République Algérienne Démocratique et Populaire
Ministère de l'enseignement Supérieur et de la Recherche Scientifique
Université de Mohamed El Bachir El Ibrahimi de Bordj Bou Arréridj
Faculté des Mathématiques et d'Informatique
Département d'informatique

MEMOIRE

Présenté en vue de l'obtention du diplôme

Master en informatique

Spécialité : Réseau et Multimédia

THEME

Nouvelle heuristique pour la protection des données sensibles lors de la fouille de motifs fréquents

les Étudiants:

1) Bahfir Sofiane.
2) Chekkal Mohammed.

Soutenu publiquement le : 19/06/2024

Devant le jury composé de :

Président : Mr.Belazoug Mouhoub.

Examinateur : Mme.Saifi Lynda.

Encadrant : Mr.Nouioua Farid.

2023/2024

Dédicace

Au nom de dieu, le très miséricordieux, je remercier dieu qui m'a aidé, mes parents pour toute leur sacrifice infini et pour leur prière précieuse afin de voir mon parcours universitaire et la présentation de mon projet de fin d'étude.

Eteins à exprimer ma gratitude envers mon professeur "Nouioua Farid" et toutes les personnes et mes amis qui m'ont aidé à accomplir cette grande réalisation. Ce fut un voyage formidable et je ne peux pas les remercier suffisamment pour leur soutien et leurs conseils.

J'espère que ce mémoire sera un beau souvenir des moments merveilleux que j'ai passés pendant mes études, je souhaite à tous bonne chance pour leur avenir.

Remerciement

Nous exprimons notre sincère Remerciements et notre gratitude au Pr Farid Nouioua, professeur au département d'informatique de l'université de Bordj Bou Arreridj, pour nous avoir donné l'opportunité de travailler sous ses ordres et pour nous avoir apporté tout le soutien possible à chaque étape de ce projet. Le niveau de flexibilité qu'il offre dans la mise en œuvre du travail du projet est hautement applaudi. Nous tenons à remercier même Mira Lefkir pour ses efforts et son coup de main à nos côtés.et nous remercions également les membres du jury d'avoir accepté de juger ce travail. Nous souhaitons également transmettre nos sincères salutations à tous les autres membres du corps professoral et du personnel du Département d'Informatique de l'université de Bordj Bou Arreridj, qui ont consacré de tout leur cœur et leurs conseils aux moments opportuns sans lesquels il aurait été très difficile de notre part. pour terminer ce travail de projet.

Résumé

La fouille de données est une technique d'analyse de données qui permet de découvrir des informations cachées et des relations entre des données. Elle est utilisée pour extraire des informations utiles à partir de grandes quantités de données non structurées. Notre étude s'est particulièrement concentrée sur la résolution du problème PPDM (Privacy preserving data mining) dans le contexte de la fouille d'itemsets fréquents à partir de bases de données transactionnelle. Le but étant d'effectuer un changement minimal sur la base afin de ne pas divulguer des informations sensibles lors du processus de fouille d'itemsets fréquents. Nous nous intéressons dans ce travail à l'étude, l'implémentation et la comparaison de deux approches heuristiques pour la résolution du problème PPDM : l'approche "Agrégée" qui supprimes certaines transactions et la nouvelle méthode que nous proposerons et qui utilise le problème combinatoire bien connu de couverture d'ensembles (en anglais SCP : **set cover problem**). L'étude expérimentale a été effectuée sur les deux bases de données : « Mushroom » et « D1 ».

Mots clés : Fouille de motifs fréquents, Bases de données transactionnelles, Fouille de données préservant la vie privée, Approche heuristique

Abstract

Data mining is a data analysis technique that uncovers hidden information and relationships between data. It is used to extract useful information from large amounts of unstructured data. Our study particularly focused on solving the PPDM (Privacy preserving data mining) problem in the context of mining frequent itemsets from transactional databases. The goal is to make a minimal change to the database in order not to disclose sensitive information during the process of mining frequent itemsets. In this work we are interested in the study, the implementation and the comparison of two heuristic approaches for solving the PPDM problem: the "Aggregated" approach which removes certain transactions and the new method that we propose bsd on the well-known combinatorial optimization problem of the **set cover problem** (**SCP**). The experimental study was carried out two databases: “Mushroom” and “D1”.

