell$. Notons que l'utilité fondée sur la norme $u_{\mathit{freq}}(\varphi) = 1$ nous permet de considérer la fréquence comme mesure d'intérêt. De plus, l'utilité $u_{\leq M}$ (resp. $u_{\geq m}$) définie comme égale 1 si $\|\varphi\| \leq M$ (resp. $\|\varphi\| \geq m$) et 0 sinon, simule une contrainte de norme maximale (resp. minimale). En effet, avec la mesure d'intérêt induite $\mathit{freq}(\varphi, \mathcal{D}) \times u_{\leq M}(\varphi)$ (resp. $\mathit{freq}(\varphi, \mathcal{D}) \times u_{\geq m}(\varphi)$), un motif de norme strictement supérieure à M (resp. inférieure à m), est jugé inutile (quel que soit sa fréquence). On dit que $\geq m$ et $\leq M$ sont des contraintes de norme (où 1 signifie *vrai* et 0 signifie *faux*). Enfin, l'utilité $u_{\mathit{decay}}(\varphi) = \alpha^{\|\varphi\|}$, avec $\alpha \leq 1$, appelée décroissance exponentielle, est utile pour pénaliser de longs motifs mais de manière continu par rapport à $u_{\leq M}$.

D'une manière intéressante, il est possible de combiner des utilités fondées sur la norme grâce à des opérations arithmétiques :

Propriété 3 (Fermeture arithmétique)**.** *La classe des utilités fondées sur la norme est fermée sous des opérations arithmétiques, c'est-à-dire que $u_1 \star u_2$ est une utilité fondée sur la norme si $u_1 \in \mathcal{U}$, $u_2 \in \mathcal{U}$ et $\star \in \{+, -, \times, /\}$.*

Cette propriété simple nous permet de combiner plusieurs utilités fondées sur la norme pour construire des utilités plus complexes. Par exemple, si q_1 et q_2 sont deux contraintes d'utilités fondées sur des normes (c'est-à-dire des utilités fondées sur la norme avec $\{0, 1\}$ comme valeurs possibles), alors $q_1 \wedge q_2$ et $q_1 \vee q_2$ sont également des utilités fondées sur des normes car $q_1 \wedge q_2 = q_1 \times q_2$ et $q_1 \vee q_2 = q_1 + q_2 - q_1 \times q_2$.

Par conséquent, dans ce qui suit, nous considérons l'utilité fondée sur la norme $u_{[m..M]} = u_{\geq m} \times u_{\leq M}$ pour concentrer le tirage sur les motifs de la théorie $\mathcal{T}h(\mathcal{L}, \mathcal{D}, m \leq \|\varphi\| \leq M)$. C'est l'ensemble de tous les motifs ayant une norme entre m et M et il est dénoté par $\mathcal{L}_{[m..M]}$. Ainsi, pour éviter de tirer les motifs trop rares et trop spécifiques de la longue traîne, notre proposition et problème consiste à tirer des motifs de $\mathcal{L}_{[m..M]}$ par rapport à une mesure d'intérêt $m \in \mathcal{M}$: $\varphi \sim m(\mathcal{L}_{[m..M]}, \mathcal{D})$ afin de contrôler leurs normes minimale et maximale.

Le problème spécifique que nous voulons résoudre ici peut finalement se formuler comme suit :

Soient $\mathcal{D}$ une base de données, u une fonction d'utilité fondée sur la norme et k un entier naturel strictement positif. Notre objectif est de calculer un échantillon de motifs $\mathcal{S}\mathit{ample}_k(\mathcal{L}, \mathcal{D}, \mathit{freq} \times u)$.

4.2 Défis de l'algorithme générique

Pour échantillonner des motifs à l'aide de l'algorithme 3 suivant une mesure d'intérêt fondée sur la norme, nous avons besoin de relever deux défis.

— **Calcul efficace des poids des instances :** à la ligne 3 de l'algorithme 3, la complexité de la méthode de pondération est exponentielle car cela nécessite d'énumérer tous les motifs du langage inclus dans une instance de la base de données. Dans ce cas, la classe de mesures d'intérêt fondées sur la norme que nous avons proposée peut être utilisée efficacement grâce à aux lemmes 2 et 3. En effet, si l'utilité est fondée sur la norme, alors dans une même instance, tous les motifs de même norme ont la même utilité. Donc le problème d'énumération de l'ensemble des généralisations d'une instance γ, $\phi(\gamma)$ de la ligne 3 de l'algorithme 3 est finalement réduit en un problème de comptage du nombre de généralisations de même norme ℓ de l'instance γ, $\Phi_\ell(\gamma)$.

— **Tirage rapide d'un motif :** la ligne 5 de l'algorithme 3 montre que la complexité de tirage d'un motif est aussi exponentielle car il faut lister toutes les généralisations de l'instance sélectionnée à la ligne 4. Avec la classe de mesures d'intérêt fondées sur la norme, nous verrons que cette étape peut être scindée en deux sous-étapes : (i) tirer une norme ℓ proportionnellement à la somme des utilités des générations de l'instance de norme ℓ, (ii) tirer uniformément une généralisation de norme ℓ d'une instance parmi l'ensemble de ses généralisations de norme exactement égale à ℓ.

Dans la suite, nous allons montrer comment formaliser efficacement le calcul des poids des instances et ceux des normes des motifs dans l'instance par rapport à une mesure d'intérêt fondée sur la norme. Ensuite, nous proposons un algorithme générique dédié à l'échantillonnage en sortie de motifs prenant en compte toute mesure d'intérêt fondée sur la norme.

4.3 Algorithme générique d'échantillonnage en sortie

Comme nous l'avons dit au début de ce chapitre, l'algorithme 4 est une instanciation de l'algorithme 3 qui s'applique à toute mesure d'intérêt fondée sur la norme. Dans [Diop *et al.*, 2019a], nous avons reformulé l'algorithme 3 de tirage en deux étapes de [Boley *et al.*, 2011] dédié à la mesure d'aire pour toute mesure d'intérêt fondée sur la norme $m \in \mathcal{M}$ afin de tirer des motifs de $\mathcal{L}$ suivant une distribution proportionnelle à une mesure d'intérêt fondée sur la norme. Rappelons que cet algorithme s'applique sur des bases de données locales qu'elles soient transactionnelles (section 4.4) ou séquentielles (chapitre 5).

4.3.1 Solution en deux étapes

Avant de tirer une instance avec une probabilité proportionnelle à son poids $\omega_u(\gamma)$, il faudrait au préalable calculer le poids de chaque instance. Bien sûr, il n'est pas possible de calculer ce poids pour chaque instance γ en sommant naïvement un à un le poids de

chaque généralisation $\varphi \preceq \gamma$ comme le suggère la ligne 3 de l'algorithme 3. En fait, il est plus efficace de décomposer ce poids selon la norme des généralisations en observant que $\omega_u(\gamma) = \sum_{\ell=0}^{\|\gamma\|} \omega_u^\ell(\gamma)$. En effet, comme toutes les généralisations de la même norme ont la même utilité, il suffit de compter le nombre de généralisations de norme ℓ et de multiplier cette quantité par l'utilité $f_u(\ell)$. Le lemme suivant formalise cette intuition :

Lemme 2 (Calcul de $\omega_u^\ell(\gamma)$)**.** *Etant donné une instance γ et une utilité fondée sur la norme u, le poids des généralisations de γ ayant ℓ comme norme, noté $\omega_u^\ell(\gamma)$, est défini par :*

$$\omega_u^\ell(\gamma) = \sum_{\varphi \preceq \gamma \wedge \|\varphi\| = \ell} u(\varphi) = \Phi_\ell(\gamma) \times f_u(\ell).$$

Preuve. En utilisant la définition 20, nous avons $\sum_{\varphi \preceq \gamma \wedge \|\varphi\| = \ell} u(\varphi) = \sum_{\varphi \preceq \gamma \wedge \|\varphi\| = \ell} f_u(\|\varphi\|)$ $= \sum_{\varphi \preceq \gamma \wedge \|\varphi\| = \ell} f_u(\ell) = (\sum_{\varphi \preceq \gamma \wedge \|\varphi\| = \ell} 1) \times f_u(\ell)$. Nous concluons que le lemme 2 est correcte car $\sum_{\varphi \preceq \gamma \wedge \|\varphi\| = \ell} 1$ est exactement $\Phi_\ell(\gamma)$ le nombre de généralisations de γ de norme ℓ. □

Lemme 3 (Poids d'une instance)**.** *Etant donné une instance γ et une utilité fondée sur la norme u, le poids de γ peut s'écrire comme suit :*

$$\omega_u(\gamma) = \sum_{\ell=0}^{\|\gamma\|} \omega_u^\ell(\gamma) = \sum_{\ell=0}^{\|\gamma\|} \Phi_\ell(\gamma) \times f_u(\ell)$$

Preuve. On sait que $\omega_u(\gamma) = \sum_{\ell=0}^{\|\gamma\|} \omega_u^\ell(\gamma)$, c'est la somme des utilités des généralisations de γ. Or d'après le lemme 2, on a $\omega_u^\ell(\gamma) = \Phi_\ell(\gamma) \times f_u(\ell)$. Donc $\omega_u(\gamma) = \sum_{\ell=0}^{\|\gamma\|} \Phi_\ell(\gamma) \times f_u(\ell)$. D'où le résultat. □

L'algorithme 4 s'applique sur une base de données $\mathcal{D}$ et une mesure d'intérêt fondée sur la norme. La première phase consiste à calculer le poids de chaque instance γ de $\mathcal{D}$ en sommant les utilités des motifs plus généraux que γ (ligne 3). Ensuite, vient la phase de tirage d'un motif.

Etape 1 : elle consiste à tirer au hasard une instance γ de la base de données déjà pondérée proportionnellement à son poids $\omega_u(\gamma)$ (ligne 4).

Etape 2 : d'abord, la ligne 5 détaille le poids $\omega_u(\gamma)$ afin de connaître pour chaque norme ℓ, le poids de toutes les généralisations de γ qui ont exactement cette norme ℓ. La ligne 6 utilise cette distribution pour tirer au hasard une norme ℓ proportionnellement à son poids $\omega_u^\ell(\gamma)$. Enfin, il suffit de retourner un motif tiré uniformément parmi l'ensemble des motifs de l'instance γ qui ont une norme exactement égale à ℓ (lignes 7 et 8).

Cet algorithme générique peut être implémenté efficacement pour différents langages $\mathcal{L}$ comme les itemsets [Diop *et al.*, 2019c] à la section 4.4 ou les sous-séquences [Diop *et al.*, 2018b] au chapitre 5. La difficulté est, en premier lieu, de calculer $\omega_u^\ell(\gamma)$ sans énumérer tous les motifs au sein de l'instance γ qui ont une norme ℓ. En second lieu, il faut aussi savoir tirer uniformément un motif de norme ℓ parmi l'ensemble des motifs de norme ℓ d'une instance donnée. Comparé à l'algorithme 3, au lieu d'être exponentiel par rapport à la taille du langage, nous verrons alors que le calcul des poids peut se faire en temps polynomial pour les bases transactionnelles et séquentielles (si les séquences considérées sont uniquement des items).

Algorithm 4 Echantillonnage en deux étapes selon une mesure d'intérêt fondée sur la norme

1: **Input :** Une base de données $\mathcal{D}$ et une fonction d'utilité fondée sur la norme $u \in \mathcal{U}$
2: **Output :** Un motif tiré aléatoirement $\varphi \sim \pi(\mathcal{L})$ où $\pi(\varphi) = freq(\varphi, \mathcal{D}) \times f_u(\|\varphi\|)/Z$
Phase de prétraitement
3: Soient les poids ω_u définis par $\omega_u(\gamma) \leftarrow \sum_{\ell=0}^{||\gamma||} \Phi_\ell(\gamma) \times f_u(\ell)$ pour tout γ de $\mathcal{D}$
Phase d'échantillonnage
Etape 1 : tirage d'une instance
4: $\gamma \sim \omega_u(\mathcal{D})$ ▷ Tirer une instance γ proportionnellement à son poids $\omega_u(\gamma)$ dans $\mathcal{D}$
Etape 2 : tirage d'un motif
5: Soient les poids définis par $\omega_u^\ell(\gamma) \leftarrow \Phi_\ell(\gamma) \times f_u(\ell)$ pour tout $\ell \in [0..\|\gamma\|]$
6: $\ell \sim \omega_u^{[0..\|\gamma\|]}(\gamma)$ ▷ Tirer un entier ℓ proportionnellement à son poids $\omega_u^\ell(\gamma)$ dans γ
7: $\varphi \sim unif(\{\varphi \preceq \gamma : \|\varphi\| = \ell\})$ ▷ Tirer un motif φ de norme ℓ de γ où *unif* est une distribution uniforme
8: **return** φ

4.3.2 Analyse théorique de la méthode

Le lemme 4 montre que la méthode de tirage de l'algorithme 4 est exacte.

Propriété 4. *Soient $\mathcal{D}$ une base de données et u une utilité fondée sur la norme, l'algorithme 4 effectue le tirage d'un motif φ de $\mathcal{L}$ proportionnellement à $freq(\varphi, \mathcal{D}) \times u(\varphi)$.*

Preuve. Soit Z la constante définie par $Z = \sum_{\gamma \in \mathcal{D}} \omega_u(\gamma)$, φ un motif du langage $\mathcal{L}$ et $\pi(\varphi)$ la probabilité de tirer le motif φ à l'aide de l'algorithme 4. Alors on a $\pi(\varphi) = \sum_{\gamma \in \mathcal{D}} \pi(\varphi, \gamma) = \sum_{\gamma \in \mathcal{D} \wedge \varphi \preceq \gamma} \pi(\gamma) \times \pi(\varphi/\gamma)$. Or d'après la ligne 4, $\pi(\gamma) = \frac{\omega_u(\gamma)}{Z}$. D'après les lignes 5 et 6, si φ est une généralisation de γ de norme égale à ℓ, alors nous avons $\pi(\varphi/\gamma) = \pi(\ell/\gamma) \times \pi(\varphi/\ell, \gamma) = \frac{\Phi_\ell(\gamma) \times f_u(\|\varphi\|)}{\omega_u(\gamma)} \times \frac{1}{\Phi_\ell(\gamma)} = \frac{f_u(\|\varphi\|)}{\omega_u(\gamma)}$. Alors $\pi(\varphi) = \sum_{\gamma \in \mathcal{D} \wedge \varphi \preceq \gamma} \frac{f_u(\|\varphi\|)}{Z} = freq(\varphi, \mathcal{D}) \times f_u(\|\varphi\|)/Z$. D'où le résultat. □

4.4 Instanciations aux bases de données transactionnelles

Pour instancier l'algorithme 4 aux bases de données transactionnelles, nous avons besoin de trouver la formule de calcul du nombre de généralisations de même norme dans une transaction donnée. Autrement dit, il nous faut trouver la formule de Φ_ℓ dans les expressions des lemmes 2 et 3 pour les lignes 3 et 5 de l'algorithme 4.

4.4.1 Pondération des transactions

Dans cette section, nous montrons comment calculer efficacement les poids des transactions en prenant en compte que les fonctions d'utilités considérées sont fondées sur la norme. D'après le lemme 3, on a $\omega_u(t) = \sum_{\ell=0}^{\|\gamma\|} \Phi_\ell(t) \times f_u(\ell)$ où $\omega_u(t)$ est le poids d'une

transaction t pour une fonction d'utilité u. Dans le cas d'une base de données transactionnelles, le nombre de motifs de même norme ℓ contenus par une transaction t est le nombre de combinaisons possibles de ℓ items pris dans l'ensemble des items de t. Formellement, nous avons $\Phi_\ell(t) = \binom{||t||}{\ell}$. Donc les formules cherchées pour la pondération des instances d'une base de données transactionnelles peuvent finalement s'écrire comme suit :

$$\omega_u(t) = \sum_{\ell=0}^{||t||} \binom{||t||}{\ell} \times f_u(\ell) \quad et \quad \omega_u^\ell(\gamma) = \binom{||t||}{\ell} \times f_u(\ell)$$

De plus, si la fonction d'utilité u est une fonction d'utilité incorporant des contraintes de taille minimale m et maximale M sur la taille des motifs extraits, i.e. u est de la forme $u' \times u_{\geq m} \times u_{\leq M}$, alors nous avons $\omega_u(t) = \sum_{\ell=m}^{min(M,||t||)} \omega_u^\ell(t)$.

Exemple 18. *Considérons l'exemple de base de données jouet $\mathcal{T}$ de 4 transactions construites à partir de $\mathcal{I} = \{A, ..., P\}$ du tableau 4.1. Soit l'utilité $u_{[1..2]} = u_{\geq 1} \times u_{\leq 2}$. Dans ce cas, nous avons par exemple $\omega_u(t_1) = \sum_{\ell=m}^{min(M,||t||)} \omega_u^\ell(t_1) = \sum_{\ell=1}^{min(2,5)} \binom{5}{\ell} = \binom{5}{1} + \binom{5}{2} = 5 + \frac{5!}{(5-2)!2!} = 5 + 10 = 15$. De même, pour $u = u_{area} \times u_{\geq 1} \times u_{\leq 2}$, nous avons $\omega_u(t_1) = \sum_{l=1}^{min(2,||t||)} \omega_u^\ell(t_1) = \sum_{\ell=1}^{min(2,5)} \binom{5}{\ell} \times \ell = \binom{5}{1} \times 1 + \binom{5}{2} \times 2 = 5 \times 1 + 10 \times 2 = 25$. Des exemples plus nombreux de poids sont présentés dans le tableau 4.1.*

TABLE 4.1 – Exemple d'une base de données transactionnelles pondérée

tid	Transaction t	$\|\|t\|\|$	$\omega_{freq}(t)$	$\omega_{u_{[1..2]}}(t)$	$\omega_{u_{[1..2]} \times u_{area}}(t)$
1	A B D E F	5	31	15	25
2	A D E	3	7	6	9
3	B D E	3	7	6	9
4	A B C D F G H I J K L M N O P	15	32767	120	225

4.4.2 Complexité théorique

Pour calculer la complexité temporelle de l'algorithme 4 sur une base de données transactionnelles définie sur un ensemble de littéraux $\mathcal{I}$, nous distinguons une complexité pour le pré-traitement de la base et celle d'un tirage uniforme d'un motif. Soient μ et M des contraintes de normes minimale et maximale. Notons d'abord que le nombre de sous-ensembles de norme ℓ d'une transaction est obtenu au pire des cas en $O(|\mathcal{I}|)$. Sans utiliser des contraintes de normes minimale et maximale, on pondère d'abord chaque transaction de la base de données par la somme des utilités des sous-ensembles qu'elle contient en $O(|\mathcal{I}|^2)$. Donc la pondération de la base de données est faite en $O(|\mathcal{T}| \times |\mathcal{I}|^2)$. Ensuite, on tire une transaction de $\mathcal{T}$ proportionnellement à son poids en $O(\log(|\mathcal{T}|))$. Enfin, on tire la norme ℓ du motif à retourner en $O(\log(|\mathcal{I}|))$ puis un motif de la transaction précédemment tirée est uniformément retourné parmi l'ensemble de ses motifs de norme ℓ en $O(\mathcal{I})$. Ainsi, pour un échantillon de k motifs, la complexité totale est en $O(|\mathcal{T}| \times |\mathcal{I}|^2 + k(\log(|\mathcal{T}|) + \log(|\mathcal{I}|) + |\mathcal{I}|))$.

Si en plus des contraintes de normes minimale μ et maximale M sont utilisées, alors le temps de pondération d'une transaction est en $O(|\mathcal{I}| \times M_{diff})$ avec $M_{diff} = M - \mu$. Ce qui fait la complexité pour la pondération de la base de données est en $O(|\mathcal{T}| \times |\mathcal{I}| \times \mu_{diff})$. Nous notons dans ce cas aussi que la complexité pour le tirage de la norme est en $O(\log(M_{diff}))$ et celle du tirage du motif en $O(M_{diff})$. Donc la complexité totale de tirage d'un échantillon de k motifs ensemblistes est linéaire en $O(|\mathcal{T}| \times |\mathcal{I}| \times M_{diff} + k(\log(|\mathcal{T}|) + \log(M_{diff}) + M_{diff}))$.

4.4.3 Analyse expérimentale

La section expérimentale étudie l'efficacité de la méthode d'échantillonnage sous contrainte de mesures d'intérêt fondées sur la norme pour les itemsets et présente les résultats des accuracies [1] des classifieurs obtenus. Les expériences ont été conduites avec 10 bases de données de l'UCI avec les versions pré-traitées [2]. La longueur moyenne des transactions des jeux de données est au moins égale à 15. Ainsi, nous pouvons bien observer l'impact des contraintes de taille sur les performances des classifieurs construits avec et sans contraintes. Le tableau 4.2 détaille les caractéristiques des benchmarks. Dans la suite, nous allons étudier les utilités $u_{freq} \times u_{[\mu..M]}$ et $u_{area} \times u_{[\mu..M]}$ qui donnent un poids non nul respectivement à un motif suivant sa fréquence tant que sa norme est comprise entre μ et M, et à un motif suivant son aire tant que sa norme est comprise entre μ et M, avec $\mu \leq M$. La valeur de la contrainte de taille minimale est fixée à $\mu = 1$ tout au long des expériences. Toutes les expérimentations sont réalisées sur un PC de 3.50 GHz 2 Core CPU avec une mémoire de 16 Go. Le prototype de notre méthode [3] est implémenté en Java.

TABLE 4.2 – Caractéristiques des jeux de données utilisés

Dataset	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert Class\rvert$	$\lvert t\rvert_{max}$	$\lvert t\rvert_{moy}$
Auto	130	205	7	25	24.0
Congres	32	435	2	16	15.0
CylBands	122	540	2	35	33.0
Hepatitis	54	155	2	19	17.0
HorseColic	83	368	2	22	16.0
Ionosphere	155	351	2	34	34.0
Mushroom	88	8124	2	22	21.0
Soybean-large	99	683	19	35	31.0
Waveform	98	5000	3	21	21.0
Zoo	35	101	7	16	16.0

Dans la suite, nous allons d'abord étudier la rapidité de notre méthode et ensuite nous passerons à la construction de classifieurs comme nous l'avons présenté au chapitre 2 à la

1. Pluriel de *accuracy*, le taux de bien classés (nombre d'entités bien classées sur nombre total d'entités)
2. http://cgi.csc.liv.ac.uk/~frans/KDD/Software/LUCS-KDD-DN/DataSets
3. Code source disponible à https://github.com/ItemsetSampling/cnria19

page 65. Enfin, nous comparerons les accuracies obtenues par la méthode avec contrainte et celles sans contrainte de [Boley *et al.*, 2011].

Rapidité de l'algorithme avec contrainte Dans cette partie, nous allons mesurer le temps d'exécution de notre méthode d'échantillonnage (prétraitement et tirage). Pour le temps de prétraitement, nous allons prendre les deux jeux de données `Mushroom` et `Waveform` qui contiennent plus de transactions, 8124 et 5000 respectivement. Pour la durée d'échantillonnage d'un motif, nous considérons les bases de données `Ionosphere` et `CylBands` qui en moyenne, contiennent les transactions les plus longues (voir tableau 4.2).

D'après les expérimentations pour $M = 10$, la durée de prétraitement de la base de données `Mushroom` est en moyenne de 14 ms et celle de `Waveform` est de 11 ms. D'autre part, pour le tirage de 1000 motifs, notre méthode nécessite environ 15 ms pour `Ionosphere` et 25 ms dans `CylBands`. La figure 4.1 montre l'évolution des temps de prétraitement des différentes bases de données et le temps de tirage par motifs au sein de chacune d'elles suivant la contrainte de taille maximale.

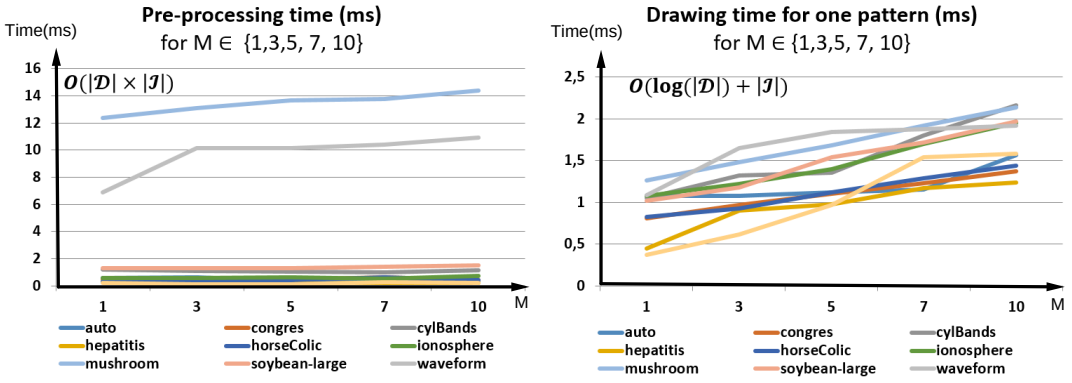


FIGURE 4.1 – Evolution des temps d'exécution suivant la contrainte de norme maximale

Comme nous pouvons le voir, le temps de prétraitement augmente légèrement avec la contrainte de norme maximale (`Mushroom` et `Waveform`). Nous notons que cette évolution est quasi-linéaire sur les autres jeux de données de taille moyenne (`Auto`, `Hepatitis`, etc.). D'après ces courbes, nous pouvons dire que la méthode avec contrainte de norme est très efficace pour échantillonner des milliers de motifs ensemblistes en quelques millisecondes.

Impact de l'usage de contraintes pour la classification Pour construire nos classifieurs, nous utilisons la méthode présentée à la section 2.3. Le reste de cette partie se focalise sur la classification des bases de données de l'UCI du tableau 4.2. En premier lieu, on évalue les performances des classifieurs construits avec notre méthode à base d'utilité fondée sur la norme en prenant des échantillons de 1000 motifs. Nous avons ainsi considéré différentes valeurs de la contrainte de norme maximale ($M \in [1..7]$) d'une part pour la fréquence ($freq$) et d'autre part pour l'aire ($area$). Ainsi, nous verrons à partir de quelle valeur de la contrainte de norme maximale le phénomène de la longue traîne va commencer à se manifester. Dans tous les cas, nous avons fait une validation croisée avec 10 folds. Le

tableau 4.3 présente les moyennes des accuracies obtenues en répétant 10 fois l'expérience. Nous avons omis les écarts-types car ils sont très faibles, au plus 1%.

TABLE 4.3 – Accuracy(%) des classifieurs construits avec 1000 motifs retournés par la méthode avec contrainte.

	M=1		M=2		M=3		M=4		M=5		Best	
Dataset	freq	area	freq	area	freq	area	freq	area	freq	area	freq	area
`Auto`	82.0	82.1	**84.2**	**83.7**	83.5	83.1	82.7	82.2	82.1	81.7	84.2	83.7
Cong.	**95.2**	**95.2**	93.8	94.0	94.1	94.1	94.5	94.3	94.1	94.1	95.2	95.2
Cyl.	75.7	75.9	76.2	75.9	**77.1**	76.9	77.1	77.1	77.0	**78.2**	77.1	78.2
Hepat.	**86.3**	**86.0**	82.0	81.5	80.7	81.5	80.1	79.8	79.9	79.6	86.3	86.0
Horse.	**80.3**	**81.2**	77.8	79.6	77.3	77.3	76.5	77.0	76.0	75.5	80.3	81.2
Ionos.	87.0	87.4	88.2	88.3	89.5	89.3	**90.0**	**89.7**	89.1	89.6	90.0	89.7
Mush.	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100
Soy.	**93.0**	92.9	92.8	**93.2**	92.5	92.5	90.8	90.1	88.9	87.9	93.0	93.2
Wave.	**79.6**	**79.5**	74.6	74.7	73.3	73.2	73.1	73.2	72.6	72.5	72.2	79.5
`Zoo`	96.6	96.7	**96.9**	**96.9**	96.4	96.2	96.3	96.6	96.8	96.6	96.9	96.9
Moy	87.6	**87.7**	86.7	86.8	86.4	86.4	86.1	86.0	85.7	85.6	88.3	88.4

Les résultats du tableau 4.3 confirment qu'avec les bases de données transactionnelles, nous parvenons à obtenir de très bonnes accuracies rien qu'en variant la contrainte maximale M entre 1 et 5.

Ainsi, nous considérons la contrainte de norme maximale comme étant un paramètre à faire varier pour obtenir la meilleure des accuracies (sachant que nous noterons **Best** la meilleure valeur de M obtenue). Par ailleurs, on note que la différence des accuracies obtenues d'une part avec l'aire et d'autre part avec la fréquence, pour une même valeur de la contrainte maximale ($M \in [1..7]$) donnée, est très faible. En utilisant **Best** comme valeur de M avec l'utilité u_{freq}, nous allons comparer les accuracies obtenues par notre méthode basée sur des utilités fondées sur la norme et celles de la méthode sans contrainte étudiée dans [Boley *et al.*, 2011] que nous avons implémentée.

La figure 4.2 montre les accuracies obtenues avec la méthode sans contrainte (en blanc) et la méthode avec contrainte (en gris) pour chacune des bases de données. Les gains (en noir) sont positifs (barres noires situées en haut) si la méthode avec contrainte donne un meilleur résultat et négatifs (barres noires situées en bas, s'il n'y a pas de gain) dans le cas contraire. On note alors qu'il est primordial de poser des contraintes de taille sur les motifs produits lorsque les transactions de la base sont très longues comme dans `Soybean-large`. Ainsi, notre méthode obtient globalement un gain en moyenne de 10.1%. Par ailleurs, sur les grandes bases de données telles que `Auto`, `CylBands`, `HorseColic`, `Ionosphere` et `Soybean-large`, les valeurs des accuracies obtenues par la méthode avec contrainte de norme sont largement au dessus de celles de la méthode sans contrainte proposée dans [Boley *et al.*, 2011].

Un autre paramètre très important pour notre approche est la taille de l'échantillon. Pour étudier l'évolution de l'accuracy suivant ce paramètre, nous avons fait une validation croisée et choisi pour chaque base de données la valeur optimale **Best** pour la contrainte

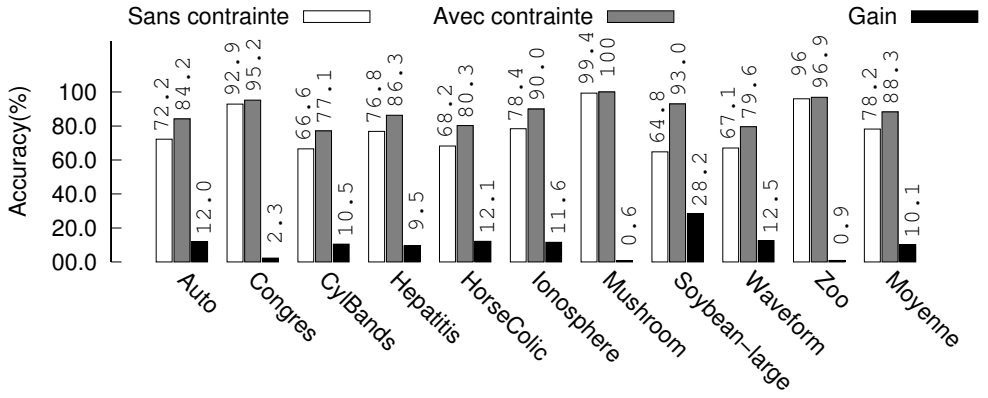


FIGURE 4.2 – Comparaison des accuracies(%) des classifieurs obtenus

de norme maximale M. Enfin, nous faisons d'une part la moyenne des accuracies obtenues avec la fréquence et d'autre part celles obtenues avec l'aire. La figure 4.3 montre l'évolution des accuracies moyennes avec différentes valeurs de $M \in \{3, 10, Best\}$.

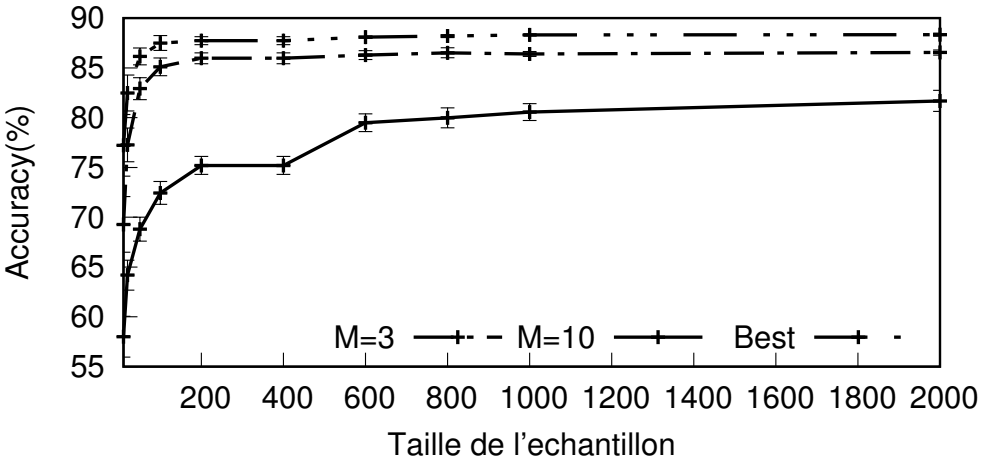


FIGURE 4.3 – Evolution de l'accuracy en fonction de la taille de l'échantillon

On peut facilement voir que la performance des classifieurs construits avec les motifs échantillonnés par notre méthode augmente avec la taille de l'échantillon. D'une façon importante, les accuracies convergent rapidement vers la valeur maximale. Ainsi, avec un échantillon de 100 motifs (dans notre cas) on obtient déjà de très bonnes accuracies. En effet, la taille de l'échantillon dépend de l'application de l'utilisateur. On note aussi que si la valeur de la contrainte de norme maximale est très grande ($M = 10$ dans nos expérimentations) alors les accuracies sont très faibles, car les motifs utilisés comme features ne sont pas pertinents, i.e. leur fréquence dans la base est trop faible.

4.5 Conclusion

Nous avons proposé un algorithme générique d'échantillonnage en sortie de motifs en deux étapes et une classe de mesures d'intérêt fondées sur la norme. L'efficacité de notre algorithme a été montrée sur plusieurs jeux de données transactionnelles avec les mesures d'intérêt (u_{freq}, u_{area} et $u_{\leq M}$). Nous avons aussi montré que notre méthode sous contrainte de norme maximale est meilleure que la méthode sans contrainte de [Boley *et al.*, 2011] pour construire des classifieurs de données ensemblistes.

En perspectives, nous pensons qu'avec les motif échantillonnés suivant une mesure d'intérêt fondée sur la norme, il serait envisageable de construire directement des règles de classification, afin de ne plus avoir à transformer le jeu données initial à l'aide des motifs échantillonnés et à utiliser sur ce fichier de donnés un algorithme de classification tel que SVM. Néanmoins, pour construire de telles règles de classification, il pourrait être nécessaire de considérer des mesures de contraste, mesures que nous n'avons pas considérées dans notre travail.

Finalement, nous pensons que la classe de mesures d'intérêt fondées sur la norme qui nous a permis de résoudre le problème de la longue traîne est prometteuse pour tirer des motifs ayant une structure plus complexe tels que les motifs séquentiels.

Chapitre 5

Echantillonnage de motifs séquentiels

Sommaire

Nous allons présenter dans ce chapitre, nos contributions sur l'échantillonnage en sortie de motifs dans des bases de données séquentielles. Dans le cadre de cette thèse, nous avons proposé CSSampling (Constrained Subsequence Sampling) [Diop *et al.*, 2018b, Diop *et al.*, 2018a] pour échantillonner des motifs séquentiels suivant une probabilité proportionnelle à leur fréquence dans une base de données et sous une contrainte de norme maximale pour éviter le phénomène de la longue traîne (voir section 3.2 à la page 79). NUSSampling (Norm-based Utility Subsequence Sampling) [Diop *et al.*, 2019d], est un algorithme générique d'échantillonnage en sortie de motifs séquentiels prenant en compte toute mesure d'intérêt fondée sur la norme. C'est l'instanciation de l'algorithme 4 présenté à la page 100 sur les données séquentielles.

Nos principales contributions sur l'échantillonnage en sortie de motifs séquentiels que nous allons présenter dans ce chapitre sont les suivantes :

— Nous proposons un nouvel algorithme générique, nommé NUSSampling (Norm-based Utility Subsequence Sampling), qui échantillonne des motifs séquentiels pro-

portionnellement à une mesure d'intérêt fondée sur la norme.

— Nous présentons un large éventail de résultats expérimentaux pour analyser le comportement de NUSSampling. Nous montrons sur plusieurs jeux de données que notre approche est suffisamment efficace pour renvoyer des centaines de motifs séquentiels par seconde. Nous soulignons également l'intérêt pratique des mesures d'intérêt fondées sur la norme pour mieux contrôler la qualité des motifs retournés et éviter la malédiction de la longue traîne.

— La classification des séquences est une tâche cruciale de la fouille de données utile dans un large domaine d'applications. Nous étudions comment l'échantillonnage de motifs séquentiels conduit à construire des classifieurs associatifs pour les séquences à l'aide de la méthode présentée à la section 2.4. Fait intéressant, la précision de ces classifieurs basés sur des échantillons construits dans un temps de réponse court est comparable à celle des méthodes de l'état de l'art. Les expériences montrent qu'il est encore essentiel d'utiliser une contrainte pour tirer des motifs généraux contenus dans la tête et non dans la distribution de la queue.

Dans la suite de ce chapitre, la section 5.1 donne quelques rappels et notions de base sur les séquences. La section 5.2 présente les différents défis que nous avons besoin de relever pour résoudre le problème de l'échantillonnage en sortie de motifs séquentiels. Nous présentons notre procédure aléatoire en deux étapes pour l'échantillonnage en sortie de motifs séquentiels selon une mesure d'intérêt fondée sur la norme dans la section 5.3. La section 5.4 évalue notre approche en effectuant une étude sur des jeux de données réels et de référence, et en comparant les accuracies des classifieurs basés sur des échantillons de motifs séquentiels avec les méthodes de l'état de l'art. Nous concluons dans la section 5.5.

5.1 Problème de l'échantillonnage en sortie de motifs séquentiels

Nous avions présenté à la section 1.1.1 les définitions liées aux séquences et aux sous-séquences d'une base de données séquentielles. Nous rappelons que $\phi(s)$ est l'ensemble des sous-séquences d'une séquence $s = \langle X_1 X_2 \dots X_l \rangle$, c'est-à-dire $\phi(s) = \{s' \in \mathcal{L}_\mathcal{S} : s' \sqsubseteq s\}$, et $\Phi(s)$ sa cardinalité, c'est-à-dire $\Phi(s) = |\phi(s)|$. Dans ce qui suit, s^l désigne le préfixe $\langle X_1 X_2 \dots X_l \rangle$ de s $(0 \leq l \leq n,\ l \in \mathbb{N})$, avec $n = |s|$ la taille de la séquence, s^0 étant la séquence vide (représentée par $\langle\rangle$) et $s[j] = X_j$ indique le j-ème itemset de s $(1 \leq j \leq n,\ j \in \mathbb{N})$.

Exemple 19. *Nous utilisons la base de données séquentielles $\mathcal{S}$ présentée dans le tableau 5.1 comme exemple courant. Cet ensemble de données contient 4 séquences s_1, s_2, s_3 et s_4 définies sur l'ensemble des éléments $\mathcal{I} = \{a, b, c, d\}$. Par exemple, la taille de $s_1 = \langle (ab)c \rangle$ est égale à 2, soit $|s_1| = 2$, tandis que sa norme est égale à 3, soit $\|s_1\| = |(ab)| + |c| = 2 + 1 = 3$. De plus, nous avons $s_1^0 = \langle\rangle$, $s_1^1 = \langle (ab) \rangle$, $s_1^2 = s_1$, $s_1[1] = (ab)$ et $s_1[2] = c$. Enfin, l'ensemble $\phi(s_1)$ des sous-séquences de s_1 est défini par $\phi(s_1) = \{\langle\rangle, \langle a \rangle, \langle b \rangle, \langle c \rangle, \langle (ab) \rangle, \langle ac \rangle, \langle bc \rangle, \langle (ab)c \rangle\}$. Ainsi, nous avons $\Phi(s_1) = |\phi(s_1)| = 8$. Le nombre de sous-séquences $\Phi(s_i)$ de toutes les séquences $s_i \in \mathcal{S}$ est détaillé dans le tableau 5.2. La notation $\Phi_{[\mu..M]}(s_i)$ représente le nombre de sous-séquences de la séquence s_i*

dont les normes sont comprises entre la contrainte de norme minimale μ et celle maximale M.

TABLE 5.1 – Une base de données séquentielles $\mathcal{S}$

Sid	**Séquence**
s_1	$\langle(ab)c\rangle$
s_2	$\langle(ab)c(ac)\rangle$
s_3	$\langle c(ac)\rangle$
s_4	$\langle(ab)(cd)\rangle$

Le tableau 5.2 fournit un ensemble de sous-séquences et leurs fréquences dans la base de données séquentielles $\mathcal{S}$. De plus, nous donnons également les utilités fondées sur la norme $u_{\in[1..2]}(s)$ et $u_{area}(s)$ pour chaque sous-séquence s. Par exemple, parce que notre problème est de tirer une sous-séquence proportionnellement à sa fréquence multipliée par une utilité fondée sur la norme, et $freq(\langle ac\rangle, \mathcal{S}) = 3 \times freq(\langle ba\rangle, \mathcal{S})$, notre objectif consiste à développer une instance de l'algorithme 4 pour échantillonner la sous-séquence $\langle ac\rangle$ avec une probabilité trois fois supérieure à celle de la sous-séquence $\langle ba\rangle$ (car $\langle ac\rangle$ et $\langle ba\rangle$ ont la même norme et $u(\langle ac\rangle) = u(\langle ba\rangle)$ quel que soit l'utilité fondée sur la norme u). Mais, même si la sous-séquence $\langle(ab)c\rangle$ a une fréquence de 3, elle ne sera pas tirée si l'on considère l'utilité $u_{\in[1..2]}$ (car sa norme est égale à 3 et supérieure à $M = 2$).

TABLE 5.2 – Exemples de sous-séquences dans $\mathcal{L}_{\mathcal{S}}$ des séquences de $\mathcal{S}$

s	$u_{\in[1..2]}(s)$	$freq(s)$	$u_{area}(s)$	s	$u_{\in[1..2]}(s)$	$freq(s)$	$u_{area}(s)$
$\langle a\rangle$	1	4	1	$\langle ac\rangle$	1	3	2
$\langle b\rangle$	1	3	1	$\langle ad\rangle$	1	1	2
$\langle c\rangle$	1	4	1	$\langle ba\rangle$	1	1	2
$\langle d\rangle$	1	1	1	$\langle bc\rangle$	1	3	2
$\langle(ab)\rangle$	1	3	2	$\langle bd\rangle$	1	1	2
$\langle(ac)\rangle$	1	2	2	$\langle ca\rangle$	1	2	2
$\langle(cd)\rangle$	1	1	2	$\langle cc\rangle$	1	2	2
$\langle aa\rangle$	1	1	2	$\langle c(ac)\rangle$	0	2	3
$\langle\rangle$	0	4	0	$\langle(ab)c\rangle$	0	3	3

Supposons que nous souhaitions échantillonner des motifs de norme inférieure ou égale à 2 avec une probabilité proportionnelle à leur fréquence à partir de la base de données séquentielles $\mathcal{S}$. La figure 5.1 illustre le problème de la longue traîne sur le jeu de données séquentielles $\mathcal{S}$ du tableau 5.1.

L'histogramme de gauche montre la fréquence des 35 motifs de la base de données jouets (c'est-à-dire les barres en gris foncé et clair). Nous observons que 23 motifs ont une fréquence de 1 (la queue). Ainsi, les barres en gris foncé de l'histogramme de droite montrent

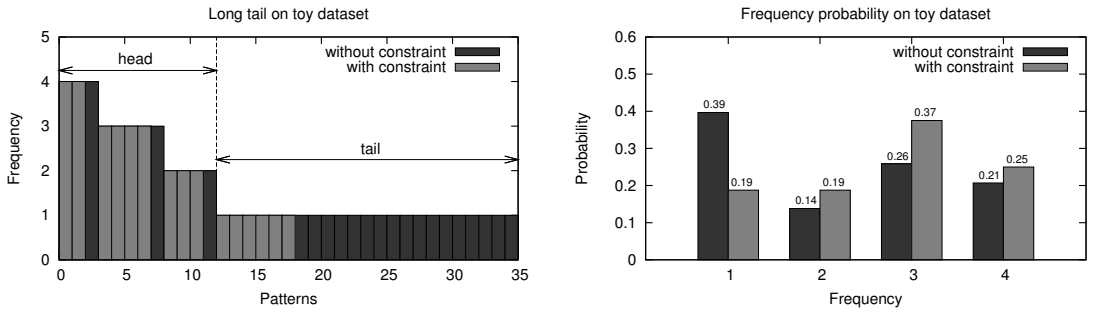


FIGURE 5.1 – Phénomène de la longue traîne sur le jeu de données séquentielles $\mathcal{S}$

que $39,6\%$ des motifs tirés selon la fréquence appartiennent à cette queue (avec une fréquence de seulement 1). En posant une contrainte de norme maximale égale à 2, nous n'avons que 19% des motifs qui se retrouvent avec une fréquence égale à 1.

D'une façon plus spécifique par rapport au problème général de la section 4.1, le problème que nous visons à résoudre dans ce chapitre peut être formulé comme suit :

Soient $\mathcal{S}$ une base de données séquentielles, u une utilité fondée sur la norme et k un entier naturel strictement positif. Notre objectif est de calculer efficacement un échantillon de motifs $\mathcal{S}ample_k(\mathcal{L}_\mathcal{S}, \mathcal{S}, freq \times u)$.

5.2 Défis de l'échantillonnage en sortie de motifs séquentielles

Les défis que nous devons relever pour échantillonner des motifs séquentiels suivant une mesure d'intérêt fondée sur la norme portent essentiellement sur les *occurrences* multiples au sein d'une même séquence. Etant donné une séquence $s = \langle X_1 \dots X_n \rangle$, une sous-séquence $s' = \langle X'_1 \dots X'_{n'} \rangle$ de s peut apparaître plusieurs fois au sein de s s'il existe plusieurs séquences d'indices $1 \leq i_1 < i_2 < \dots < i_{n'} \leq n$ telles que pour tout $j \in [1..n']$, on ait $X'_j \subseteq X_{i_j}$. Dans ce cas, on parle d'*occurrences* multiples de la sous-séquence s' au sein de s. La définition suivante précise comment ces différentes occurrences peuvent être représentées.

Définition 21 (Occurrence)**.** *Etant donné une séquence $s = \langle X_1 \dots X_n \rangle$, une liste ordonnée d'itemsets $o = \langle Z_1 \dots Z_n \rangle$ de même taille que s est une occurrence d'une sous-séquence $s' = \langle X'_1 \dots X'_{n'} \rangle$ de s s'il existe une séquence d'indices $1 \leq i_1 < \dots < i_{n'} \leq n$ telle que pour tout $j \in \{i_1, \dots, i_{n'}\}$, on ait $Z_{i_j} = X'_j$, et tout $j \in \{1, \dots, n\} \setminus \{i_1, \dots, i_{n'}\}$, on ait $Z_j = \emptyset$. Cette séquence d'indices, appelée signature de o, est unique par définition.*

Exemple 20. *Pour $s = \langle (ab)c(ac) \rangle$, $o_1 = \langle ac\emptyset \rangle$ et $o_2 = \langle a\emptyset c \rangle$ sont deux occurrences de $s' = \langle ac \rangle$ avec pour signature respective $\langle 1, 2 \rangle$ et $\langle 1, 3 \rangle$.*

Dans ce cas, chaque motif φ, quel que soit son nombre d'occurrences dans la séquence s, n'est compté qu'une et une seule fois dans l'ensemble des sous-séquences de s. Cette pro-

priété des séquences est un obstacle que l'on rencontre aussi bien à la phase de pondération qu'à la phase de tirage. Ainsi, les défis à relever sont :

— Comment calculer efficacement le nombre de sous-séquences d'une séquence qui respectent une contrainte de norme tout en tenant en compte les occurrences multiples ?

— Comment tirer uniformément une sous-séquence parmi l'ensemble des sous-séquences ayant une même norme et en présence d'occurrences multiples ?

Voilà deux questions que nous allons traiter dans la section suivante en se basant sur des éléments théoriques solides.

5.3 Echantillonnage de motifs séquentiels : NUSSampling

Dans cette section, nous allons montrer comment échantillonner efficacement des motifs séquentiels avec NUSSampling. D'abord, nous proposons une méthode efficace de pondération des séquences d'une base de données séquentielles afin d'avoir un tirage exact lors de la première étape de l'algorithme 4. Ensuite, nous montrons comment tirer un motif séquentiel à partir d'une séquence proportionnellement à son utilité dans la séquence pour satisfaire la deuxième étape du tirage. Enfin, nous analysons la complexité de NUSSampling.

5.3.1 Comptage du nombre de sous-séquences distinctes d'une séquence

Dans cette section, nous montrons comment calculer le nombre de sous-séquences d'une séquence s sous contrainte de norme en généralisant la proposition de [Egho *et al.*, 2015]. La principale difficulté est de ne pas compter plusieurs fois une même sous-séquence même si elle possède plusieurs occurrences dans s.

Sans contrainte sur la norme Soient une séquence $s = \langle X_1 \dots X_n \rangle$ et un itemset X. Par la suite, nous notons $s \circ X$ la concaténation de s et X définie par : $s \circ X = \langle X_1 \dots X_n X \rangle$. Intuitivement, si X est disjoint de tous les itemsets de la séquence s, il est aisé de vérifier que le nombre de sous-séquences distinctes de $s \circ X$ est égal au nombre de sous-séquences de s multiplié par le nombre de sous-ensembles de X, i.e. $\Phi(s \circ X) = \Phi(s) \times 2^{\|X\|}$. Si X n'est pas disjoint des itemsets de s, $\Phi(s \circ X)$ sera inférieur à $\Phi(s) \times 2^{\|X\|}$ et [Egho *et al.*, 2015] introduisent un terme correcteur $R(s, X)$ pour calculer le nombre exact de sous-séquences distinctes. Pour ce faire, ils commencent par introduire un ensemble de positions précisant où les répétitions d'items de l'itemset X sont localisées dans la séquence s :

Définition 22 (Ensemble de positions - [Egho *et al.*, 2015])**.** *Soient une séquence s et un itemset X. $L(s, X) = \{i \in \mathbb{N} : i \leq |s| \wedge s[i] \cap X \neq 0 \wedge (\forall j > i)(s[i] \cap X \not\subseteq s[j] \cap X)\}$ est l'ensemble des positions où l'itemset X a une intersection maximale avec les différents itemsets de s.*

Exemple 21. *Soit la séquence $s = \langle (ab)c(ac) \rangle$. Nous avons $s^1 = \langle (ab) \rangle$, $s[2] = c$ et $L(s^1, s[2]) = \emptyset$ car $s[2]$ n'intersecte aucun itemset de s^1. Calculons maintenant $L(s^2, s[3])$. $s[3] = (ac)$ intersecte à la fois le premier itemset $s[1] = (ab)$ de s ($s[1] \cap s[3] = a$) et le*

second itemset $s[2] = c$ de s ($s[2] \cap s[3] = c$). De plus, ces deux intersections sont disjointes. Par conséquent, nous avons $L(s^2, s[3]) = \{1, 2\}$, ce qui indique qu'en concaténant des sous-ensembles de $s[3]$ à des sous-séquences de s^2 des répétitions de sous-séquences de s^2 pourront être générées du fait que des items de $s[3]$ se retrouvent aux postions 1 et 2 de s^2.

A partir de cet ensemble de positions, il est possible de calculer le nombre de sous-séquences distinctes d'une séquence s grâce à la formule récursive suivante.

Théorème 1 (Nombre de sous-séquences - [Egho *et al.*, 2015])**.** *Etant donnés une séquence s et un itemset X, le nombre de sous-séquences distinctes de $s \circ X$, noté $\Phi(s \circ X)$, est défini par $\Phi(s \circ X) = \Phi(s) \times 2^{\|X\|} - R(s, Y)$ où $R(s, Y)$ est un terme correcteur :*

$$R(s, Y) = \sum_{\emptyset \subset K \subseteq L(s,Y)} (-1)^{|K|+1} (\Phi(s^{min(K)-1}) \times (2^{|s[K] \cap Y|} - 1))$$

avec $s[K] = \cap_{k \in K} s[k]$ pour toute séquence s et un ensemble d'indices K.

L'exemple suivant permet de donner une intuition de la formule récursive introduite précédemment, et en particulier de son terme correcteur.

Exemple 22. *Poursuivons l'exemple 21. L'ensemble $\phi(s^1)$ des sous-séquences de $s^1 = \langle (ab) \rangle$ est défini par $\phi(s^1) = \{\langle\ \rangle, \langle a \rangle, \langle b \rangle, \langle (ab) \rangle\}$. Nous avons donc $\Phi(s^1) = 4$. Comme $L(s^1, s[2]) = 0$ et $R(s^1, s[2]) = 0$, nous avons $\Phi(s^2) = \Phi(s^1) \times 2^{|c|} = 4 \times 2 = 8$. En effet, les sous-séquences de s^2 sont obtenues par simple concaténation de l'itemset vide ou de l'itemset (c) avec une sous-séquence de s^1. Nous détaillons maintenant le calcul de $\Phi(s^3) = \Phi(s^2) \times 2^{|(ac)|} - R(s^2, s[3]) = 8 \times 4 - R(s^2, s[3])$ et du terme correcteur $R(s^2, s[3])$. Comme $L(s^2, s[3]) = \{1, 2\}$, nous avons $R(s^2, s[3]) = (-1)^2 \Phi(s^0) \times (2^{|a|} - 1) + (-1)^2 \Phi(s^1) \times (2^{|c|} - 1) = 1 + 4 = 5$. Le premier terme de $R(s^2, s[3])$ permet de ne pas recompter la sous-séquence $\langle a \rangle$ de s^2 en la construisant par concaténation de l'itemset (a) (inclus dans $s[3]$) à la sous-séquence vide de s^0. Quant au second terme de $R(s^2, s[3])$, il permet de ne pas recompter les sous-séquences $\langle c \rangle, \langle ac \rangle, \langle bc \rangle, \langle (ab)c \rangle$ de s^2 en les construisant par concaténation de l'itemset (c) (inclus dans $s[3]$) aux sous-séquences $\langle \rangle, \langle a \rangle, \langle b \rangle, \langle (ab) \rangle$ de s^1. Nous avons finalement $\Phi(s^3) = 32 - R(s^2, s[3]) = 27$.*

Avec contrainte sur la norme Nous proposons une généralisation du théorème 1 permettant de calculer le nombre de sous-séquences de norme inférieure ou égale à M d'une séquence s.

Théorème 2 (Nombre de sous-séquences de norme bornée)**.** *Etant donnés une séquence s, un itemset X et un entier $j \leq \|s\|$, le nombre de sous-séquences distinctes de norme inférieure ou égale à j de $s \circ X$, noté $\Phi_{\leq j}(s \circ X)$, est défini ci-dessous :*

$$\Phi_{\leq j}(s \circ X) = \left(\sum_{k=0}^{min\{j, |X|\}} \binom{|X|}{k} \times \Phi_{\leq j-k}(s) \right) - R_{\leq j}(s, X)$$

où $R_{\leq j}(s, X)$ est un terme correcteur défini par :

$$R_{\leq j}(s, X) = \sum_{\emptyset \subset K \subseteq L(s,X)} (-1)^{|K|+1} \left(\sum_{k=1}^{j} \binom{|s[K] \cap X|}{k} \times \Phi_{\leq j-k}(s^{min(K)-1}) \right)$$

sachant que $R_{\leq j}(s,X)=0$ *si* $L(s,X)=\emptyset$ *et* $s[K]=\cap_{k\in K}s[k]$.

Ce théorème 2 étend la proposition de [Egho *et al.*, 2015] limitée au cas particulier où $j=\infty$. Notez que $\Phi_\ell(s)$ peut se calculer à l'aide de la différence $\Phi_{\leq\ell}(s)-\Phi_{\leq\ell-1}(s)$ si $\ell>0$ et 1, sinon.

Preuve. Soit s une séquence et X un ensemble d'éléments. Nous expliquons déjà que pour construire une sous-séquence de $s\circ X$ ayant une norme inférieure à j, nous pouvons concaténer n'importe quel sous-ensemble de taille k de X à une sous-séquence de s ayant une norme inférieure à $j-k$. En effet, on est sûr d'obtenir une sous-séquence de $s\circ X$ ayant une norme inférieure à $k+(j-k)=j$. Ainsi, nous avons $\phi_{\leq j}(s\circ X)=\cup_{k=0}^{j}\phi_{\leq j-k}(s)\circ\mathcal{P}_{=k}(X)$ et $\Phi_{\leq j}(s\circ X)=\sum_{k=0}^{j}\Phi_{\leq j-k}(s)\times\binom{|X|}{k}-R_{\leq j}(s,X)$ où $R_{\leq j}(s,X)$ est un terme de correction (pour compter le nombre de sous-séquences *distinctes*).

Soit $s=\langle X_1\dots X_m\rangle$ avec $|X_m|=k$ une séquence qui est comptée plusieurs fois, c'est-à-dire $s\in\phi_{\leq j}(s)\cap(\phi_{\leq j}(s)\circ\mathcal{P}_{\geq 1}(X))$ où $\mathcal{P}_{\geq 1}(X)=\{X\subseteq X:|X|\geq 1\}$. Parce que $t\in(\phi_{\leq j}(s)\circ\mathcal{P}_{\geq 1}(X))$, nous avons nécessairement $T_m\in\mathcal{P}_{\geq 1}(X)$, c'est-à-dire $X_m\subseteq X$. De plus, comme $s\in\phi_{\leq j}(s)$, il existe un entier $i\leq|s|$ tel que $X_m\subseteq s[i]$. Soit $l=max\{i\leq|s|:X_m\subseteq s[i]\}$. Depuis $X_m\subseteq X$, nous avons également $l=max\{i\leq|s|:X_m\subseteq(s[i]\cap X)\}$. Nous montrons maintenant que $l\in L(s,X)$. Premièrement, parce que $X_m\neq\emptyset$, nous avons $s[l]\cap X\neq\emptyset$. Supposons maintenant qu'il existe $l'>l$ tel que $s[l]\cap X\subseteq s[l']\cap X$. Ensuite, nous aurions $X_m\subseteq s[l']\cap X$, ce qui contredit que l est maximal, et complète la preuve que $l\in L(s,X)$. A ce stade, nous avons prouvé que $X\in\phi_{\leq j-k}(s^{l-1})\circ\mathcal{P}_{=k}(s[l]\cap X)$ pour un entier $l\in L(s,X)$. Ainsi, nous avons $R_{\leq j}(s,X)=|\bigcup_{l\in L(s,X)}(\cup_{k=1}^{j}\phi_{\leq j-k}(s^{l-1})\circ\mathcal{P}_{=k}(s[l]\cap X))|$.

En utilisant le principe d'inclusion-exclusion, nous réécrivons $R_{\leq j}(s,X)$ comme suit $\sum_{\emptyset\subset K\subseteq L(s,X)}(-1)^{|K|+1}R^K_{\leq j}(s,X)$ avec $R^K_{\leq j}(s,X)=|\bigcap_{l\in K}(\cup_{k=1}^{j}\phi_{\leq j-k}(s^{l-1})\circ\mathcal{P}_{=k}(s[l]\cap X))|$. Maintenant, soit $s=\langle X_1\dots X_m\rangle$ une séquence dans l'ensemble $\bigcap_{l\in K}(\cup_{k=1}^{j}\phi_{\leq j-k}(s^{l-1})\circ\mathcal{P}_{=k}(s[l]\cap X))$. Nous avons nécessairement $t^{m-1}\in\phi_{\leq j-k}(s^{min(K)-1})$ et $X_m\in\cap_{l\in K}\mathcal{P}_{=k}(s[l]\cap X)$, c'est-à-dire $X_m\in\mathcal{P}_{=k}(s[K]\cap X)$ avec $s[K]=\cap_{l\in K}s[l]$. Il s'ensuit que $R^K_{\leq j}(s,X)=|\cup_{k=1}^{j}\phi_{\leq j-k}(s^{min(K)-1})\circ\mathcal{P}_{=k}(s[K]\cap X))|$. Enfin, parce que les ensembles $\phi_{\leq j-k}(s^{min(K)-1})\circ\mathcal{P}_{=k}(s[K]\cap X))$ sont disjoints, nous avons $R^K_{\leq j}(s,X)=\sum_{k=1}^{j}\Phi_{\leq j-k}(s^{min(K)-1})\times\binom{|s[K]\cap X|}{k}$, qui complète la preuve du théorème 2. □

En poursuivant l'exemple 22, l'exemple suivant illustre le principe de fonctionnement de la formule du théorème 2.

Exemple 23. *L'ensemble* $\phi_{\leq 2}(s^1)$ *des sous-séquences de* $s^1=\langle(ab)\rangle$ *de norme inférieure à* 2 *est défini par* $\phi_{\leq 2}(s^1)=\{\langle\,\rangle,\langle a\rangle,\langle b\rangle,\langle(ab)\rangle\}$. *Nous avons donc* $\Phi_{\leq 2}(s^1)=4$, *et on voit aussi aisément que* $\Phi_{\leq 1}(s^1)=3$ *(la sous-séquence* $\langle(ab)\rangle$ *étant de norme strictement supérieure à* 1*). Comme* $L(s^1,s[2])=\emptyset$, *nous avons* $R_{\leq 2}(s^1,s[2])=0$ *et* $\Phi_{\leq 2}(s^2)=\sum_{k=0}^{|c|}\binom{|c|}{k}\times\Phi_{\leq 2-k}(s^1)=\binom{1}{0}\times\Phi_{\leq 2}(s^1)+\binom{1}{1}\times\Phi_{\leq 1}(s^1)=4+3=7$. *Le premier terme de la somme correspond aux* 4 *sous-séquences de* s^3 *obtenues par concaténation du sous-ensemble vide aux sous-séquences de* s^2, *alors que le deuxième terme correspond aux* 3 *sous-séquences de* s^3 *obtenues par concaténation de l'itemset* (c) *aux*

sous-séquences de s^2 *de norme inférieure à* 1. *Détaillons maintenant le calcul de* $\Phi_{\leq 2}(s^3) = \sum_{k=0}^{|(ac)|} \binom{|(ac)|}{k} \times \Phi_{\leq 2-k}(s^2) - R_{\leq 2}(s^2, s[3]) = \Phi_{\leq 2}(s^2) + 2 \times \Phi_{\leq 1}(s^2) + \Phi_{\leq 0}(s^2) - R_{\leq 2}(s^2, s[3]) = 7 + 2 \times 4 + 1 - R_{\leq 2}(s^2, s[3])$. *Le second terme de* $\Phi_{\leq 2}(s^3)$, *égal à* 2×4, *correspond par exemple au nombre de sous-séquences de* s^3 *pouvant être obtenues par concaténation d'un sous ensemble de taille* 1 *de* (ab) *(au nombre de* 2*) avec une sous-séquence de* s^2 *de norme inférieure à* 1. *Pour finir, le calcul du terme correcteur* $R_{\leq 2}(s^2, s[3])$ *se présente comme suit :* $R_{\leq 2}(s^2, s[3]) = (-1)^2 \binom{|a|}{1} \times \Phi_{\leq 1}(s^0) + (-1)^2 \binom{|c|}{1} \times \Phi_{\leq 1}(s^1) = 1 + 3 = 4$. *On en déduit ainsi que* $\Phi_{\leq 2}(s^3) = 7 + 2 \times 4 + 1 - 4 = 12$.

La formule présentée au théorème 2 est récursive. Néanmoins, étant donné une séquence s et une borne $M \leq \|s\|$, cette récursivité peut facilement être supprimée en calculant ligne par ligne les matrices T et R définies par :

— $T[i][j] = \Phi_{\leq j}(s^i)$ pour $i \in [0..|s|]$ et $j \in [0..M]$. $T[i][j]$ représente le nombre de sous-séquences de norme inférieure ou égale à j de la séquence s^i.

— $R[i][j] = R_{\leq j}(s^{i-1}, s[i])$ pour $i \in [2..|s|]$ et $j \in [0..M]$. Ce terme correcteur représente le terme à soustraire quand on souhaite calculer le nombre de sous-séquences de norme inférieure à j de $s^i = s^{i-1} \circ s[i]$ à partir du nombre de sous-séquences de norme inférieure à j de s^i en y concaténant des sous-ensembles de $s[i]$.

L'algorithme 5 montre comment on peut calculer ligne par ligne T et R. Après une phase d'initialisation où on calcule les deux premières lignes de la matrice T (étapes 1 à 5), il calcule ligne par ligne les valeurs de $R[i][j]$ (étapes 8 à 11) et $T[i][j]$ (étapes 12 et 13) en se basant directement sur les formules du théorème 2. Notons que par définition $max_{s \in \mathcal{L}}(L(s^{i-1}, s[i])) = i - 1$. Il en découle que $min(K) - 1 \leq (i - 2)$, et donc que $R[i][j]$ peut être calculé à l'étape 10 en utilisant uniquement des lignes de T précédemment calculées. Des exemples de matrices T et R sont donnés à la figure 5.2 pour la séquence $s = \langle (ab)c(ac) \rangle$, sachant que l'exemple 23 illustre en particulier comment calculer $R[3][2] = R_{\leq 2}(s^2, s[3])$ et $T[3][2] = \Phi_{\leq 2}(s^3)$.

T[i][j]	$\leq$ **0**	$\leq$ **1**	$\leq$ **2**	$\leq$ **3**
$s^{\mathbf{0}} = \langle\rangle$	1	1	1	1
$s^{\mathbf{1}} = \langle (ab) \rangle$	1	3	4	4
$s^{\mathbf{2}} = \langle (ab)c \rangle$	1	4	7	8
$s^{\mathbf{3}} = \langle (ab)c(ac) \rangle$	1	4	12	21

R[i][j]	$\leq$ **0**	$\leq$ **1**	$\leq$ **2**
$s^1, s[\mathbf{2}] = c$	0	0	0
$s^2, s[\mathbf{3}] = (ac)$	2	4	5

FIGURE 5.2 – Exemples de matrices T et R

Pour conclure cette section, notons qu'à partir du théorème 2, étant donné une séquence s et deux entiers naturels strictement positifs μ et M tels que $\mu \leq M \leq \|s\|$, il est possible de calculer le nombre de sous-séquences distinctes de s de norme comprise entre μ et M. En effet, nous avons $\Phi_{[\mu,M]}(s) = \Phi_{\leq M}(s) - \Phi_{\leq \mu-1}(s)$. Dans l'algorithme 4, cette formule permet de calculer à l'étape 1 le poids initial $\omega_u(s)$ des séquences s de la base de données séquentielles $\mathcal{S}$.

Algorithm 5 Construction des matrices T&R

Input : Une séquence s et une borne $M \leq \|s\|$
Output : Une matrice T de dimension $|s| \times M$ telle que $T[i][j] = \Phi_{\leq j}(s^i)$
$T[0][0] \leftarrow T[1][0] \leftarrow 1$
for $j = 1$ to M **do**
    $T[0][j] \leftarrow 1$ and $T[1][j] \leftarrow T[1][j-1] + \binom{|s[1]|}{j}$
for $i = 2$ to $|s|$ **do**
    for $j = 1$ to M **do**
        $R[i][j] \leftarrow T[i][j] \leftarrow 0$
        for $K \in \mathcal{P}_{\geq 1}(L(s^{i-1}, s[i]))$ **do**
            $m \leftarrow min(K)$ and $k_{max} \leftarrow |s[K] \cap s[i]|$
            for $k = 1$ to j **do**
                $R[i][j] + \leftarrow (-1)^{|K|+1} T[m-1][j-k] \times \binom{k_{max}}{k}$
        for $k = 0$ to $min\{j, |s[i]|\}$ **do**
            $T[i][j] + \leftarrow T[i-1][j-k] \times \binom{|s[i]|}{k}$
        $T[i][j] \leftarrow T[i][j] - R[i][j]$
Return (T)

5.3.2 Echantillonnage d'une séquence

Pour échantillonner une séquence proportionnellement à son poids, nous allons utiliser le lemme 3 du chapitre 4 qui, dans le cas des séquences, repose fortement sur le théorème 2 fournissant le nombre de sous-séquences ayant la norme ℓ i.e, $\Phi_\ell(s) = \Phi_{\leq \ell}(s) - \Phi_{\leq \ell-1}(s)$. Ainsi, la formule du lemme 2 permet de calculer le poids initial $\omega_u(s) = \sum_{\ell=0}^{\|s\|} \omega_u^\ell(s)$ pour chaque séquence s de la base de données séquentielles $\mathcal{S}$ (voir la ligne 3 de l'algorithme 4).

Exemple 24. *Revenons à l'exemple du tableau 5.3. Comme $f_{u_{freq}}(\ell) = 1$ pour tous $\ell \in \mathbb{N}$, le poids $\omega_{freq}(s_i)$ est égal au nombre de sous-séquences distinctes $\Phi_\ell(s_i)$. Nous détaillons maintenant le calcul des poids de la première séquence s_1 avec trois autres utilités :*

- *Avec $f_{u_{area}}(\ell) = l$, nous avons $\omega_{area}(s_1) = \Phi_0(s_1) \times 0 + \Phi_1(s_1) \times 1 + \Phi_2(s_1) \times 2 + \Phi_0(s_3) \times 3 = 1 \cdot 0 + 3 \cdot 1 + 3 \cdot 2 + 1 \cdot 3 = 12$.*
- *Avec $f_{u_{\in[1..2]}}(\ell) = 1$ ssi $l \in [1..2]$ (0 sinon), nous avons $\omega \in [1..2](s_1) = \Phi_0(s_1) \times 0 + \Phi_1(s_1) \times 1 + \Phi_2(s_1) \times 1 + \Phi_0(s_3) \times 0 = 1 \cdot 0 + 3 \cdot 1 + 3 \cdot 2 + 1 \cdot 0 = 6$.*
- *Avec $f_{u_{decay}}(\ell) = \alpha^\ell$ et $\alpha = 0.5$, nous avons $\omega_{decay}(s_1) = \Phi_0(s_1) \times 0.5^0 + \Phi_1(s_1) \times 0.5^1 + \Phi_2(s_1) \times 0.5^2 + \Phi_0(s_3) \times 0.5^3 = 1 \cdot 1 + 3 \cdot 0.5 + 3 \cdot 0.25 + 1 \cdot 0.075 = 3.375$.*

Le tableau 5.3 donne les poids des séquences de la base de données séquentielles $\mathcal{S}$ suivant différentes mesures d'intérêt fondées sur la norme. Nous avons aussi le nombre total d'occurrences de motifs dans chaque séquence.

Calculer le nombre de sous-séquences distinctes plus petites qu'une longueur donnée ℓ est très coûteux comme nous le verrons dans la section 5.3.4. Heureusement, avec l'utilité fondée sur la norme $u_{\leq M}$ (que nous recommandons d'utiliser pour éviter le problème de la longue traîne), nous avons $f_{\leq M}(\ell) = 0$ pour tout $\ell > M$ et par conséquent, $\omega(s) = \sum_{\ell=0}^{\|s\|} \omega_u^\ell(s) = \sum_{\ell=0}^{M} \omega_u^\ell(s)$.

TABLE 5.3 – Un ensemble de données séquentielles $\mathcal{S}$ avec 4 séquences

Sid	Séquence	#occurrences	$\Phi(s_i)$	$\omega_{freq}(s_i)$	$\omega_{area}(s_i)$	$\omega_{\in[1..2]}(s_i)$	$\omega_{decay}(s_i)$
s_1	$\langle(ab)c\rangle$	8	8	8	12	6	3.38
s_2	$\langle(ab)c(ac)\rangle$	32	25	25	65	11	5.72
s_3	$\langle c(ac)\rangle$	8	7	7	11	5	2,88
s_4	$\langle(ab)(cd)\rangle$	16	16	16	32	10	5.06

5.3.3 Echantillonnage de sous-séquences par rejet

Après avoir tiré au hasard une séquence $s \in \mathcal{S}$ proportionnellement à son poids $\omega_u(s)$ (ligne 4 de l'algorithme 4) et un entier ℓ entre 0 et $\|s\|$ selon la distribution $\omega_u^\ell(s)$ (ligne 6 de l'algorithme 4), NUSSAMPLING vise à renvoyer une sous-séquence de norme ℓ tirée uniformément de la séquence s (ligne 7 de l'algorithme 4). La difficulté est de ne pas privilégier les sous-séquences qui ont plusieurs occurrences dans la séquence.

Pour faire face à cette difficulté, nous utilisons une méthode de rejet en tirant uniformément une occurrence de la séquence s et en la rejetant si cette occurrence n'est pas la première. Comme chaque sous-séquence a une première occurrence unique, cette approche garantit un tirage uniforme des sous-séquences de norme ℓ. Nous commençons par formaliser la notion de première occurrence :

Définition 23 (Première occurrence)**.** *Étant donné une séquence s, soient o_1 et o_2 deux occurrences d'une sous-séquence s' dans s, dont les signatures sont $\langle i_1^1, i_2^1, \ldots, i_m^1\rangle$ et $\langle i_1^2, i_2^2, \ldots, i_m^2\rangle$ respectivement. o_1 est inférieure à o_2, noté $o_1 < o_2$, s'il existe un indice $k \in [1..m]$ tel que pour tout $j \in [1..k-1]$, on a $i_j^1 = i_j^2$, et $i_k^1 < i_k^2$. Enfin, nous appelons la* première occurrence *de s' dans s sa plus petite occurrence par rapport à l'ordre défini précédemment.*

Exemple 25. *Continuons l'exemple 20 où $\langle 1,2\rangle$ et $\langle 1,3\rangle$ sont les signatures des occurrences $o_1 = \langle ac\emptyset\rangle$ et $o_2 = \langle a\emptyset c\rangle$ de la sous-séquence $s' = \langle ac\rangle$ dans $s = \langle(ab)(cd)(ce)\rangle$. Comme $\langle 1,2\rangle$ est inférieure à $\langle 1,3\rangle$, nous obtenons alors $o_1 < o_2$. Enfin, comme o_1 et o_2 sont les deux seules occurrences de s' dans s, cela signifie que o_1 est la première occurrence de s' dans s.*

En pratique, on vérifie notamment si une occurrence de la sous-séquence $s' \sqsubseteq s$ est la première occurrence de s' dans la séquence s. Cela peut être fait efficacement en utilisant la propriété 5 :

Propriété 5. *Étant donné une occurrence o de la sous-séquence $s' \sqsubseteq s$ dont la signature est $\sigma = \langle i_1, i_2, \ldots, i_m\rangle$, o est la première occurrence de s' si et seulement si pour tous $i_j \in \sigma$, il n'y a pas d'indice $l \in [i_{j-1}+1..i_j-1]$ tel que $o[i_j] \subseteq s[l]$ (avec $i_0 = 0$).*

Preuve. Soit $\sigma = \langle i_1, \ldots, i_m\rangle$ la signature d'une occurrence o de $s' \sqsubseteq s$. Nous montrons d'abord que s'il existe $i_j \in \sigma$ et $l \in [i_{j-1}+1..i_j-1]$ tels que $o[i_j] \subseteq s[l]$, alors o n'est pas la première occurrence de s'. Soit $1 \leq i'_1 < i'_2 < \cdots < i'_m \leq n$ la séquence d'indices

définie par $i'_j = l$ et pour tout $k \in [1..m] \setminus \{j\}$, $i'_k = i_k$. Considérons maintenant la liste ordonnée o' de n itemsets définie par $o'[l] = o[i_j]$, $o'[i_j] = \emptyset$ et pour tous $k \in [1..n] \setminus \{l, i_j\}$, $o'[k] = o[k]$. Comme o' est une occurrence de $s' \sqsubseteq s$ et $o' < o$, cela prouve que o n'est pas la première occurrence de s'. Inversement, nous montrons que si o avec la signature σ n'est pas la première occurrence de $s' \sqsubseteq s$, alors il existe $i_j \in \sigma$ et $l \in [i_{j-1} + 1..i_j - 1]$ tel que $o[i_j] \subseteq s[l]$. Par définition, si o n'est pas la première occurrence de s, alors il existe une autre occurrence o' de s' tel que $o' < o$. Donc, nous savons qu'il existe $k \in [1..n]$ tel que $i'_k < i_k$ et pour tout $j \in [1..k-1]$, $i'_j = i_j$. Ainsi, il existe des index $i_k \in \sigma$ et $l = i'_k \in [i'_{k-1}+1..i_k-1] = [i_{k-1}+1..i_k-1]$ tel que $o[i_k] = o[i'_k] \subseteq s[i'_k]$, soit $o[i_k] \subseteq s[l]$. □

Grâce à la propriété 5, il est maintenant facile de tirer uniformément une sous-séquence de norme ℓ dans une séquence s. En tirant au hasard ℓ des positions distinctes entre 1 et $\|s\|$, nous commençons par tirer uniformément une occurrence contenant ℓ items à partir de s. Si cette occurrence est une première occurrence, elle est acceptée et renvoyée. Sinon, nous la rejetons et effectuons un autre tirage aléatoire d'une nouvelle occurrence de s. Bien que NUSSampling repose sur une technique d'échantillonnage par rejet, nous montrons dans la section suivante que le nombre moyen de tirages avant acceptation est calculable. La section expérimentale montre également que ce nombre moyen de tirages s'avère être extrêmement faible pour des bases de données réelles.

Exemple 26. *Dans l'exemple 20, supposons que nous avons tiré les positions d'élément* 1 *et* 5 *dans le séquence* $s = \langle (\mathbf{a}b)(cd)(\mathbf{c}e)\rangle$ *afin de construire une occurrence d'une sous-séquence de* s *de norme* $k = 2$. *De cette façon, nous obtenons l'occurrence* $o = \langle a\emptyset c\rangle$ *de la signature* $\langle 1, 3\rangle$ *de la sous-séquence* $s' = \langle ac\rangle$ *dans* s. *Dans ce cas, comme il existe* $l = 2$ *dans* $[1 + 1..3 - 1]$ *tel que* $o[3] = c \subseteq s[2] = (cd)$, *c'est sûr que* o *n'est pas la première occurrence de* s' *et cette occurrence est donc rejetée.*

5.3.4 Analyse théorique de NUSSampling

Nous étudions maintenant la complexité de notre méthode en distinguant deux phases principales : le prétraitement (où la distribution des sous-séquences selon la norme est vérifiée pour chaque séquence) et le tirage des sous-séquences.

Complexité du prétraitement Le prétraitement est effectué en temps en $O(|\mathcal{S}| \cdot L \cdot M^2 \cdot 2^P \cdot T)$ où L est la longueur maximale de séquence, M est la norme maximale des sous-séquences tirées (s'il n'y a pas de contrainte de norme, M est égale à la norme maximale des séquences dans $\mathcal{S}$), P est la taille maximale des ensembles de positions $L(s^{i-1}, s[i])$ et T est la taille maximale d'un ensemble dans une séquence. Il est important de noter que $P \leq L$ peut être très petit en pratique (voir la section suivante) et que ce prétraitement (ligne 1 de l'algorithme 4) n'est réalisé qu'une seule fois avant la phase de tirage (où un grand nombre de sous-séquences sont tirées de $\mathcal{S}$). De plus, il est facile de voir que si l'ensemble des données $\mathcal{S}$ ne contient que des séquences d'*items* (et non des séquences d'*itemsets*), alors nous avons $P = 1$. Ainsi, dans ce cas, le prétraitement peut être effectué en temps polynômial $O(|\mathcal{S}| \cdot L \cdot M^2 \cdot T)$.

Complexité du tirage Le tirage des sous-séquences est moins coûteux. Tout d'abord, le tirage d'une séquence (ligne 4 de l'algorithme 4) est réalisé en $O(\log|\mathcal{S}|)$. Il est plus difficile d'estimer la complexité dans le pire des cas pour le tirage d'une sous-séquence car le nombre de rejets n'est pas borné. Néanmoins, un bon moyen de mesurer l'efficacité de l'approche est de calculer le nombre moyen de tirages requis, noté $\eta_u(\mathcal{S})$, pour tirer une sous-séquence de $\mathcal{S}$ suivant une utilité fondée sur la norme u. Intuitivement, $\eta_u(\mathcal{S})$ dépend à la fois de la probabilité qu'une séquence $s \in \mathcal{S}$ soit tirée et du nombre moyen de tirages requis, noté $\eta_u(s)$, pour trouver une première occurrence d'une sous-séquence de s. La propriété suivante montre comment ces termes peuvent être calculés :

Propriété 6 (Nombre moyen de tirages)**.** *Etant donné une fonction d'utilité fondée sur la norme u, le nombre moyen de tirages pour l'acceptation d'une sous-séquence dans la base de données séquentielles $\mathcal{S}$ est défini par :* $\eta_u(\mathcal{S}) = \sum_{s\in\mathcal{S}} \frac{\omega_u(s)}{\sum_{s'\in\mathcal{S}}\omega_{util}(s')} \times \eta_u(s)$ *où* $\eta_u(s) = \frac{\sum_{k=0}^{\|s\|}\binom{\|s\|}{k}\times f_u(k)}{\omega_u(s)}$ *et* $\omega_u(s) = \sum_{k=0}^{\|s\|}\Phi_k(s)\times f_u(k)$.

Preuve. En utilisant l'algorithme 4, il est clair que $\eta_u(\mathcal{S}) = \sum_{s\in\mathcal{S}}\pi(s)\times\eta_u(s)$ avec $\pi(s) = \frac{\omega(s)}{\sum_{s'\in\mathcal{S}}\omega(s')}$. Donc, nous avons $\eta_u(s) = \sum_{k\in[0..\|s\|]}\pi(k/s)\times N_k(s)$ où $N_k(s)$ est le nombre moyen de tirages nécessaires pour obtenir une sous-séquence s' de s telle que $\|s'\| = k$. Quand nous tirons une sous-séquence s' de norme k, la probabilité que cette sous-séquence soit acceptée (parce qu'elle est une première occurrence) est $\pi_a^k(s) = \frac{\Phi_k(s)}{\binom{\|s\|}{k}}$. Ainsi, nous avons $N_k(s) = \sum_{i=1}^{\infty} i\times(1-\pi_a^k(s))^{i-1}\times\pi_a^k(s) = \pi_a^k(s)\times\sum_{i=1}^{\infty} i\times(1-\pi_a^k(s))^{i-1} = \pi_a^k(s)\times\frac{1}{\pi_a^k(s)^2} = \frac{1}{\pi_a^k(s)}$. Il s'en suit que $\eta_u(s) = \sum_{k\in[0..\|s\|]}\pi(k/s)\times N_k(s) = \sum_{k\in[0..\|s\|]}\frac{\omega_k(s)}{\omega(s)}\times\frac{\binom{\|s\|}{k}}{\Phi_k(s)}$. Puisque $\omega_k(s) = \Phi_k(s)\times f_u(k)$, nous obtenons finalement $\eta_u(s) = \frac{\sum_{k\in[0..\|s\|]}\binom{\|s\|}{k}\times f_u(k)}{\omega(s)}$. □

5.4 Expérimentations

Cette étude expérimentale vise à évaluer l'efficacité de notre approche et l'intérêt des sous-séquences échantillonnées à l'aide de différentes mesures d'intérêt fondées sur la norme. Plus précisément, nous considérons les trois mesures d'intérêt ci-dessous suivantes :

- M**-freq** combine la mesure de fréquence avec la contrainte d'utilité basée sur la norme définie pour chaque séquence s par $u_{\geq 1}(s)\times u_{\leq M}(s)$,
- M**-area** combine la mesure de fréquence avec la fonction d'utilité définie pour chaque séquence s par $u_{area}(s)\times u_{\geq 1}(s)\times u_{\leq M}(s)$, et
- α**-freq** combine la mesure de fréquence avec l'utilité de décroissance exponentielle $u_{decay}(s) = \alpha^{\|s\|}$.

La section 5.4.1 se concentre sur la rapidité de NUSSampling et sa capacité à tirer des motifs qui n'appartiennent pas à la longue traîne. En particulier, nous comparons l'impact de l'utilisation de l'utilité fondée sur la norme $u_{\leq M}$ ou de l'utilité de décroissance exponentielle u_{decay}. La section 5.4.4 compare les motifs séquentiels échantillonnés par NUSSampling dans le contexte de la classification de séquence, où l'accuracy effectue une mesure objective.