Keywords: Mining frequent patterns, Transactional databases, Privacy preserving data mining, Heuristic approach.

ملخص

استخراج البيانات هو أسلوب تحليل البيانات الذي يكشف عن المعلومات والعلاقات المخفية بين البيانات. يتم استخدامه لاستخراج معلومات مفيدة من كميات كبيرة من البيانات غير المنظمة. ركزت دراستنا بشكل خاص على حل مشكلة PPDM في سياق استخراج مجموعات العناصر المتكررة من قواعد بيانات المعاملات. الهدف هو إجراء تغيير بسيط على قاعدة البيانات حتى لا يتم الكشف عن المعلومات الحساسة أثناء عملية التنقيب في مجموعات العناصر المتكررة. نهتم في هذا العمل بدراسة وتنفيذ ومقارنة طريقتين إرشاديتين لحل مشكلة PPDM: النهج "المجمع" الذي يزيل بعض المعاملات والطريقة الجديدة التي سنقترحها ;التي تستعمل مشكلة الأمثلية التوافقية المعروفة بتغطية المجموعات (بالإنجليزية: SCP : Set Cover Problem) (SCP). أجريت الدراسة التجريبية على قاعدتي بيانات: "Mushroom" و "D1".

الكلمات المفتاحية: التنقيب عن الأنماط المكررة، قواعد بيانات المعاملات، التنقيب في البيانات مع الحفاظ على الخصوصية، النهج الإرشادي.

Table des matières

Liste des abréviations x

Liste des figures xi

Liste des tableaux xiii

Liste des algorithmes xiv

Chapitre 1 : Introduction Générale 1

1.1 Contexte 1
1.2 Objectifs 1
1.3 Structure du rapport 2

Chapitre 2 : Concepts et généralités sur la fouile de données 3

2.1 Introduction 3
2.2 KDD (Knowledge Discovery from Database) 3
2.2.1 Définition 3
2.2.2 Les étapes essentielles du KDD 4
2.2.3 Fouille de données et ECD 5
2.3 Fouille de données (Data Mining) 5
2.3.1 Définition de la fouille de données 5
2.3.2 Les tâches de Fouille de données 6
2.4 Apprentissage supervisé 7
2.4.1 Tâches d'apprentissage supervisé 7
2.4.2 Quelques techniques et modèles pour l'apprentissage supervisé 8
2.5 Apprentissage non supervisé 11
2.5.1 Classification non supervisée (clustering) 11
2.5.2 Recherche d'associations 13
2.6 Conclusion 13

Chapitre 3 : Fouille des motifs pertinents..... 15

3.1 Introduction.....15
3.2 Fouille des motifs fréquents 15
3.2.1 L'algorithme Apriori 16
3.2.2 L'algorithme FP-Growth :(Frequent-Pattern Growth).....17
3.3 Autres types de fouille de motifs pertinents.....19
3.3.1 Fouille des motifs périodiques 19
3.3.2 Fouille des motifs à haute utilité 20
3.3.3 Fouille des motifs séquentiels 20
3.3.4 Fouille d'épisodes 21
3.4 Problème de PPDM (Privacy Preservation Data Mining) 22
3.4.1 Dimensions de PPDM 23
3.4.2 Différentes techniques de PPDM.....25
3.4.3 L'approche que nous suivons pour le PPDM 26
3.5 Conclusion 27

Chapitre 4 : Heuristiques existantes pour résoudre le problème du PPDM 29

4.1 Introduction.....29
4.2 Approches heuristiques de nettoyage des données.....29
4.2.1 Approche "Agrégée".....30
4.2.2 Approche "Désagrégé".....34
4.3 Exemple illustratif.....38
4.4 Conclusion 40

Chapitre 5 : Nouvelle heuristique pour le PPDM 41

5.1 Introduction.....41
5.2 Position du problème.....41
5.3 Problème de couverture d'ensemble (SCP: Set Cover Problem) 43
5.4 Modélisation du problème PPDM par un problème SCP.....44
5.5 Heuristique proposée basée sur le SCP.....45
5.5.1Principe 45
5.5.2 Algorithme 45
5.6 Conclusion 50