Notez que le prototype de notre méthode est implémenté en Python et toutes les expériences sont effectuées sur un CPU 2 coeurs à 2,71 GHz avec 12 Go de RAM. Tous les jeux de données expérimentaux utilisés, ainsi que le code source (sous la licence GPLv3), sont disponibles sur https://github.com/LDIOPBSF/NUSSampling.

5.4.1 Analyse de la méthode NUSSampling

Cette section expérimentale évalue la rapidité de notre méthode et l'impact de différentes utilités fondées sur la norme sur les motifs échantillonnés. A cette fin, nous utilisons 15 jeux de données, dont 11 jeux de données réels [1] et 4 jeux de données synthétiques générés par le générateur de données IBM [2]. Le principal intérêt de l'utilisation de jeux de données synthétiques est d'avoir des exemples de séquences avec des itemsets au lieu de séquences contenant uniquement des items (c'est-à-dire avec $T > 1$). Le tableau 5.4 répertorie les statistiques de base de tous les jeux de données : le nombre de séquences, le nombre d'items, la norme maximale/moyenne des séquences, la taille maximale des ensembles de positions P, la taille maximale des itemsets T et le nombre de classes. Le tableau 5.5 compare le nombre moyen de tirages par sous-séquence requis pour extraire un motif lorsqu'il existe une mesure d'intérêt fondée sur la norme $u_{\geq 1} \times u_{\leq M}$ avec $M \in \{1, 2, 3, 5, 7\}$. Le tableau 5.5 est obtenu en utilisant la propriété 6. Il montre que les jeux de données `bms` et `sign` ne contiennent pas plusieurs occurrences dans une même sous-séquence, tandis que le nombre d'occurrences multiples augmente avec M pour les autres jeux de données.

Table 5.4 – Statistiques des jeux de données utilisés

$\mathcal{S}$	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{I}\rvert$	$\lVert S\rVert_{max}$	$\lVert S\rVert_{mean}$	P	T	$\lvert\mathcal{C}\rvert$
bms	59,601	497	267	2.5	1	1	–
sign	730	267	94	52.0	1	1	–
D10K5S2T6I	10,000	6	70	10.3	7	6	–
D10K6S3T10I	10,000	10	92	15.9	10	6	–
D100K5S2T6I	100,000	6	72	8.5	7	6	–
D100K6S2T6I	100,000	6	83	10.4	8	9	–
aslbu	441	132	27	7.5	1	1	7
aslgt	3,493	87	88	22.8	1	1	40
auslan	200	12	24	10.0	1	1	10
blocks	210	8	12	6.7	1	1	8
context	240	48	123	45.2	1	1	5
pioneer	160	92	50	21.1	1	1	3
skater	530	41	120	25.1	1	1	6
speed	530	41	260	64.5	1	1	7
reuters	5 459	14 577	533	67.3	1	1	8

1. Les jeux de données `bms` et `sign` sont disponibles sur http://www.philippe-fournier-viger.com/spmf et d'autres sur http://www.mybytes.de/#data
2. https://github.com/zakimjz/IBMGenerator

TABLE 5.5 – Nombre moyen de tirages par sous-séquence

$\mathcal{S}$	M=1	M=2	M=3	M=5	M=7
bms	1.0	1.0	1.0	1.0	1.0
sign	1.0	1.0	1.0	1.0	1.0
D10K5S2T6I	4.0	7.0	11.4	23.5	38.4
D10K6S3T10I	3.9	6.7	10.4	18.5	25.7
D100K5S2T6I	3.6	5.8	8.5	14.9	23.9
D100K6S2T6I	4.0	7.0	11.1	21.4	32.4
aslbu	1.2	1.3	1.3	1.4	1.5
aslgt	1.4	1.9	2.8	5.8	12.6
auslan	2.6	3.5	4.8	8.2	10.4
blocks	1.4	1.9	2.4	3.7	4.5
context	2.5	6.3	17.5	159.5	1,652.7
pioneer	1.1	1.3	1.5	2.0	2.6
skater	1.7	3.3	8.3	81.3	837.0
speed	3.3	7.0	15.9	99.4	554.2
reuters	1.6	2.4	3.4	5.9	8.6

5.4.2 Rapidité de NUSSAMPLING pour le prétraitement et le tirage

Dans cette section, nous analysons la vitesse à laquelle NUSSAMPLING effectue le prétraitement d'un jeu de données et le tirage d'un motif séquentiel en considérant trois mesures d'intérêt différentes : M-freq, M-area et α-freq.

Contrainte de norme La figure 5.3 trace le temps d'exécution moyen de notre méthode en distinguant le temps de prétraitement (côté gauche) et le nombre moyen de tirages d'un motif séquentiel (côté droit) en utilisant M-freq mesuré avec $M \in \{1, 2, 3, 5, 7\}$. Notez que nous ne rapportons pas ces résultats pour la mesure M-area car elle a exactement le même comportement que la mesure M-freq pour toutes les expériences. Comme prévu, le temps de prétraitement augmente avec la taille du jeu de données, la taille maximale P des ensembles de positions, la taille maximale T d'un itemset dans une séquence et la norme maximale M des sous-séquences tirées. Cependant, même pour la base de données D100K6S2T6I qui est grande, le temps d'exécution du prétraitement (qui peut être préparé hors ligne) est tout à fait raisonnable (moins de 80 secondes). En ce qui concerne la phase d'échantillonnage, quel que soit le jeu de données, la mesure d'intérêt et la norme maximale M, le temps d'exécution est toujours inférieur à 1.5 milliseconde, malgré un nombre moyen de tirages $\eta_{[\mu,M]}(\mathcal{S}) > 1$ (et donc du rejet).

Décroissance exponentielle La figure 5.4 trace le temps de prétraitement (côté gauche) et d'échantillonnage (côté droit) de notre méthode en utilisant la mesure de fréquence α-freq avec différentes valeurs de $\alpha \in \{0.05, 0.1, 0.2,\ 0.3, 0.4, 0.5\}$. Étant donné un jeu de données, nous voyons d'abord que le temps de prétraitement reste constant avec α. En ef-

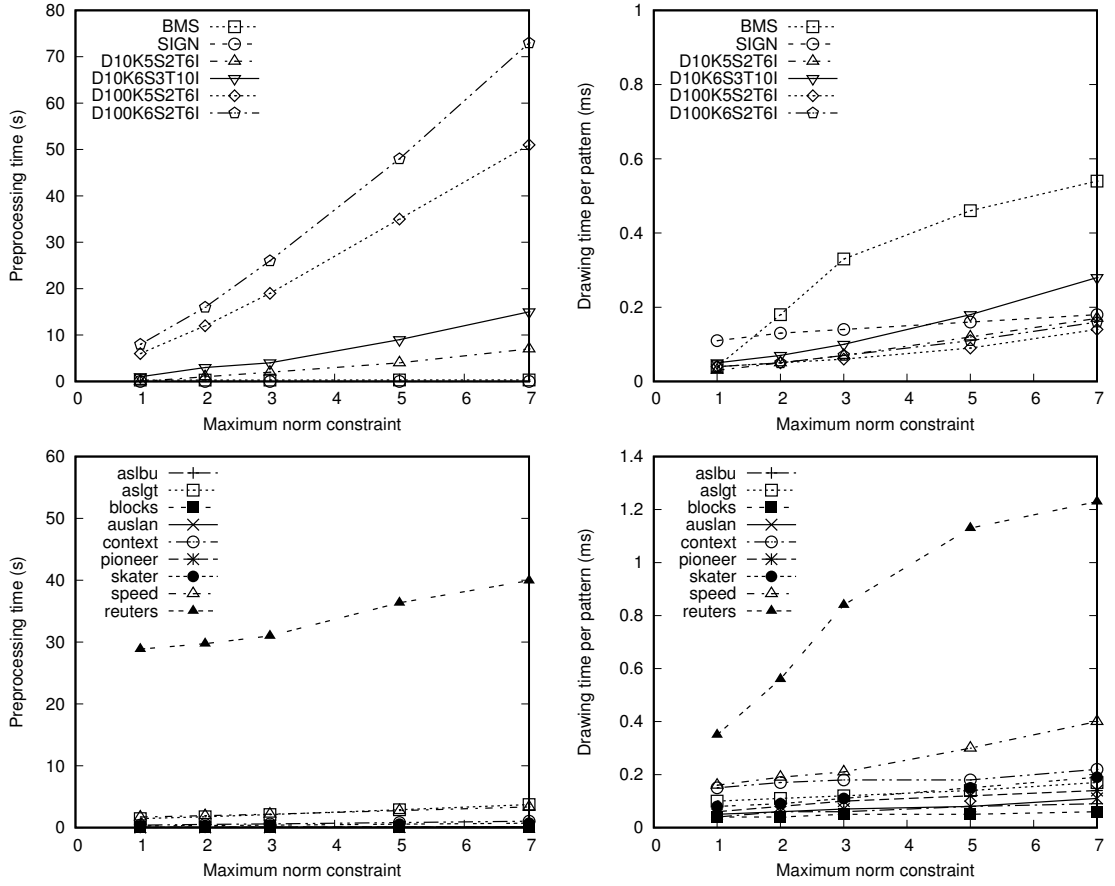


FIGURE 5.3 – Temps d'exécution pour l'échantillonnage en sortie de motifs séquentiel avec M-frequency

fet, le calcul du poids d'une séquence dans un jeu de données est clairement indépendant de α, c'est-à-dire sans contrainte. Pour chaque séquence $s \in \mathcal{S}$, il suffit de calculer le nombre de sous-séquences distinctes de s. Inversement, le côté droit de la figure 5.4 montre que la vitesse d'échantillonnage varie avec α. Lorsque α augmente, la probabilité de tirer une sous-séquence avec une norme plus élevée augmente, et le temps de calcul pour tirer une occurrence d'une sous-séquence augmente avec sa norme. Cependant, lorsque nous tirons une occurrence, la probabilité qu'il ne s'agisse pas d'une première occurrence (c-à-d qu'il y ait rejet) diminue avec la norme de la séquence tirée. Cela explique pourquoi le temps d'échantillonnage commence à augmenter avec α, puis diminue. Enfin, il est intéressant de voir que les temps de prétraitement et d'échantillonnage sont plus faibles lorsque nous utilisons une contrainte de norme. Plus précisément, en comparant la figure 5.3 avec la figure 5.4, nous observons qu'ils sont au moins deux fois plus faibles lorsqu'une contrainte de norme maximale $u_{\leq M}$ est utilisée avec $M \in [1..7]$.

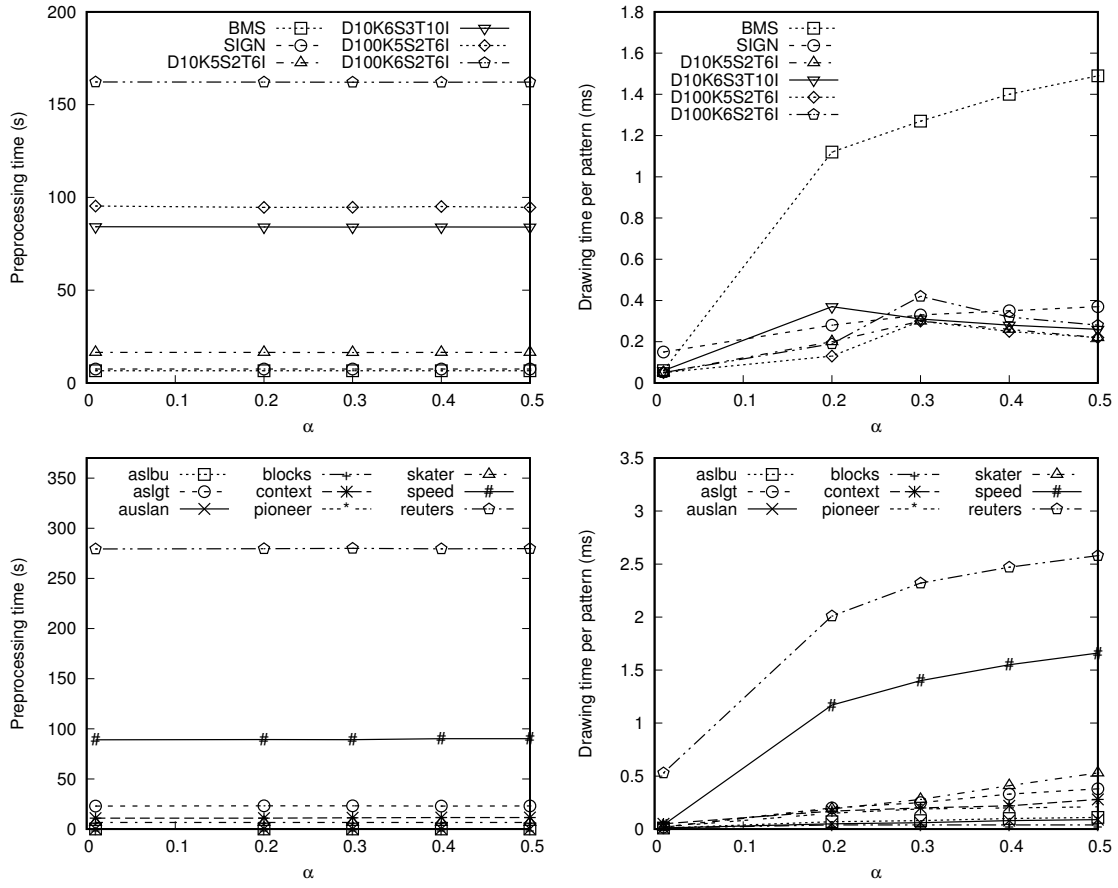


FIGURE 5.4 – Temps d'exécution pour l'échantillonnage en sortie de motifs séquentiels avec α-frequency

5.4.3 Distribution des motifs échantillonnés

Nous considérons maintenant l'avantage d'utiliser une fonction d'utilité basée sur des normes pour limiter l'effet de la longue traîne.

Contrainte de norme Nous comparons d'abord la distribution des motifs échantillonnés en utilisant les mesures de fréquence et d'aire avec ou sans utiliser une contrainte de norme maximale $u_{\leq M}$ (avec $M \in \{4, 7\}$). Les figures 5.5 et 5.6 représentent la distribution de 10,000 motifs séquentiels échantillonnés selon les mesures M-freq et M-area pour différents jeux de données, avec ou sans contrainte.

Pour tous les jeux de données, sans contrainte, la méthode d'échantillonnage ne renvoie que des motifs présentant un intérêt très faible. Avec la mesure de fréquence, on remarque sur la figure 5.5 que sans contrainte, la plupart des séquences échantillonnées ont une fréquence égale à 1. De même, avec la mesure d'aire, on voit aussi que dans la plupart des cas, sans contrainte, la méthode d'échantillonnage renvoie des séquences avec des aires très basses. Notez que ce n'est pas le cas avec le jeu de données `bms`, car cet ensemble de données contient une séquence très longue, et que nous échantillonnons principalement des sous-séquences très longues, donc avec une grande aire, de cette séquence. Ainsi, même si la fréquence de ces sous-séquences est très faible, la moyenne de leur norme est grande. Inversement, avec une contrainte de norme maximale, la méthode d'échantillonnage renvoie des motifs séquentiels avec des fréquences ou des aires nettement plus élevées, ce qui montre l'importance d'introduire des contraintes sur la norme pour éviter le problème de la longue traîne. Notez que pour `sign`, la norme maximale de 7 n'est pas suffisante pour renvoyer des motifs échantillonnés avec une fréquence supérieure à 1. Une norme d'au plus 4 est nécessaire pour que les fréquences des sous-séquences de l'échantillon augmentent.

Décroissance exponentielle La figure 5.7 montre la distribution de 10,000 motifs séquentiels échantillonnés selon la mesure α-freq avec différentes valeurs de α. Lorsque α est égal à 1.0, cela signifie qu'il n'y a pas de filtrage de décroissance exponentielle. Par conséquent, les résultats sont aussi mauvais que ceux obtenus sans contrainte de norme maximale. En outre, il est facile de voir que lorsque α diminue, la fréquence des motifs séquentiels tirés augmente comme c'était le cas lorsque M diminue. Cependant, les sous-séquences tirées n'ont pas exactement la même distribution que celles obtenues avec des contraintes de norme. En effet, la figure 5.8 compare la distribution de 10,000 motifs échantillonnés par rapport à leur norme en considérant M-freq, M-area ($M \in \{4, 7\}$) et α-freq ($\alpha \in \{0.10, 0.05, 0.01\}$) comme mesures d'intérêt pour deux bases de données séquentielles (`sign` et `D10K6S3T10I`). Tout d'abord, nous voyons sur les graphiques des deux premières lignes qu'avec M-freq et M-area, la plupart des motifs échantillonnés ont une norme égale à la contrainte de norme maximale M. De plus, il n'y a presque pas de différence entre les distributions des motifs avec M-freq et M-area. Par comparaison, nous observons dans la figure 5.8 (dernière ligne) qu'avec la mesure de α-freq, la diversité des normes des sous-séquences échantillonnées est plus élevée lorsque α n'est pas trop faible (i.e, 0.10 ou 0.05). Si la valeur de α est faible (c'est-à-dire 0.01 dans nos expériences), nous n'obtenons que des sous-séquences avec des normes très basses entre 1 et 4.

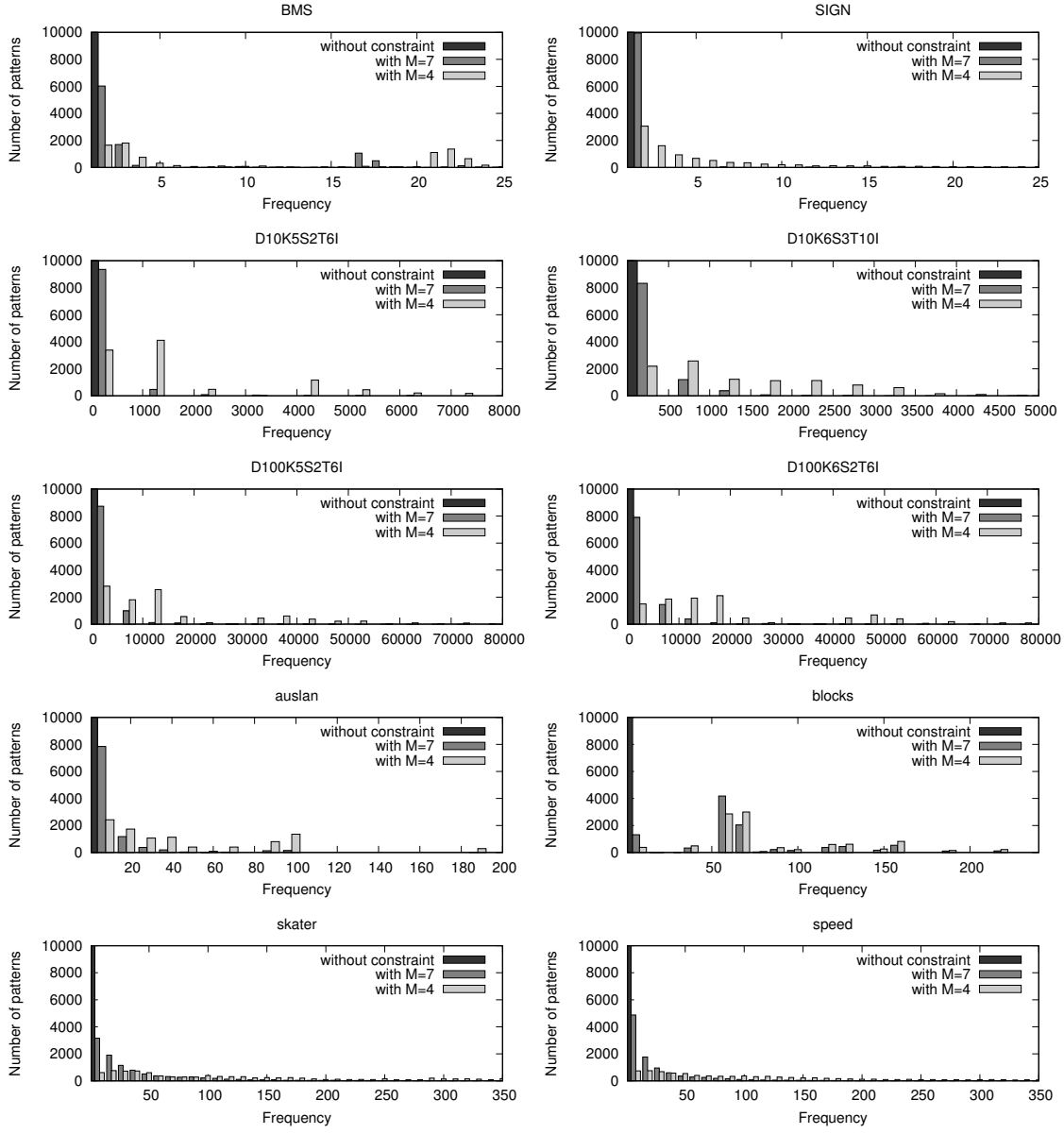


FIGURE 5.5 – Distribution de 10,000 motifs séquentiels selon M-freq

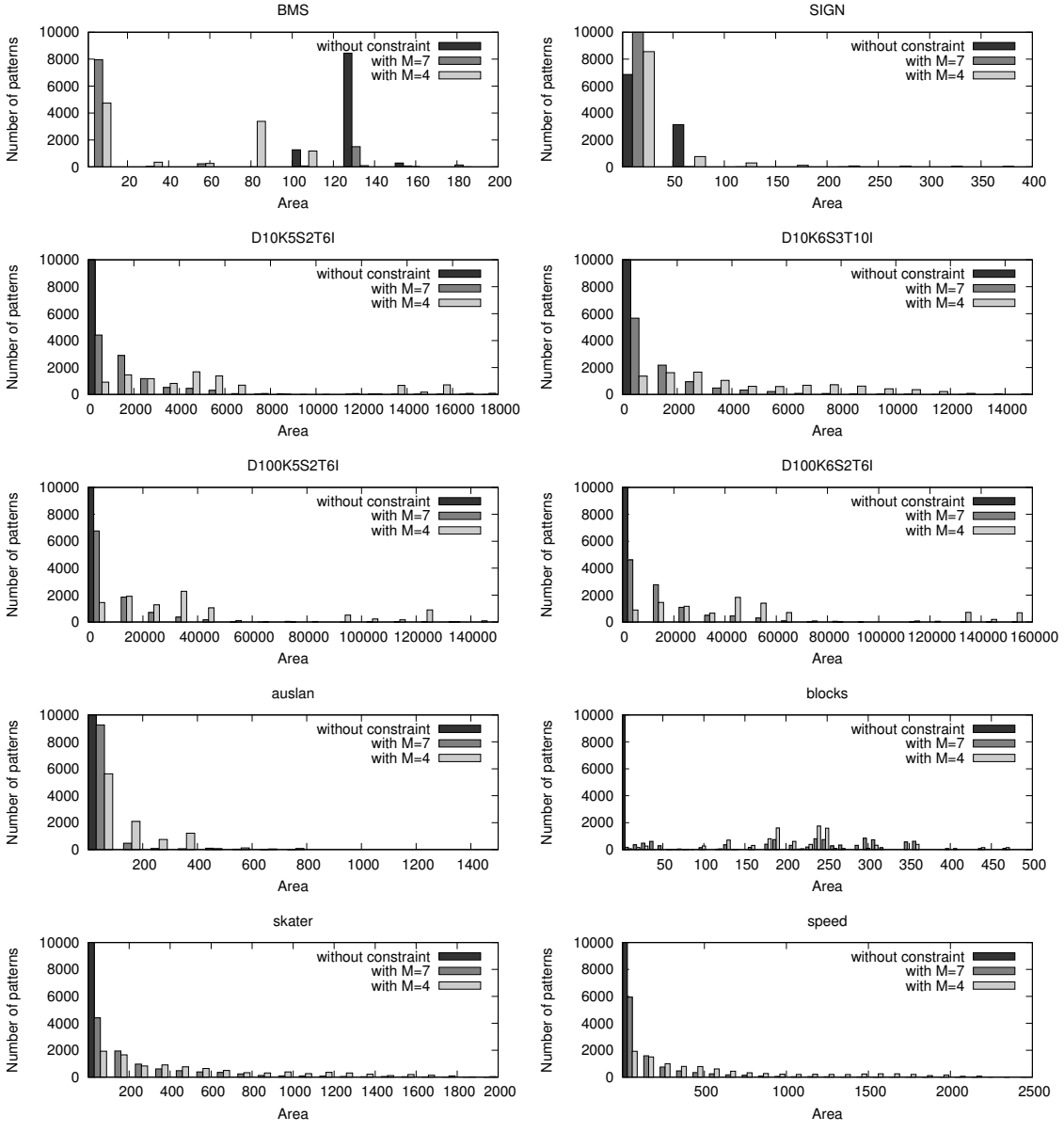


FIGURE 5.6 – Distribution de 10,000 motifs séquentiels selon M-area

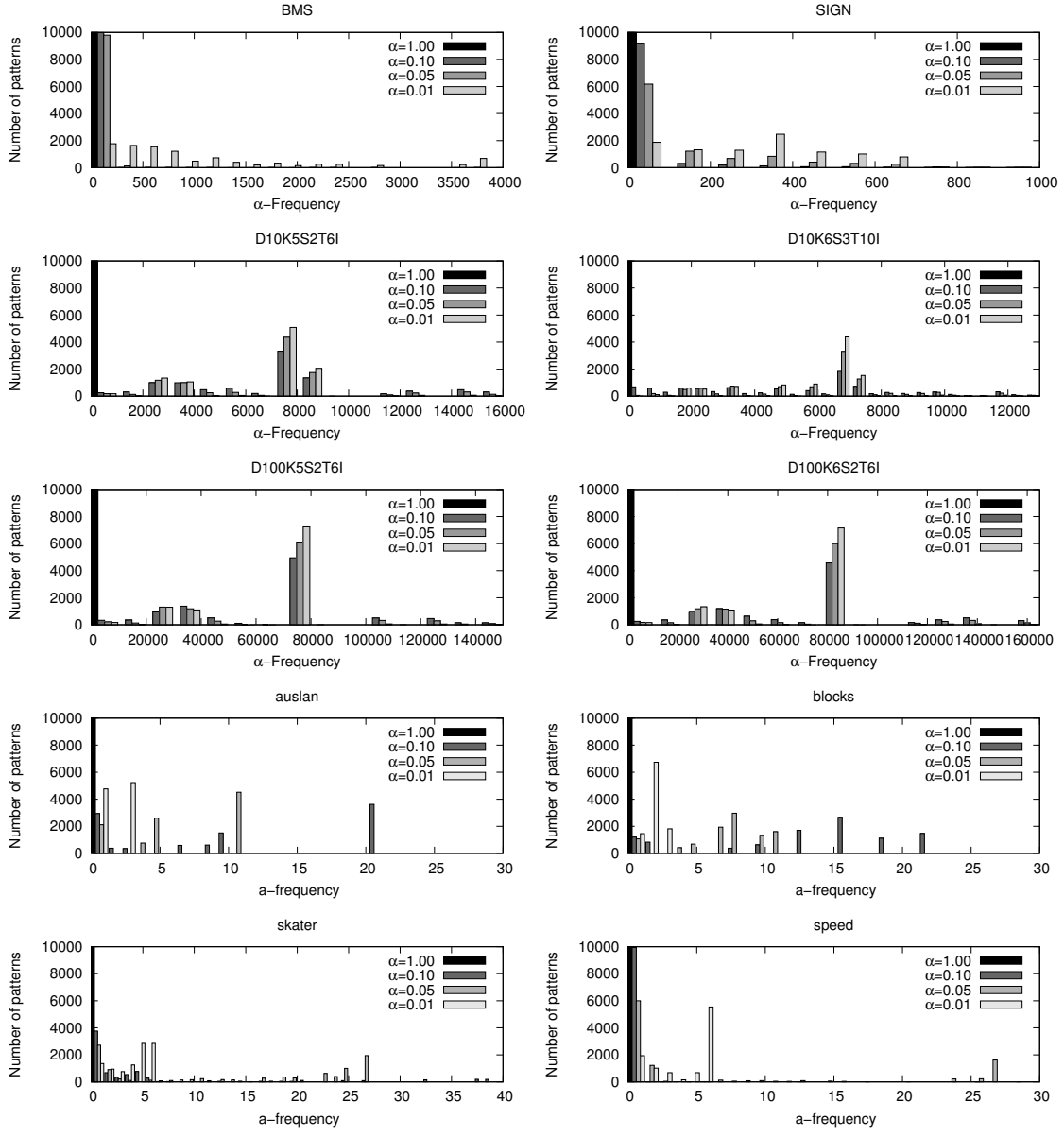


FIGURE 5.7 – Distribution de 10,000 motifs séquentiels selon α-freq

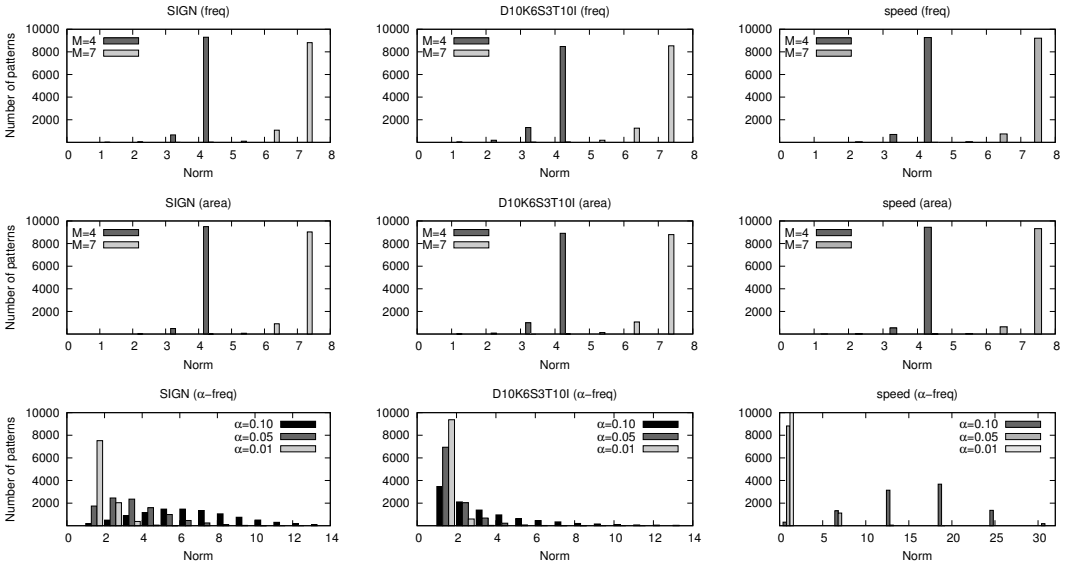


FIGURE 5.8 – Distribution de 10,000 motifs séquentiels selon la norme avec différentes mesures d'intérêt fondées sur la norme

5.4.4 Performance des classifieurs basés sur l'échantillonnage en sortie de motifs

Cette section montre comment les sous-séquences échantillonnées peuvent être utilisées pour construire des classifieurs associatifs dédiés aux séquences et l'avantage d'utiliser des contraintes de norme ou des décroissances exponentielles pour obtenir de meilleurs modèles de classification. Notre méthode de classification, appelée NUSSAMPLING+SVM, est l'approche reposant sur deux phases présentées dans la section 2.3.1.

Importance de la contrainte de norme et de la décroissance exponentielle Comme décrit dans les sections précédentes, la contrainte d'utilité fondée sur la norme $u_{\leq M}$ et u_{decay} peut être utilisée pour limiter la longueur maximale des sous-séquences échantillonnées. Maintenant, nous évaluons l'impact de l'utilisation de ces utilités fondées sur la norme pour améliorer les performances de nos classifieurs. Plus précisément, nous construisons d'abord des classifieurs en utilisant des sous-séquences échantillonnées avec des mesures M-freq ou M-area en considérant différentes valeurs de $M \in [1..10]$. Ensuite, nous construisons également des classifieurs en utilisant des sous-séquences échantillonnées avec la mesure α-freq en considérant les valeurs de $\alpha \in \{0.01, 0.025, 0.05, 0.075, 0.1, 0.5\}$. Notez que nous réduisons le temps de prétraitement pour l'utilité de décroissance exponentielle en ajoutant une contrainte de norme maximale $u_{\leq 10}$. Cette contrainte de norme n'a aucun effet sur les sous-séquences échantillonnées car même si aucune contrainte n'était prise en compte, la probabilité de tirer une sous-séquence avec une norme supérieure à 10 serait presque nulle avec $\alpha \leq 0.5$.

Pour toutes les expériences et chaque jeu de données, nous utilisons un t-test de Student

apparié différent (avec un niveau de confiance $\beta = 95\%$) pour évaluer si la meilleure accuracy moyenne (obtenue avec la valeur optimale de M ou α) est significativement différente des autres accuracies moyennes (obtenues avec des valeurs non optimales des paramètres M ou α). En effet, les valeurs optimales des paramètres M ou α ne sont pas nécessairement les mêmes pour chaque jeu de données.

TABLE 5.6 – Impact de la contrainte de norme avec M-freq

Dataset	M=1	M=2	M=3	M=5	M=7	M=10	**Best**
aslbu	**0.649***	0.601	**0.608**	0.539	0.396	0.373	0.649
aslgt	0.668	**0.688***	**0.680**	0.634	0.505	0.364	0.688
auslan	0.230	0.250	**0.320**	**0.320**	**0.330***	**0.330***	0.330
blocks	0.857	**1.000***	**0.995**	**0.995**	**0.995**	**0.995**	1.000
context	**0.984***	**0.984**	**0.971**	**0.975**	**0.967**	**0.959**	0.984
pioneer	**1.000***	**0.975**	**0.969**	0.858	0.691	0.656	1.000
skater	0.883	**0.930**	**0.944***	0.919	0.889	0.874	0.944
speed	0.257	0.281	0.306	**0.366***	**0.326**	0.301	0.366
reuters	**0.949***	0.901	0.765	0.531	0.523	0.519	0.949
Moyenne	0.720	**0.734**	0.729	0.682	0.625	0.597	0.734

TABLE 5.7 – Impact de la contrainte de norme avec M-area

Dataset	M=1	M=2	M=3	M=5	M=7	M=10	**Best**
aslbu	**0.649***	**0.623**	0.588	0.452	0.370	0.375	0.649
aslgt	0.668	**0.688***	0.680	0.634	0.505	0.359	0.688
auslan	0.250	0.255	**0.355***	**0.350**	**0.345**	**0.325**	0.355
blocks	0.857	**1.000***	**0.995**	**0.990**	**0.995**	**0.991**	1.000
context	**0.964**	**0.984***	**0.966**	**0.983**	**0.971**	**0.971**	0.984
pioneer	**0.994***	**0.975**	0.962	0.801	0.701	0.645	0.994
skater	0.887	0.930	**0.945***	**0.925**	0.866	0.805	0.945
speed	0.266	0.273	0.339	**0.371***	**0.329**	0.272	0.371
reuters	**0.952***	0.904	0.545	0.522	0.520	0.518	0.952
Moyenne	0.721	**0.737**	0.708	0.670	0.622	0.584	0.737

Contrainte de norme : Compte tenu des mesures M-freq ou M-area, les tableaux 5.6 et 5.7 montrent que l'accuray de NUSSAMPLING+SVM dépend clairement de la contrainte de norme. La meilleure accuracy moyenne est indiquée pour chaque jeu de données par une étoile, tandis que les accuracies comparables (identifiées avec un test de Student apparié) sont en gras. Alors que la taille totale de l'échantillon est fixe (ici, 10,000 motifs), nous pouvons voir que la meilleure performance de classification est généralement obtenue lorsque le seuil de norme maximale est strictement supérieur à 1 et inférieur à 10. Cependant, ce résultat est moins significatif lorsque le problème de classification à résoudre est

assez simple (pour une accuracy supérieure à 95%), c'est-à-dire pour les jeux de données `blocks`, `context`, `pioneer` et `reuters`. Pour ces jeux de données, une bonne performance peut souvent être obtenue avec une contrainte de norme maximale M égale à 1 (c'est-à-dire que les variables sont des sous-séquences avec un seul élément). Pour les autres ensembles de données, les performances des classifieurs diminuent avec M lorsque M est supérieur à sa valeur optimale. Pour cette raison, il est important de considérer des seuils de norme maximale pour construire des classifieurs efficaces. En particulier, les performances des classifieurs qui seraient obtenues sans tenir compte des contraintes de norme (c'est-à-dire avec M tendant vers l'infini) seraient donc très faibles.

Décroissance exponentielle : Pour la mesure α-freq, le tableau 5.8 montre également que les performances de NUSSampling+SVM dépendent clairement de la valeur du paramètre α. Tout d'abord, nous pouvons observer que si la valeur de α est trop élevée (c'est-à-dire $\alpha = 0.5$ ou plus), alors les performances des classifieurs ne sont généralement pas satisfaisantes. Ce phénomène peut être facilement expliqué car lorsque la valeur de α est trop élevée, nous échantillonnons principalement des sous-séquences très longues avec des fréquences très basses. Dans ce cas, par rapport à la recherche de la contrainte de norme maximale optimale M, on pourrait penser que trouver la valeur optimale de α est plus difficile car c'est un paramètre réel. Cependant, nous pouvons observer dans le tableau 5.8 que l'accuracy n'est pas très sensible à la valeur de α entre 0.01 et 0.1.

Table 5.8 – Impact de la décroissance exponentielle avec α-freq

Dataset	α=0.010	α=0.025	α=0.050	α=0.075	α=0.100	α=0.500	**Best**
`aslbu`	**0.635***	0.602	**0.618**	**0.618**	**0.611**	**0.592**	0.635
`aslgt`	**0.683***	0.681	**0.683**	**0.679**	**0.682**	0.471	0.683
`auslan`	0.265	**0.335**	**0.335**	**0.355***	**0.345**	**0.325**	0.355
`blocks`	**0.995**	**0.995**	**0.990**	**0.995**	**1.000***	**0.995**	1.000
`context`	**0.987***	**0.983**	**0.975**	**0.975**	**0.975**	**0.975**	0.987
`pioneer`	**0.994***	**0.994***	**0.981**	**0.975**	**0.987**	0.688	0.994
`skater`	0.925	**0.934**	**0.947***	**0.940**	**0.943**	0.843	0.947
`speed`	0.289	**0.319**	**0.345**	**0.347***	**0.328**	0.260	0.347
`reuters`	**0.950***	0.899	0.544	0.520	0.519	0.519	0.950
Moyenne	0.747	**0.749**	0.713	0.712	0.710	0.630	0.749

Comparaison avec les méthodes Top-k Cette section compare NUSSampling avec M-freq avec l'algorithme TKS [Fournier-Viger *et al.*, 2013b], qui renvoie les k motifs séquentiels les plus fréquents ayant une norme inférieure à M. Fait intéressant, ces deux approches utilisent un paramètre k pour contrôler le nombre de motifs extraits et un paramètre M pour limiter leur norme. Pour construire des classifieurs, les mêmes jeux de données sont utilisés pour extraire des descripteurs par TKS et NUSSampling+SVM. Ensuite, comme avec notre approche, nous utilisons l'algorithme SMO pour construire un classifieur SVM. Les approches de classification sont donc extrêmement comparables car seules les caractéristiques extraites diffèrent.

TABLE 5.9 – Comparaison d'accuracy entre NUSSAMPLING (M-freq) et TKS

	NUSSAMPLING (M-freq)		*TKS*	
Dataset	Optimal M	Best accuracy	Optimal M	Best accuracy
`aslbu`	M=1	**0.649***	M=1	**0.623**
`aslgt`	M=2	**0.688***	M=2	0.659
`auslan`	M=7	**0.330***	M=1	0.230
`blocks`	M=2	**1.0***	M=1	**0.976**
`context`	M=1	**0.984***	M=1	**0.980**
`pioneer`	M=1	**1.0***	M=1	**0.994**
`skater`	M=3	**0.944***	M=1	0.870
`speed`	M=5	**0.366***	M=1	0.249
`reuters`	M=1	**0.950***	M=1	**0.950***

Le tableau 5.9 rapporte les accuracies de notre approche et celles de TKS sur tous les jeux de données de classification. Plus précisément, pour les deux approches, en utilisant la validation croisée, nous sélectionnons la valeur optimale du paramètre M (M variant entre 1 et 7) et rapportons la meilleure précision obtenue. Notez que pour TKS, la valeur optimale du paramètre M est toujours 1 sauf sur `aslgt`. Ensuite, en utilisant un test de Student apparié (avec un niveau de confiance $\beta = 95\%$), nous évaluons si les meilleures accuracies obtenues avec NUSSAMPLING+SVM et TKS sont comparables ou non.

Tout d'abord, nous notons que NUSSAMPLING+SVM a toujours une meilleure accuracy que TKS (sauf sur `reuters` où les meilleures accuracies sont égales). Ensuite, nous observons que l'écart d'accuracy entre les deux méthodes est généralement plus élevé lorsque la meilleure accuracy de NUSSAMPLING+SVM est pour une valeur M supérieure à 1 (sauf sur `blocks`). Dans ce cas, une bonne classification nécessite des motifs complexes basées sur des combinaisons d'items. Cet espace de variables devient alors plus grand et l'échantillonnage en sortie de motifs couvre mieux cet espace de variables que TKS. Pour cette raison, dans 4 jeux de données où la meilleure accuracy nécessite de définir $M > 1$ (`aslgt`, `auslan`, `skater` et `speed`), l'accuracy de NUSSAMPLING+SVM est nettement meilleure que l'accuracy obtenue par TKS.

Comparaison avec les méthodes de l'état de l'art Nous comparons enfin l'accuracy de NUSSAMPLING+SVM avec les résultats de 7 méthodes de classification de séquence de l'état de l'art rapportées dans [Egho *et al.*, 2017] comme références par rapport aux mêmes jeux de données : MISERE, SQS, GOKRIMP, CSPADE, SCII et DEFFED. En utilisant les trois mesures M-freq, M-area et α-freq, la figure 5.9 montre que les meilleures précisions obtenues par NUSSAMPLING+SVM (colonne **Best** du tableau 5.6, 5.7 et 5.8) sont comparables à d'autres méthodes de classification de séquences basées sur des algorithmes d'extraction de motifs rapportées dans [Egho *et al.*, 2017], même mieux pour les jeux de données `auslan`, `context`, `speed` et `skater`.

Pour comparer plus précisément les différentes méthodes, nous appliquons le test de Friedman et un test Nemenyi post-hoc comme suggéré par [Demšar, 2006] pour des compa-

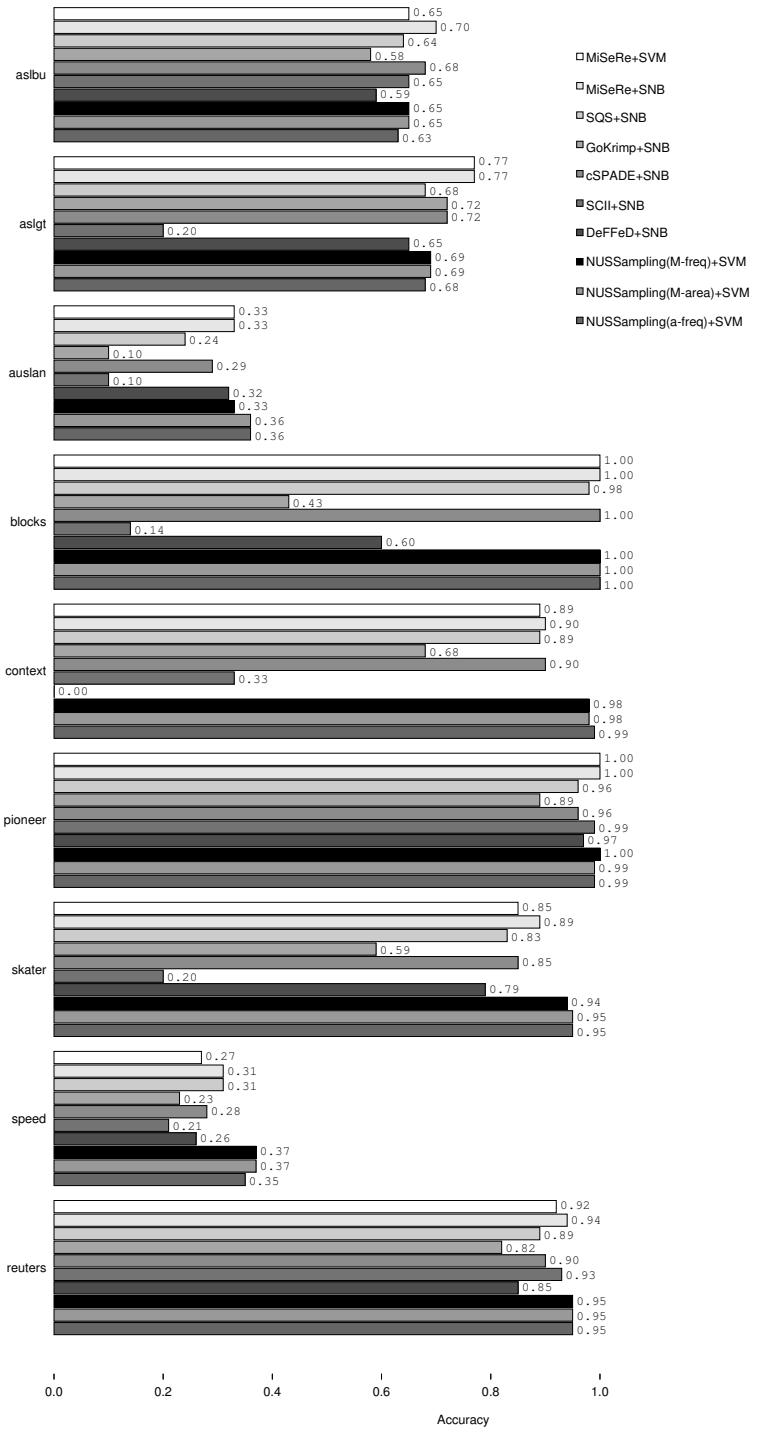


FIGURE 5.9 – Comparaison des accuracies entre NUSSAMPLING avec SVM et les méthodes de classsification de données séquentielles de l'état de l'art

raisons de classifieurs sur plusieurs jeux de données (avec un niveau de confiance $\beta = 95\%$ pour tous les tests). Premièrement, le test de Friedman est utilisé pour évaluer si les rangs moyens des différents classifieurs sont significativement différents du rang moyen attendu sous l'hypothèse nulle. Dans notre cas, puisque $F_F = 11.06$ est supérieur à la valeur critique 2.01 (obtenue pour $\beta = 95\%$), nous rejetons l'hypothèse nulle que toutes les méthodes sont comparables. Ensuite, nous utilisons le test de Nemenyi pour des comparaisons par paire. Avec $p = 0.05$, la différence critique CD est égale à 4.52. Ainsi, nous pouvons distinguer dans la figure 5.10 deux groupes de méthodes : NUSSampling+SVM, MiSeRe, cSPADE et SQS construisent de meilleurs classifieurs que SCII, GoKrimp et DeFFeD.

Ainsi, même si le but de cette thèse n'est pas de proposer une nouvelle méthode de classification de séquences, ces expériences montrent comment l'échantillonnage de sous-séquences peut être utilisé pour construire des classifieurs, et que notre méthode NUSSampling+SVM est compétitive avec les meilleures méthodes de la littérature. Enfin, il est important de rappeler que la complexité de notre méthode d'échantillonnage est très faible (uniquement linéaire avec $|\mathcal{S}|$ lors de l'étape de prétraitement, et logarithmique avec $|\mathcal{S}|$ pendant la phase de tirage), ce qui signifie que notre méthode pourrait être utilisée avec des jeux de données plus grands que les jeux de données de notre benchmark.

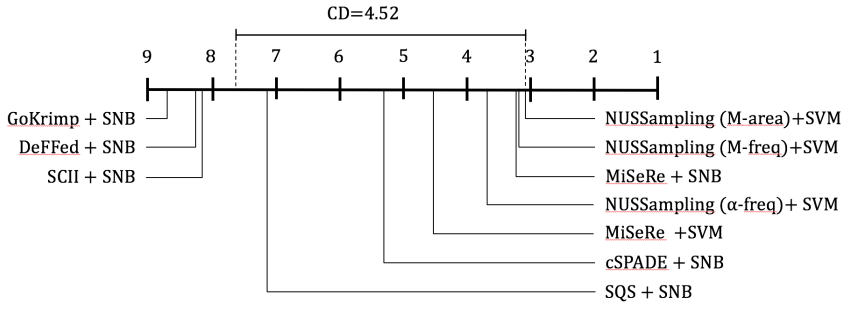


FIGURE 5.10 – Différence critique de performances entre les différents classifieurs.

Impact de la taille de l'échantillon : En fonction des applications, notamment des tâches de classification, l'impact de la taille de l'échantillon ne doit pas être ignoré avec notre méthode de classification. De toute évidence, l'accuracy de la classification augmente avec la taille de l'échantillon car les séquences sont plus susceptibles d'être couvertes par au moins une sous-séquence. Dans cette section, afin d'évaluer l'impact de la taille de l'échantillon, nous effectuons uniquement des expériences en utilisant la mesure M-freq. Dans ce contexte, la figure 5.11 montre les performances de classification, considérées comme des valeurs d'accuracy moyennes sur tous les jeux de données, obtenues par différentes tailles d'échantillon par rapport aux valeurs de contrainte de norme maximale 1, 10 et **Best** mentionnées dans le tableau 5.6. Il est facile d'observer que les performances de classification augmentent continuellement lorsque davantage de motifs séquentiels échantillonnés sont impliqués (ce qui est utile pour développer une approche "anytime"). Fait intéressant, l'accuracy augmente très rapidement avec la taille de l'échantillon. Ainsi, un classifieur construit dans un temps court ne considérant qu'un échantillon de 1,000 motifs séquentiels est en concurrence avec les méthodes de l'état de l'art où tout l'espace de recherche

de motifs est exploré comme nous l'avions souligné au chapitre 1.

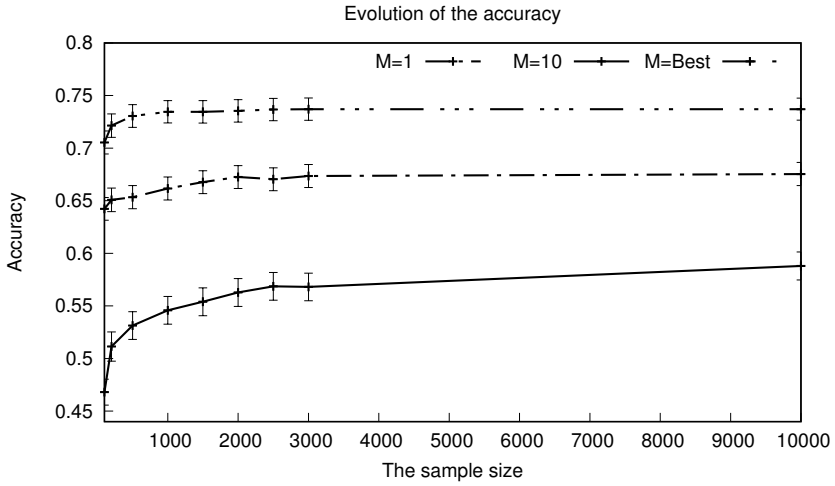


FIGURE 5.11 – Impact de la taille de l'échantillon sur les performances de classification avec M-freq

5.5 Conclusion

Ce chapitre propose notre méthode d'échantillonnage en sortie de motifs séquentiels prenant en compte notre classe de mesures d'intérêt fondées sur la norme. NUSSAMPLING permet de spécifier une contrainte d'intervalle sur la norme des motifs séquentiels pour mieux contrôler les motifs renvoyés et éviter le phénomène de la longue traîne. Nous avons estimé son efficacité par rapport au nombre moyen de rejets qui augmente avec le nombre d'occurrences dans une séquence. L'étude expérimentale montre que l'approche est très efficace sur des jeux de données réels où le nombre de répétitions est faible. Par ailleurs, les expériences montrent que, aussi bien pour la fréquence que pour l'aire, l'ajout de contraintes sur la norme ou la décroissance exponentielle évite de renvoyer trop de motifs trop rares, et concentre l'échantillonnage sur les motifs de la "tête" comme souhaité. Enfin, nous avons montré qu'avec seulement un échantillon de 1,000 motifs, nous parvenons à construire des classifieurs très performants pour les données séquentielles. Quelle que soit la mesure (fréquence ou aire), ces motifs ont toujours une accuracy comparable à certaines méthodes qui se basent sur une énumération complète de l'espace de recherche de motifs.

Cependant, la méthode de rejet employée lors du tirage des motifs peut être très coûteuse en temps dans le cas où la base de données contient des séquences avec trop de répétitions d'items. Une piste prometteuse pour éviter le rejet est d'utiliser les tries qui permettent de représenter l'espace des occurrences de motifs.

En perspectives, nous pensons qu'il est possible avec la forme canonique (*la première occurrences*) d'attaquer d'autres langages structurés pour le tirage uniforme d'un motif. Comme nous l'avons remarqué avec les itemsets, la construction de règles de classification

par échantillonnage en sortie de motifs est aussi une piste prometteuse pour construire efficacement des classifieurs de données séquentielles. Enfin, la fouille de motifs séquentiels à base de mesures d'intérêt discriminatives sous contraintes de normes maximale et minimale que nous avons abordée théoriquement dans [Diop *et al.*, 2019d], est aussi une piste de recherches prometteuse. Toutefois, les mesures d'intérêt discriminatives sont théoriquement plus coûteuses en temps que les mesures d'intérêt de la classe $\mathcal{M}$.

Chapitre 6

Echantillonnage de motifs dans une base de données distribuée

Sommaire

Ce chapitre propose une nouvelle méthode de la classe des "Two-Step" (voir section 2.2.3) pour échantillonner des motifs à partir des bases de données distribuées (voir section 3.4) afin de surmonter les limites des méthodes existantes de fouille de motifs dédiées aux bases de données distribuées. L'échantillonnage en sortie de motifs à partir de bases de données distribuées n'est pas trivial. Pour le faire efficacement, il ne faut surtout pas centraliser les données des fragments car cela peut s'avérer très coûteux en temps et en espace mémoire. Ainsi, pour appliquer une méthode d'échantillonnage en deux étapes sur une base de données distribuée, nous avons besoin de répondre aux défis

suivants : i) Comment calculer le poids de chaque instance de la base de données suivant une contrainte de norme maximale M ? ii) Comment tirer uniformément un motif de norme ℓ parmi l'ensemble des motifs de même norme de la base de données ?

Nous proposons un algorithme d'échantillonnage en deux étapes nommé DDSampling (pour Distributed Database Sampling)[Diop *et al.*, 2019b, Diop *et al.*, 2020b], qui tire un motif suivant une mesure d'intérêt fondée sur la norme dans l'ensemble de la base de données distribuée. DDSampling est générique dans le sens où il peut être appliqué sur tout type de partitionnement d'une base de données distribuée. Dans ce chapitre, nous montrons comment échantillonner efficacement des motifs à partir d'une base de données distribuée sans utiliser la capacité de calcul des fragments ni centraliser les données des différents sites.

Nos principales contributions sont les suivantes :

— Nous proposons un algorithme générique appelé DDSampling (Distributed Database Sampling) qui tire au hasard un motif d'une base de données distribuée suivant une mesure d'intérêt fondée sur la norme.

— Nous démontrons que DDSampling effectue un échantillonnage exact et analysons sa complexité en moyenne. Les expérimentations montrent que DDSampling est très rapide et que le coût de communication de notre proposition est bien inférieur à celui de la centralisation des données pour tirer quelques milliers de motifs.

— Nous illustrons l'intérêt de DDSampling sur un cas d'utilisation en détectant les données aberrantes dans deux triplestores du monde réel : *DBpedia* et *Wikidata*. Ces expérimentations montrent l'importance d'utiliser une contrainte de norme maximale et que l'échantillonnage *en sortie* est plus efficace que l'échantillonnage *en entrée*.

Le reste de ce chapitre est structuré comme suit : la section 6.1 présente quelques définitions de base supplémentaires à celle de la section 3.1 et formalise le problème de l'échantillonnage en sortie de motifs dans les bases de données distribuées. La section 6.3 évalue les performances de DDSampling sur les benchmarks en simulant des défaillances sur le réseau ainsi que des pannes de sites. Enfin, la section 6.4 illustre l'intérêt de l'échantillonnage de motifs pour détecter les données aberrantes dans les triplestores du Web.

6.1 Formulation du problème

Nous avions introduit quelques notions élémentaires sur les bases de données distribuées à la section 3.1. Ensuite, la section 3.4.3 avait énuméré les différents défis que nous devons résoudre pour échantillonner des motifs dans des bases de données distribuées. Reconsidérons l'exemple 16 de base de données distribuée à la figure 6.1 où nous considérons $K = 4$ fragments répartis dans K sites $\{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4\}$. $\mathcal{P}^*$ correspond à la base de données centralisée.

Pour commencer, nous considérons un environnement avec une base de données distribuée où il est seulement possible de demander à un fragment k, la taille d'une transaction d'identifiant j et l'item à la position i d'une transaction j :

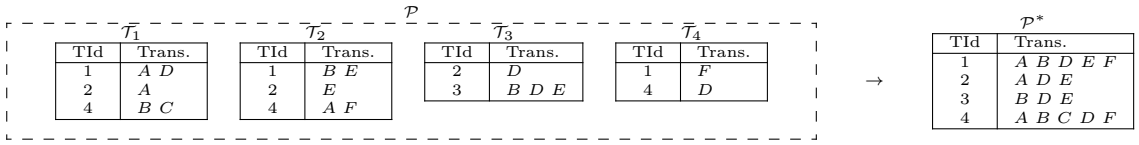


FIGURE 6.1 – Exemple de base de données distribuée

1. La requête $\texttt{lengthOf}(j, \mathcal{T}_k)$ retourne la taille de la transaction d'identifiant j du fragment $\mathcal{T}_k$ i.e., $\texttt{lengthOf}(j, \mathcal{T}_k) = |\mathcal{T}_k[j]|$. Dans notre exemple, on a $\texttt{lengthOf}(4, \mathcal{T}_1) = 2$ car $|\mathcal{T}_1[4]| = |BC| = 2$.
2. La requête $\texttt{itemAt}(i, j, \mathcal{T}_k)$ retourne le $i$ème item de la transaction d'identifiant j du fragment $\mathcal{T}_k$ en supposant un ordre arbitraire sur les items de $\mathcal{I}$. Par exemple, avec l'ordre lexicographique sur les items, nous avons $\texttt{itemAt}(2, 4, \mathcal{T}_1) = C$.

Nous avons proposé les opérations élémentaires `lengthOf` et `itemAt` dans un soucis de généricité. Dans la pratique, on peut disposer d'opérations plus sophistiquées qui composent ces opérations élémentaires (par exemple, récupérer une transaction complète présente sur une base de données permet d'inférer à la fois la taille de la transaction et l'item de chaque position). Notre démarche était d'identifier les opérations élémentaires au coeur de notre cadre théorique. Dans le contexte du Web sémantique, nous utilisons des requêtes SPARQL pour interroger les données. Ce langage d'interrogation est très expressif mais en émulant les seules opérations `lengthOf` et `itemAt`, nous avons une architecture très performante. Par exemple, on peut obtenir directement une transaction de longueur ℓ d'un fragment à l'aide de ℓ requêtes de la primitive `itemAt`.

Dans ce chapitre, le problème que nous voulons résoudre peut se formuler comme suit :

Soient une base de données transactionnelles distribuée $\mathcal{P} = \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$ et une utilité fondée sur la norme $u \in \mathcal{U}$. Notre objectif est de tirer un motif φ de $\mathcal{L}_\mathcal{I}$ suivant une distribution π telle que $\pi(\mathcal{S}ampl\mathcal{e}_1(\mathcal{L}_\mathcal{I}, \mathcal{P}^\star, freq \times u) = \varphi) = freq(\varphi, \mathcal{P}^\star) \times u(\varphi)/Z$, où $\mathcal{P}^\star$ est la base de données centralisée de $\mathcal{P}$ et en utilisant uniquement des requêtes `lengthOf` et `itemAt` et $Z = \sum_{\varphi \in \mathcal{L}_\mathcal{I}} freq(\varphi, \mathcal{P}^\star) \times u(\varphi)$ est une constante de normalisation.

6.2 Méthode d'échantillonnage par fragments

Cette section présente notre algorithme DDSAMPLING (pour Distributed Database Sampling) [Diop *et al.*, 2019b] dont la force est de ne centraliser aucun item (sauf ceux qui seront tirés pour construire les motifs). Il prend en entrée une base de données distribuée et une mesure d'intérêt fondée sur la norme afin de retourner un itemset φ tiré proportionnellement suivant son intérêt $freq(\varphi, \mathcal{P}^\star) \times u(\varphi)$. L'idée clé de notre approche consiste à centraliser la taille de chacune des transactions des différents fragments (plutôt que tous les items). Ces seules informations permettent ensuite de calculer aisément les différentes distributions de tirage et permettent de localiser les fragments utiles à la construction de l'itemset à tirer.

6.2.1 Calcul de motifs à la demande : DDSampling

L'algorithme 6 prend en entrée les adresses des fragments d'une base de données distribuée et une mesure d'intérêt fondée sur la norme. La phase de prétraitement consiste à construire la matrice de pondération $\mathbb{M}$ (voir définition 24) en centralisant les tailles des transactions à l'aide de la requête `lengthOf` (ligne 3). Ensuite, à partir de cette matrice, il calcule le poids de chaque ligne de la matrice (ligne 4). Après cette phase de prétraitement, pour tirer un motif à partir des fragments, on commence à la ligne 5 par tirer un indice j entre 1 et le nombre de lignes de $\mathbb{M}$ proportionnellement à son poids. On pondère chaque entier ℓ entre 0 et la taille de la transaction (la somme des valeurs de la ligne j), notée par $\mathbb{M}_{j\bullet}$, par l'utilité des motifs de taille exactement égale à ℓ (ligne 6). Ensuite, un entier ℓ est tiré proportionnellement à son poids $\omega^{\ell}(j)$ (ligne 7). Enfin, un motif de taille ℓ appartenant à la transaction d'identifiant j est tiré uniformément parmi l'ensemble des motifs de cette transaction ayant une taille égale à ℓ (lignes 8-13) puis retourné à la ligne 14.

Algorithm 6 DDSampling

1: **Input** Une base de données distribuée $\mathcal{P} = \{\mathcal{T}_1, \cdots, \mathcal{T}_K\}$ et une utilité fondée sur la norme u
2: **Output** Un itemset tiré aléatoirement $\varphi \sim \pi(\mathcal{L}_{\mathcal{I}})$ où $\pi(\varphi) = freq(\varphi, \mathcal{P}^{\star}) \times u(\varphi)$
Phase de prétraitement
3: Soit $\mathbb{M}$ la matrice définie par $\mathbb{M}_{jk} \leftarrow \texttt{lengthOf}(j, \mathcal{T}_k)$ pour tout $j \in [1..|\mathcal{P}^{\star}|]$ et $k \in [1..K]$
4: Soient les poids définis par $\omega_u(j) \leftarrow \sum_{\ell=0}^{\mathbb{M}_{j\bullet}} \binom{\mathbb{M}_{j\bullet}}{\ell} \times f_u(\ell)$ pour tout $j \in [1..|\mathcal{P}^{\star}|]$
Phase de tirage
Etape 1 : tirage d'un identifiant de transaction
5: Tirer un indice j compris entre 1 et $|\mathcal{P}^{\star}|$ proportionnellement à ω_u : $j \sim \omega_u(\mathcal{P}^{\star})$
Etape 2 : tirage fragmenté d'un itemset
6: Soient les poids définis par $\omega_u^{\ell}(j) \leftarrow \binom{\mathbb{M}_{j\bullet}}{\ell} \times f_u(\ell)$ pour tout $\ell \in [0..\mathbb{M}_{j\bullet}]$
7: Tirer un entier ℓ proportionnellement à $\omega_u^{\ell}(j)$: $\ell \sim \omega_u^{[0..\mathbb{M}_{j\bullet}]}(j)$
8: $\vartheta \leftarrow \emptyset$ et $\varphi \leftarrow \emptyset$
9: **while** $||\varphi|| < \ell$ **do**
10: $\quad i \sim unif([1..\mathbb{M}_{j\bullet}] \setminus \vartheta)$ ▷ Tirage uniforme d'un indice i parmi ceux non visités
11: $\quad k \leftarrow \min\{l \in [1..K] : i \leq \sum_{m=1}^{k} \mathbb{M}_{jm}\}$ ▷ Trouver l'adresse du fragment correspondant
12: $\quad i' \leftarrow \sum_{m=1}^{k} \mathbb{M}_{jm} - i + 1$ ▷ Trouver l'indice dans le fragment d'identifiant k
13: $\quad \varphi \leftarrow \varphi \cup \{\texttt{itemAt}(i', j, \mathcal{T}_k)\}$ et $\vartheta \leftarrow \vartheta \cup \{i\}$
14: **return** φ ▷ Retourner le motif correspondant

Matrice de pondération Nous commençons par introduire la matrice de pondération qui quantifie pour chaque transaction combien d'items sont stockés sur chaque fragment :

Définition 24 (Matrice de pondération)**.** *La matrice de pondération* $\mathbb{M}$ *de la base de données distribuée* $\mathcal{P} = \{\mathcal{T}_1, \ldots, \mathcal{T}_K\}$ *est une matrice d'entiers de dimension* $|\mathcal{P}^{\star}| \times |\mathcal{P}|$ *où* $\mathbb{M}_{jk} = \texttt{lengthOf}(j, \mathcal{T}_k)$ *pour tout* $j \in [1..|\mathcal{P}^{\star}|]$ *et* $k \in [1..K]$

En pratique, la matrice de pondération $\mathbb{M}$ est pré-calculée hors ligne (ligne 1) et utilisée à volonté par la suite pour tirer de nombreux échantillons au sein de l'algorithme 6.

Par ailleurs, nous définissons la somme des valeurs de la ligne j, i.e. $\mathbb{M}_{j\bullet} = \sum_{k=1}^{K} \mathbb{M}_{jk}$, qui correspond à la taille de la transaction d'identifiant j. Par exemple, le tableau 6.1 présente la matrice de pondération pour la base de données jouet de la figure 6.1, et on a $\mathbb{M}_{1\bullet} = 2 + 2 + 1 = |\mathcal{P}^{\star}[1]|$.

j	$\mathbb{M}$				$\mathbb{M}_{j\bullet}$	$\sum_{\ell=0}^{\mathbb{M}_{j\bullet}} \omega_{freq}^{\ell}(j) =$	$\omega_{freq}(j)$	$\omega_{area}(j)$	$\omega_{\leqslant 2}(j)$
1	2	2	0	1	5	$1 + 5 + 10 + 10 + 5 + 1 =$	32	70	16
2	1	1	1	0	3	$1 + 3 + 3 + 1 =$	8	12	7
3	0	0	3	0	3	$1 + 3 + 3 + 1 =$	8	12	7
4	2	2	0	1	5	$1 + 5 + 10 + 10 + 5 + 1 =$	32	70	16

TABLE 6.1 – Matrice de pondération $\mathbb{M}$ et poids pour le tirage

Calcul des poids $\omega_{\mathbf{u}}(\mathbf{j})$ et $\omega_{\mathbf{u}}^{\ell}(\mathbf{j})$ La première utilité de la matrice de pondération est de permettre un calcul aisé des poids $\omega_u(j)$ et $\omega_u^{\ell}(j)$. Pour rappel, ces poids correspondent respectivement à la somme des utilités des motifs de la transaction j et à la somme des utilités des motifs de norme ℓ dans la transaction j. Tout d'abord notons que la somme total $\omega_u(j)$ correspond à la somme de tous les poids $\omega_u^{\ell}(j)$: $\omega_u(j) = \sum_{\ell=0}^{\mathbb{M}_{j\bullet}} \omega_u^{\ell}(j)$. Concentrons-nous maintenant sur le calcul de la somme des utilités des motifs de norme ℓ dans la transaction j. Intuitivement, comme tous les motifs de même norme ont la même utilité, il suffit de dénombrer le nombre de motifs de norme ℓ et de multiplier cette quantité par l'utilité $f_u(\ell)$. La propriété suivante formalise ce calcul :

Propriété 7. *Soient $\mathbb{M}$ la matrice de pondération d'une base de données distribuée $\mathcal{P} = \{\mathcal{T}_1, \ldots, \mathcal{T}_K\}$ et une fonction d'utilité fondée sur la norme u. Le poids des itemsets de longueur ℓ de la transaction j est défini par : $\omega_u^{\ell}(j) = \sum_{\varphi \subseteq \mathcal{T}[j] \wedge |\varphi| = \ell} u(\varphi) = \binom{\mathbb{M}_{j\bullet}}{\ell} \times f_u(\ell)$.*

Preuve. Triviale. □

L'utilisation de cette propriété est illustrée dans le tableau 6.1 pour trois fonctions d'utilités fondées sur la norme, u_{freq}, u_{area} et $u_{\leq 2}$. Par exemple, comme $f_{u_{freq}}(\ell) = 1$ pour toute norme ℓ, on a $\omega_{freq}(2) = \sum_{\ell=0}^{3} \binom{3}{\ell} = 1 + 3 + 3 + 1 = 8 = 2^3$ (où $\binom{3}{\ell}$ est le nombre d'itemsets de norme ℓ dans une transaction de taille 3). En considérant l'aire, on a $\omega_{area}(2) = \sum_{\ell=0}^{3} \binom{3}{\ell} \times \ell = (1 \cdot 0) + (3 \cdot 1) + (3 \cdot 2) + (1 \cdot 3) = 3 + 6 + 3 = 12 = 3 \cdot 2^{3-1}$ car $f_{u_{area}}(\ell) = \ell$. Enfin, avec la contrainte de cardinalité maximale de 2, on a $\omega_{\leqslant 2}(2) = \sum_{\ell=0}^{2} \binom{3}{\ell} = 1 + 3 + 3 = 7$.

Construction de l'itemset Les lignes 4 à 11 détaillent le tirage de l'itemset sur l'ensemble des fragments. L'idée est de tirer sans remise une position d'item i dans la transaction d'identifiant j (ligne 8) et de rechercher le fragment k disposant de cet item (ligne 9). Il faut alors calculer la position i' de cet item au sein du fragment k (ligne 10) avant de l'interroger (ligne 11). On répète cette procédure ℓ fois (ligne 7) et pour éviter de tirer deux fois le même item, on maintient l'ensemble des positions déjà tirées ϑ. Avec l'exemple du tableau 6.1, si on a le tirage de la position 2 (i.e., $i = 3$) dans la transaction 1, le fragment retenu est $k = 2$ avec $i' = 3 - 2 = 1$, et l'item $\texttt{itemAt}(1, 1, \mathcal{T}_2) = B$ est ajouté à φ.

6.2.2 Analyse théorique de DDSampling

La propriété suivante démontre que DDSampling retourne un échantillon de manière exacte :

Propriété 8 (Correction)**.** *Soient $\mathcal{P} = \{\mathcal{T}_1, ..., \mathcal{T}_K\}$ une base de données distribuée et u une mesure d'intérêt fondée sur la norme. L'algorithme 6 effectue le tirage aléatoire d'un motif φ de $\mathcal{L}$ proportionnellement à $freq(\varphi, \mathcal{P}^\star) \times u(\varphi)$.*

Preuve. Soit Z la constante définie par $Z = \sum_{j \in [1..|\mathcal{P}^\star|]} \omega_u(j)$, φ un itemset de $\mathcal{L}_\mathcal{I}$ tel que $||\varphi|| = \ell$ et π la probabilité de tirer φ à l'aide de l'algorithme 6. Alors on a : $\pi(\mathcal{S}amp\ell e_1(\mathcal{L}_\mathcal{I}, \mathcal{P}^\star, freq \times u) = \varphi) = \sum_{j \in [1..|\mathcal{P}^\star|]} \pi(j) \times \pi(\varphi/j) = \sum_{j \in [1..|\mathcal{P}^\star|] \wedge \varphi \subseteq \mathcal{P}^\star[j]} \pi(j) \times \pi(\varphi/j)$. D'après la ligne 5, on a $\pi(j) = \omega_u(j)/Z$. Or d'après les lignes 7 à 13 on a $\pi(\varphi/j) = \frac{\Phi_\ell(j) \times f_u(\ell)}{\omega_u(j)} \times \frac{f_u(\ell)}{\Phi_\ell(j) \times f_u(\ell)} = f_u(\ell)/\omega_u(j)$. Ce qui implique que $\pi(\mathcal{S}amp\ell e_1(\mathcal{L}_\mathcal{I}, \mathcal{P}^\star, freq \times u) = \varphi) = \sum_{j \in [1..|\mathcal{P}^\star|] \wedge \varphi \subseteq \mathcal{P}^\star[j]} \frac{\omega_u(j)}{Z} \times \frac{f_u(\ell)}{\omega_u(j)} = \sum_{j \in [1..|\mathcal{P}^\star|] \wedge \varphi \subseteq \mathcal{P}^\star[j]} \frac{f_u(\ell)}{Z}$. Ce qui nous donne finalement $\pi(\mathcal{S}amp\ell e_1(\mathcal{L}_\mathcal{I}, \mathcal{P}^\star, freq \times u) = \varphi) = \frac{freq(\varphi, \mathcal{P}^\star) \times u(\varphi)}{Z}$. D'où le résultat. □

La complexité de la méthode DDSampling se décompose en deux phases : celle du prétraitement correspondant à la pondération des lignes de la matrice, et celle du tirage d'un motif. Pour chaque phase, il est pertinent de déterminer la complexité en temps et en communication.

Complexité temporelle Pour le prétraitement, DDSampling remplit la matrice $\mathbb{M}$ avec une complexité en $O(|\mathcal{P}^\star| \times |\mathcal{P}|)$. Ensuite, il pondère les lignes de la matrice avec une complexité en $O(|\mathcal{P}^\star| \times |\mathcal{I}|)$ à cause de la fonction binomiale. La complexité temporelle du prétraitement est donc en $O(|\mathcal{P}^\star| \times (|\mathcal{P}| + |\mathcal{I}|))$. Pour le tirage d'un motif, la première étape consiste à sélectionner une ligne de la matrice, soit une complexité en $O(\log(|\mathcal{P}^\star|))$. Deuxièmement, on tire uniformément un itemset de la transaction ayant pour identifiant le numéro de la ligne précédemment tiré avec une complexité en $O(|\mathcal{I}|)$. Finalement, nous obtenons une complexité temporelle de tirage d'un motif égale à $O(\log(|\mathcal{P}^\star|) + |\mathcal{I}|)$.

Complexité en communication Pour rappel, les coûts de communication correspondent aux requêtes `lengthOf` et `itemAt`. Dès lors, pour le prétraitement, la construction de la matrice de pondération $\mathbb{M}$ requiert $O(|\mathcal{T}_1| + \cdots + |\mathcal{T}_K|)$ échanges ce qui est souvent très inférieur au coût de la centralisation complète en $O(|\mathcal{T}_1| + \cdots + |\mathcal{T}_K|) = O(|\mathcal{P}^\star|)$ échanges. Dans notre exemple jouet, le calcul de la matrice de pondération requiert 10 requêtes contre 17 pour la centralisation complète de $\mathcal{P}^\star$. Pour le tirage d'un motif, il est nécessaire d'effectuer autant de requêtes `itemAt` que d'items contenus dans l'itemset à retourner. La complexité temporelle moyenne est donc égale à la longueur moyenne $\overline{\ell}$ des itemsets tirés, cette longueur étant donnée par la formule suivante : $\overline{\ell} = \sum_{\ell=1}^{\ell_{max}} \frac{\sum_{j \in [1..|\mathcal{P}^\star|]} \omega_u^\ell(j)}{\sum_{j \in [1..|\mathcal{P}^\star|]} \omega_u(j)} \times \ell$ où $\ell_{max} = max_{j \in [1..|\mathcal{T}|]} |\mathcal{P}^\star[j]|$. Considérant notre exemple jouet dans le tableau 6.1 et la fonction d'utilité de norme maximale $u_{\leq 2}$, nous avons $\pi(L = 1) = \frac{3+6+3+6+3}{7+11+7+11+7} = \frac{21}{43}$ et $\pi(L = 2) = \frac{3+4+3+4+3}{7+11+7+11+7} = \frac{17}{43}$. Ainsi, le coût moyen de communication pour tirer un échantillon de motifs est égal à $\overline{\ell} = (\frac{21}{43} \times 1) + (\frac{17}{43} \times 2) \approx 1.28$. Notez que sans contrainte de longueur (en utilisant la fonction d'utilité u_{freq}), ce coût est plus élevé et égal à $\overline{\ell} = (\frac{21}{56} \times 1) + (\frac{17}{56} \times 2) + (\frac{11}{56} \times 3) + (\frac{2}{56} \times 4) \approx 1.71$.

Complexité en espace mémoire Pour échantillonner des motifs, nous avons besoin de stocker les tailles des transactions de chaque fragment $\mathcal{T}_k$ avec $k \in [1..K]$ sous la forme d'une matrice appelée matrice de pondération. Le coût de stockage de la matrice de pondération est en $O(K \times |\mathcal{P}^\star|)$.

Taux de rejet Deux problèmes principaux se posent dans les bases de données distribuées : les erreurs de communication réseau (*panne réseau*) et l'inaccessibilité des nœuds (*défaillance des noeuds*). Ces phénomènes induisent un biais dans les tirages réalisés car nous devons rejeter un motif φ dès qu'une requête `itemAt` échoue lors de son tirage. Dans le cas de *panne réseau*, soit ϵ_{net} la probabilité d'échec d'une requête `itemAt`. En supposant que les défaillances sont indépendantes, nous pouvons montrer que $\mathbf{P}(\text{reject}) = \sum_\ell \pi(\ell) \times \left(1 - (1 - \epsilon_{net})^\ell\right)$. Ainsi, si ϵ_{net} est un petit nombre, nous avons $\mathbf{P}(\text{reject}) \approx \epsilon_{net} \times \overline{\ell}$ (où $\overline{\ell}$ est la longueur moyenne des itemsets échantillonnés). Maintenant, dans le cas de *panne de noeud*, soit ϵ_{node} la probabilité qu'un nœud tombe en panne. Si la base de données distribuée est un partitionnement horizontal d'une base de données centralisée, il est clair que $\mathbf{P}(\text{reject}) = \epsilon_{node}$. Sinon, si la base de données distribuée est un partitionnement vertical ou hybride de $\mathcal{P}^*$, en supposant que les items soient uniformément répartis sur les noeuds, nous pouvons montrer que $\mathbf{P}(\text{reject}) = \sum_\ell \pi(\ell) \times \left(1 - (1 - \epsilon_{node})^\ell\right)$. Ainsi, si ϵ_{node} est un petit nombre, nous avons $\mathbf{P}(\text{reject}) \approx \epsilon_{node} \times \overline{\ell}$. De nos jours, ϵ_{net} et ϵ_{node} sont très petits. Il s'ensuit donc que le rejet est négligeable.

6.3 Evaluation expérimentale de DDSampling

Après avoir présenté le protocole expérimental, cette section expérimentale commence par évaluer le coût de communication de DDSampling par rapport à une solution centralisée et l'impact de défaillances sur sa performance (problèmes de communication ou de pannes de sites). Ensuite, elle présente un cas d'utilisation pour la détection de données aberrantes dans les données du Web à l'aide de la mesure FPOF.

6.3.1 Protocole expérimental

Dans nos premières expérimentations, nous utilisons 5 jeux de données de l'UCI (`archive.ics.uci.edu/ml`). Pour obtenir des bases de données distribuées [1], nous avons fragmenté uniformément chaque jeu de données en $K = 10$ fragments. Pour générer un partitionnement horizontal, chaque transaction est placée aléatoirement dans un fragment donné avec la même probabilité $1/K$. Pour un partitionnement hybride, tous les items d'une transaction sont placés indépendamment et aléatoirement (avec la même probabilité $1/K$) dans un fragment donné. Enfin, pour générer un partitionnement vertical, chaque item est placé aléatoirement sur un site donné (avec la même probabilité $1/K$). Les premières colonnes du tableau 6.2 affichent des informations statistiques sur tous les jeux de données. Dans toutes les expérimentations, nous utilisons des contraintes de normes minimale $u_{\geq 1}$ et maximale $u_{\leq M}$ (avec $M = \{3, 5\}$). Ce choix évite de tirer des motifs trop

1. Jeux de données et code source disponibles à `https://github.com/DDSampling/DDSampling`

peu fréquents, en particulier pour les jeux de données contenant de longues transactions. Toutes les expérimentations sont réalisées sur un PC de 2.71 GHz 2 Core CPU avec une RAM de 12 Go.

TABLE 6.2 – Caractéristiques de 4 jeux de données fragmentés de l'UCI

Données centralisées			
$\mathcal{T}$	$\|\mathcal{I}\|$	$\|\mathcal{T}\|$	$\|\|\mathcal{P}^\star\|\|$
`Chess`	75	3,196	118,252
`Connect`	129	67,557	2,904,951
`Iris`	15	150	750
`Mushroom`	119	8,124	186,852
`Waveform`	67	5,000	110,000

6.3.2 Evaluation du temps d'exécution et du coût de communication

Le tableau 6.3 indique les temps d'exécution de notre méthode en distinguant la phase de prétraitement et d'échantillonnage, dans les différents cas de partitionnement pour $M = 3$. Comme prévu, le temps de prétraitement (qui peut être préparé hors ligne) augmente avec la taille de la base de données. Cependant, il est très petit (moins de 3 secondes). Concernant la phase d'échantillonnage, quel que soit le jeu de données, le temps de tirage d'un motif (avec $M = 3$) est toujours inférieur à 0,02 milliseconde.

TABLE 6.3 – Evaluation du temps d'exécution en seconde

	Horizontal		Hybride		Vertical	
$\mathcal{T}$	Preprocessing	Sampling	Preprocessing	Sampling	Preprocessing	Sampling
`Chess`	0.11	$1 \cdot 10^{-5}$	0.13	$1 \cdot 10^{-5}$	0.13	$1 \cdot 10^{-5}$
`Connect`	2.12	$2 \cdot 10^{-5}$	2.42	$2 \cdot 10^{-5}$	2.59	$2 \cdot 10^{-5}$
`Iris`	0.01	$1 \cdot 10^{-5}$	0.01	$1 \cdot 10^{-5}$	0.01	$1 \cdot 10^{-5}$
`Mushroom`	0.15	$1 \cdot 10^{-5}$	0.20	$1 \cdot 10^{-5}$	0.21	$1 \cdot 10^{-5}$
`Waveform`	0.11	$1 \cdot 10^{-5}$	0.14	$1 \cdot 10^{-5}$	0.16	$1 \cdot 10^{-5}$

Pour les coûts de communication, nous considérons les trois types de partitionnements. Dans la phase de prétraitement, le coût de communication correspond au nombre d'appels `lengthOf` pour construire la matrice de poids. Ce coût est naturellement plus élevé pour les partitionnements hybrides et verticaux car les éléments d'une transaction peuvent ne pas être dans le même fragment. Dans la phase de tirage, le coût de communication correspond au nombre d'appels `itemAt`, et le tableau 6.4 montre le nombre moyen d'appels pour tirer un motif. Ce coût ne dépend pas du type de partitionnement, mais de la longueur maximale ($M \in \{3, 5\}$). Enfin, nous comparons le coût de communication entre les approches distribuées et centralisées en évaluant le nombre N_{max} de motifs tirés dans le pire des cas (lorsque $M = 5$ pour le partitionnement vertical) qui sont nécessaires pour que l'approche par échantillonnage soit aussi coûteuse que l'approche par centralisation

des données. Pour tous les jeux de données, nous pouvons voir que DDSampling peut tirer quelques milliers de motifs avec un coût de communication inférieur à celui d'une centralisation des données.

Table 6.4 – Evaluation du coût de communication

	Nb d'appels de `lengthOf`			#`itemAt`		
	Données distribuées			Distribuée		N_{max}
$\mathcal{T}$	Horizontal	Hybride	Vertical	M=3	M=5	vertical avec $u_{[1..5]}$
`Chess`	3,196	31,312	31,427	2.91	4.83	17,976
`Connect`	67,557	668,296	668,547	2.92	4.86	460,165
`Iris`	150	614	618	2.19	2.58	132
`Mushroom`	8,124	74,036	74,180	2.85	4.70	23,973
`Waveform`	5,000	45,081	45,324	2.84	4.68	13,820

6.3.3 Altération de l'exactitude du tirage

Avec les bases de données distribuées, deux des principaux problèmes sont les erreurs de communication réseau ou l'inaccessibilité d'un site. Chacun de ces phénomènes induit un biais dans les tirages réalisés. En effet, lors du tirage d'un motif φ, nous le rejetons systématiquement en cas d'échec d'une des requêtes `itemAt` nécessaires à sa construction. Pour mesurer le biais résultant d'une telle stratégie, nous avons utilisé le jeu de données `Iris` pour tirer un million de motifs de norme maximale $M = 5$. Ensuite, nous avons évalué si les motifs de l'échantillon ont effectivement été tirés avec une probabilité proportionnelle à leur mesure d'intérêt. Pour ce faire, nous comptons pour chaque tranche de fréquence $[\theta, \theta + 0.05[$ le nombre moyen de répétitions (ainsi que son écart type) avec lequel un motif apparaît dans l'échantillon alors que sa fréquence réelle est dans la tranche considérée. Si le tirage est exact, les motifs sont tirés proportionnellement à leur mesure d'intérêt et la courbe doit être une droite.