Chapitre 6 : Implémentation et Résultats 53

6.1 Introduction 53
6.2 Configuration matérielle 53
6.3 Outil logiciel : Python 54
6.4 Résultats et discussions 54
6.4.1 La Base Mushroom 54
6.4.2 Résultats sur la base Mushroom 55
6.4.2.1. Temps d'exécution 55
6.4.2.2. Consommation de mémoire 56
6.4.2.3. Fonctions objectives (Side effects) 57
6.4.3. La BDT D1 60
6.4.3.1. Temps d'exécution 60
6.4.3.2. Consommation de mémoire 62
6.4.3.3. Fonctions objectives (Side effects) 63
6.4.4. Discussion générale 64
6.5 Conclusion 65

Chapitre 7 : Conclusion générale 67

Références 69

Liste des abréviations

K-PPV Les K Plus Proches Voisins.

KDD Knowledge Discovery in Databases.

DM Data Mining.

ECD Extraction Connaissance à partir de Données.

FP-Growth Frequent-Pattern Growth.

PFPM Periodic frequent Patterns with novel periodicity Measures.

PHM Periodic High utility itemsets Mining.

EFIM Efficient high-utility Itemset Mining.

UP-Growth An efficient algorithm for high utility Itemset mining.

GSP Generalization and performance improvement.

SPAM Sequential Pattern Mining using a bitmap representation.

EMMA Episode Mining using Memory Anchor.

TKE Top-k Episodes.

PPDM Privacy Preservation Data Mining.

BDT Basée Données Transactionnelle.

SCP cet cover problème

Liste des figures

Figure 2.1 Les étapes essentielles du KDD…..……...…...4

Figure 2.2 Tâches de la fouille de données…..……...…….……..….6

Figure 2.3 Exemple de K-NN ……..…….....9

Figure 2.4 Exemple d'arbre hiérarchique …….…….……..12

Figure 3.1 Algorithme Apriori ……17

Figure 3.2 Exemple d'application de l'algorithme Apriori ….…….....18

Figure 3.3 Les techniques de PPDM……...25

Figure 5.1 Nouvelle heuristique à base de SCP……...……42

Figure 5.2 : Exemple du problème SCP ……...……...41

Figure 6.1 Configuration matérielle ……..…..…….49

Figure 6.2 Temps d'exécution de l'heuristique agrégée et l'heuristique à base de SCP dans la base Mushroom .....….56

Figure 6.3 Consommation mémoire de l'heuristique agrégée et l'heuristique à base de SCP dans la base Mushroom ..…….....57

Figure 6.4 Missing Cost de l'heuristique agrégée et l'heuristique à base de SCP dans la base Mushroom ……58

Figure 6.5 Dissimularité de l'heuristique agrégée et l'heuristique à base de SCP dans la base Mushroom …….…….....59

Figure 6.6 Temps d'exécution de l'heuristique agrégée et l'heuristique à base de SCP dans la base D1 …….....…61

Figure 6.7 Consommation mémoire de l'heuristique agrégée et l'heuristique à base de SCP dans la base D1 …….....62

Figure 6.8 Missing Cost de l'heuristique agrégée et l'heuristique à base de SCP dans la base D1 ……..…….....……63

Figure 6.9 Dissimilarité de l'heuristique agrégée et l'heuristique à base de SCP dans la base D1 ….....….64

Liste des tableaux

Tableau 4.1 Exemple de base de données transactionnelle 34

Tableau 4.2 Itemsets fréquents non-singleton avec *minsup = 4*…….....35

Tableau 6.1: Caractéristiques de la base de données Mushroom…….....55

Tableau 6.2: Caractéristiques de la base de données D1.....60

Liste des algorithmes

Algorithme 4.1 Partie initialisation de l'heuristique Agrégée.....31

Algorithme 4.2 Boucle "While" de l'approche agrégée.....33

Algorithme 4.3 Partie Sélection.....33

Algorithme 4.4 Mise à jour.....33

Algorithme 4.5 Mise à jour du support….....34

Algorithme 4.6 Partie initialisation de l'heuristique Désagrégée.....36

Algorithme 4.7 Boucle "While" de l'heuristique Désagrégée.....37

Algorithme 4.8 Mise à jour …….38

Algorithme 5.1 Partie initialisation de l'heuristique à base de SCP.....46

Algorithme 5.2 Boucle "While" de l'approche Agrégée……...……...…...48

Algorithme 5.3 Boucle "While" de l'approche à base de SCP……..…….49

Algorithme 5.4 Mise à jour……...49

Algorithme 5.5 Mise à jour le support…….50

Chapitre 1 : Introduction Générale

1.1 Contexte

La fouille de données a émergé au milieu des années 1990 à l'intersection des statistiques et des technologies de l'information : bases de données, intelligence artificielle et apprentissage automatique. Aujourd'hui, les données collectées sont stockées dans différentes bases de données. Elles contiennent un grand nombre d'attributs, parfois plus d'un million d'enregistrements. Par conséquent, il est nécessaire de développer et/ou d'automatiser des outils efficaces pour explorer de grandes quantités d'informations afin de glaner des informations utiles. L'idée de base de l'exploration de données est d'extraire des connaissances cachées des données disponibles. Tout le travail consiste à appliquer des méthodes intelligentes pour extraire ces connaissances.

La fouille de données est un sous-processus général d'extraction de connaissances à partir de données (ECD, en anglais KDD), composé d'une variété d'algorithmes et de techniques pour mieux analyser ces énormes bases de données.

Il existe plusieurs domaines qui ont largement utilisé les techniques de fouille de données. En particulier, la fouille données préservant la vie privée (en Anglais, PPDM : Privacy Preserving data Mining) est un domaine émergent qui vise à protéger les informations personnelles des individus tout en permettant l'utilisation de leurs données à des fins d'analyse statistiques. Cette approche concerne une quantité croissante de données sur les individus, ce qui peut compromettre leur vie privée.

1.2 Objectifs

Dans ce cadre de recherche nous essayons de trouver une solution à ce problème. Pour ce faire nous utilisons d'abord l'algorithme « a priori » qui permet l'extraction des motifs fréquents. Ensuite, nous cherchons à modifier la base de données de façon aussi minimale que possible afin d'éviter que les informations sensibles soient détectées comme motifs fréquents.

Chapitre 1 : Introduction Générale

Le but de notre projet est d'étudier en détail deux méthodes heuristiques utilisées pour cacher des ensembles d'éléments sensibles dans des bases de données transactionnelles. La première méthode est une méthode déjà proposée auparavant et le deuxième est une méthode que nous proposons dans ce travail. Ensuite, nous essayons d'implémenter ces deux méthodes sur une base de données réelle.

Nous utilisons pour la validation, les bases de données : Mushroom et D1) qui sont publiées et largement utilisées dans le domaine de la fouille de motifs.

1.3 Structure du rapport

La suite de ce mémoire est organisée en cinq chapitres :

- **Le deuxième chapitre** se focalise sur les concepts et généralités sur la fouille de données et et l'ECD.
- **Le troisième chapitre** s'articule sur la fouille des motifs pertinents et le problème de PPDM.
- **Le quatrième chapitre** sera consacré à l'étude de deux différentes heuristiques pour résoudre le problème de PPDM.
- **Dand le Cinquième chapitre, nous présentons** la nouvelle heuristique que nous proposons (**SCP)**
- **Le dernier chapitre** met en évidence les étapes essentielles pour implémenter, vérifier et interpréter les résultats obtenus.

Enfin, ce mémoire se termine par une conclusion et des perspectives de travaux futurs.

Chapitre 2 : Concepts et généralités sur la fouille de données

2.1 Introduction

Les techniques de fouille de données dans le domaine de l'intelligence artificielle, sont essentielles pour traiter de grandes quantités de données et en extraire des informations cruciales pour la prise de décision. La connaissance résulte d'un ensemble de relations entre les données, telles que des règles, des phénomènes, des anomalies ou des tendances.