En suivant ce protocole, nous avons évalué la qualité des échantillons construits en faisant varier la probabilité $p \in \{0.00, 0.10, 0.20\}$ qu'une communication avec un site échoue, ou qu'un nombre $z \in \{0, 2, 5\}$ de sites soient en panne (les sites en panne étant tirés aléatoirement). Les résultats obtenus sont représentés à la figure 6.2. De manière générale, les courbes obtenues montrent que les défaillances n'altèrent pas significativement l'exactitude du tirage si leur niveau reste modéré ($p \leq 10\%$ et au plus $z = 2$ sites sur $K = 10$ sont en panne), et ceci quel que soit le type de partitionnement. Plus précisément, dans le cas de problèmes de communication, les motifs les plus fréquents sont les plus tirés car ils sont généralement les plus courts et ont ainsi moins de chance d'être rejetés. Ils seront donc sur-représentés dans l'échantillon construit. Dans le cas de sites en panne permanente, on constate que la probabilité de tirage d'un motif reste proportionnelle à sa mesure d'intérêt. Pour le cas vertical, tous les motifs tirés le restent de manière exacte (leurs items étant toujours accessibles au cours du temps). Plus le nombre de sites en panne est important, moins il y a de motifs disponibles. Du coup, les motifs restants sont plus fréquemment tirés augmentant la pente des courbes.

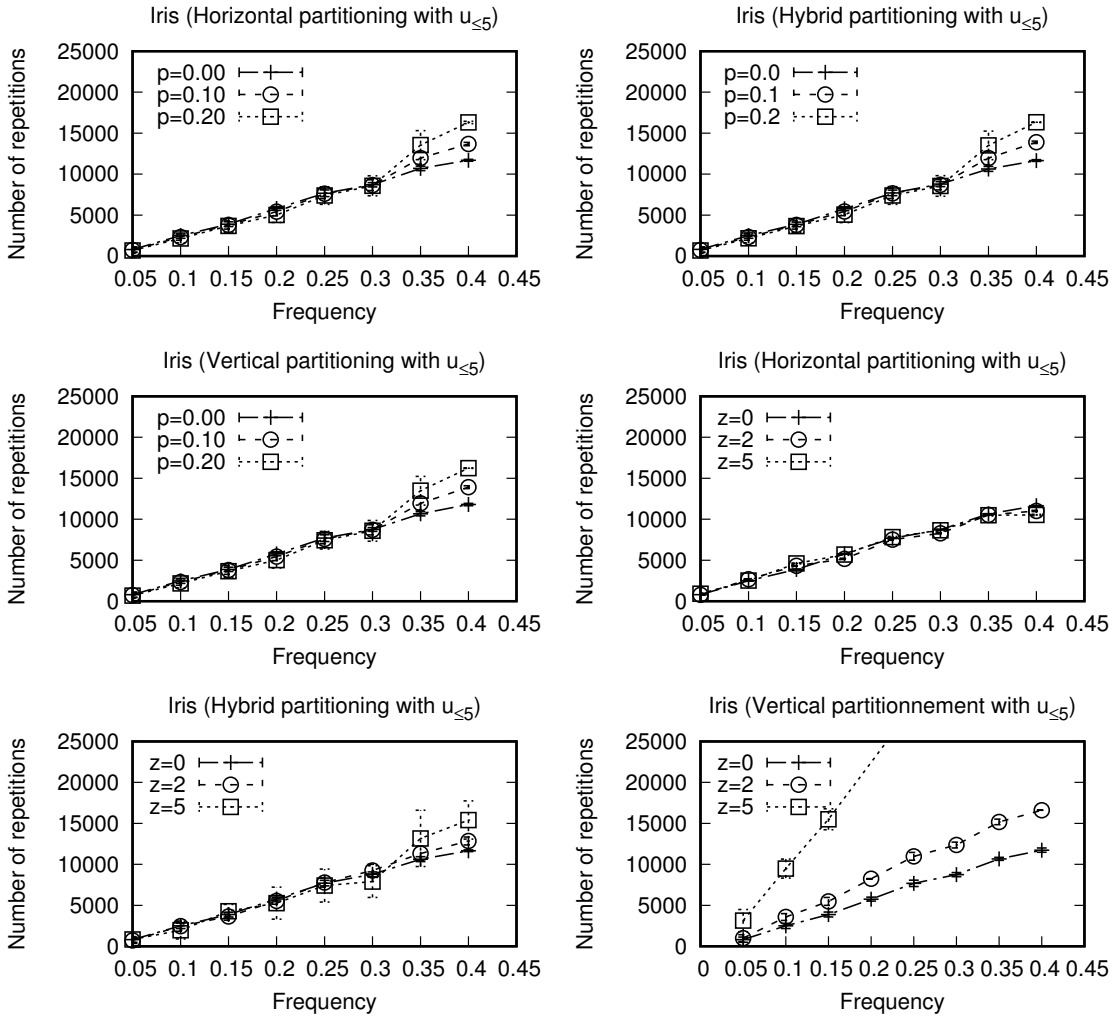


FIGURE 6.2 – Qualité de DDSAMPLING face aux défaillances de communication ou aux pannes

6.3.4 Evaluation du taux de motifs rejetés

Nous évaluons maintenant le taux de rejet moyen pour tirer 10,000 motifs en variant p et z. L'expérience est répétée 100 fois en faisant varier aléatoirement le partitionnement des bases et les sites en panne. La figure 6.3 montre les taux de rejets moyens et leurs écarts-types en calculant la moyenne des taux de rejets obtenus pour les 5 bases : `Chess`, `Connect`, `Iris`, `Mushroom` et `Waveform`. En effet, ces taux sont indépendants d'un jeu de données particulier, les partitionnements générés étant aléatoires et uniformes. Les taux de rejets étant indépendants d'un type particulier de partionnement, la partie gauche de la figure 6.3 représente l'évolution du taux de rejet moyen en fonction de p pour $M \in \{1, 2, 3, 4, 5\}$. Pour une valeur de p fixe, on constate que ce taux de rejet augmente avec M. Néanmoins, il reste inférieur à 50% si p est inférieur à 10% ce qui constitue déjà un niveau élevé de défaillance. La partie droite de la figure 6.3 représente les taux de rejets en cas de panne de sites. Tout d'abord, ils sont plus faibles dans le cas d'un partitionnement horizontal où le taux de rejet est égal à z/K. En moyenne, ils sont plus élevés dans le cas de partitionnements hybride ou vertical. Enfin, l'écart-type des taux de rejets moyen est plus élevé dans le cas d'un partitionnement vertical. En effet, avec un partitionnement vertical, les taux de rejets seront plus ou moins élevés si les items des motifs les plus fréquents sont ou pas sur les sites en panne. Pour finir, on note que le taux de rejet moyen reste inférieur à 50% si on a moins de 10% des sites sont en panne. Comme pour le cas de défaillances de communication, ce taux de rejet reste acceptable du point de vue du temps de calcul nécessaire pour construire un échantillon.

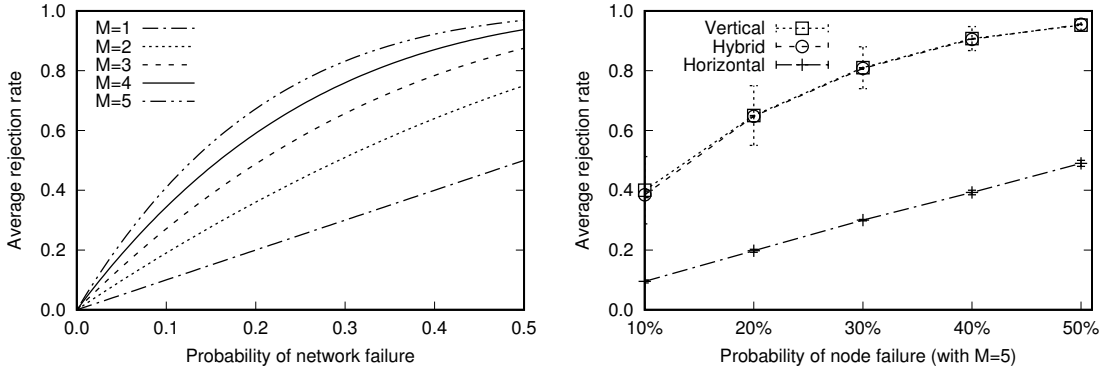


FIGURE 6.3 – Nombre moyen de rejets lors d'une défaillance

6.4 Détection de données aberrantes dans les triplestores

Cette section vise à détecter les données aberrantes dans les bases de connaissances du Web sémantique en estimant le FPOF(Frequent Pattern Outlier Factor) d'entité du Web sémantique avec des motifs obtenus par échantillonnage (méthode que nous avions présentée à la section 2.3.2). Plus précisément, nous utilisons cette mesure pour identifier les entités mal classées de deux classes (`Person` et `Organisation`) décrites dans *DBpedia* et *Wikidata*. Pour ce faire, nous allons appliquer DDSAMPLING sur les entités conjointes à

DBpedia et *Wikidata* et en particulier sur les classes `Person` et `Organisation`. Le tableau 6.5 rappelle les caractéristiques des bases de données du Web de la section 3.1.3 et fournit les temps d'exécution de DDSampling pour le prétraitement et pour le tirage d'un motif.

Table 6.5 – Caractéristiques et temps d'exécution de `Person` et `Organisation`

							Time (s)	
$\mathcal{T}$	$\lvert\mathcal{P}^\star\rvert$	$\lvert\mathcal{I}\rvert_{DBpedia}$	$\lvert\mathcal{I}\rvert_{Wikidata}$	$\lvert t\rvert_{min}$	$\lvert t\rvert_{max}$	$\lvert t\rvert_{avg}$	Prep.	Sampl.
`Person`	772,432	13,142	6,213	8	552	50.02	8,400.80	0.34
`Organisation`	338,402	19,022	5,504	8	328	36.22	1,847.88	0.27

On peut noter que les temps d'exécution sont plus longs que ceux des benchmarks de l'UCI de la section 6.3 car nous utilisons des points d'accès publics SPARQL. La section 6.4.1 formalise les primitives `itemAt` et `lengthOf` pour l'interrogation des triplestores *DBpedia* et *Wikidata*.

6.4.1 Formalisation des primitives SPARQL `itemAt` et `lengthOf`

Dans cette section, nous allons formaliser les requêtes SPARQL nécessaires pour lancer DDSampling [2] dans les données du Web. La base de données distribuée considérée est composée de 2 fragments *DBpedia* accessible via le service SPARQP http://dbpedia.org/sparql et *Wikidata* accessible via https://query.wikidata.org/sparql.

Dans la pratique, nous ne savons pas a priori les identifiants des instances. Ainsi, nous appliquons en premier lieu la requête `lengthOf`$(j, \mathcal{T}_{DBpedia})$ de la figure 6.4 sur l'ensemble des entités de *DBpedia* appartenant à la classe `Person` par exemple avec j variant entre 0 et le nombre d'instances de la base de données `Person`. Précisément, nous récupérons pour chaque identifiant d'une instance ($?s$), le nombre d'items (ou de propriétés) qui décrivent l'instance dans *DBpedia* ainsi que son identifiant dans *Wikidata*.

```
1  select ?s ?l (count(distinct ?item) as ?count) where {
2      ?s ?item ?o.
3      {
4          select distinct ?s ?l where{
5              ?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?c.
6              ?c <http://www.w3.org/2000/01/rdf-schema#subClassOf>
7                  <http://dbpedia.org/ontology/person>
8              ?s <http://www.w3.org/2002/07/owl#sameAs> ?l
9              filter(strstarts(str(?l),"http://wikidata"))
10         } limit 1 offset j
11     }
12 } group by ?s ?l;
```

Figure 6.4 – Requête SPARQL `lengthOf`$(j, \mathcal{T}_{DBpedia})$

2. Code source disponible à https://github.com/DDSampling/ddsamplingRDF

En appliquant la requête `lengthOf`$(j, \mathcal{T}_{DBpedia})$ de la figure 6.4, nous obtenons une matrice de pondération, nommée *mat*, de deux colonnes dont le nombre de lignes est égal au nombre d'instances de la classe `Person` dans *DBpedia*. La première colonne contient les identifiants des instances et la deuxième, le nombre de propriétés décrivant l'instance dans *DBpedia*.

A l'aide des identifiants stockés dans les cases $mat[1][j]$, nous utilisons la requête `lengthOf`$(j, \mathcal{T}_{Wikidata})$ de la figure 6.5 dédiée à Wikidata pour récupérer les tailles des itemsets appartenant à la transaction d'identifiant j et contenus par *Wikidata*. Autrement dit, pour chaque instance de la classe `Person` dans *DBpedia*, nous récupérons son nombre d'items qui la décrivent dans *Wikidata*. Cela peut éventuellement être égal à 0.

```
1  select (count(distinct ?item) as ?count) where {
2      <http://www.wikidata.org/entity/mat[1][j]> ?item ?o.
3  };
```

FIGURE 6.5 – Requête SPARQL `lengthOf`$(j, \mathcal{T}_{Wikidata})$

Après avoir récupéré et stocké dans *mat* les tailles des itemsets à l'aide de la requête `lengthOf`$(j, \mathcal{T}_{Wikidata})$ de la figure 6.5, la matrice *mat* a maintenant une colonne de plus, mais le nombre de lignes reste inchangé.

Une fois la matrice de pondération construite, DDSAMPLING utilise la requête `itemAt` pour former les motifs tirés en envoyant des requêtes. A la figure 6.6, nous avons la requête `itemAt`$(i, j, \mathcal{T}_{dataset})$ qui récupère le i-ème item de la transaction d'identifiant j contenu par le fragment $\mathcal{T}_{dataset}$ avec $dataset \in \{DBpedia, Wikidata\}$.

```
1  select distinct ?item where {
2      dataset[j] ?item ?o
3  } limit 1 offset i;
```

FIGURE 6.6 – Requête SPARQL `itemAt`$(i, j, \mathcal{T}_{dataset})$

6.4.2 Répartition des FPOF sous contrainte de norme maximale

L'intérêt d'utiliser une contrainte de norme maximale $u_{\leq M}$ pour calculer les FPOF approchés (formules 2.1 et 2.2 de la section 2.3.2) à l'aide d'un échantillon de motifs est illustré à la figure 6.7. Elle montre les distributions des FPOF de toutes les entités des classes `Person` et `Organisation` sans contrainte ($M = \infty$) ou avec une contrainte de norme maximale $M \in \{1, 2, 3, 4, 5, 10\}$.

Les mesures exactes des FPOF sous contrainte de norme maximale sont approximées à l'aide d'un échantillon de 10^6 motifs. Nous pouvons voir que sans contrainte ($M = \infty$) ou avec une valeur élevée pour M ($M \geq 5$), les valeurs des FPOF sont majoritairement égales à zéro. Ainsi, il est presque impossible de distinguer les valeurs aberrantes des entités normales. D'autre part, une contrainte de norme maximale égale à $M = 1$ n'est pas

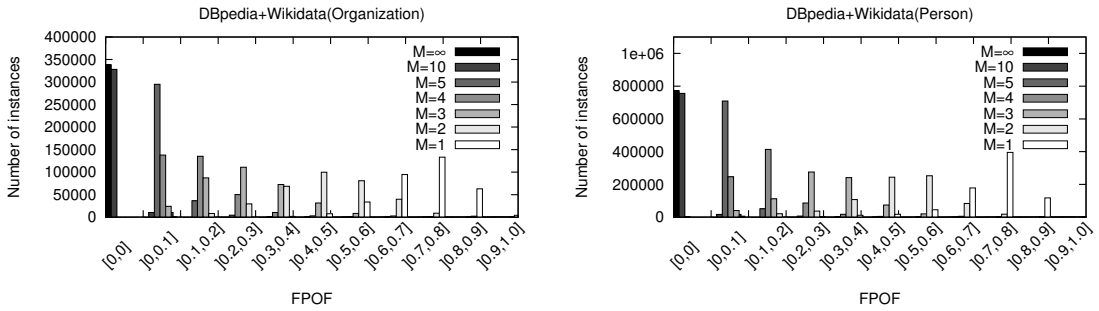


FIGURE 6.7 – Phénomène de la longue traîne

pertinente pour la détection d'anomalies car on ne tire que des items (motif de norme égale à 1). Par conséquent, dans les expériences suivantes, nous n'utiliserons l'approximation du FPOF qu'avec une contrainte de norme maximale $M = 3$.

6.4.3 Evaluation du temps de calcul des FPOF des méthodes exacte et approchée

Nous allons évaluer le temps de calcul des FPOF indépendamment du réseau. Pour ce faire, nous utilisons la version centralisée des données. En plus des jeux de données du Web, nous allons aussi considérer les jeux de données `Chess` et `Connect` de l'UCI (voir chapitre 6.3) afin d'élargir le champs de comparaison entre la méthode de calcul des FPOF exacts et la méthode approchée. Pour évaluer le temps de calcul des FPOF (indépendamment du réseau), nous avons appliqué notre méthode présentée dans [Diop *et al.*, 2020a] [3] sur les jeux de données. C'est une version de DDSAMPLING appliquée sur un seul fragment centralisé. Le tableau 6.6 montre les temps d'exécution des méthodes exhaustive et k-échantillonnée, avec $k \in \{1,000; 3,000; 5,000; 10,000\}$. Chaque expérience est répétée 10 fois afin d'avoir les écarts types. Rappelons que nous utilisons la mesure d'intérêt $u_{\leq 3}$.

TABLE 6.6 – Evaluation des temps d'exécution des méthodes en seconde

		FPOF k-échantillonné			
$\mathcal{T}$	FPOF exact	$k = 1,000$	$k = 3,000$	$k = 5,000$	$k = 10,000$
`Person`	–	164.36 ± 6.33	457.60 ± 12.03	736.26 ± 12.87	1,412.86 ± 27.34
`Organisation`	–	57.35 ± 2.57	174.57 ± 6.25	276.20 ± 6.93	528.92 ± 7.77
`Chess`	441.90 ± 8.27	0.94 ± 0.03	2.31 ± 0.01	3.53 ± 0.04	6.44 ± 0.12
`Connect`	3,475.09 ± 18.37	14.60 ± 0.67	43.48 ± 3.22	69.28 ± 4.35	122.28 ± 3.24

On note que la méthode approchée est beaucoup plus rapide que la méthode exhaustive pour le calcul des FPOF. En effet, quand la taille de la base est très grande ou les normes des transactions très élevées, la méthode exacte devient très coûteuse en temps. Ainsi, la durée pour le calcul des scores exacts sur la base `Organisation` est égale à environ

3. Code source disponible ici https://github.com/ItemsetSampling/cnria2020

27 *heures*. Pour la base `Person`, la méthode exacte a duré plus de 72 *heures* sans résultat. En revanche, les temps de calcul obtenus avec la méthode approchée restent raisonnables surtout avec la base `Person` où le nombre d'instances dépasse 700,000 et la taille moyenne d'une transaction est égale à 50.02.

6.4.4 Comparaison des FPOF k-échantillonnés avec les FPOF des échantillons en entrée

Cette expérience compare l'échantillonnage en entrée et l'échantillonnage en sortie pour déterminer quelle méthode est la meilleure pour le même coût (même nombre de motifs, même coût de communication). Pour construire un nombre de motifs k, nous commençons par tirer un échantillon $\mathbb{S}_k^{out}$ de motifs sous la contrainte $q(\cdot) = (1 \leq ||\cdot|| \wedge ||\cdot|| \leq 3)$ avec DDSampling et nous calculons son coût de communication $Cost_{out}$. Ensuite, nous tirons un échantillon de transactions $\widetilde{\mathcal{T}}$ nécessitant le même coût de communication $Cost_{in} = Cost_{out}$. Enfin, nous tirons un échantillon $\mathbb{S}_k^{in}$ de k motifs à partir de $\widetilde{\mathcal{T}}$. Étant donné un échantillon de k motifs $\mathbb{S}_k$, nous évaluons la qualité de son FPOF approché en utilisant comme erreur la distance euclidienne $\epsilon(\mathbb{S}_k, \mathcal{T}, q)$ définie comme suit :

$$\epsilon(\mathbb{S}_k, \mathcal{T}, q) = \sqrt{\sum_{t \in \mathcal{T}} (\widetilde{fpof_k}(t, \mathcal{T}, q) - fpof(t, \mathcal{T}, q))^2}.$$

La figure 6.8 rapporte $\epsilon(\mathbb{S}_k^{out}, \mathcal{T}, q)$ et $\epsilon(\mathbb{S}_k^{in}, \mathcal{T}, q)$ par rapport à la taille de l'échantillon de motifs (chaque mesure est la moyenne arithmétique obtenue avec 100 échantillons répétés ; l'écart type est également précisé). Bien sûr, les deux erreurs tendent vers zéro lorsque la taille de l'échantillon tend vers l'infini. Mais, il est clair que la convergence est plus rapide et plus stable avec l'échantillonnage en sortie (par exemple, la qualité des FPOF avec un échantillon de 10,000 motifs $\mathbb{S}^{out}$ est égale à celle de $\mathbb{S}^{in}$ avec un échantillon de 100,000 motifs).

Cette expérimentation montre que l'utilisation des motifs échantillonnés par DDSampling pour calculer les valeurs approchées des FPOF est plus efficace que celle des motifs obtenus à partir d'un échantillon de la base de données (à coût de communication égaux).

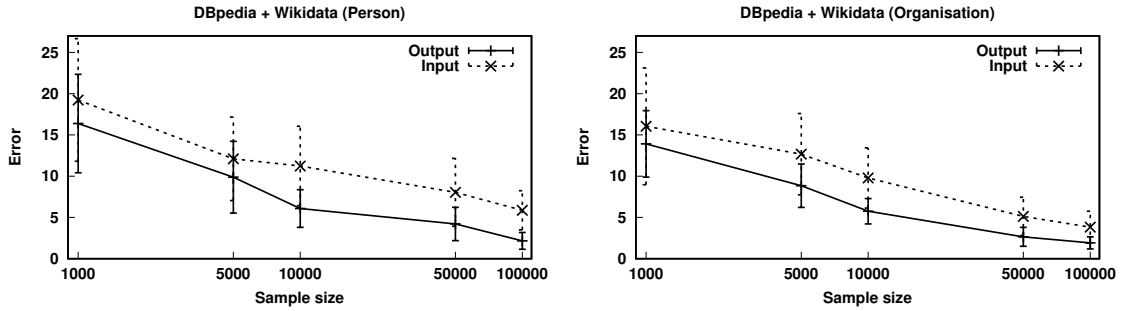


FIGURE 6.8 – Evolution de l'erreur de norme euclidienne pour l'échantillonnage en entrée et en sortie

6.4.5 Evaluation qualitative

Enfin, nous analysons qualitativement certains résultats obtenus pour la classe `Person` de DBpedia. Rappelons tout d'abord que cette classe contient plus de 700,000 entités (voir le tableau 3.1). Pour ces expérimentations, nous continuons à utiliser la méthode FPOF avec la mesure d'intérêt fondée sur la norme $u_{\leq 3}$ et à approximer le FPOF exact en utilisant un échantillon de $10,000$ motifs. Ensuite, nous calculons le sous-ensemble $L_{\texttt{Person}}$ (resp. $H_{\texttt{Person}}$) de $K = 100$ instances de la classe `Person` ayant le plus petit FPOF approximé (respectivement le plus élevé FPOF approximé).

En premier lieu, nous analysons manuellement $L_{\texttt{Person}}$ (resp. $H_{\texttt{Person}}$) selon le FPOF. Pour chacun des deux cas, nous rangeons dans une liste les instances de la base de données correspondante suivant l'ordre décroissant des FPOF. De ce fait, les données aberrantes sont en bas de la liste et ont donc des scores très faibles. A l'inverse, les vraies personnes sont en haut de la liste avec des scores très élevés.

Tout d'abord, il est intéressant de voir que toutes les entités de $H_{\texttt{Person}}$ sont de vraies personnes. La plupart d'entre elles sont des artistes (48%), avec Hugo, Tolstoï, Picasso, Beckett et Tolkien dans le top-5 ou des hommes politiques (40%), avec Roosevelt, Hitler, Churchill, Lénine et Staline aux top-5. Ensuite, nous évaluons si les entités dans $L_{\texttt{Person}}$ sont vraiment des personnes ou non. Premièrement, nous notons que 48% d'entre elles sont de vraies personnes. Cependant, ces personnes réelles sont décrites avec un très petit nombre de propriétés (moins de 24, 15 en moyenne). Ensuite, nous voyons que 33% d'entre elles ne sont pas des personnes réelles, mais des personnages de fiction (qui peuvent être considérés comme des anomalies dans un ensemble contenant un grand nombre de personnes réelles). Enfin, 19% des instances dans $L_{\texttt{Person}}$ sont de véritables anomalies. Plus précisément, 9% d'entre elles devraient être classées dans d'autres sous-classes de la classe *Agent* de DBpedia que la classe `Person`, et 10% dans une classe de DBpedia qui n'est pas une sous-classe de classe *Agent*.

Maintenant nous analysons manuellement les 50 pires entités de $L_{\texttt{Person}}$ toujours selon le FPOF. Encore mieux, nous notons que seules 36 % des entités ayant les FPOF les plus faibles sont de vraies personnes (décrites avec peu de propriétés), et 64 % d'entre elles peuvent être considérées comme des données aberrantes. En effet, 44% des entités sont des personnages fictifs et, plus important encore, 8% d'entre elles doivent être classées dans `Organisation` (une classe soeur de `Person`), et même, 12% d'entre elles doivent être classées dans une autre classe de *DBpedia* (par exemple, *Event*).

D'après ces analyses, nous avons vu que les motifs retournés par DDSampling sont très intéressants pour bien approximer les FPOF des instances afin de trouver les anomalies.

6.5 Conclusion

Dans ce chapitre nous avons proposé une méthode générique d'échantillonnage de motifs dans des bases de données distribuées. Il permet de considérer tous les types de partitionnement d'une base de données (horizontal, vertical et hybride) et toutes mesures d'intérêt fondées sur la norme. Puisqu'on ne centralise que les longueurs des transactions, les coûts de communication de DDSampling sont faibles car seuls les items nécessaires à

la construction des motifs de l'échantillon sont centralisés. Les expérimentations ont montré ce faible coût sur plusieurs jeux de données quel que soit le type de partitionnement. Nous avons aussi illustré l'intérêt des motifs échantillonnés en détectant des anomalies sur les classes `Person` et `Organisation` des triplestores du Web sans centraliser les données. Par ailleurs, nous avons revisité le calcul du FPOF en ajoutant des contraintes de norme sur les motifs échantillonnés. La méthode ainsi proposée a permis de réduire considérablement la malédiction de la longue traîne dont celle proposée dans [He *et al.*, 2005] souffrait face aux gros jeux de données atteints de la malédiction de la longue traîne.

Cependant, il est clair que la construction de la matrice de pondération peut être très coûteuse en temps et en espace mémoire car elle dépend exclusivement du nombre d'instances et du nombre de fragments de la base de données. Dans les travaux futurs, nous envisageons de remplacer le tirage exact des transactions par une méthode stochastique afin de ne plus avoir à centraliser les tailles de toutes les transactions de chacun des fragments. Par ailleurs, il serait également intéressant de proposer un mécanisme de correction des poids afin de contre-balancer la panne d'un site.

Chapitre 7

Echantillonnage de motifs par trie

Sommaire

Nous avons présenté à la section 3.3 la notion de trie comme un moyen efficace pour le stockage d'une base de données transactionnelles. Dans ce chapitre, nous présentons une nouvelle méthode d'échantillonnage en sortie de motifs qui se base sur une structure de données compressée. L'objectif principal est donc de proposer un algorithme générique d'échantillonnage en sortie de motifs qui tire des motifs proportionnellement à une mesure d'intérêt fondée sur la norme à partir d'un trie d'occurrences de motifs. La grande différence entre la méthode que nous allons proposer et l'algorithme 4, présenté à la page 100, résulte de la structure de données à partir de laquelle les motifs vont être tirés.

Nos principales contributions dans ce chapitre sont les suivantes :

— Nous introduisons une nouvelle structure appelée *trie d'occurrences de motifs* puis nous proposons un algorithme pour sa construction. C'est un trie où chaque noeud a un ensemble d'informations relatives aux poids pour le tirage exact d'un motif. Dans notre cas, nous pondérons chaque noeud suivant les normes des occurrences de motifs dans le sous-trie dont il est la racine.

— Nous proposons TPSampling (Trie based Pattern Sampling), un algorithme générique d'échantillonnage en sortie de motifs à partir d'un trie d'occurrences de motifs suivant une probabilité proportionnelle à une mesure d'intérêt fondée sur la norme. La généricité de TPSampling vient du fait qu'il peut prendre en compte toute mesure d'intérêt fondée sur la norme.

— Nous évaluons théoriquement et expérimentalement la complexité de TPSampling. Notamment, nous allons montrer que TPSampling parvient à faire un tirage exact suivant une mesure d'intérêt fondée sur la norme choisie par l'utilisateur et directement à partir du trie. En outre, nous évaluons la complexité en stockage mémoire du trie d'occurrences de motifs sur différents jeux de données transactionnelles puis nous évaluons sa rapidité suivant différentes mesures d'intérêt fondées sur la norme.

Le reste de ce chapitre est scindé comme suit : la section 7.1 commence par présenter quelques notions de base pour bien comprendre notre approche sur l'échantillonnage en sortie de motifs à partir d'un trie. Ensuite, elle pose les défis que nous devons résoudre dans ce chapitre. La section 7.2 présente notre première contribution sur les tries qui consiste à montrer comment construire un trie d'occurrences de motifs τ. La section 7.3 décrit notre algorithme générique d'échantillonnage en sortie de motifs proportionnellement à une mesure d'intérêt fondée sur la norme. La section 7.4 analyse théoriquement notre méthode en détaillant le temps de la construction d'un trie d'occurrences de motifs et celui du tirage d'un motif par TPSampling. Enfin, la section 7.5 présente les résultats expérimentaux de notre approche en les comparant avec ceux de la méthode en deux étapes de l'algorithme 4.

7.1 Notions élémentaires et formalisation du problème

Avant de formaliser la problématique, nous allons commencer par définir quelques notions de base.

7.1.1 Notions de base

A la section 3.3, la figure 3.6-(a) à la page 84 montre que 2 occurrences d'un même motif peuvent se retrouver dans différentes branches du trie, par exemple BE au sein du chemin $D \rightarrow B \rightarrow E$ et BE au sein du chemin $D \rightarrow A \rightarrow B \rightarrow E$. Mais elles peuvent être confondues, le cas des occurrences du motif DA (apparaissant dans les transactions d'identifiant 1, 2 et 4) qui sont représentées dans une seule et même branche du trie. D'après cette analyse, l'approche que nous allons présenter dans ce chapitre se base sur deux notions fondamentales qui sont celles d'*occurrence de motif* et de *langage d'occurrences de motifs*.

Définition 25 (Occurrence et langage d'occurrences de motifs)**.** *Soit $\mathcal{T}$ une base de données transactionnelles. S'il existe une transaction t d'identifiant i dans $\mathcal{T}$ contenant le motif $\varphi \in \mathcal{L}_{\mathcal{I}}$, alors nous notons φ_i l'occurrence du motif φ dans la transaction t_i. L'ensemble des occurrences de motifs de la base de données $\mathcal{T}$ forme un langage d'occurrences de motifs noté par $\mathcal{L}_o$. Formellement, $\mathcal{L}_o = \{\varphi_i : (\exists(\varphi, t_i) \in \mathcal{L}_{\mathcal{I}} \times \mathcal{T})(\varphi \subseteq t_i)\}$. La norme d'une occurrence de motif φ_i d'un motif $\varphi \in \mathcal{L}_{\mathcal{I}}$ est égale à la norme de φ.*

On peut noter que $\mathcal{L}_o$ est un ensemble d'occurrences de motifs alors que $\mathcal{L}_{\mathcal{I}}$ est un ensemble de motifs. Notons que contrairement à un motif, une occurrence de motif appartient à une et une seule transaction.

Dans le reste de ce chapitre, nous allons considérer la base de données transactionnelles $\mathcal{T}$ de la figure 7.1 pour donner des exemples avec son trie. Les valeurs stockées dans les noeuds représentent le nombre de fois que leurs labels terminent une transaction de la base de données.

$\mathcal{T}$

tid	Transaction
1	A B
2	A C
3	B C
4	A B C D

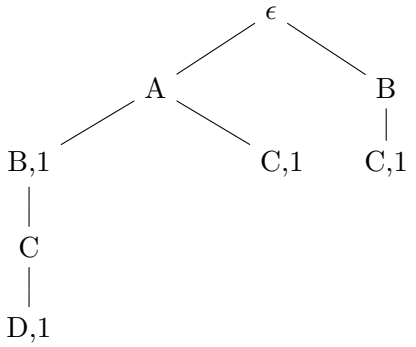


FIGURE 7.1 – Représentation d'une base de données $\mathcal{T}$ sous la forme d'un trie

Exemple 27. *Si on considère les motifs de norme 2, alors on note que AB_1 et AB_4 sont les 2 occurrences du motif AB dans la base de données $\mathcal{T}$ car AB est contenu dans les transactions d'identifiant 1 et 4. La fréquence de AB est donc égale à 2. De même, AC_2 et AC_4 sont les deux occurrences du motif AC. BC_3 et BC_4 sont les deux occurrences de BC. AD, BD et CD ont chacun une seule occurrence dans la transaction t_4. Dans cet exemple, pour tirer un motif de norme 2 proportionnellement à sa fréquence dans la base de données $\mathcal{T}$, il suffit de tirer uniformément une occurrence de motif dans l'ensemble $E = \{AB_1, AB_4, AC_2, AC_4, BC_3, BC_4, AD_4, BD_4, CD_4\}$. Ainsi, la probabilité de tirer le motif AB dans l'ensemble E est égale à $\pi(\mathcal{S}amp\ell e_1(\mathcal{L}_{\mathcal{I}}, \mathcal{T}, freq \times u_{[2,2]}) = AB) = \pi(\mathcal{S}amp\ell e_1(\mathcal{L}_o, \mathcal{T}, u_{[2..2]}) = AB_1) + \pi(\mathcal{S}amp\ell e_1(\mathcal{L}_o, \mathcal{T}, u_{[2..2]}) = AB_4) = \frac{1}{9} + \frac{1}{9} = \frac{2}{9}$.*

7.1.2 Défis de l'échantillonnage en sortie à base de trie

D'après cet exemple, nous savons tirer un motif de norme ℓ proportionnellement à sa fréquence parmi l'ensemble des motifs de même norme à l'aide d'un tirage uniforme d'occurrences de motifs. Dans ce cas, si on sait tirer une norme ℓ proportionnellement à la somme des utilités des motifs de norme ℓ, alors on sait tirer un motif proportionnellement à son intérêt dans la base de données.

Les défis que nous voulons résoudre dans ce chapitre peuvent finalement se poser comme suit :

Soient une base de données transactionnelles $\mathcal{T}$ et une utilité fondée sur la norme $u \in \mathcal{U}$.

1. **Quelles informations supplémentaires faut-il ajouter à un trie (classique) pour être ensuite capable de faire un échantillonnage sans avoir besoin d'utiliser la base de données transactionnelles sous-jacente ?**
2. **Comment tirer un motif φ du langage de motifs $\mathcal{L}_\mathcal{I}$ proportionnellement à $freq(\varphi, \mathcal{T}) \times f_u(||\varphi||)$ directement à partir du trie enrichi, appelé trie d'occurrences de motifs ?**

7.2 Pondération du trie d'occurrences de motifs

Dans cette section, nous allons commencer par présenter les éléments nécessaires pour définir un trie d'occurrences de motifs. Nous terminons avec un exemple illustratif pour la construction d'un trie d'occurrences de motifs.

7.2.1 Définition d'un trie d'occurrences de motifs

Nous avons vu qu'au chapitre 3, la figure 3.6 à la page 84 montre qu'un trie d'une base de données transactionnelles change de forme suivant la relation d'ordre total considérée sur les items. Ainsi, nous allons commencer par définir les relations d'ordre total utilisées dans ce chapitre de la thèse avant de parler de l'identifiant et du contenu d'un noeud.

Définition 26 (Relation d'ordre total entre les items)**.** *Soit $\mathcal{I}$ un ensemble d'items ou littéraux sur lequel est défini la base de données transactionnelles $\mathcal{T}$.*

- *Une relation d'ordre total sur les littéraux est dite fondée sur l'ordre lexicographique et notée par $>_\mathcal{I}^{lexico}$, si elle ordonne les éléments de $\mathcal{I}$ suivant l'ordre lexicographique.*
- *Une relation d'ordre total sur les littéraux est dite fondée sur la fréquence et notée par $>_\mathcal{I}^{freq}$, si elle ordonne les éléments de $\mathcal{I}$ suivant l'ordre décroissant de leurs fréquences dans $\mathcal{T}$ et suivant l'ordre lexicographique en cas d'égalité de fréquence.*

Dans la suite nous désignons $>_\mathcal{I}$ une relation d'ordre total entre les items d'une base de données. Notons qu'il existe d'autres types de relations d'ordre total dans la littérature qui peuvent être appliquées sur les littéraux. Cependant, nous allons nous limiter aux deux relations d'ordre total précédemment définies en guise d'exemples, même si notre approche marche avec toute relation d'ordre total sur les littéraux.

Exemple 28. *En considérant la base de données $\mathcal{T}$, nous avons $freq(A, \mathcal{T}) = 3$, $freq(B, \mathcal{T}) = 3$, $freq(C, \mathcal{T}) = 3$ et $freq(D, \mathcal{T}) = 1$. Dans ce cas, nous pouvons déjà dire que $A >_\mathcal{I}^{freq} D$, $B >_\mathcal{I}^{freq} D$ et $C >_\mathcal{I}^{freq} D$ car $freq(A, \mathcal{T}) = freq(B, \mathcal{T}) = freq(C, \mathcal{T}) > freq(D, \mathcal{T})$. Maintenant, si on tient compte de l'ordre lexicographique entre les items, nous avons $A >_\mathcal{I}^{lexico} B >_\mathcal{I}^{lexico} C$. Donc, en continuant avec la relation d'ordre $>_\mathcal{I}^{freq}$, nous avons $A >_\mathcal{I}^{freq} B >_\mathcal{I}^{freq} C >_\mathcal{I}^{freq} D$.*

Nous allons maintenant définir la notion d'identifiant d'un noeud dans un trie. C'est un concept qui nous permettra d'enrichir le trie d'occurrences de motifs depuis une base de données transactionnelles.

Définition 27 (Identifiant d'un noeud)**.** *Etant donnés un ensemble d'items* $\mathcal{I} = \{e_1, \cdots, e_n\}$ *et un symbole* $\epsilon \notin \mathcal{I}$*, un trie* τ *défini sur* $\mathcal{I}$ *est un arbre où tout noeud* $\eta \in \tau$ *excepté la racine contient un label noté* $\eta.label$ *appartenant à* $\mathcal{I}$*, i.e.* $\eta.label \in \mathcal{I}$*, et où la racine* r *de* τ *contient le label* ϵ*, i.e.* $r.label = \epsilon$*. Ainsi, tout noeud* $\eta \in \tau$ *peut être identifié par la séquence des labels des noeuds sur le chemin allant de la racine de* τ *au noeud* η*. Si* $P = \epsilon e_{i_1} \dots e_{i_k}$ *est cette séquence, on la note plus simplement* $P = e_{i_1} \dots e_{i_k}$ *pour un noeud non racine, et on note* $\widetilde{P} = e_{i_k}$ *le label du noeud* η *identifié, i.e.* $\widetilde{P} = \eta.label$*.*

Exemple 29. *En considérant le trie de la base de données* $\mathcal{T}$ *de la figure 7.1, le noeud contenant le label* B *issu de la transaction* t_4 *a pour identifiant* $P_1 = AB$*. Ce même noeud d'identifiant* $P_1 = AB$ *représente l'item* B *issu de* t_1*. Par contre, l'item* B *de la transaction* t_3 *est représenté par le noeud d'identifiant* $P_2 = B$*.*

Dans la suite, nous allons souvent parler de la notion de sous-trie d'un trie. La définition 28 introduit la notion de sous-trie.

Définition 28 (Sous-trie)**.** *Soit* τ *un trie et* P *l'identifiant d'un noeud de* τ*. On note* τ_P *le sous-trie de* τ *dont la racine est le noeud d'identifiant* P*.*

Exemple 30. *Le trie de la figure 7.2 est un sous-trie du trie* τ *de la figure 7.1. La racine de ce sous-trie est le noeud d'identifiant* $P = A$ *dans le trie* τ*.*

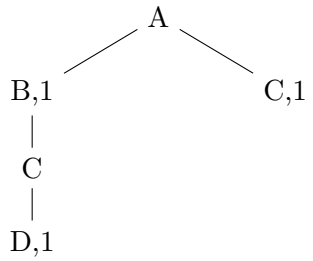


FIGURE 7.2 – Sous-trie τ_A d'occurrences de motifs du trie τ de la base de données $\mathcal{T}$

Dans la suite, nous allons définir l'opérateur de concaténation $\circ$ comme suit :

Définition 29 (Opérateur de concaténation $\circ$)**.** *Soient* X *et* Y *deux itemsets définis dans* $\mathcal{I}$ *et ordonnés suivant la relation d'ordre total* $>_{\mathcal{I}}$*,* $X \circ Y = X \cup \{y \in Y : (\forall x \in X)(x >_{\mathcal{I}} y)\}$*.*

Si $>_{\mathcal{I}}$ est l'ordre lexicographique, alors on a $B \circ AC = BC$. Par contre, on a $A \circ BC = ABC$.

Cet opérateur de concaténation nous permet de définir la notion de préfixe que nous allons utiliser dans la définition d'une base projetée.

Définition 30 (Préfixe)**.** *Soient* t *une transaction définie dans* $\mathcal{I}$ *et* P *un itemset défini dans* $\mathcal{I}$ *et ordonné suivant la relation d'ordre total* $>_{\mathcal{I}}$*.* P *est un préfixe de la transaction* t *s'il existe un itemset* $Y \subseteq \mathcal{I}$ *tel que* $t = P \circ Y$*.*

Dans la base de données $\mathcal{T}$ de la figure 7.1, $P = AB$ est un préfixe de la transaction $t_4 = ABCD$ mais P n'est pas un préfixe de la transaction $t_3 = BC$.

D'après la définition 30, les transactions de la base de données ayant un préfixe commun peuvent être regroupées. Pour identifier où sont représentées dans un trie les occurrences d'un motif, i.e. dans quel sous-trie, nous allons maintenant introduire la notion de base projetée. Notre définition est une adaptation de la notion de base projetée introduite par [Han *et al.*, 2004].

Définition 31 (Base projetée)**.** *Soient $>_{\mathcal{I}}$ une relation d'ordre total sur l'ensemble des items $\mathcal{I}$, $\mathcal{T}$ une base de données transactionnelles et P l'identifiant d'un noeud. Une base projetée de $\mathcal{T}$ sur P, dénotée par $\mathcal{T}_P$, est une base de données transactionnelles qui contient pour toute transaction t de $\mathcal{T}$ de préfixe P une copie de cette transaction sans les items de t précédant $\widetilde{P}$. Formellement, la base projetée de $\mathcal{T}$ sur le préfixe P est définie par :*

$$\mathcal{T}_P = \{(i, \widetilde{P} \circ \varphi) \in \mathbb{N} \times \mathcal{L}_{\mathcal{I}} : (i, t) \in \mathcal{T} \wedge t = P \circ \varphi\}$$

Exemple 31. *Considérons le trie de la base de données transactionnelles $\mathcal{T}$ de la figure 7.1. Les occurrences de motifs dans la base projetée $\mathcal{T}_{AB}$ sont les occurrences de motifs stockées dans le sous-trie dont la racine est identifiée par le préfixe AB : B_1, B_4, BC_4, BD_4, CD_4, BCD_4. Les occurrences de motifs dans la base projetée $\mathcal{T}_{AC}$ sont les occurrences de motifs stockées dans le sous-trie dont la racine est identifiée par le préfixe AC : C_2.*

Maintenant, nous allons définir précisément quelles sont les occurrences de motifs représentées au niveau de la base projetée de $\mathcal{T}$ sur le préfixe P, en partitionnant ces ensembles d'occurrences par leur norme.

Définition 32 (Calcul des poids Φ_ℓ^- et Φ_ℓ^+)**.** *Soient $\mathcal{T}$ une base de données transactionnelles et P un préfixe. L'ensemble des occurrences de motifs de la base projetée sur le préfixe P et ayant une norme égale à ℓ est défini par :*

$$\phi_\ell(P, \mathcal{T}) = \{(i, \varphi) \in \mathbb{N} \times \mathcal{L}_{\mathcal{I}} : (i, X) \in \mathcal{T}_P \wedge \varphi \subseteq X \wedge ||\varphi|| = \ell\}$$

$\Phi_\ell(P, \mathcal{T})$ désigne le nombre total d'occurrences de motifs de norme égale à ℓ dans la base projetée $\mathcal{T}_P$.

- *L'ensemble des occurrences de motifs de la base projetée sur le préfixe P, ayant une norme égale à ℓ et contenant l'item $\widetilde{P}$ est défini par :*
  $\phi_\ell^+(P, \mathcal{T}) = \{(i, \varphi) \in \phi_\ell(P, \mathcal{T}) : \widetilde{P} \in \varphi\}$. Sa cardinalité est notée par $\Phi_\ell^+(P, \mathcal{T}) = |\phi_\ell^+(P, \mathcal{T})|$.
- *L'ensemble des occurrences de motifs de la base projetée sur le préfixe P, ayant une norme égale à ℓ et ne contenant pas l'item $\widetilde{P}$ est défini par :*
  $\phi_\ell^-(P, \mathcal{T}) = \{(i, \varphi) \in \phi_\ell(P, \mathcal{T}) : \widetilde{P} \notin \varphi\}$. Sa cardinalité est notée par $\Phi_\ell^-(P, \mathcal{T}) = |\phi_\ell^-(P, \mathcal{T})|$.

Par ailleurs, comme $\widetilde{P}$ n'est pas défini si P est le préfixe vide, on considère par convention que $\phi_\ell^-(\epsilon, \mathcal{T}) = \phi_\ell(\epsilon, \mathcal{T})$, alors que $\phi_\ell^+(\epsilon, \mathcal{T}) = \emptyset$.

Exemple 32. *En poursuivant avec la base projetée* $\mathcal{T}_A = \{(1, AB), (2, AC), (4, ABCD)\}$ *de l'exemple 31, nous avons* $\phi_2^+(A, \mathcal{T}) = \{(1, AB), (2, AC), (4, AB), (4, AC), (4, AD)\}$. *Donc* $\Phi_2^+(A, \mathcal{T}) = 5$. *D'autre part, nous avons aussi* $\phi_2^-(A, \mathcal{T}) = \{(4, BC), (4, BD), (4, CD)\}$. *Donc* $\Phi_2^-(A, \mathcal{T}) = 3$.

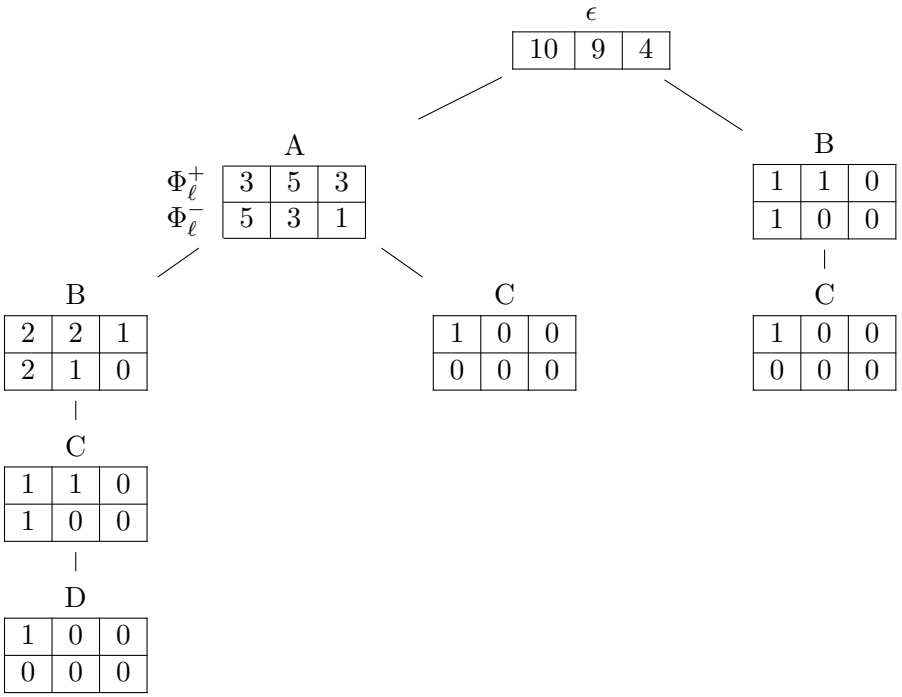


FIGURE 7.3 – Trie enrichi de la base de données transactionnelles $\mathcal{T}$ de la figure 7.1

Par la suite, au niveau du noeud d'identifiant P d'un trie, nous stockerons comme poids les cardinalités de ces ensembles, en distinguant les sous-ensembles d'occurrences de norme ℓ contenant ou pas l'item $\widetilde{P}$.

Après avoir introduit les notions et notations nécessaires à l'enrichissement d'un trie classique en un trie d'occurrences de motifs, nous allons définir explicitement ce dernier. Dans le reste de ce chapitre, la notation τ désigne un trie d'occurrences de motifs.

Définition 33 (Trie d'occurrences de motifs)**.** *Etant donné une base de données transactionnelles, un trie d'occurrences de motifs pour* $\mathcal{T}$*, noté* τ*, est un arbre où chaque noeud* $\eta \in \tau$ *contient :*

- *un label noté* $\eta.label$ *appartenant à* $\mathcal{I} \cup \{\epsilon\}$, ϵ *étant le label réservé à la racine.*
- *une liste de fils notée* $\eta.fils$*. Par la suite, on note* $|\eta.fils|$ *le nombre de fils du noeud* η *et* $\eta.fils[i]$ *le i-ième fils de* η *pour* $i \in [1..k]$ *avec* $k = |\eta.fils|$.
- *un tableau de poids positifs noté* $\eta.\Phi^+$*. Si* P *est l'identifiant du noeud* η *dans* τ*, alors* $\eta.\Phi^+[\ell] = \Phi_\ell^+(P, \mathcal{T})$ *pour* $\ell \in [\mu..M]$, μ *et* M *étant les contraintes de normes minimale et maximale considérées pendant la construction du trie.*

— *un tableau de poids négatifs noté $\eta.\Phi^-$. Si P est l'identifiant du noeud η dans τ, alors $\eta.\Phi^-[\ell] = \Phi^-_\ell(P, \mathcal{T})$ pour $\ell \in [\mu..M]$, μ et M étant les contraintes de normes minimale et maximale considérées pendant la construction du trie.*

Exemple 33. *En reprenant la base de données transactionnelles de la figure 7.1, les contraintes de normes minimale $\mu = 1$ et maximale $M = 3$ nous construisons le trie enrichi suivant la relation d'ordre total $\mathcal{T}$ à la figure 7.3.*

Dans cet exemple, l'ensemble des labels des fils de la racine r est $\{A, B\}$. Le nombre de motifs de norme $\ell = 2$ dans le trie τ est égal à $r.\Phi^-[2] = 9$. Soit η le noeud d'identifiant $P = \epsilon A$. Nous avons alors $\eta.\Phi^+[2] = 5$ et $\eta.\Phi^-[2] = 3$ pour dire que le sous-trie τ_A contient 5 occurrences de motifs de norme 2 contenant l'item $\widetilde{P} = A$ (AD_4, AC_4, AB_4, AB_1, AC_2) et 3 occurrences de motifs de norme 2 ne contenant pas l'item $\widetilde{P} = A$ (CD_4, BD_4, BC_4).

Nous allons maintenant écrire l'algorithme de construction d'un trie d'occurrences de motifs en se basant sur les définitions de la section 7.2.1.

7.2.2 Algorithme de construction d'un trie d'occurrences de motifs

Nous allons à présent montrer comment construire un trie d'occurrences de motifs à partir d'une base de données transactionnelles afin de prendre efficacement en compte tout type de mesure d'intérêt fondée sur la norme. Notons d'abord que cette construction se fait itérativement en ajoutant une à une les transactions dans le trie. Dans notre cas, nous avons besoin de calculer les apports positifs et négatifs de chaque transaction t au sein de chaque noeud d'identifiant P tel que P est un préfixe de t.

Propriété 9. *Soit $\mathcal{T} = \{t_1, \cdots, t_n\}$ une base de données transactionnelles. Notons $\mathcal{T}_i$ le sous-ensemble de transactions défini par $\mathcal{T}_i = \{t_k \in \mathcal{T} : 1 \leq k \leq i\}$. Si P un préfixe de t_i, alors on a :*

$$\Phi^+_\ell(P, \mathcal{T}_i) = \Phi^+_\ell(P, \mathcal{T}_{i-1}) + \binom{|t_i| - |P|}{\ell - 1}$$

$$\Phi^-_\ell(P, \mathcal{T}_i) = \Phi^-_\ell(P, \mathcal{T}_{i-1}) + \binom{|t_i| - |P|}{\ell}$$

Par convention, $\Phi^-_\ell(P, \mathcal{T}_0) = 0$ quel que soit l'identifiant P. Si P est l'identifiant de la racine, $P = \epsilon$, alors $|P| = 0$, et dans ce cas, nous convenons que $\Phi^+_\ell(P, \mathcal{T}) = 0$.

Preuve. D'après la définition 32, on sait que d'une part $\Phi^+_\ell(P, \mathcal{T}_i) = \Phi^+_\ell(P, \mathcal{T}_{i-1}) + \Phi^+_\ell(P, \{t_i\})$ et d'autre part $\Phi^-_\ell(P, \mathcal{T}_i) = \Phi^-_\ell(P, \mathcal{T}_{i-1}) + \Phi^-_\ell(P, \{t_i\})$. Si P est un préfixe de t_i, alors d'après la définition 30, il existe un itemset Y tel que $t_i = P \circ Y$. Ainsi, $\Phi^+_\ell(P, \{t_i\}) = |\{\widetilde{P} \circ \varphi : \varphi \subseteq Y \wedge ||\varphi|| = \ell - 1\}|$. Donc, nous avons $\Phi^+_\ell(P, \{t_i\}) = \binom{|t_i|-|P|}{\ell-1}$. D'autre part, $\Phi^-_\ell(P, \{t_i\}) = |\{\varphi \subseteq Y : ||\varphi|| = \ell\}|$. Donc, nous avons aussi $\Phi^-_\ell(P, \{t_i\}) = \binom{|t_i|-|P|}{\ell}$. D'où le résultat. □

Lors de l'ajout de la transaction t_i dans le trie τ, les termes $\binom{|t_i|-|P|}{\ell-1}$ et $\binom{|t_i|-|P|}{\ell}$ sont appelés respectivement l'apport positif et l'apport négatif de la transaction t_i aux occurrences de motifs de norme ℓ du noeud identifié par le préfixe P.

Exemple 34. *Poursuivons avec l'exemple 32 en calculant les poids* $\Phi_2^+(A,\mathcal{T})$ *et* $\Phi_2^-(A,\mathcal{T})$ *mais cette fois-ci à l'aide de la propriété 9. Par définition, on a* $\mathcal{T}_A = \{(1,AB),(2,AC),(4,ABCD)\}$.
D'après la propriété 9, on a : $\Phi_2^+(A,\mathcal{T}_1) = 0 + \binom{|t_1|-1}{2-1} = \binom{2-1}{1} = \binom{1}{1} = 1$. *Ensuite, en ajoutant* t_2 *on a* $\Phi_2^+(A,\mathcal{T}_2) = \Phi_2^+(A,\mathcal{T}_1) + \binom{|t_2|-1}{2-1} = 1 + \binom{2-1}{1} = 1 + \binom{1}{1} = 1+1 = 2$. *L'ajout de la transaction* t_3 *n'affecte pas les poids du noeud d'identifiant* $P = A$ *car dans ce cas,* P *n'est pas un préfixe de* t_3. *Donc on a* $\Phi_2^+(A,\mathcal{T}_3) = \Phi_2^+(A,\mathcal{T}_2) = 2$. *Enfin, en ajoutant la transaction* t_4, *on a* $\Phi_2^+(A,\mathcal{T}_4) = \Phi_2^+(A,\mathcal{T}_3) + \binom{|t_4|-1}{2-1} = 2 + \binom{4-1}{1} = 2 + \binom{3}{1} = 2+3 = 5$.

On a aussi $\Phi_2^-(A,\mathcal{T}_1) = 0 + \binom{|t_1|-1}{2} = \binom{2-1}{2} = \binom{1}{2} = 0$. *Avec l'ajout de la transaction* t_2 *on a* $\Phi_2^-(A,\mathcal{T}_2) = \Phi_2^-(A,\mathcal{T}_1) + \binom{|t_2|-1}{2} = 0 + \binom{2-1}{2} = \binom{1}{2} = 0$. *De même, l'ajout de la transaction* t_3 *n'affecte pas les poids du noeud d'identifiant* $P = A$. *Donc on a* $\Phi_2^-(A,\mathcal{T}_3) = \Phi_2^-(A,\mathcal{T}_2) = 0$. *Finalement, en ajoutant la transaction* t_4, *on a* $\Phi_2^-(A,\mathcal{T}_4) = \Phi_2^-(A,\mathcal{T}_3) + \binom{|t_4|-1}{2} = 0 + \binom{4-1}{2} = \binom{3}{2} = 3$.

Nous avons aussi besoin d'introduire quelques fonctions de base pour la création, l'ajout ou la recherche d'un noeud lors de l'insertion des items d'une transaction dans un trie.

— Soit la fonction *CreateNode* définie par $\eta \leftarrow CreateNode(e)$ où η est un noeud tel que $\eta.label = e$, $\eta.fils = \emptyset$ où $\emptyset$ représente ici une liste vide de noeuds, et $\eta.\Phi^+[\ell] = \eta.\Phi^-[\ell] = 0$ pour $\ell \in [\mu..M]$.
— Soit la fonction *SearchChild* définie par $\eta.fils[i] \leftarrow SearchChild(e,\eta)$ s'il existe i tel que $\eta.fils[i].label = e$, *null* sinon.
— Soit *AddChild* la fonction permettant d'ajouter un fils à un noeud. Plus précisément, si η est un noeud tel que $k = |\eta.fils|$, on considérera qu'après exécution de $AddChild(c,\eta)$, on a $|\eta.fils| = k+1$ et $\eta.fils[k+1] = c$.

Dans la suite, $t[j]$, avec j un entier naturel, est le $j-ième$ item de la transaction t suivant la relation d'ordre total $>_{\mathcal{I}}$.

L'algorithme 7 décrit l'algorithme de création d'un trie d'occurrences de motifs d'une base de données en entrée $\mathcal{T}$ suivant une relation d'ordre total $>_{\mathcal{I}}$. On commence par initialiser le trie d'occurrences de motifs (ligne 3) en créant un noeud vide avec la fonction *CreateNode*. Pour chaque transaction t de la base de données en entrée dont les items suivent la relation d'ordre $>_{\mathcal{I}}$, on se place à la racine puis, à l'aide de la propriété 9, on calcule son apport total dans le trie suivant les normes que l'on ajoute à la racine (ligne 6). Ensuite, pour chaque item $t[j]$ de la transaction en cours d'insertion dans le trie, s'il n'existe pas un noeud fils c ayant pour label l'item $t[j]$ d'après la fonction *SearchChild* (ligne 9), on le crée à l'aide de la fonction *CreateNode* (ligne 11) puis on l'ajoute parmi les fils de η avec la fonction *AddChild* (ligne 12). Enfin, on ajoute les apports positif et négatif de la transaction t au noeud c (lignes 13 à 15) à l'aide de la propriété 9. On se place maintenant au niveau du noeud c (ligne 16) et le processus recommence avec l'item à la position $j+1$ dans t. Finalement, à la ligne 17, on retourne le trie d'occurrences de motifs τ pour la base de données transactionnelles $\mathcal{T}$.

Algorithm 7 Construction d'un trie d'occurrences de motifs

Input : Une base de données transactionnelles $\mathcal{T}$, les contraintes de normes minimale μ et maximale M
Output : Un trie d'occurrences de motifs τ
$\tau \leftarrow CreateNode(\epsilon)$ ▷ Création du noeud racine du trie
for $t \in \mathcal{T}$ **do**
  for $\ell \leftarrow \mu$ to M **do**
    $\tau.\Phi^{-}[\ell] \leftarrow \tau.\Phi^{-}[\ell] + \binom{||t||}{\ell}$ ▷ Ajouter l'apport de t à la racine
  $\eta \leftarrow \tau$
  for $j \leftarrow 1$ to $||t||$ **do**
    $c \leftarrow SearchChild(t[j], \eta)$
    if $c = null$ **then** ▷ Si c n'est pas fils du noeud η
      $c \leftarrow CreateNode(t[j])$ ▷ alors on le crée
      $AddChild(c, \eta)$
    for $\ell \leftarrow \mu$ to M **do**
      $c.\Phi^{+}[\ell] \leftarrow c.\Phi^{+}[\ell] + \binom{||t||-j}{\ell-1}$ ▷ Ajouter l'apport positif de t au noeud c
      $c.\Phi^{-}[\ell] \leftarrow c.\Phi^{-}[\ell] + \binom{||t||-j}{\ell}$ ▷ Ajouter l'apport négatif de t au noeud c
    $\eta \leftarrow c$
return τ

7.2.3 Exemple de construction d'un trie d'occurrences de motifs

En considérant la relation d'ordre total $>_{\mathcal{I}}^{freq}$ entre les items, construisons le trie d'occurrences de motifs de la base de données transactionnelles $\mathcal{T}$ de la figure 7.1 reportée au tableau 7.1.

D'abord, nous allons calculer la fréquence de chaque item puis nous fixons l'ordre des items. En scannant la base de données $\mathcal{T}$, nous avons $A >_{\mathcal{I}}^{freq} B >_{\mathcal{I}}^{freq} C >_{\mathcal{I}}^{freq} D$ d'après l'exemple 28. Tous les items de chaque transaction de la base de données doivent suivre cet ordre. Notons que cet ordonnancement est important puisque chaque branche du trie le suit aussi.

TABLE 7.1 – Exemple d'une base de données transactionnelles $\mathcal{T}$

tid	Transaction
1	A B
2	A C
3	B C
4	A B C D

En supposant maintenant que nous voulons échantillonner des motifs de norme comprise entre $\mu = 1$ et $M = 3$, nous allons commencer la construction du trie en créant la racine avec la fonction $CreateNode(\epsilon)$. Nous ne mentionnons pas la liste exhaustive des

calculs des poids, mais la figure 7.4 donne tous les résultats intermédiaires pour la construction du trie d'occurrences de motifs τ correspondant à la base de données $\mathcal{T}$. Maintenant, nous allons montrer pas à pas comment insérer les transactions dans le trie en calculant le poids de chaque noeud à chaque fois qu'une occurrence de motif y est compté.

1. *insert*(t_1) : On calcule les poids de la racine r : $r.\Phi^-[\ell] \leftarrow \binom{||AB||}{\ell}$, avec $\ell \in [1..3]$ (lignes 5 et 6). On a les poids suivants $r.\Phi^-[1] \leftarrow 2$, $r.\Phi^-[2] \leftarrow 1$ et $r.\Phi^-[3] \leftarrow 0$. L'insertion de la transaction $t_1 = AB$, suivant la relation d'ordre total $>_{\mathcal{I}}^{freq}$, ajoute deux noeuds au trie. Etant donné que la racine n'a pas encore de fils, alors la fonction *SearchChild*(A, r) renvoie "null" (ligne 9). Ce qui fait que, juste après la racine, un noeud portant le label A est créé à l'aide de la fonction *CreateNode*, $c \leftarrow$ *CreateNode*(A) (ligne 11) est ajouté comme premier fils de la racine à l'aide de la fonction *AddChild*(c, r) (ligne 12). Dans ce noeud, on a $c.\Phi^+[1] \leftarrow \binom{||AB||-1}{0} = 1$, $c.\Phi^-[1] \leftarrow \binom{||AB||-1}{1} = 1$, $c.\Phi^+[2] \leftarrow \binom{||AB||-1}{2-1} = 1$ et les autres poids sont nuls (lignes 13 à 15). On se place au noeud c en y pointant la variable η, $\eta \leftarrow c$ (ligne 16). Le noeud η n'admet pas de fils portant le prochain item de la transaction car *SearchChild*$(B, \eta) = null$, donc on crée le noeud $c \leftarrow$ *CreateNode*(B) (ligne 11) puis on l'ajoute aux fils de η avec la fonction *AddChild*(c, η) (ligne 12). Dans ce noeud, on a $c.\Phi^+[1] \leftarrow \binom{||AB||-2}{1-1} = 1$ et les autres poids sont nuls.
2. *insert*(t_2) : On ajoute les nombres d'occurrences de norme égale à $\ell \in [1..3]$ aux poids de la racine du trie τ (lignes 5 et 6) : $r.\Phi^-[1] \leftarrow r.\Phi^-[1] + \binom{||AC||}{1} = 4$, $r.\Phi^-[2] \leftarrow r.\Phi^-[2] + \binom{||AC||}{2} = 2$ et $r.\Phi^-[3] \leftarrow r.\Phi^-[3] + \binom{||AC||}{3} = 0$. Ensuite, on a *SearchChild*$(A, r) \neq null$ (ligne 9), alors les poids du noeud c sont mis à jour comme suit : $c.\Phi^+[1] \leftarrow c.\Phi^+[1] + \binom{||AC||-1}{1-1} = 1 + 1 = 2$ et $c.\Phi^-[1] \leftarrow c.\Phi^-[1] + \binom{||AC||-1}{1} = 1 + 1 = 2$ (lignes 13 et 15). Pour le prochain et dernier item de t_2, on a $\eta \leftarrow c$ (ligne 16) et *SearchChild*$(C, \eta) = null$ (ligne 9), donc on crée le noeud fils du noeud η avec le label C, $c \leftarrow$ *CreateNode*(C) (ligne 11). On ajoute le noeud c aux fils de η, *AddChild*(c, η) (ligne 12), puis on calcule ses poids (lignes 13 à 15).
3. *insert*(t_3) : Après avoir ajouté le nombre d'occurrences de motifs de norme égale à $\ell \in [1..3]$ de la transaction t_3 aux poids de la racine du trie τ (lignes 5 et 6), nous notons que la racine n'a pas un fils contenant le label B car *SearchChild*$(B, r) = null$ (ligne 9). Alors on crée le noeud $c =$ *CreateNode*(B) (ligne 11) puis on l'ajoute aux fils de la racine *AddChild*(c, r) (ligne 12). Enfin, on calcule les poids de c (lignes 13 à 15). En posant $\eta \leftarrow c$ (ligne 16), on a *SearchChild*$(C, \eta) = null$ (ligne 9), alors on crée le noeud ayant pour label l'item C, $c \leftarrow$ *CreateNode*(C) (ligne 11), et on l'ajoute aux fils de η avec la fonction *AddChild*(c, η) (ligne 12). Pour terminer, on calcule les poids du noeud c ($c.\Phi^+[1] \leftarrow 1$ et les autres poids sont égaux à 0) (lignes 13 à 15).
4. *insert*(t_4) : Après avoir ajouté les nombres d'occurrences de norme égale à $\ell \in [1..3]$ aux poids de la racine du trie τ (lignes 5 et 6), on note que *SearchChild*$(A, r) \neq null$ (ligne 9), alors on met à jour les poids du noeud c d'identifiant $\epsilon \rightarrow A$ (lignes 13 à 15) : d'une part $c.\Phi^+[1] \leftarrow c.\Phi^+[1] + \binom{4-1}{1-1} = 2 + 1 = 3$, $c.\Phi^+[2] \leftarrow c.\Phi^+[2] + \binom{4-1}{2-1} = 2 + 3 = 5$ et $c.\Phi^+[3] \leftarrow c.\Phi^+[3] + \binom{4-1}{3-1} = 0 + 3 = 3$, d'autre part $c.\Phi^-[1] \leftarrow c.\Phi^-[1] + \binom{4-1}{1} = 2 + 3 = 5$, $c.\Phi^-[2] \leftarrow c.\Phi^-[2] + \binom{4-1}{2} = 0 + 3 = 3$ et $c.\Phi^-[3] \leftarrow$

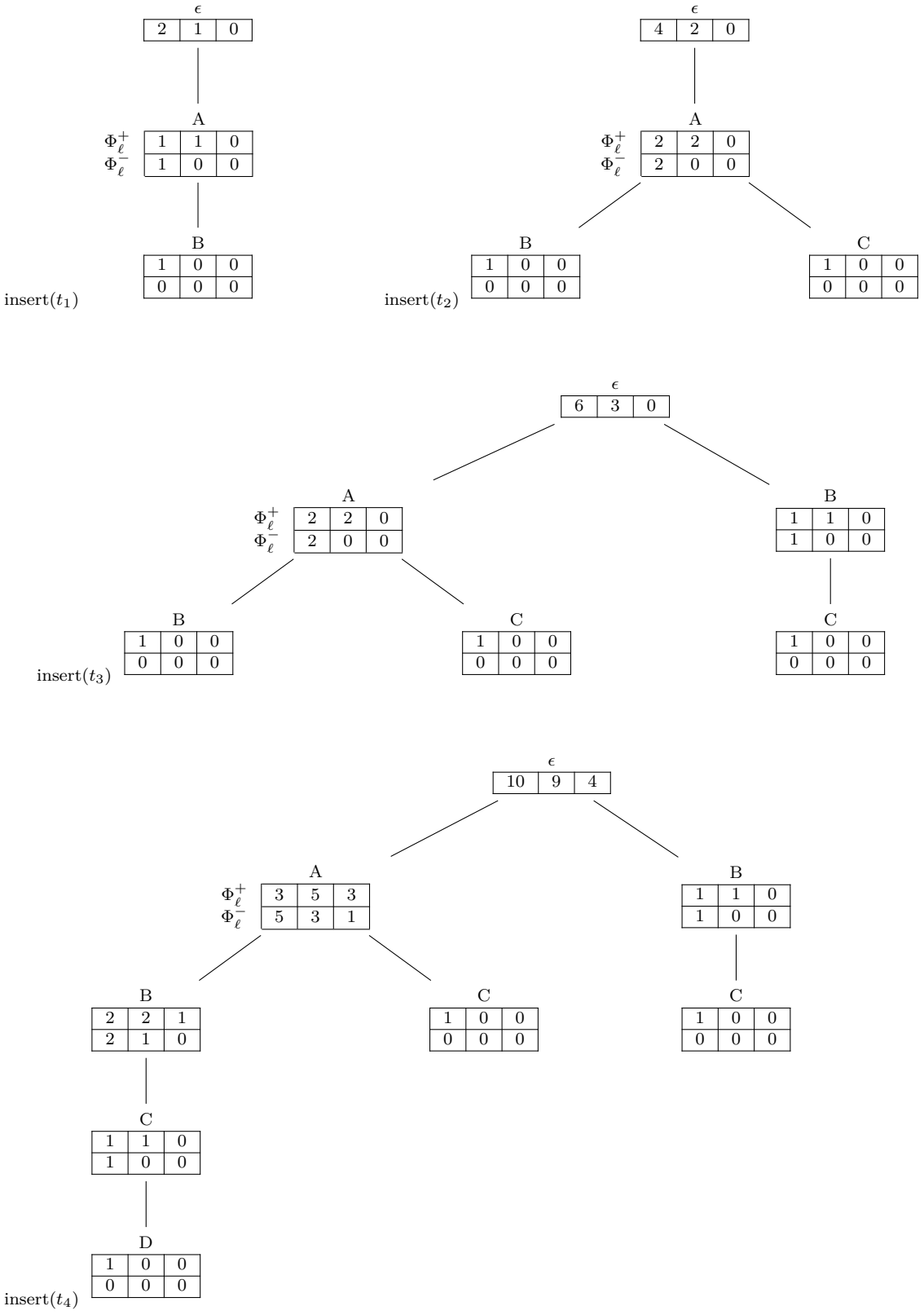


FIGURE 7.4 – Etapes de construction du trie d'occurrences de motifs de $\mathcal{T}$ avec $\ell \in [1..3]$

$c.\Phi^{-}[3] + \binom{4-1}{3} = 0 + 1 = 1$. Ensuite, on calcule les poids du noeud d'identifiant $\epsilon \rightarrow A \rightarrow B$ en suivant le même raisonnement appliqué au noeud précédent. A l'aide des fonctions *CreateNode* (ligne 11) et *AddChild* (ligne 12), on crée le noeud d'identifiant $\epsilon \rightarrow A \rightarrow B \rightarrow C$ et celui d'identifiant $\epsilon \rightarrow A \rightarrow B \rightarrow C \rightarrow D$ qui n'existent pas encore dans le trie d'après la fonction *SearchChild* (ligne 9). Enfin, on ajoute ces derniers noeuds à leur place tout en calculant leurs poids respectifs (ligne 13 à 15).

5. Pour terminer, on renvoie le trie τ obtenu (ligne 16).

Au final, nous avons au total 10 occurrences de motifs ayant une norme égale à 1, 9 occurrences de motifs ont une norme égale à 2 et 4 occurrences de motifs ont une norme égale à 3. Tel que nous avons construit le trie d'occurrences de motifs, si on garde les contraintes de normes minimale et maximale déjà utilisées lors de la construction, nous verrons dans la section suivante que l'utilisateur peut appliquer toute mesure d'intérêt fondée sur la norme $m \in \mathcal{M}$ sans pour autant reconstruire le trie.

7.3 Tirage d'un motif selon une utilité fondée sur la norme

Cette section commence par introduire quelques notions de base relatives au trie d'occurrences de motifs afin de mieux présenter notre approche. Ensuite, elle présente l'algorithme TPSampling pour le tirage exact d'un motif proportionnellement à une mesure d'intérêt choisie par l'utilisateur.

7.3.1 Approche du tirage

Pour tirer un motif de norme ℓ proportionnellement à une utilité fondée sur la norme multipliée par sa fréquence dans la base de données, nous pouvons tirer uniformément une occurrence de motif parmi l'ensemble des occurrences de motifs de norme ℓ. Pour ce faire, nous avons besoin en premier lieu de tirer un entier $\ell \in [\mu..M]$ proportionnellement à $\Phi_\ell(\epsilon, \mathcal{T}) \times f_u(\ell)$ avec u une utilité fondée sur la norme. En second lieu, nous tirons uniformément une occurrence de motif de norme ℓ parmi l'ensemble des occurrences de motifs de norme ℓ de la base de données, mais directement à partir du trie.

Le tirage d'une occurrence de motif de norme donnée n'est pas trivial car on ne sait pas a priori où est-ce qu'elle se trouve dans le trie. Cela veut dire qu'il faut parcourir intelligemment les noeuds du trie pour trouver l'occurrence de motif cherchée. Nous remarquons qu'il existe déjà des systèmes de numérotation des noeuds d'un arbre (en ordre préfixe ou postfixe), mais dans notre cas, l'objectif est de numéroter non pas les noeuds du trie, mais les occurrences de motifs représentées par le trie. L'intuition de la numérotation que nous avons utilisée peut être caractérisée comme suit :

— c'est une numérotation définie de manière récursive, de type postfixe (en profondeur et de gauche à droite). Au niveau des occurrences représentées dans un sous-trie, on les numérote de gauche à droite, les fils d'une racine d'un sous-trie étant ordonnés,

— au niveau d'un sous-trie, de racine r identifiée par un préfixe P, on donne un rang plus faible aux occurrences ne contenant pas le label de cette racine, qu'aux occurrences

le contenant.

Définition 34 (Rang d'une occurrence de norme donnée dans un trie)**.** *Si φ_j est une occurrence de norme ℓ d'une base de données $\mathcal{T}$ et τ un trie construit à partir de $\mathcal{T}$, on note $rank_\ell(\varphi_j, \tau) = rank_\ell(\varphi_j, \tau_\epsilon)$ le rang de cette occurrence par rapport au trie τ. Ce rang peut être défini de manière récursive comme suit :*

- *Si $\varphi_j \in \phi_\ell^-(P, \mathcal{T})$, et plus précisément si $\varphi_j \in \phi_\ell(P \circ e_i, \mathcal{T})$ où $e_i = \tau_P.fils[i]$, alors $rank_\ell(\varphi_j, \tau_P) = \sum_{k=1}^{i-1} \Phi_\ell(P \circ e_k, \tau) + rank_\ell(\varphi_j, \tau_{P \circ e_i})$.*
- *Si $\varphi_j \in \phi_\ell^+(P, \mathcal{T})$, alors $\varphi_j = \widetilde{P} \circ \varphi'_j$. Et dans ce cas, si $\varphi'_j \in \phi_{\ell-1}(P \circ e_i, \mathcal{T})$ où $e_i = \tau_P.fils[i]$, alors $rank_\ell(\varphi_j, \tau_P) = \Phi_\ell^-(P, \tau) + \sum_{k=1}^{i-1} \Phi_{\ell-1}(P \circ e_k, \tau) + rank_{\ell-1}(\varphi'_j, \tau_{P \circ e_i})$.*
- *Enfin, si φ_j est une occurrence de norme 1, on définit $rank_1(\varphi_j, \tau_P)$ par : $rank_1(\varphi_j, \tau_P) = rank_{>_\mathcal{I}}(\varphi_j, \phi_1(P, \mathcal{T}))$ où l'ordre $>_\mathcal{I}$ défini sur les items est étendu aux occurrences de norme 1 comme suit : $(i, x) >_\mathcal{I} (j, y)$ si $y >_\mathcal{I} x$ ou $x = y$ et $i > j$.*

Exemple 35. *En considérant seulement les motifs de norme égale à 2 et la relation d'ordre total $>_\mathcal{I}^{freq}$, la liste du tableau 7.2 donne le rang de chaque occurrence de motif dans le trie de la figure 7.4 :*

Table 7.2 – Rangs des occurrences de motifs du trie de la figure 7.4 de norme 2

φ_i	CD_4	BD_4	BC_4	AD_4	AC_4	AB_4	AB_1	AC_2	BC_3
$rank_2$	1	2	3	4	5	6	7	8	9

Nous avons $\phi_2(\epsilon, \mathcal{T}) = \{CD_4, BD_4, BC_4, AD_4, AC_4, AB_4, AB_1, AC_2, BC_3\}$. Nous savons que $CD_4 \in \phi_2^-(A, \mathcal{T})$, alors $rank_2(CD_4, \tau_\epsilon) = rank_2(CD_4, \tau_A) = rank_2(CD_4, \tau_{AB}) = rank_2(CD_4, \tau_{ABC}) = \Phi_2^-(ABC, \tau) + rank_1(D, \tau_{ABC}) = 0 + rank_{>_\mathcal{I}^{freq}}(D, \{D\}) = 1$.

Calculons $rank_2(AB_4, \tau_\epsilon)$. On sait que $AB_4 \in \phi_2^+(A, \mathcal{T})$, alors $rank_2(AB_4, \tau_\epsilon) = \Phi_2^-(A, \tau) + 0 + rank_1(B_4, \tau_{AB}) = 3 + rank_{>_\mathcal{I}^{freq}}(B_4, \{B_1, B_4, C_4, D_4\})$. Or $(4, D) >_\mathcal{I}^{freq} (4, C) >_\mathcal{I}^{freq} (4, B) >_\mathcal{I}^{freq} (1, B)$, alors $rank_{>_\mathcal{I}^{freq}}(B_4, \{B_1, B_4, C_4, D_4\}) = 3$, donc $rank_2(AB_4, \tau_\epsilon) = 3 + 3 = 6$.

Dans cette liste d'occurrences de motifs, pour tirer un motif proportionnellement à sa fréquence, il suffit de tirer uniformément un rang entre 1 et le rang le plus élevé, et retourner l'occurrence qui possède ce rang. Par exemple, nous avons 2 occurrences de motifs de AB de rangs 6 et 7. Ce qui veut dire que la probabilité de tirer le motif AB parmi l'ensemble des motifs de norme exactement égale à 2 est $\frac{2}{9}$, donc proportionnelle à sa fréquence. Si l'utilisateur choisit une mesure d'intérêt égale à la fréquence, alors d'après le trie d'occurrences de motifs, on tire $\ell = 2$ avec une probabilité de $\frac{9}{23}$. De ce fait, nous pouvons remarquer que le motif AB est tiré dans le trie avec une probabilité égale à $\frac{9}{23} \times \frac{2}{9} = \frac{2}{23}$, donc proportionnellement à sa fréquence dans la base de données.

7.3.2 TPSampling : échantillonnage en sortie de motifs à base de trie

L'algorithme 8 prend en entrée un trie d'occurrences de motifs τ, une utilité fondée sur la norme $u \in \mathcal{U}$ et des contraintes de normes minimale μ et maximale M. Il retourne

un motif φ tiré proportionnellement à son intérêt dans la base de données correspondant au trie. Le tirage d'un motif se fait un deux-sous étapes :

— *Tirage d'une norme ℓ entre μ et M* : A la ligne 4, on tire un entier ℓ compris entre μ et M proportionnellement au nombre d'occurrences de motifs de norme ℓ, i.e. $\tau.\Phi^-[\ell]$ multiplié par l'utilité d'un motif de norme ℓ, $f_u(\ell)$.

— *Tirage uniforme d'une occurrence de motif de norme ℓ* : Pour débuter le tirage d'une occurrence de motif de norme ℓ, on tire uniformément un rang x dans l'intervalle $[1..\tau.\Phi^-[\ell]]$ (ligne 5). Pour chercher l'occurrence de motif correspondant à x, on fait un parcours en profondeur et de gauche à droite dans le trie en cherchant les noeuds qui vérifient le système d'inéquations à la ligne 7 qui se base sur la définition 34. Ce système d'inéquations permet de retrouver le rang de l'occurrence de motif dans le trie ayant pour racine τ. A chaque fois qu'on rencontre un noeud vérifiant le système d'inéquations, on teste si l'item qu'il contient est candidat du motif à retourner en sortie (ligne 9), et on l'ajoute au motif au cas échéant (ligne 10). A la ligne 13, on considère le sous-trie dont le noeud vérifiant le système d'inéquations est la racine. Ainsi, le nouveau rang à visiter est celui obtenu en soustrayant de l'ancienne valeur de x la somme des poids des $i-1$ premiers fils du noeud courant, père de η_i, (ligne 8) et l'apport négatif au noeud η_i (ligne 11). On va ensuite chercher les $\ell-1$ items restant du motif à retourner dans le sous-trie de racine η_i. Le processus est itéré jusqu'à ce que la valeur courante de ℓ soit égale à 0. L'ensemble des items sélectionnés au différents noeuds visités forment le motif à retourner à la ligne 14.

Algorithm 8 TPSampling

1: **Input :** Un trie τ d'une base de données $\mathcal{T}$, une utilité fondée sur la norme $u \in \mathcal{U}$ et des contraintes de normes minimale μ et maximale M
2: **Output :** Un motif φ tiré proportionnellement à son intérêt $\varphi \sim f_u(||\varphi||) \times freq(\varphi, \mathcal{T})$
3: $\varphi \leftarrow \emptyset$
4: Tirer une norme ℓ proportionnellement à $\tau.\Phi^-[\ell] \times f_u(\ell)$ où $\ell \in [\mu..M]$
5: Tirer uniformément un rang x dans l'intervalle $[1..\tau.\Phi^-[\ell]]$
6: **while** $(\ell > 0)$ **do**
7: Trouver le i-ième fils $\eta_i \in \tau.fils$ tel que :

$$\sum_{1 \leq k < i} \Big(\tau.fils[k].\Phi^+[\ell] + \tau.fils[k].\Phi^-[\ell]\Big) < x \leq \sum_{1 \leq k \leq i} \Big(\tau.fils[k].\Phi^+[\ell] + \tau.fils[k].\Phi^-[\ell]\Big)$$

8: $x \leftarrow x - \sum_{1 \leq k < i} \left(\tau.fils[k].\Phi^+[\ell] + \tau.fils[k].\Phi^-[\ell]\right)$
9: **if** $(x > \eta_i.\Phi^-[\ell])$ **then** ▷ Vérifier si le label du noeud courant est un item du motif
10: $\varphi \leftarrow \varphi \cup \eta_i.label$ ▷ Ajouter le label du noeud au motif à retourner
11: $x \leftarrow x - \eta_i.\Phi^-[\ell]$
12: $\ell \leftarrow \ell - 1$
13: $\tau \leftarrow \eta_i$
14: **return** φ ▷ Retourner l'union des items récupérés

L'exemple 36 montre un cas d'exécution de l'algorithme 8 pour le tirage exact d'un motif proportionnellement à la fréquence.