La fouille de données a émergé au début des années 1990 grâce à une combinaison de facteurs technologiques, économiques et sociopolitiques. Les énormes volumes de données sont devenus des mines d'informations stratégiques pour les décideurs et les utilisateurs. Cette discipline offre une multitude de techniques et d'algorithmes de fouille de données, provenant de diverses sources telles que les statistiques (régression), l'intelligence artificielle (réseaux de neurones, arbres de décision), la théorie de l'évolution (algorithmes génétiques, essaims de particules) ou l'éthologie (colonies d'abeilles, colonies de fourmis). Cette combinaison de technologies facilite la résolution, la compréhension, la modélisation et l'anticipation des problèmes. Ce chapitre se concentrera sur les concepts fondamentaux de la fouille de données.

2.2 KDD (Knowledge Discovery from Database)

2.2.1 Définition

Le processus KDD, ou Knowledge Discovery in Databases (Découverte de Connaissances dans les Bases de Données), est un processus qui utilise des méthodes de fouille de données pour extraire ce qui est considéré comme une connaissance, en respectant des mesures et des seuils spécifiques, à partir d'une base de données ainsi que de tous les éléments nécessaires. Cela implique généralement des étapes telles que le prétraitement, le sous-échantillonnage et la transformation de la base de données [1].

Chapitre 2 : Concepts et généralités sur la fouille de données

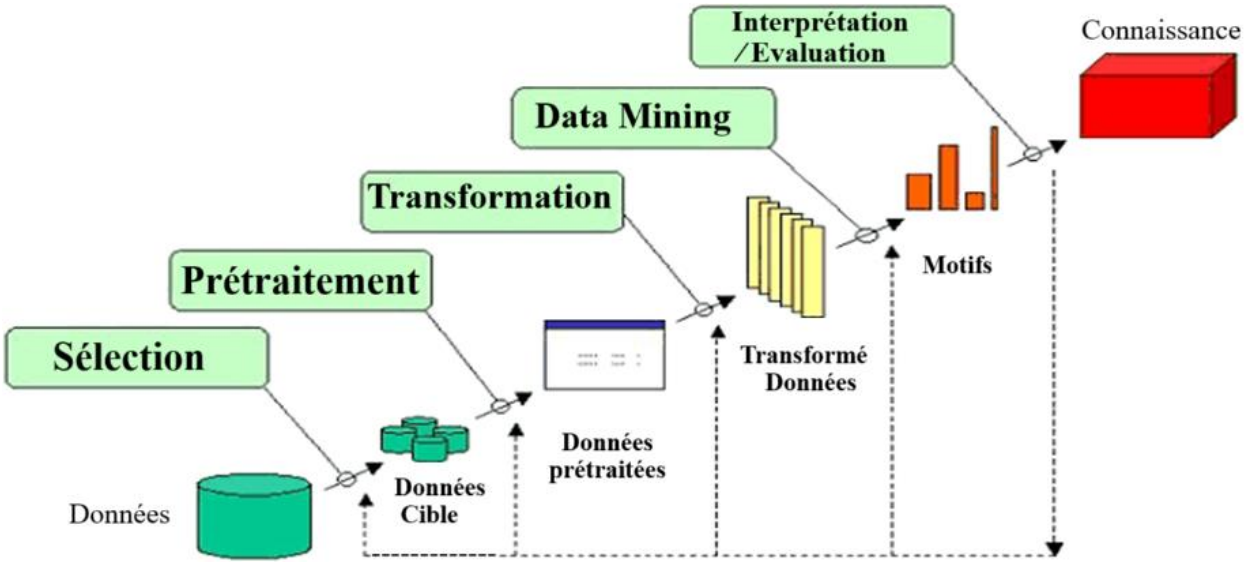

Figure 2.1: Les étapes essentielles du KDD

2.2.2 Les étapes essentielles du KDD

Cinq étapes sont envisagées, à savoir : [1]

- **Sélection.** Cette étape consiste à créer un jeu de données cible, ou à se focaliser sur un sous-ensemble de variable ou d'échantillons de données, sur lequel la découverte doit être effectuée ;
- **Prétraitement.** Cette étape consiste en un nettoyage et un prétraitement des données cibles afin d'obtenir des données cohérentes.
- **Transformation.** Cette étape consiste en la transformation des données à l'aide de méthodes de réduction ou de transformation de dimensionnalité.
- **Data Mining.** Cette étape consiste à rechercher des modèles d'intérêt sous une forme de représentation particulière, en fonction de l'objectif (souvent, la prédiction),
- **Interprétation / Evaluation :** Cette étape consiste en l'interprétation et l'évaluation des modèles minés.