Exemple 36. *Ayant tiré uniformément un rang $x = 5$ d'une occurrence de motif parmi les motifs de norme 2, nous allons montrer à la figure 7.5, avec l'algorithme 8, comment retrouver l'occurrence correspondante dans le trie d'occurrences de motifs de la figure 7.4.*

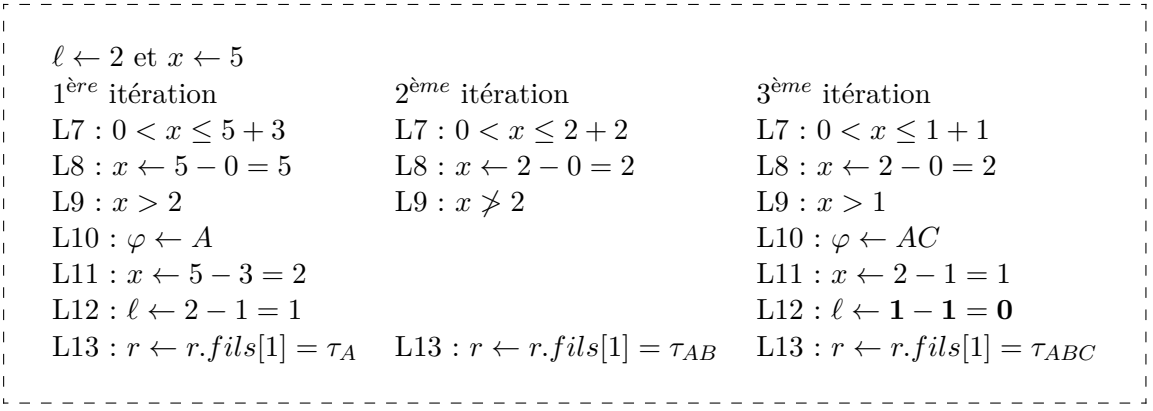


FIGURE 7.5 – Exemple d'un tirage d'une occurrence de motif sous contrainte de norme retournant l'occurrence AC_4

Pour retrouver la 5-ème occurrence de motif parmi l'ensemble des occurrences de motifs de norme égale à 2, nous avons fait 3 itérations (voir figure 7.5). Lors de la première itération, nous sélectionnons le premier fils c de la racine du trie τ d'identifiant $P = A$ car $0 < x \leq (c.\Phi^-[2] + c.\Phi^+[2] = 3 + 5)$ (ligne 7), et aussi on a $x > c.\Phi^-[2]$ à la ligne 9. De ce fait, l'item A fait partie de l'occurrence de motif à retourner en sortie et le contenu courant de l'occurrence de motif est égal à $\varphi = A$. La valeur de ℓ devient 1 car on va maintenant regarder sur les motifs de norme $\ell - 1 = 1$ en considérant le sous-trie τ_A puis retourner la $5 - 3 = 2$ ème occurrence de motif de norme 1 dans τ_A. A la deuxième itération, nous sélectionnons le premier fils c de la racine du sous-trie τ_A d'identifiant $P = AB$ car $0 < x \leq (c.\Phi^-[1] + c.\Phi^+[1] = 2 + 2)$ (ligne 7), et aussi on a $x < c.\Phi^-[1]$ à ligne 9 et donc l'item B ne fait pas partie de l'occurrence de motif à retourner, la valeur de ℓ est toujours la même. Lors de la 3-ème itération, nous sélectionnons le premier fils c de la racine du sous-trie τ_{AB} d'identifiant $P = ABC$ car $0 < x \leq (c.\Phi^-[1] + c.\Phi^+[1] = 1 + 1)$ (ligne 7). On note ensuite que $x > c.\Phi^-[1]$ et donc l'item C fait partie de l'occurrence de motif à retourner. Le contenu courant de l'occurrence de motif est égal à $\varphi = AC$. La valeur de ℓ est maintenant égale à 0 (ligne 12) ce qui veut dire que TPSAMPLING *s'arrête puis retourne le motif $\varphi = AC$. On peut aussi remarquer que la 5-ème occurrence de motif de norme 2 est bien AC d'après le tableau 7.2.*

7.4 Analyse théorique de la méthode

Cette section fait une étude théorique de notre approche pour l'échantillonnage à partir d'un trie d'occurrences de motifs. En premier lieu, elle nous donne un aperçu sur la correction du tirage d'un motif. En second lieu, elle montre la complexité en espace mémoire du trie d'occurrences de motifs. En troisième et dernier lieu, elle présente les complexités temporelles de nos deux algorithmes pour la construction d'un trie d'occurrences de motifs

et pour le tirage d'un motif à partir d'un trie d'occurrences de motif par TPSampling.

7.4.1 Correction

La propriété 10 montre que notre méthode d'échantillonnage TPSampling fait un tirage exact d'un motif.

Propriété 10. *Soit τ un trie d'occurrences de motifs d'une base de données transactionnelles et u une utilité fondée sur la norme, l'algorithme 8 tire un motif φ suivant une probabilité proportionnelle à sa fréquence pondérée par son utilité.*

Preuve. Soit φ_i une occurrence d'un motif φ de norme égale à ℓ. La probabilité que l'occurrence de motif φ_i soit retournée est égale à $\pi(\ell) \times \pi(\varphi_i/\ell)$. Soit $Z = \sum_\ell f_u(\ell) \times \tau.\Phi^-[l]$. On sait d'après la ligne 4 que $\pi(\ell) = f_u(\ell) \times \tau.\Phi^-[l]/Z$. D'après la ligne 5, le rang d'une occurrence de motif de norme ℓ est tiré uniformément parmi les rangs des occurrences de motifs de même norme. Les lignes 9 à 12 exploitent la définition 34 afin d'accéder au motif d'un rang donné. Donc d'après les lignes 6 à 13, nous avons $\pi(\varphi_i/\ell) = 1/\tau.\Phi^-[l]$. Il en résulte que $\pi(\varphi_i) = \frac{f_u(\ell)\times\tau.\Phi^-[l]}{Z} \times \frac{1}{\tau.\Phi^-[l]}$. Donc $\pi(\varphi_i) = \frac{f_u(\ell)}{Z}$. Ce qui nous donne finalement $\pi(\mathcal{S}amp\ell e_1(\mathcal{L}_\mathcal{I}, \tau, freq \times u) = \varphi) = \frac{f_u(\ell)}{Z} \times freq(\varphi, \mathcal{T})$. D'où le résultat. □

7.4.2 Complexité en espace mémoire

La taille mémoire occupée par le trie d'occurrences de motifs n'est pas négligeable, surtout avec les gros jeux de données. Cependant, elle peut être optimisée grâce à un prétraitement pour réduire le nombre de noeuds. Par exemple, en tenant compte que les items des transaction sont ordonnés suivant l'ordre décroissant de leurs fréquences, le nombre de noeuds dans le trie est largement inférieur à la borne supérieure qui est de l'ordre de $2^{|\mathcal{I}|}$.

La taille en mémoire d'un trie d'occurrences de motifs dépend aussi de l'information stockée dans les noeuds. Dans notre cas, plus la contrainte de norme maximale est élevée, plus les tableaux sont grands et plus la taille mémoire est élevée. Cela veut dire que si le nombre de noeuds dans le trie d'occurrences de motifs est en $O(Z)$, μ et M les contraintes de normes minimale et maximale respectivement, alors la taille en mémoire du trie est en $O(Z \times 2 \times (M - \mu))$. Heureusement, la contrainte de norme maximale doit généralement être petite pour éviter le phénomène de la longue traîne (voir section 4.4.3). Il est aussi important de noter que, pour avoir une très bonne complexité en espace mémoire dans la pratique, nous ne matérialisons pas les colonnes des tableaux qui ne contiennent que des valeurs nulles.

Exemple 37. *La figure 7.6 est une version du trie d'occurrences de motifs de la base de données $\mathcal{T}$ que nous avions représentée à la figure 7.4. Dans cette exemple, nous avons omis les colonnes qui ne contiennent que des zéros (0).*

Dans cet exemple, si maintenant nous prenons une contrainte de norme maximale M égale à 4, alors seuls le tableau du noeud d'identifiant A et celui de la racine seront impactés. Nous aurons notamment une colonne de plus au noeud d'identifiant A contenant

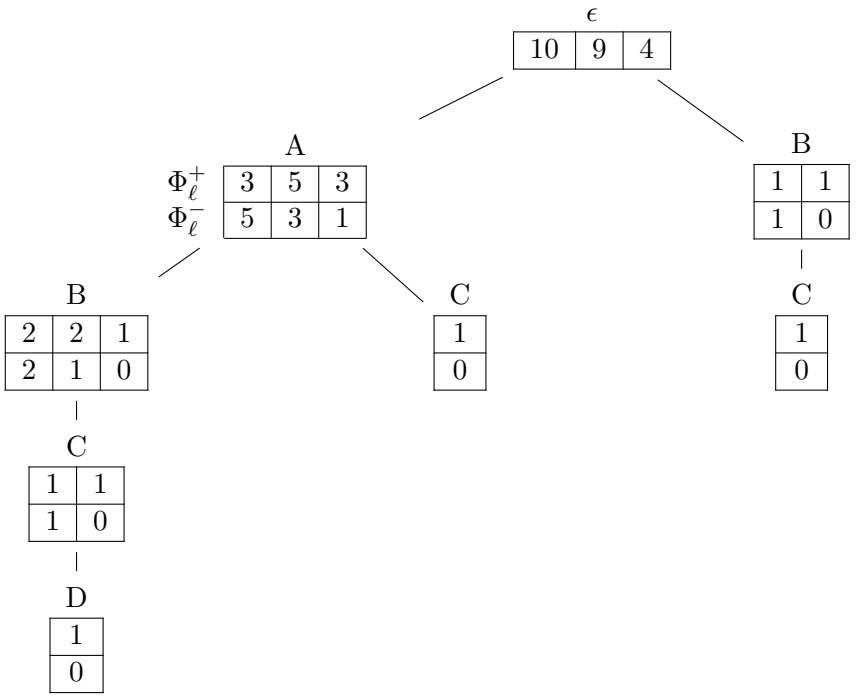


FIGURE 7.6 – Représentation en pratique du trie d'occurrences de motifs τ de la base de données $\mathcal{T}$

1 *à l'indice* 1 *(pour comptabiliser le motif* $ABCD$ *de taille* 4*) et* 0 *à l'indice* 2*. A la racine, nous aurons une case de plus pour compter le motif de norme* 4*. Si maintenant, on a* $M = 5$*, alors le trie reste inchangé car aucun motif de norme* 5 *n'apparait dans aucune transaction de la base de données* $\mathcal{T}$*.*

Comme on peut le voir, cette optimisation peut permettre d'avoir un gain significatif sur la taille en mémoire d'un trie d'occurrences de motifs par rapport à la représentation théorique où toutes les colonnes sont matérialisées. Tel que le trie est représenté dans l'exemple 37, nous avons l'intuition qu'en pratique, la taille en mémoire d'un trie d'occurrences de motifs sera moins impactée par la contrainte de norme maximale.

7.4.3 Complexité temporelle

La complexité temporelle de notre méthode peut être divisée en deux temps : le temps de prétraitement ou de la construction du trie d'occurrences de motifs et le temps de tirage d'une occurrence de motif.

— *Le temps de prétraitement* : c'est la phase la plus coûteuse de l'algorithme 8. Une première passe sur la base de données est nécessaire pour récupérer les items de la base de données $\mathcal{T}$ et calculer leurs fréquences en $O(||\mathcal{T}||)$ où $||\mathcal{T}||$ est la somme des longueurs des transactions de la base de données $\mathcal{T}$. Les items précédemment récupérés sont ordonnés suivant la relation d'ordre total choisie $>_\mathcal{I}$ en $O(|\mathcal{I}| \times \log(|\mathcal{I}|))$.

Ensuite, avant d'ajouter une transaction dans le trie, on range ses items suivant la relation d'ordre total $>_{\mathcal{I}}$ en $O(T_{\max} \times \log(T_{\max}))$ où $T_{\max}$ est la longueur maximale des transactions de la base de données $\mathcal{T}$. Enfin, si Z est le nombre total de noeuds dans le trie suivant la relation d'ordre total $>_{\mathcal{I}}$, μ et M les contraintes de normes minimale et maximale respectivement, alors la pondération des noeuds se fait en $O(Z \times 2 \times (M-m))$. Ainsi, la complexité totale pour la construction du trie d'occurrences de motifs de la base de données transactionnelles $\mathcal{T}$ construite sur l'ensemble des littéraux $\mathcal{I}$ est en $O(||\mathcal{T}|| + |\mathcal{I}| \times \log(|\mathcal{I}|) + |\mathcal{T}| \times T_{\max} \times \log(T_{\max}) + Z \times (M-m))$. Toutefois, rappelons que ce calcul pour le prétraitement d'une base de données n'est fait qu'une seule fois.

— *Le temps de tirage d'une occurrence de motif* : notons par d le degré (nombre de fils) d'un noeud du trie et par d_{max} le degré maximal du trie, $d_{\max} \leq |\mathcal{I}|$. A la ligne 7, TPSampling retrouve i-ième noeud en $O(\log(d_{max}))$. Ainsi, en parcourant en profondeur le trie d'occurrences de motifs, l'algorithme TPSampling tire une occurrence de motif en $O(T_{\max} \times \log(d_{max}))$. Donc un échantillon de k motifs est obtenu par TPSampling en $O(k \times T_{\max} \times \log(d_{max}))$. Cette complexité est comparable à celle de la méthode en deux étapes de l'algorithme 4 qui tire un échantillon de k motifs en $O(k \times T_{\max} \times \log(|\mathcal{T}|))$.

7.5 Expérimentations

Cette section expérimentale vise à évaluer l'efficacité de notre approche face aux gros jeux de données transactionnelles. La section 7.5.1 évalue la taille mémoire occupée par le trie d'occurrences de motifs en fonction de la contrainte de norme maximale et de la relation d'ordre sur les items $>_{\mathcal{I}} \in \{>_{\mathcal{I}}^{freq}, >_{\mathcal{I}}^{lexico}\}$. La section 7.5.2 étudie le temps de construction d'un trie d'occurrences de motifs suivant la contrainte de norme maximale sur différents jeux de données puis évalue la rapidité du tirage d'un motif par TPSampling. La section 7.5.3 montre le passage à l'échelle de notre méthode d'échantillonnage TPSampling sur de grosses bases de données.

Les expériences ont été conduites avec 7 bases de données de l'UCI avec les versions prétraitées [1] pour `Adult`, `Connect`, `Mushroom`, `Pumsb` et `Chess`, ou venant du site SPMF [2] pour `Susy` et `USCensus`. Le tableau 7.3 détaille d'une part les caractéristiques des benchmarks suivant le nombre d'items, de transactions, la taille maximale et moyenne des transactions. D'autre part, il présente le nombre de noeuds dans le trie correspondant à chaque base de données et suivant l'ordre total posé sur les items. Pour l'échantillonnage avec contrainte de norme, la valeur de la contrainte de norme minimale est fixée à $\mu = 1$ tout au long des expérimentations. Le prototype de notre méthode est implémenté en Python version 3 et toutes les expérimentations sont réalisées sur un PC de 2.71 GHz 2 Core CPU avec une RAM de 12 Go. Tous les jeux de données expérimentaux utilisés, ainsi que le code source, sont disponibles sur https://github.com/TPSampling/TPSampling.

1. http://cgi.csc.liv.ac.uk/~frans/KDD/Software/LUCS-KDD-DN/DataSets
2. http://www.philippe-fournier-viger.com/spmf

TABLE 7.3 – Caractéristiques des jeux de données

$\mathcal{T}$	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert t\rvert_{\min}$	$\lvert t\rvert_{\max}$	$\lvert t\rvert_{moy}$	Nb noeuds dans le trie $>_{\mathcal{I}}^{freq}$	Nb noeuds dans le trie $>_{\mathcal{I}}^{lexico}$	Gain (en %) $\frac{Nb_{lexico}-Nb_{freq}}{Nb_{lexico}}$
Adult	97	48,842	12	15	14.88	26,365	61,131	56.87
Connect	129	67,557	43	43	43.00	359,291	1,014,837	64.60
Mushroom	90	8,124	22	23	22.69	20,953	33,738	37.89
Pumsb	2,113	49,096	74	74	74.00	1,126,154	2,629,640	57.17
Chess	58	28,056	7	7	7.00	65,465	75,675	13.49
USCensus	396	1,000 K	25	68	68.00	312,808	607,611	48.52
Susy	190	5,000 K	19	19	19.00	10,424,240	12,630,372	17.47

7.5.1 Coût de stockage en mémoire du trie d'occurrences de motifs

Le coût de stockage d'un trie d'occurrences de motifs d'une base de données dépend à la fois de la relation d'ordre total $>_{\mathcal{I}}$ et de la contrainte de norme maximale.

Comme nous pouvons le noter dans les trois dernières colonnes du tableau 7.3, le nombre de noeuds dépend de l'ordre dans lequel les items sont insérés dans le trie. Globalement, nous avons remarqué que le nombre de noeuds est beaucoup plus petit avec la relation $>_{\mathcal{I}}^{freq}$ qu'avec la relation $>_{\mathcal{I}}^{lexico}$ d'après le gain obtenu à la dernière colonne. Par exemple, avec la base de données Connect, le nombre de noeuds du trie construit avec la relation d'ordre $>_{\mathcal{I}}^{freq}$ est 64.6% plus petit que le nombre de noeuds du trie construit avec la relation d'ordre total $>_{\mathcal{I}}^{lexico}$. Cette caractéristique est très importante dans notre proposition car chaque noeud doit contenir deux tableaux dont leur taille dépend de la contrainte de norme maximale. Ainsi, plus le nombre de noeud est petit plus le coût de stockage est faible. Avec le gain obtenu à la dernière colonne du tableau 7.3, on peut déjà remarquer que la relation d'ordre total joue un rôle très important sur l'optimisation du coût de stockage en mémoire surtout avec les gros jeux de données.

La figure 7.7 montre l'évolution de la taille en mémoire des tries de différentes bases de données suivant la contrainte de norme maximale $M \in [1..10]$ selon la relation d'ordre choisie. Nous prenons comme baseline le coût de stockage des représentations tabulaires des bases de données pondérées suivant les contraintes de norme maximale de la méthode en deux étapes de l'algorithme 4 ("Two-Step").

Nous pouvons tout d'abord remarquer que la taille en mémoire du trie d'occurrences de motifs augmente légèrement suivant la contrainte de norme maximale. En effet, les tableaux qui permettent de comptabiliser les motifs par norme augmentent de taille suivant la contrainte de norme maximale. Cette augmentation est plus notoire dans le cas où on a considéré la relation d'ordre total $>_{\mathcal{I}}^{lexico}$ car le nombre de noeuds dans le trie construit avec la relation d'ordre total $>_{\mathcal{I}}^{lexico}$ est plus élevé que celui du trie construit avec la relation $>_{\mathcal{I}}^{freq}$. De ce fait, la taille en mémoire du trie est plus coûteuse avec la relation $>_{\mathcal{I}}^{lexico}$ qu'avec la relation $>_{\mathcal{I}}^{freq}$. Ainsi, dans le cas où la base de données est très volumineuse, par exemple avec Connect, Pumsb, USCensus et Susy, il est préférable d'insérer les items des instances de la base de données dans le trie suivant la relation d'ordre total $>_{\mathcal{I}}^{freq}$. On peut noter qu'avec cette dernière, la taille du trie d'occurrences de motifs en mémoire est

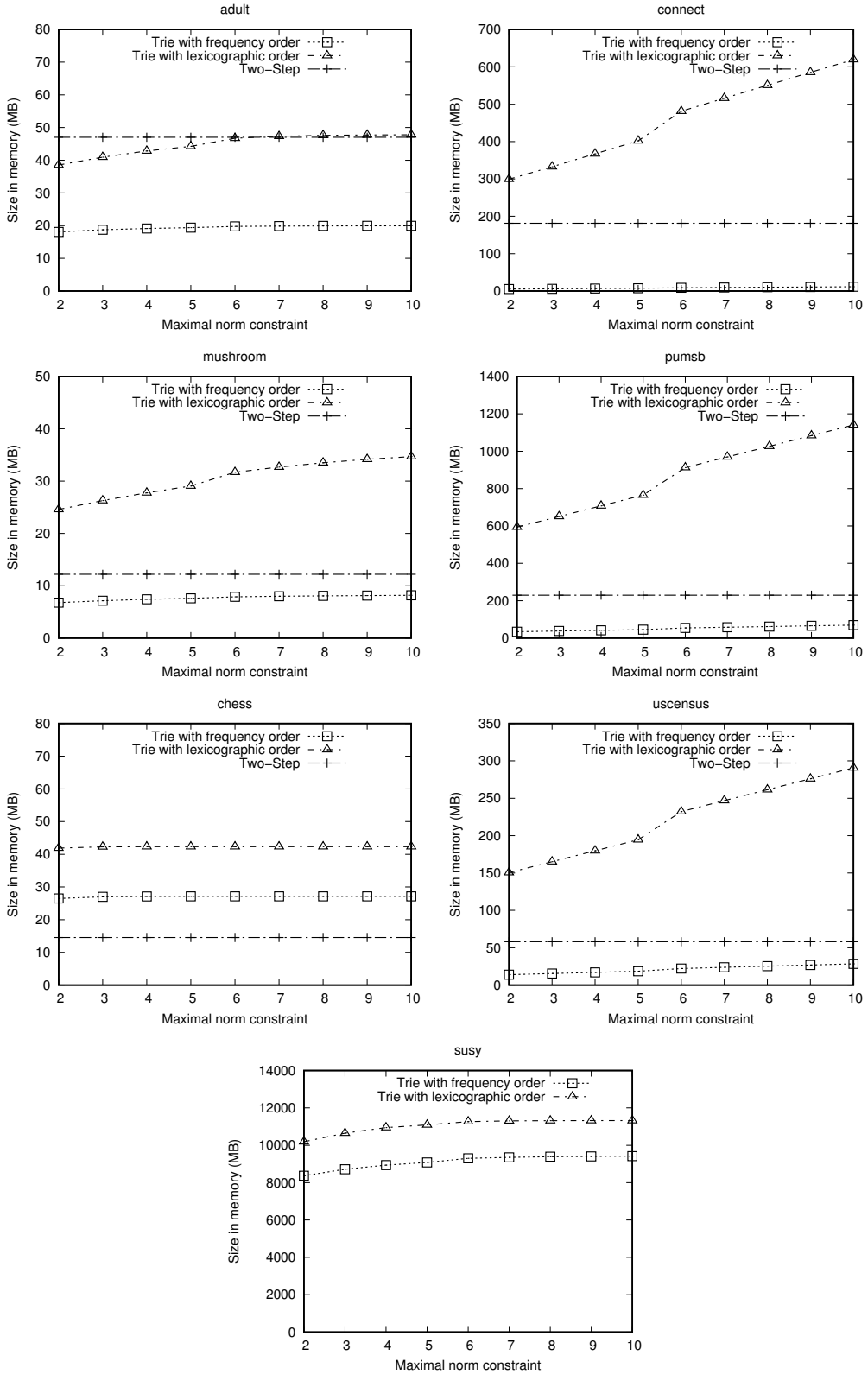


FIGURE 7.7 – Evolution en mémoire de la taille du trie suivant la contrainte de norme

presque constante.

Nous avons remarqué aussi que si la base de données est très grande `Connect` (67,557 instances ayant une norme moyenne de 43.0) et `Pumsb` (49,096 instances ayant une norme moyenne de 74.0), la taille en mémoire du trie d'occurrences de motifs construit selon la relation d'ordre $>_{\mathcal{I}}^{freq}$ est 200 fois plus petite que la taille en mémoire de la représentation de la base de données utilisée par l'approche Two-step, et en particulier l'algorithme 4 du chapitre 4. Nous notons aussi qu'avec la base de données `Chess`, la représentation de la base de données en mémoire de la méthode en deux étapes est moins coûteuse que celle du trie d'occurrences de motifs construit suivant la relation $>_{\mathcal{I}}^{freq}$. Cela est dû au fait que non seulement les longueurs des transactions de `Chess` sont très petites (7 items en moyenne), mais également qu'elle est de petite taille (28,056 transactions). L'implémentation de la méthode Two-Step génère une exception de type "Out of memory" avec la base de données `Susy`. Pour cette même base de données, la taille du trie d'occurrences de motifs atteint les 9 Go avec la relation d'ordre total $>_{\mathcal{I}}^{freq}$ contre 11 Go avec la relation d'ordre total $>_{\mathcal{I}}^{lexico}$.

7.5.2 Rapidité de l'approche

Cette section étudie la rapidité de notre méthode d'échantillonnage à base de trie appelée TPSampling en distinguant le temps de prétraitement et celui du tirage d'un motif. Nous allons commencer par évaluer la durée de prétraitement de chaque jeu de données du tableau 7.3 puis la comparer avec celle de la méthode en deux étapes de l'algorithme 4. Nous terminons par calculer la durée moyenne du tirage d'un motif par TPSampling dans les 6 premiers jeux de données suivant la contrainte de norme maximale et la mesure d'intérêt considérée.

Calcul du temps de prétraitement D'une façon intéressante, le temps de construction d'un trie d'occurrences de motifs est indépendant d'une quelconque mesure d'intérêt fondée sur la norme. Dans nos expérimentations, nous considérons les contraintes de normes maximales $M \in \{2, 6\}$. Ces valeurs nous paraissent suffisantes pour éviter de tirer des motifs de la longue traîne. Le tableau 7.4 présente les temps de prétraitement pour la construction des tries d'occurrences de motifs des bases de données et ceux de la pondération des bases de données avec l'algorithme 4 suivant la contrainte de norme maximale. Chaque expérience est répétée 10 fois afin d'avoir les durées moyennes de prétraitement et les écarts-types.

Comme nous pouvons le voir, la méthode en deux étapes est plus rapide que celle à base de trie pour la pondération d'une base de données. Cela peut s'expliquer par le fait qu'avec la méthode en deux étapes, le prétraitement se fait en une passe sur la base de données avec une complexité linéaire par rapport à la taille de la base de données. Au contraire, avec notre méthode de prétraitement pour les tries, il nous faut 2 passes dans le cas où nous voulons insérer les items suivant une relation d'ordre total $>_{\mathcal{I}}$. A cela s'ajoute le temps d'insertion des items de chaque transaction dans le trie d'occurrences de motifs suivant la relation d'ordre total choisie. Rappelons qu'avec la base de données `Susy`, une exception de type "Out of memory" a été lancée par l'implémentation de la méthode en deux étapes

TABLE 7.4 – Temps de prétraitement en seconde des méthodes suivant les mesures (A=Area, D=Decay, F=Freq) et la contrainte de norme maximale ($M \in \{2, 6\}$)

		Two-Step (Algorithme 4)		TPSAMPLING ($>_{\mathcal{I}}^{freq}$)		TPSAMPLING ($>_{\mathcal{I}}^{lexico}$)	
$\mathcal{T}$	m	M=2	M=6	M=2	M=6	M=2	M=6
`Adult`	A	0.14 ± 0.01	0.14 ± 0.01	2.23 ± 0.21	2.88 ± 0.69	2.83 ± 0.21	4.45 ± 0.21
	D	0.14 ± 0.01	0.14 ± 0.01				
	F	0.13 ± 0.01	0.14 ± 0.01				
`Connect`	A	0.44 ± 0.02	0.44 ± 0.04	9.98 ± 0.30	13.33 ± 3.45	15.57 ± 0.37	25.72 ± 0.69
	D	0.44 ± 0.03	0.44 ± 0.04				
	F	0.43 ± 0.04	0.45 ± 0.03				
`Mushroom`	A	0.03 ± 0.01	0.03 ± 0.01	0.59 ± 0.06	0.77 ± 0.20	0.73 ± 0.02	1.22 ± 0.05
	D	0.03 ± 0.01	0.03 ± 0.01				
	F	0.03 ± 0.01	0.03 ± 0.01				
`Pumsb`	A	0.54 ± 0.04	0.53 ± 0.03	18.66 ± 0.40	24.12 ± 5.62	78.89 ± 1.65	94.79 ± 3.24
	D	0.54 ± 0.04	0.54 ± 0.04				
	F	0.56 ± 0.04	0.53 ± 0.04				
`Chess`	A	0.05 ± 0.01	0.05 ± 0.01	0.99 ± 0.05	1.35 ± 0.09	1.13 ± 0.07	1.45 ± 0.10
	D	0.05 ± 0.01	0.05 ± 0.01				
	F	0.05 ± 0.01	0.05 ± 0.01				
`Susy`	A	–	–	347.99 ± 8.01	593.01 ± 44.24	532.01 ± 41.65	910.45 ± 65.47
	D	–	–				
	F	–	–				
`USCensus`	A	0.01 ± 0.00	0.01 ± 0.00	5.26 ± 1.85	7.39 ± 0.17	8.09 ± 0.32	12.27 ± 1.27
	D	0.01 ± 0.00	0.01 ± 0.00				
	F	0.01 ± 0.00	0.01 ± 0.00				

de l'algorithme 5 à la page 100 du fait de sa très grande taille qui ne tient pas en mémoire. Avec cette même base de données, la construction du trie d'occurrences de motifs suivant la relation d'ordre total $>_{\mathcal{I}}^{lexico}$ dure en moyenne 9 et 15 minutes avec les contraintes de normes maximales $M = 2$ et $M = 6$ respectivement. Nous avons vu qu'avec la grosse base de données `Susy`, le prétraitement pour la construction du trie d'occurrences de motifs suivant la relation d'ordre total $>_{\mathcal{I}}^{freq}$ a atteint 5 *minutes* et 9 *minutes* respectivement pour les contraintes de normes maximales $M = 2$ et $M = 6$. Dans tous les autres cas, la phase de prétraitement pour la construction des tries d'occurrences de motifs des jeux de données suivant la relation d'ordre total $>_{\mathcal{I}}^{freq}$ ne dure généralement que quelques secondes. Il est aussi intéressant de noter que nous ne faisons ce prétraitement qu'une seule fois.

Evaluation du temps de tirage d'un motif Maintenant, nous allons évaluer la rapidité du tirage d'un motif de notre approche sur les 6 premiers jeux de données du tableau 7.3. Précisément, nous allons calculer le temps de tirage moyen d'un motif suivant la contrainte de norme maximale appartenant à l'intervalle [1..10] et une décroissance exponentielle dont la valeur est fixée à $\alpha = 0.1$. La figure 7.8 montre l'évolution du temps moyen de tirage d'un motif par TPSAMPLING comparé à celui de la méthode en deux étapes présentée à la page 100. Chaque valeur moyenne est obtenue en répétant, pour chaque valeur de $M \in [1..10]$, 100 fois le tirage d'un motif dans chaque jeu de données. Les écart-types sont omis du fait qu'ils sont très faibles.

Comme nous pouvons le voir, avec la méthode TPSAMPLING, les temps de tirages sont inférieurs à 0.05 milliseconde sur les jeux de données `Adult`, `Chess` et `Mushroom` que nous avons utilisés avec une contrainte de norme maximale $M \in [1..10]$. Avec les

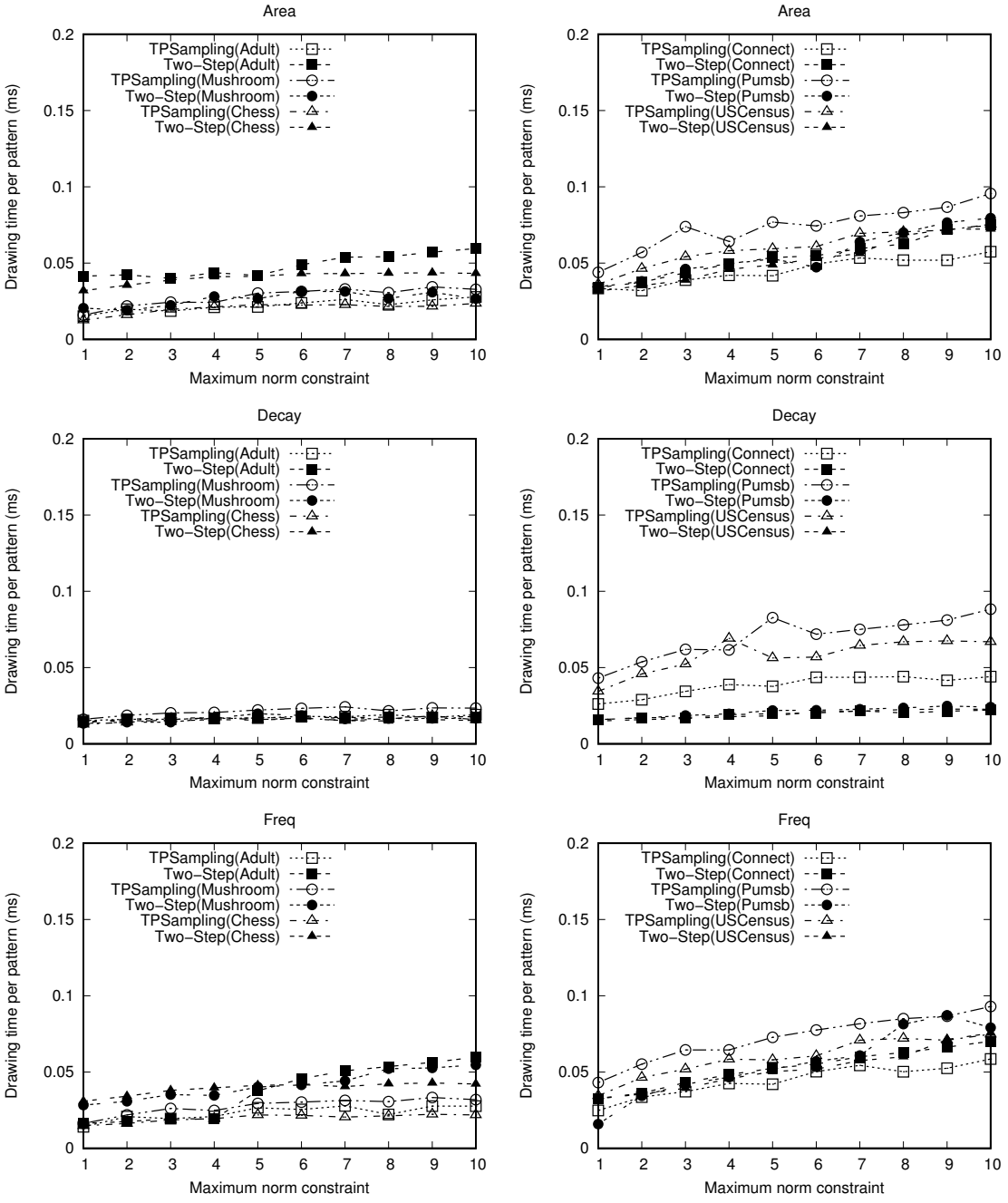


FIGURE 7.8 – Evolution de la durée moyenne du tirage d'un motif suivant la contrainte de norme maximale ($\alpha = 0.1$ pour la décroissance exponentielle)

bases de données de grande taille `Connect`, `Pumsb` et `USCensus`, on peut remarquer que la méthode en deux étapes de l'algorithme 4 est un plus rapide que TPSampling. Le jeu de données `Susy` n'est pas utilisé dans cette section du fait que la méthode en deux étapes de l'algorithme 4 ne s'y applique pas à cause de sa très grande taille. Sur les jeux de données `Adult`, `Chess` et `Mushroom`, on peut noter que les rapidités des deux méthodes sont quasiment égales. En effet, pour le tirage d'un motif de norme ℓ au sein d'une transaction t, l'algorithme 4 visite exactement ℓ items de t. A contrario, TPSampling peut visiter plus de ℓ noeuds (conteneurs des items) avant de retourner le motif correspondant. Nous notons d'une part que les temps de tirage de TPSampling + $(M - Area)$ et TPSampling + $(M - Freq)$ évoluent presque de la même manière. D'autre part, le temps de tirage d'un motif augmente légèrement suivant la contrainte de norme maximale.

D'une manière générale, nous pouvons dire que TPSampling est presque aussi rapide que la méthode en deux étapes de l'algorithme 4 pour tirer des milliers de motifs en quelques secondes.

7.5.3 Passage à l'échelle de TPSampling

Cette section étudie les performances de TPSampling sur les gros jeux de données `USCensus` et `Susy` contenant respectivement $1,000,000$ et $5,000,000$ de transactions. Cette étude se fait en fixant une contrainte de norme maximale $M = 5$ pour l'aire et la fréquence et en combinant une fonction de décroissance exponentielle en $\alpha = 0.1$ avec une contrainte de norme maximale $M = 5$ pour la mesure de décroissance exponentielle. Avec ces paramètres, nous avons évalué à la figure 7.9 le nombre moyen de motifs tirés par TPSampling en une durée $d \in [1..10]$ secondes.

D'une manière générale, nous notons que TPSampling tire en moyenne des milliers de motifs en une seconde sur les gros jeux de données (`USCensus` et `Susy`). Avec le jeu de données `USCensus`, la méthode TPSampling parvient à tirer en moyenne des centaines de milliers de motifs en 10 secondes. Précisément, suivant la fréquence, elle tire en moyenne $175,514$ motifs dont $175,335$ motifs distincts, $196,830$ motifs dont $179,788$ motifs distincts suivant la décroissance exponentielle en $\alpha = 0.1$ et enfin, $179,960$ motifs dont $179,751$ motifs distincts suivant l'aire. Avec le jeu de données `Susy`, nous notons que TPSampling parvient à tirer en moyenne des dizaines de milliers de motifs en 10 secondes. Précisément, suivant la fréquence, TPSampling tire en moyenne $42,329$ motifs dont $39,405$ motifs distincts, $53,848$ motifs dont $14,939$ motifs distincts suivant la décroissance exponentielle en $\alpha = 0.1$ et enfin, $46,869$ motifs dont $43,547$ motifs distincts suivant l'aire.

Nous avons aussi noté qu'il existe globalement une très grande diversité sur les motifs retournés (cette diversité est calculée par rapport au nombre de motifs distincts dans l'échantillon). Comparée à celle obtenue avec le jeu de données `USCensus`, cette diversité est plus faible dans le cas de la décroissance exponentielle en $\alpha = 0.1$ pour la base de données `Susy`. Cela s'explique par le fait que la décroissance exponentielle favorise les motifs courts qui sont aussi les plus fréquents dans la base de données et donc dans l'échantillon de motifs tirés. Or les transactions de la base de données `USCensus` sont plus longues que celles de la base de données `Susy`, en moyenne 68.0 contre 19.0. Pour augmenter la diversité des motifs retournés par TPSampling avec la base de données `Susy` dans le cas de la décroissance exponentielle, on peut augmenter la valeur de α.

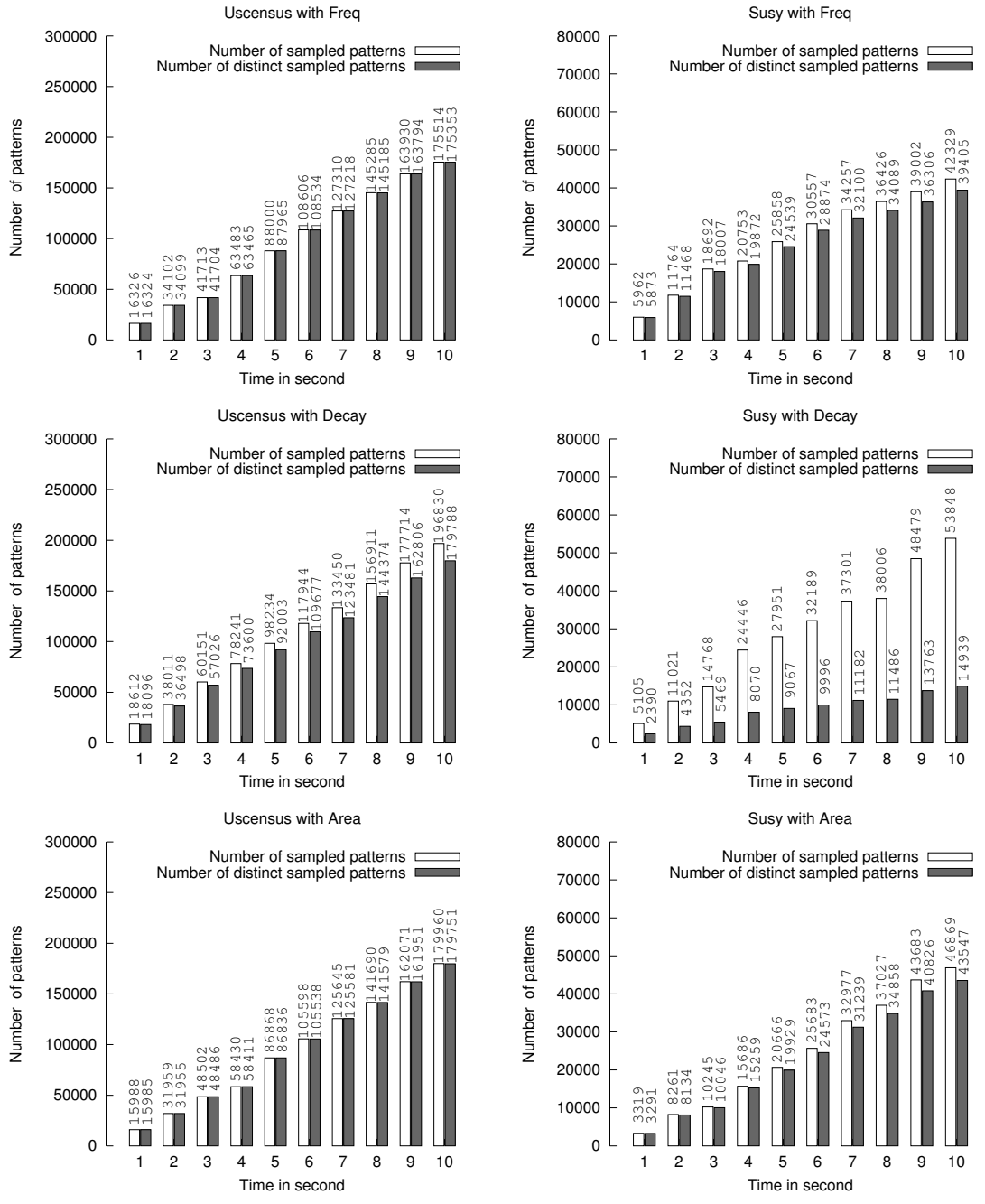


FIGURE 7.9 – Passage à l'échelle de TPSAMPLING (avec $M = 5$ et $\alpha = 0.1$ pour la décroissance exponentielle)

7.6 Conclusion

Nous avons proposé une nouvelle méthode d'échantillonnage en sortie de motifs ensemblistes basée sur une structure de données compacte appelée "trie". Nous avons précisément montré comment échantillonner des motifs ensemblistes directement à partir d'un trie et suivant n'importe quelle mesure d'intérêt fondée sur la norme en proposant TPSampling. C'est un algorithme générique qui tire des motifs à partir d'un trie d'occurrences de motifs suivant une distribution proportionnelle à une mesure d'intérêt fondée sur la norme choisie par l'utilisateur. Les expérimentations ont montré que TPSampling est aussi performant que l'algorithme 4 en deux étapes pour tirer des milliers de motifs en quelques secondes.