Chapitre 2 : Concepts et généralités sur la fouille de données

2.2.3 Fouille de données et ECD

La fouille de données fait partie d'un processus plus global appelé extraction de connaissances à partir de données (ECD), également connu sous le nom de Knowledge Discovery from Databases (KDD) en anglais. Son objectif est de produire de nouvelles informations utiles pour les utilisateurs en utilisant des algorithmes efficaces, tout en tenant compte des contraintes imposées par le contexte des bases de données [2].

La fouille de données et l'ECD partagent l'utilisation de mégabases ou Data Warehouses (DW), des entrepôts de données orientés utilisateurs. Bien que dans l'esprit, l'ECD englobe le cycle complet de traitement des données (la fouille de données faisant plutôt référence à leur analyse statistique), leur finalité reste cependant fondamentalement identique : fournir une aide à la prise de décision aux gestionnaires à partir de vastes bases de données [3].

2.3 Fouille de données (Data Mining)

2.3.1 Définition de la fouille de données

La fouille de données, en gros, c'est extraire des informations des données stockées électroniquement, un processus automatique ou semi-automatique qui cherche des connaissances jusque-là inconnues mais utiles. Ces connaissances, sous forme de modèles dans les données, peuvent aussi servir à prédire ou répondre dans le futur.

David Hand (1998) le décrit comme la découverte de structures intéressantes, inattendues ou précieuses dans de grandes collections de données. Pour SAS-INSTITUTE, c'est un processus de sélection, exploration, modification et modélisation de grandes bases de données pour trouver des relations entre les données jusque-là inconnues. Le terme "data mining" vient du forage de données. Comme dans tout forage, le but est d'extraire quelque chose de précieux, ici la connaissance. Ces idées reposent sur l'idée qu'il y a des infos cachées dans le gisement de données, et grâce à différentes techniques, on peut les révéler. La fouille de donnéesest le fruit du succès du concept de data Warehouse, combiné à la multiplication des bases de données décisionnelles dans les entreprises, comme cité dans [4] .