Parmi les obstacles que rencontre notre méthode TPSampling, nous avons la durée du prétraitement qui n'est pas négligeable pour l'ordonnancement des items. Le temps de tri des items d'une transaction est très élevé dans le cas des jeux de données ayant de très longues transactions. Nous avons aussi noté que ses performances en terme de stockage diminuent face aux grosses bases de données ne contenant que des items rares. Dans ce cas, le nombre de noeuds dans le trie d'occurrences de motifs est très élevé.

En perspective, nous voudrions dans l'immédiat tirer des motifs séquentiels à partir d'un trie d'occurrences de motifs. En effet, le système de numérotation des motifs que nous avons utilisé pour tirer des motifs ensemblistes proportionnellement à une mesure d'intérêt fondée sur la norme est une piste prometteuse pour éviter la méthode de tirage par rejet de CSSampling et NUSSampling.

Conclusion générale

Bilan et perspectives

Ce dernier chapitre conclut cette thèse qui porte sur "l'échantillonnage sous contraintes de motifs structurés". Il commence par une synthèse de nos différentes contributions sur l'échantillonnage en sortie de motifs en soulignant les limites et termine par une présentation de nos perspectives dans le domaine de la fouille de motifs par échantillonnage en sortie.

Bilan

Les travaux présentés dans cette thèse constituent nos différentes contributions dans le domaine de l'échantillonnage en sortie de motifs. Nous avons en particulier proposé une classe de mesures d'intérêt fondées sur la norme afin de surmonter l'écueil de la longue traîne que rencontre généralement les méthodes d'échantillonnage en sortie et en particulier celles en deux étapes. En se basant sur cette classe de mesures d'intérêt fondées sur la norme, nous avons proposé des méthodes d'échantillonnage en deux étapes pour tirer des motifs ensemblistes et séquentiels puis des motifs à la demande dans des bases de données distribuées. Nous avons aussi élargi notre étude en faisant de l'échantillonnage en sortie de motifs dans les bases de données compressées sous la forme d'un trie.

Cadre générique des mesures d'intérêt fondées sur la norme La classe dans laquelle s'inscrivent les mesures d'intérêt que nous avons utilisées dans cette thèse est dite "fondée sur la norme". Elle nous a permis de surmonter le phénomène de la longue traîne qui fait partie des principaux problèmes de l'échantillonnage en sortie. Avec des contraintes de normes minimale μ et maximale M, nous sommes parvenus à tirer des motifs ayant une mesure d'intérêt élevée.

D'une façon intéressante, plusieurs fonctions d'utilité fondées sur la norme peuvent être combinées pour en former une autre. Par exemple, en prenant une décroissance exponentielle en α et une contrainte de norme maximale M, nous avons une mesure qui favorise les motifs avec une petite norme au dépit de ceux avec une grande norme. En outre, les motifs de normes plus grandes que M ont tous une utilité égale à 0. Une propriété majeure que cache cette classe de mesures d'intérêt est que tous les motifs de même norme ont la même utilité au sein d'une même instance. Cette propriété facilite considérablement l'utilisation

de la classe de mesures sur d'autres langages de motifs et d'autres structures de données tels que le langage des bases de données séquentielles et les tries. Elle nous a aussi permis de proposer des algorithmes génériques dans le sens où ils peuvent être utilisés avec toute mesure d'intérêt fondée sur la norme ou avec tout type de partitionnement d'une base de données distribuée (vertical, horizontal et hybride).

Cependant, il existe d'autres types de contraintes qui ne font pas partie de notre classe et qui permettent de bien gérer le phénomène de la longue traîne telle que la contrainte de fréquence minimale. Excepté le cas de la méthode naïve qui se base sur le rejet, le contrôle de la fréquence des motifs tirés reste une limite des méthodes existantes d'échantillonnage en sortie de motifs.

Echantillonnage en deux étapes selon une mesure d'intérêt fondée sur la norme Nous avons proposé une méthode générique d'échantillonnage en sortie en deux étapes qui tire des motifs suivant une mesure d'intérêt fondée sur la norme.

Notre première contribution algorithmique consiste à la mise en place d'un algorithme d'échantillonnage en sortie de motifs ensemblistes générique dans le sens où il s'applique à n'importe quelle mesure d'intérêt fondée sur la norme. Plus précisément, nous avons proposé une version générique de la méthode en deux étapes de base en introduisant des contraintes basées sur la norme afin d'éviter de tirer des motifs à partir de la longue traîne.

Notre deuxième contribution algorithmique concerne les bases de données séquentielles, là où les motifs longs et rares de la longue traîne sont très abondants dans la base de données. NUSSampling est l'algorithme générique en deux étapes que nous avons proposé pour échantillonner des motifs séquentiels. Après avoir pondéré les séquences de la base de données suivant la mesure d'intérêt fondée sur la norme choisie par l'utilisateur, il tire des motifs séquentiels en se basant sur le principe des deux étapes. Les expérimentations effectuées sur certains jeux de données séquentielles ont toutefois montré que le taux de rejet peut alors être très significatif.

Echantillonnage de motifs à la demande à partir de bases de données distribuées Une autre contribution que nous avons apportée dans cette thèse est l'extension de l'échantillonnage en sortie aux bases de données distribuées. Contrairement aux bases de données locales où toutes les transactions sont stockées dans un même support, les transactions et/ou les items d'une transaction peuvent être stockées dans des fragments différents. Autrement dit, les items d'un même motif peuvent eux aussi se retrouver dans des endroits différents. Pour résoudre efficacement ce problème, nous avons proposé DDSampling, un algorithme générique d'échantillonnage en deux étapes qui n'a besoin de centraliser que les normes des transactions, normes qu'il stocke sous la forme d'une matrice appelée matrice de pondération. Ainsi, lors du tirage d'un motif, seuls les items nécessaires à sa construction sont rapatriés. Ainsi, DDSampling est une méthode parcimonieuse en coût de communication, et donc, propice à la découverte interactive de motifs dans des bases de données distribuées.

Nous avons aussi noté qu'il y a une faible altération de l'exactitude du tirage d'un motif lorsque certains sites tombent en panne. Il serait donc intéressant de proposer un mécanisme de correction des poids afin de contre-balancer la panne d'un site. Dans le

cas des bases de données répliquées, on pourrait mettre l'adresse des fragments dupliqués comme une alternative.

Echantillonnage de motifs par trie Notre dernière contribution est basée sur une structure de données compacte, appelée "trie", qui nous a permis d'introduire la notion de trie d'occurrences de motifs. Autrement dit, l'ensemble des occurrences de motifs d'une base de données sont stockés sous la forme d'un trie dans lequel les noeuds sont pondérés. Le système de numérotation défini de manière récursive, de type postfixe (en profondeur et de gauche à droite) nous a permis de donner un rang unique à chaque occurrence de motifs. Ensuite, l'algorithme d'échantillonnage basé sur le trie d'occurrences de motifs nommé TPSampling, permet de tirer efficacement des motifs suivant toute mesure d'intérêt fondée sur la norme. Il est intéressant de noter que la pondération des occurrences de motifs dans le trie ne dépend que des contraintes de normes minimale et maximale. Ainsi, changer la mesure d'intérêt, par exemple "l'aire" à la place de "la décroissance exponentielle" n'oblige pas à la reconstruction du trie.

Il est finalement important de noter que nous n'avons pas encore tiré pleinement profit des possibilités du trie. Autrement dit, nous pensons que d'autres tâches de la recherche de motifs pourraient être résolues efficacement avec les tries d'occurrences de motifs. Par exemple, si $\mathcal{I} = \{A, B, C, D\}$, on peut facilement échantillonner des motifs qui respectent des contraintes syntaxiques du type "ne contenant pas les items A et D à la fois" à l'aide d'une méthode naïve par rejet. Mais, peut-on prendre efficacement en compte ce type de contraintes avec les trie ? Par rapport à ce que nous avons présenté dans le chapitre des tries, on peut répondre par "non". Cependant, un enrichissement des informations stockées dans les tries d'occurrences pourrait peut-être permettre à l'utilisateur de choisir des mesures d'intérêt plus complexes que celles de notre classe de mesures fondées sur la norme.

Cas d'usage des motifs échantillonnés L'usage des motifs échantillonnés a été illustré dans différents domaines au cours de nos travaux. Dans le domaine de la classification, nous avons montré que les motifs échantillonnés (avec l'algorithme 4 en deux étapes pour les données ensemblistes ou NUSSampling pour les séquences) combinés avec une méthode de classification de type SVM permettent de construire de très bons classifieurs. En particulier, nous avons vu qu'avec NUSSampling+SVM, les accuracies des classifieurs obtenus pour la classification de données séquentielles sont meilleures que celles des méthodes de l'état de l'art qui font de la fouille exhaustive. Même en la comparant avec les méthodes "Top-k" qui utilisent les motifs les plus fréquents, NUSSampling+SVM donne de meilleures performances du fait de la diversité des motifs qu'il produit.

Avec les triplestores du Web, nous avons montré comment détecter des données aberrantes à l'aide des motifs échantillonnés à la demande par DDSampling. Notamment, nous avons retrouvé des entités du Web considérées comme étant des personnes alors qu'en réalité, elles ne font pas partie de la classe `Person`.

Dans tous les cas, nous notons que les performances de nos méthodes aussi bien pour la classification que pour la détection de données aberrantes, sont fortement liées à la contrainte de norme maximale des motifs échantillonnés. Généralement, nous avons re-

marqué que les motifs les plus intéressants ont une norme comprise entre 1 et 10. Cette contrainte de norme est souvent bien plus facile à déterminer que la contrainte de seuil de support minimal qui est un nombre réel variant entre 0 (exclu) et 1. Evidemment, la bonne valeur de α variant entre 0 et 1 pour la décroissance exponentielle n'est pas aussi facile à déterminer. Cependant, nous avons noté dans nos expérimentations que combiné avec la contrainte de norme maximale, $\alpha = 0.01$ donne des motifs ni trop courts, ni trop longs.

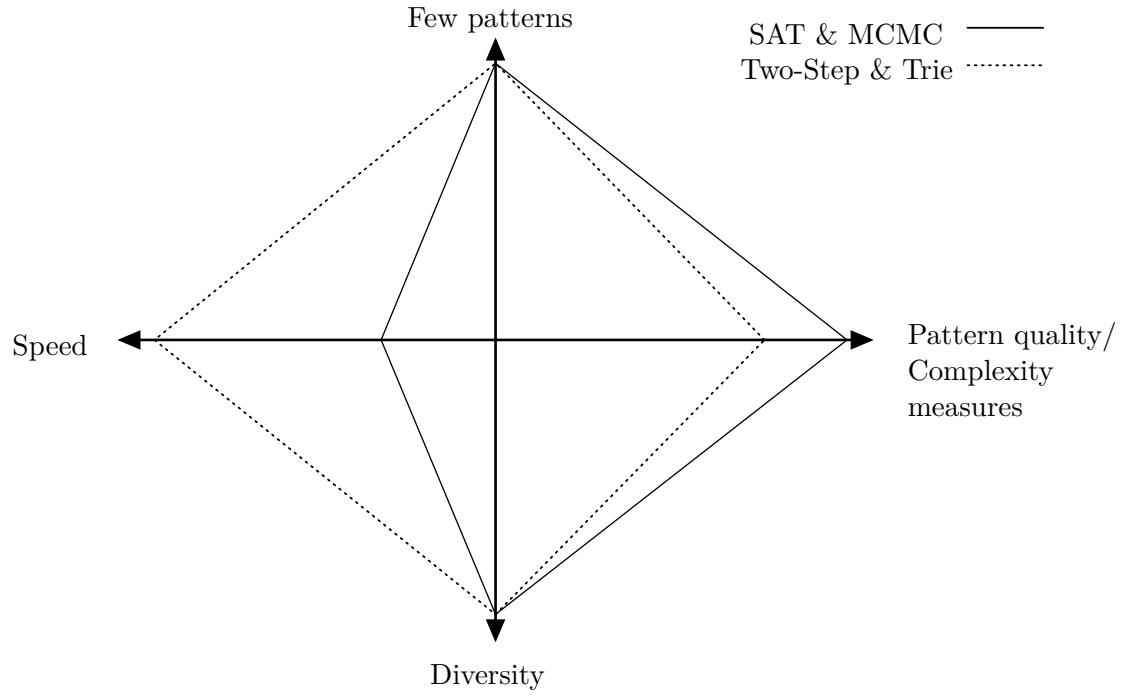


FIGURE 7.10 – Caractéristiques des méthodes d'échantillonnage en sortie

De manière plus générale, la figure 7.10 positionne nos contributions dans le domaine de l'échantillonnage en sortie en les comparant avec celles de l'état de l'art. Par exemple pour la classe de méthodes en deux étapes, nous avons contribué dans l'axe de la qualité et de la complexité des mesures à l'aide de la classe de mesures d'intérêt fondées sur la norme. Ensuite, nous avons la méthode d'échantillonnage en sortie de motifs basée sur les tries qui est aussi performante que la méthode en deux étapes tout en étant moins coûteuse en espace mémoire. Cependant, il reste beaucoup de choses à apporter en terme de mesures d'intérêt. Comparées aux méthodes basées sur les MCMC ou sur SAT, les méthodes en plusieurs étapes que nous avons développées sont moins génériques, car elles se limitent à l'utilisation de mesures fondées sur la norme.

Perspectives

Au vu des avantages et des inconvénients notés dans la section précédente, plusieurs perspectives se déclinent. En voici quelques unes :

Echantillonnage de motifs séquentiels à base de trie Nous rappelons que dans cette thèse, l'échantillonnage d'un motif séquentiel est basé sur la méthode par rejet combinée avec la notion de première occurrence. Autrement dit, si l'occurrence tirée n'est pas la première occurrence d'un motif, alors on la rejette puis on tire une autre occurrence. Dans certains jeux de données, le nombre de rejets pourrait être très élevé et rendrait ainsi la méthode inefficace (par exemple, pour des séquences ADN). Nous pensons que ce rejet pourrait être évité en ayant recours aux tries. Dans le cadre des données transactionnelles, toute occurrence d'un itemset contenue dans une transaction est identifiée dans un trie par un rang précis, ce qui permet un tirage efficace d'un motif de norme donnée. Nous pensons que ce principe pourrait être généralisé aux bases de données séquentielles en attribuant également un rang unique à chaque première occurrence d'un motif séquentiel.

Vers d'autres contraintes, d'autres fonctions d'utilités et d'autres langages de motifs Nous souhaiterions étendre notre classe de mesures d'intérêt fondées sur la norme avec d'autres fonctions d'utilité et d'autres contraintes afin d'explorer des langages plus complexes. Notons d'abord qu'avec les bases de données séquentielles, il serait intéressant d'introduire des contraintes de gap pour éviter d'avoir des gaps significatifs entre deux itemsets d'un motif séquentiel [Srikant et Agrawal, 1996]. Comme dans le cas des itemsets, nous pensons aussi que les motifs séquentiels échantillonnés peuvent être utiles pour la détection de données aberrantes dans des données séquentielles [Giacometti et Soulet, 2016] ou pour concevoir des systèmes interactifs dédiés à la découverte de modèles séquentiels [Bhuiyan *et al.*, 2012]. Avec plus d'ambition, il serait intéressant de considérer le cadre des *set systems* [Arimura et Uno, 2009], qui est un cadre flexible de langages de motifs tel que le langage des graphes relationnels sans contraintes de connectivité. En effet, le tirage uniforme au sein de structures complexes rendu possible par une forme canonique (ici la première occurrence) peut être envisagé avec d'autres langages structurés.

Dans certains cas, il est possible d'avoir une base de données où chaque transaction a son utilité, mais aussi chaque item, et enfin de devoir tenir compte du nombre de répétitions d'un item dans une transaction. L'extraction de "HUI" (High Utility Itemset) [Yao *et al.*, 2006] pour les bases de données transactionnelles peut ainsi être une piste de recherche prometteuse pour l'échantillonnage en sortie de motifs sous contrainte de norme. Nous avons aussi vu dans des expérimentations (non présentées dans cette thèse) pour la classification avec les bases de données distribuées du LOD, que l'idéal serait d'extraire des contrastes (motifs extraits avec des mesures discriminatives) afin de construire directement de bons classifieurs (sans passer par l'utilisation d'une méthode de classification annexe telle que SVM).

Tries et fouille de motifs centré sur l'utilisateur La fouille interactive de motifs requiert un temps de réponse très court pour avoir un couplage fort entre le système et

l'utilisateur. Avec une méthode d'échantillonnage en sortie de motifs, on a besoin de mettre à jour les poids des transactions et/ou des motifs à chaque fois que l'utilisateur change de mesure d'intérêt. Ainsi, il faut nécessairement optimiser ce temps de prétraitement pour rendre le système performant. Nos expérimentations faites dans le chapitre 7 pour les tries montre l'importance notoire de ne pas pondérer toutes les transactions de la base de données à chaque fois que l'utilisateur change de mesure d'intérêt. En effet, avec les mesures d'intérêt fondées sur la norme, il est possible de fixer les contraintes de normes minimale et maximale évitant de tirer des motifs à partir de la longue traîne. De plus, nous avons vu qu'on construit le trie d'occurrences de motifs indépendamment de toute autre utilité considérée. Ainsi, le temps de prétraitement quand un utilisateur change de mesure d'intérêt est quasiment nul.

Pourquoi ne pas utiliser une contrainte de fréquence minimale plutôt qu'une longueur maximale pour éviter la malédiction de la longue traîne ? La principale cause du phénomène de la longue traîne est que les motifs rares sont très abondants dans la base de données. Si nous parvenions à tirer un échantillon de motifs à partir de l'ensemble des motifs ayant un support supérieur à un seuil de support minimal, alors le problème de la longue traîne serait résolu. Cela écarterait tous les motifs rares appartenant à la longue traîne. Une piste prometteuse dans ce sens est l'utilisation des tries. On sait que si les items des transactions sont insérés dans le trie suivant l'ordre décroissant de leurs fréquences dans la base de données considérée, alors les motifs les plus fréquents sont proches du sommet. Par contre les motifs peu fréquents ont certains de leurs items en bas du trie. Si on élague les noeuds du trie dont les fréquences d'apparition de leurs labels sont inférieures à un seuil de support minimal, alors on est certain d'avoir des motifs fréquents dans le trie. En effet, si l'identifiant P d'un noeud est fréquent, alors tous les sous-ensembles de P seront aussi fréquents. Mais, le principal problème est que plusieurs occurrences d'un même motif peuvent être retrouvées dans plusieurs sous-tries du trie, ce qui fait que certains motifs fréquents risquent d'être élagués aussi. Toutefois, il serait tentant de tester cette méthode afin d'évaluer la qualité des échantillons de motifs obtenus. Une autre piste de recherche serait d'étendre les tries d'occurrences de motifs aux FP-trees.

Publications issues de cette thèse

Publications

Au cours de ces trois années de thèse, nous avons fait les publications suivantes dans le domaine de l'échantillonnage en sortie de motifs structurés. Parmi ces publications, nous avons deux distinctions à la conférence nationale sur l'Extraction et Gestion des Connaissances (EGC) en 2018 et en 2019 comme étant 5 des meilleurs papiers académiques. Notre papier à la conférence internationale ICDM'18 (International Conference on Data Mining) a reçu une distinction qui nous a permis de recevoir une invitation à la soumission d'une version étendue au journal KAIS'19 (Knowledge and Information Systems).

Publications internationales

[Diop *et al.*, 2018b]**Diop, L.**, Diop, C. T., Giacometti, A., Li, D. et Soulet, A. (2018b). Sequential pattern sampling with norm constraints. In 2018 IEEE International Conference on Data Mining (ICDM), pages 89–98. IEEE.

[Diop *et al.*, 2019d] **Diop, L.**, Diop, C. T., Giacometti, A., Li, D. et Soulet, A. (2019d). Sequential pattern sampling with norm-based utility. Knowledge and Information Systems, 62(5) :2029–2065.

[Diop *et al.*, 2020b] **Diop, L.**, Diop, C. T., Giacometti, A. et Soulet, A. (2020b). Pattern sampling in distributed databases. In Advances in Databases and Information Systems, ADBIS'2020, pages –A paraître–.

Publications nationales

[Diop *et al.*, 2018a] **Diop, L.**, Diop, C. T., Giacometti, A., Li, D. et Soulet, A. (2018a). Echantillonnage de motifs séquentiels sous contrainte sur la norme. Revue des Nouvelles Technologies de l'Information, Extraction et Gestion des Connaissances, RNTI-E-34 :35–46.

[Diop *et al.*, 2019a] **Diop, L.**, Diop, C. T., Giacometti, A., Li, D. et Soulet, A. (2019a). Construction de variables pour la classification par échantillonnage de motifs. Actes des 26èmes Rencontres, Société Francophone de Classification (SFC) :89–94.

[Diop *et al.*, 2019b] **Diop, L.**, Diop, C. T., Giacometti, A., Li, D. et Soulet, A. (2019b).

Découverte de motifs à la demande dans une base de données distribuée. In Extraction et Gestion des connaissances, EGC 2019, Metz, France, January 21-25, 2019, pages 21–32.

[Diop *et al.*, 2019c] **Diop, L.**, Diop, C. T., Giacometti, A., Li, D. et Soulet, A. (2019c). Echantillonnage de motifs ensemblistes selon une utilité fondée sur la taille. In ConféreNce sur la Recherche en Informatique et ses Applications, CNRIA'2019, pages 104–115.

[Diop *et al.*, 2020a] **Diop, L.**, Diop, C. T., Giacometti, A. et Soulet, A. (2020a). Fpof et malédiction de la longue traîne. In ConféreNce sur la Recherche en Informatique et ses Applications, CNRIA'2020, pages –A paraître–.

Distinctions

[Diop *et al.*, 2019b] Nominé pour le prix du meilleur papier académique

[Diop *et al.*, 2018b] Un des meilleurs papiers pour soumission d'une version étendue au journal KAIS'2019

[Diop *et al.*, 2018a] Nominé pour le prix du meilleur papier académique

Bibliographie

[Aggarwal *et al.*, 2014] Aggarwal, C. C., Bhuiyan, M. A. et Hasan, M. A. (2014). *Frequent Pattern Mining Algorithms : A Survey*, pages 19–64. Springer International Publishing, Cham.

[Agrawal *et al.*, 1993] Agrawal, R., Imieliński, T. et Swami, A. (1993). Mining association rules between sets of items in large databases. *In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, SIGMOD '93, pages 207–216.

[Agrawal et Srikant, 1994] Agrawal, R. et Srikant, R. (1994). Fast algorithms for mining association rules in large databases. *In Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Agrawal et Srikant, 1995] Agrawal, R. et Srikant, R. (1995). Mining sequential patterns. *In Proc. of ICDE 95*, pages 3–14.

[Ahlers, 2013] Ahlers, D. (2013). Assessment of the accuracy of geonames gazetteer data. *In* Jones, C. B. et Purves, R., éditeurs : *Proceedings of the 7th Workshop on Geographic Information Retrieval, GIR 2013, 5th November, 2013, Orlando, Florida, USA*, pages 74–81. ACM.

[Al Hasan et Zaki, 2009] Al Hasan, M. et Zaki, M. J. (2009). Output space sampling for graph patterns. *Proc. of the VLDB Endowment*, 2(1):730–741.

[Anderson, 2004] Anderson, C. (2004). The long tail. *Wired magazine*, 12(10):170–177.

[Arimura et Uno, 2009] Arimura, H. et Uno, T. (2009). Polynomial-delay and polynomial-space algorithms for mining closed sequences, graphs, and pictures in accessible set systems. *In Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 1088–1099. SIAM.

[Auer *et al.*, 2007] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R. et Ives, Z. G. (2007). Dbpedia : A nucleus for a web of open data. *In* Aberer, K., Choi, K., Noy, N. F., Allemang, D., Lee, K., Nixon, L. J. B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G. et Cudré-Mauroux, P., éditeurs : *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*, volume 4825 de *Lecture Notes in Computer Science*, pages 722–735. Springer.

[Bendimerad *et al.*, 2016] Bendimerad, A. A., Plantevit, M. et Robardet, C. (2016). Unsupervised exceptional attributed sub-graph mining in urban data. *In Proc. of ICDM 2016*, pages 21–30. IEEE.

[Bhuiyan et Hasan, 2016] Bhuiyan, M. et Hasan, M. A. (2016). Interactive knowledge discovery from hidden data through sampling of frequent patterns. *Statistical Analysis and Data Mining : The ASA Data Science Journal*, 9(4):205–229.

[Bhuiyan *et al.*, 2012] Bhuiyan, M., Mukhopadhyay, S. et Hasan, M. A. (2012). Interactive pattern mining on hidden data : a sampling-based solution. *In Proc. of ACM CIKM*, pages 95–104.

[Bodon et Rónyai, 2003] Bodon, F. et Rónyai, L. (2003). Trie : An alternative data structure for data mining algorithms. *Mathematical and Computer Modelling*, 38(7):739 – 751. Hungarian Applied Mathematics.

[Boley *et al.*, 2010] Boley, M., Gärtner, T. et Grosskreutz, H. (2010). Formal concept sampling for counting and threshold-free local pattern mining. *In Proc. of SDM 2010*, pages 177–188. SIAM.

[Boley *et al.*, 2011] Boley, M., Lucchese, C., Paurat, D. et Gärtner, T. (2011). Direct local pattern sampling by efficient two-step random procedures. *In Proc. of the 17th ACM SIGKDD*, pages 582–590.

[Boley *et al.*, 2012] Boley, M., Moens, S. et Gärtner, T. (2012). Linear space direct pattern sampling using coupling from the past. *In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–77. ACM.

[Burdick *et al.*, 2001] Burdick, D., Calimlim, M. et Gehrke, J. (2001). Mafia : a maximal frequent itemset algorithm for transactional databases. *In Proceedings 17th International Conference on Data Engineering*, pages 443–452.

[Calders *et al.*, 2006] Calders, T., Rigotti, C. et Boulicaut, J.-F. (2006). A survey on condensed representations for frequent sets. *In* Boulicaut, J.-F., De Raedt, L. et Mannila, H., éditeurs : *Constraint-Based Mining and Inductive Databases*, pages 64–80, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Chakraborty *et al.*, 2014] Chakraborty, S., Fremont, D. J., Meel, K. S., Seshia, S. A. et Vardi, M. Y. (2014). Distribution-aware sampling and weighted model counting for SAT. *CoRR*, abs/1404.2984.

[Cheung *et al.*, 1996] Cheung, D. W., Ng, V. T., Fu, A. W. et Fu, Y. (1996). Efficient mining of association rules in distributed databases. *IEEE transactions on Knowledge and Data Engineering*, 8(6):911–922.

[Cook et Holder, 1994] Cook, D. J. et Holder, L. B. (1994). Substructure discovery using minimum description length and background knowledge. *J. Artif. Int. Res.*, 1(1): 231–255.

[Crémilleux *et al.*, 2019] Crémilleux, B., Giacometti, A. et Soulet, A. (2019). How your supporters and opponents define your interestingness. *In* Berlingerio, M., Bonchi, F., Gärtner, T., Hurley, N. et Ifrim, G., éditeurs : *Machine Learning and Knowledge Discovery in Databases*, pages 373–389, Cham. Springer International Publishing.

[Cule *et al.*, 2009] Cule, B., Goethals, B. et Robardet, C. (2009). A new constraint for mining sets in sequences. *In Empty*, volume 1, pages 317–328.

[Demšar, 2006] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30.

[Diop *et al.*, 2018a] Diop, L., Diop, C. T., Giacometti, A., Li, D. et Soulet, A. (2018a). Echantillonnage de motifs séquentiels sous contrainte sur la norme. *Revue des Nouvelles Technologies de l'Information*, Extraction et Gestion des Connaissances, RNTI-E-34:35–46.

[Diop *et al.*, 2019a] Diop, L., Diop, C. T., Giacometti, A., Li, D. et Soulet, A. (2019a). Construction de variables pour la classification par échantillonnage de motifs. *Actes des 26èmes Rencontres*, Société Francophone de Classification (SFC):89–94.

[Diop *et al.*, 2019b] Diop, L., Diop, C. T., Giacometti, A., Li, D. et Soulet, A. (2019b). Découverte de motifs à la demande dans une base de données distribuée. *In Extraction et Gestion des connaissances, EGC 2019, Metz, France, January 21-25, 2019*, pages 21–32.

[Diop *et al.*, 2019c] Diop, L., Diop, C. T., Giacometti, A., Li, D. et Soulet, A. (2019c). Echantillonnage de motifs ensemblistes selon une utilité fondée sur la taille. *In ConféreNce sur la Recherche en Informatique et ses Applications, CNRIA'2019*, pages 104–115.

[Diop *et al.*, 2019d] Diop, L., DIOP, C. T., Giacometti, A., Li, D. et Soulet, A. (2019d). Sequential pattern sampling with norm-based utility. *Knowledge and Information Systems (KAIS)*.

[Diop *et al.*, 2018b] Diop, L., Diop, C. T., Giacometti, A., Li Haoyuan, D. et Soulet, A. (2018b). Sequential Pattern Sampling with Norm Constraints. *In IEEE International Conference on Data Mining (ICDM)*, Singapore, Singapore.

[Diop *et al.*, 2020a] Diop, L., Diop, C. T., Giacometti, A. et Soulet, A. (2020a). Fpof et malédiction de la longue traîne. *In ConféreNce sur la Recherche en Informatique et ses Applications, CNRIA'2020*, pages –A paraître–.

[Diop *et al.*, 2020b] Diop, L., Diop, C. T., Giacometti, A. et Soulet, A. (2020b). Pattern sampling in distributed databases. *In* Darmont, J., Novikov, B. et Wrembel, R., éditeurs : *Advances in Databases and Information Systems*, pages 60–74, Cham. Springer International Publishing.

[Dzyuba *et al.*, 2014] Dzyuba, V., Leeuwen, M. v., Nijssen, S. et De Raedt, L. (2014). Interactive learning of pattern rankings. *Int. Journal on Artificial Intelligence Tools*, 23(06):32 pages.

[Dzyuba *et al.*, 2017] Dzyuba, V., van Leeuwen, M. et De Raedt, L. (2017). Flexible constrained sampling with guarantees for pattern mining. *Data Mining and Knowledge Discovery*, 31(5):1266–1293.

[Egho *et al.*, 2017] Egho, E., Gay, D., Boullé, M., Voisine, N. et Clérot, F. (2017). A user parameter-free approach for mining robust sequential classification rules. *Knowl. Inf. Syst.*, 52(1):53–81.

[Egho *et al.*, 2015] Egho, E., Raïssi, C., Calders, T., Jay, N. et Napoli, A. (2015). On measuring similarity for sequences of itemsets. *Data Mining and Knowledge Discovery*, 29(3):732–764.

[Ferrández et Peral, 2019] Ferrández, A. et Peral, J. (2019). Mergedtrie : Efficient textual indexing. *PLOS ONE*, 14(4):1–19.

[Fournier-Viger *et al.*, 2013a] Fournier-Viger, P., Gomariz, A., Gueniche, T., Mwamikazi, E. et Thomas, R. (2013a). Tks : Efficient mining of top-k sequential patterns. *In* Motoda, H., Wu, Z., Cao, L., Zaiane, O., Yao, M. et Wang, W., éditeurs : *Advanced Data Mining and Applications*, pages 109–120, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Fournier-Viger *et al.*, 2013b] Fournier-Viger, P., Gomariz, A., Gueniche, T., Mwamikazi, E. et Thomas, R. (2013b). Tks : efficient mining of top-k sequential patterns. *In International Conference on Advanced Data Mining and Applications*, pages 109–120. Springer.

[Giacometti *et al.*, 2014] Giacometti, A., Li, D. H., Marcel, P. et Soulet, A. (2014). 20 years of pattern mining : A bibliometric survey. *SIGKDD Explor. Newsl.*, 15(1):41–50.

[Giacometti et Soulet, 2016] Giacometti, A. et Soulet, A. (2016). Anytime algorithm for frequent pattern outlier detection. *International Journal of Data Science and Analytics*, 2(3-4):119–130.

[Giacometti et Soulet, 2017] Giacometti, A. et Soulet, A. (2017). Interactive pattern sampling for characterizing unlabeled data. *In Proc. of IDA 2017*, pages 99–111. Springer.

[Giacometti et Soulet, 2018] Giacometti, A. et Soulet, A. (2018). Dense neighborhood pattern sampling in numerical data. *In Proc. of SDM 2018*, pages 756–764.

[Gueguen *et al.*, 2019] Gueguen, M., Sentieys, O. et Termier, A. (2019). Accelerating itemset sampling using satisfiability constraints on FPGA. *In IEEE/ACM Design, Automation and Test in Europe (DATE)*.

[Guns *et al.*, 2011] Guns, T., Nijssen, S. et Raedt, L. D. (2011). Itemset mining : A constraint programming perspective. *Artif. Intell.*, 175(12-13):1951–1983.

[Hamrouni, 2012] Hamrouni, T. (2012). Key roles of closed sets and minimal generators in concise representations of frequent patterns. *Intell. Data Anal.*, 16(4):581–631.

[Han *et al.*, 2007] Han, J., Cheng, H., Xin, D. et Yan, X. (2007). Frequent pattern mining : current status and future directions. *Data Mining and Knowledge Discovery*, 15(1):55–86.

[Han *et al.*, 2000] Han, J., Pei, J. et Yin, Y. (2000). Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2):1–12.

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

[Han *et al.*, 2002] Han, J., Wang, J., Lu, Y. et Tzvetkov, P. (2002). Mining top-k frequent closed patterns without minimum support. *In 2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 211–218.

[Hawkins, 1980] Hawkins, D. (1980). *Identification of outliers*. Chapman and Hall.

[He *et al.*, 2005] He, Z., Xu, X., Huang, J. Z. et Deng, S. (2005). Fp-outlier : Frequent pattern based outlier detection. *Computer Science and Information Systems*, 2(1):103–118.

[Inokuchi *et al.*, 2000] Inokuchi, A., Washio, T. et Motoda, H. (2000). An apriori-based algorithm for mining frequent substructures from graph data. *In* Zighed, D. A., Komorowski, J. et Żytkow, J., éditeurs : *Principles of Data Mining and Knowledge Discovery*, pages 13–23, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Jin et Agrawal, 2006] Jin, R. et Agrawal, G. (2006). Systematic approach for optimizing complex mining tasks on multiple databases. *In 22nd International Conference on Data Engineering (ICDE'06)*, pages 17–17.

[Khiari *et al.*, 2010] Khiari, M., Boizumault, P. et Crémilleux, B. (2010). Combining csp and constraint-based mining for pattern discovery. *In Empty*, pages 432–447.

[Knuth, 1997] Knuth, D. E. (Third edition, 1997). *the Art of Computer Programming*. Reading, Massachusetts : Addison–Wesley, 1968.

[Kuchar et Svatek, 2017] Kuchar, J. et Svatek, V. (2017). Spotlighting anomalies using frequent patterns. *In Proceedings of the KDD 2017 Workshop on Anomaly Detection in Finance*, volume 71 de *Proceedings of Machine Learning Research*, Halifax, Nova Scotia, Canada. PMLR.

[Kum *et al.*, 2006] Kum, H.-C., Chang, J. H. et Wang, W. (2006). Sequential pattern mining in multi-databases via multiple alignment. *Data Mining and Knowledge Discovery*, 12(2-3):151–180.

[Kuramochi et Karypis, 2001] Kuramochi, M. et Karypis, G. (2001). Frequent subgraph discovery. *In Proceedings of the 2001 IEEE International Conference on Data Mining*, ICDM '01, pages 313–320, Washington, DC, USA. IEEE Computer Society.

[Li et Zaki, 2012] Li, G. et Zaki, M. J. (2012). Sampling minimal frequent boolean (DNF) patterns. *In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 87–95. ACM.

[Lin *et al.*, 2010] Lin, F., Le, W., Bo, J., Lin, F., Le, W. et Bo, J. (2010). Research on maximal frequent pattern outlier factor for online high- dimensional time-series outlier detection.

[Mannila et Toivonen, 1997] Mannila, H. et Toivonen, H. (1997). Levelwise search and borders of theories in knowledge discovery. *Data mining and knowledge discovery*, 1(3): 241–258.

[Moens et Boley, 2014] Moens, S. et Boley, M. (2014). Instant exceptional model mining using weighted controlled pattern sampling. *In Proc. of IDA 2014*, pages 203–214. Springer.

[Moens et Goethals, 2013] Moens, S. et Goethals, B. (2013). Randomly sampling maximal itemsets. *In Proc. of IDEA Workshop 2013*, pages 79–86.

[Otey *et al.*, 2003] Otey, M. E., Wang, C., Parthasarathy, S., Veloso, A. et Meira, W. (2003). Mining frequent itemsets in distributed and dynamic databases. *In Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 617–620. IEEE.

[Özsu et Valduriez, 2011] Özsu, M. T. et Valduriez, P. (2011). *Principles of distributed database systems*. Springer Science & Business Media.

[Ren *et al.*, 2009] Ren, J., Wu, Q., Hu, C. et Wang, K. (2009). An approach for analyzing infrequent software faults based on outlier detection. *In Proceedings of the 2009 International Conference on Artificial Intelligence and Computational Intelligence - Volume 04*, AICI '09, pages 302–306.

[Savasere *et al.*, 1995] Savasere, A., Omiecinski, E. et Navathe, S. B. (1995). An efficient algorithm for mining association rules in large databases. *In Proceedings of the 21th International Conference on Very Large Data Bases*, VLDB '95, pages 432–444, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Soulet, 2006] Soulet, A. (2006). Un cadre générique de découverte de motifs sous contraintes fondées sur des primitives. *In Thèse de doctorat*.

[Soulet, 2017] Soulet, A. (2017). Two decades of pattern mining : Principles and methods. *In* Marcel, P. et Zimányi, E., éditeurs : *Business Intelligence*, pages 59–78, Cham. Springer International Publishing.

[Soulet *et al.*, 2011] Soulet, A., Raïssi, C., Plantevit, M. et Cremilleux, B. (2011). Mining dominant patterns in the sky. *In 2011 IEEE 11th International Conference on Data Mining*, pages 655–664.

[Srikant et Agrawal, 1996] Srikant, R. et Agrawal, R. (1996). Mining sequential patterns : Generalizations and performance improvements. *In Proc. of EDBT 96*, pages 3–17.

[Suchanek *et al.*, 2007] Suchanek, F. M., Kasneci, G. et Weikum, G. (2007). Yago : a core of semantic knowledge. *In* Williamson, C. L., Zurko, M. E., Patel-Schneider, P. F. et Shenoy, P. J., éditeurs : *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 697–706. ACM.

[Tang *et al.*, 2009] Tang, X., Li, G. et Chen, G. (2009). Fast detecting outliers over online data streams. *In 2009 International Conference on Information Engineering and Computer Science*, pages 1–4.

[Toivonen *et al.*, 1996] Toivonen, H. *et al.* (1996). Sampling large databases for association rules. *In Proc. of VLDB 96*, volume 96, pages 134–145.

[Ullmann, 1976] Ullmann, J. R. (1976). An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42.

[van Leeuwen, 2014] van Leeuwen, M. (2014). *Interactive Data Exploration Using Pattern Mining*, pages 169–182. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Vrandecic et Krötzsch, 2014] Vrandecic, D. et Krötzsch, M. (2014). Wikidata : a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

[Xifeng Yan et Jiawei Han, 2002] Xifeng Yan et Jiawei Han (2002). gspan : graph-based substructure pattern mining. *In 2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 721–724.

[Yao *et al.*, 2006] Yao, H. M., Hamilton, H. J. et Geng, L. (2006). A unified framework for utility based measures for mining itemsets. *In Empty*.

[Zaki *et al.*, 1997a] Zaki, M. J., Parthasarathy, S., Ogihara, M. et Li, W. (1997a). New algorithms for fast discovery of association rules. *In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, KDD'97, pages 283–286. AAAI Press.

[Zaki *et al.*, 1997b] Zaki, M. J., Parthasarathy, S., Ogihara, M. et Li, W. (1997b). Parallel algorithms for discovery of association rules. *Data Mining and Knowledge Discovery*, 1(4):343–373.

[Zhang et Zaki, 2006] Zhang, S. et Zaki, M. J. (2006). Mining multiple data sources : local pattern analysis. *Data Mining and Knowledge Discovery*, 12(2-3):121–125.

[Zhang *et al.*, 2010] Zhang, W., Wu, J. et Yu, J. (2010). An improved method of outlier detection based on frequent pattern. *In Proceedings of the 2010 WASE International Conference on Information Engineering - Volume 02*, ICIE '10, pages 3–6.

[Zhu *et al.*, 2011] Zhu, X., Li, B., Wu, X., He, D. et Zhang, C. (2011). Clap : Collaborative pattern mining for distributed information systems. *Decision support systems*, 52(1):40–51.

[Zhu et Wu, 2007] Zhu, X. et Wu, X. (2007). Discovering relational patterns across multiple databases. *In Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 726–735. IEEE.