Chapitre 2 : Concepts et généralités sur la fouille de données

2.3.2 Les tâches de Fouille de données

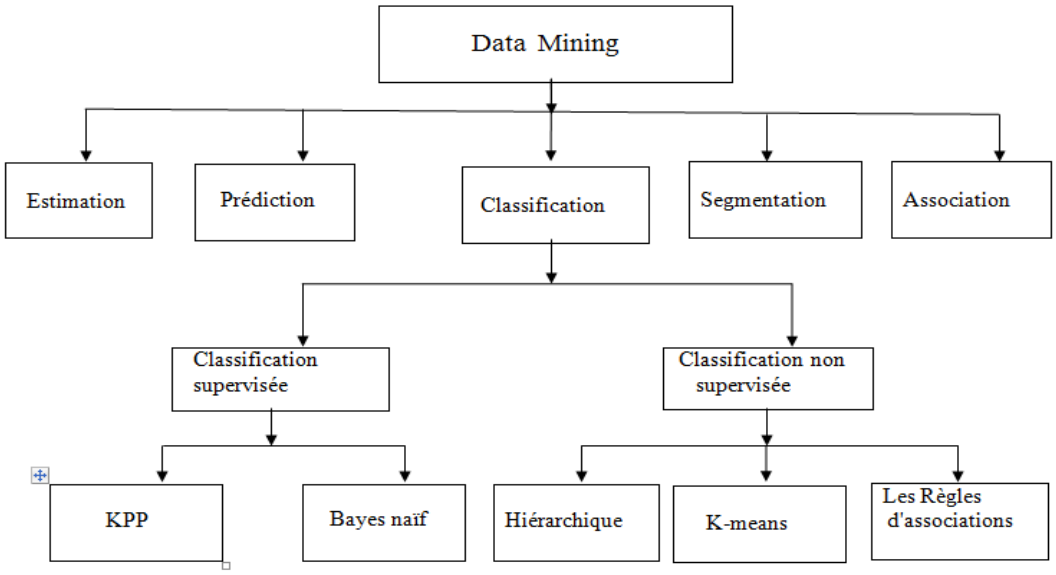

Figure2.2 Tâches de la fouille de données

La fouille de données est en fait un ensemble de techniques complémentaires spécialisées pour différentes tâches. Ces techniques sont principalement partagées entre la classification automatique (supervisée et non supervisée) et la recherche associative. Contrairement aux idées reçues, la fouille de données n'est pas une panacée capable de résoudre toutes les difficultés ou tous les besoins d'une entreprise. Cependant, un grand nombre de problèmes intellectuels, économiques ou commerciaux peuvent être classés dans l'une des tâches suivantes lorsqu'ils sont formalisés (voir Figure 2.2) :

- Classification.
- Estimation.
- Prédiction.
- Segmentation (ou clustérisassions).

- Association.

Ces tâches s'inscrivent globalement dans deux grandes catégories d'apprentissage automatique : l'apprentissage supervisé et l'apprentissage non supervisé.

2.4 Apprentissage supervisé

L'apprentissage supervisé, en gros, c'est quand une machine apprend à faire quelque chose à partir d'une bibliothèque d'exemples déjà traités. Chaque exemple dans cette bibliothèque consiste en une paire entrée-sortie. En gros, on connaît l'entrée et on veut déterminer la sortie, ce qui en fait surtout une méthode de classification des données. Mais ça fonctionne aussi dans l'autre sons : on connaît la sortie et on veut retrouver l'entrée, ce qui s'appelle la régression.

2.4.1 Tâches d'apprentissage supervisé

- **Classification**

La classification, en gros, c'est regarder les caractéristiques des nouveaux éléments pour les placer dans des catégories préexistantes. Autrement dit, c'est apprendre à classer des objets dans des catégories déjà définies en se basant sur des exemples déjà classés [5].

- **Estimation**

L'estimation est similaire à la classification, sauf que la variable de sortie est numérique plutôt que catégorique. Les estimations incluent le remplissage des valeurs manquantes dans des champs spécifiques, en fonction d'autres champs de l'enregistrement. Par exemple, nous voulons estimer la lecture de la pression artérielle systolique d'un patient dans un hôpital en fonction de l'âge, du sexe, de l'indice de masse corporelle et du taux de sodium sanguin du patient. La relation entre la pression artérielle systolique et d'autres données fournira un modèle estimé. Nous pouvons ensuite appliquer ce modèle dans d'autres contextes. Quelques exemples de l'utilisation des tâches d'estimation sont les suivants :

- ✓ Estimer le nombre d'enfants dans une famille.
- ✓ Estimant le montant d'argent qu'une famille de quatre membres choisis aléatoirement dépensera pour la rentrée scolaire.
- ✓ Estimer la valeur d'une pièce immobilière.

Souvent, la classification et l'estimation sont utilisées ensemble, comme lorsque l'exploration de données est utilisée pour prédire qui est susceptible de répondre à une offre de transfert de solde de carte de crédit et pour estimer la taille du solde (solde) à transférer.

- **Prédiction**

La prédiction ressemble à la classification et à l'estimation, mais elle concerne une échelle de temps différente. Tout comme ces tâches précédentes, elle se base sur des données passées et présentes, mais elle vise un résultat dans un futur déterminé. La qualité d'une prédiction ne peut être mesurée que par l'attente et la comparaison avec les événements réels. En outre, on parle de régression lorsque l'objectif est de prédire des valeurs de variables continues plutôt que des variables discrètes (classes). Ainsi, la régression est similaire à la prédiction, à part la nature de la variable cible. Les techniques les plus appropriées pour la prédiction incluent les arbres de décision et les réseaux de neurones

2.4.2 Quelques techniques et modèles pour l'apprentissage supervisé

- **Les k plus proches voisins (k-PPV)**

Connue en anglais sous le nom de K-nearest neighbor (K-NN), la méthode des K plus proches voisins est l'un des algorithmes les plus simples d'apprentissage automatique. Son principe général consiste à déterminer, pour chaque nouvel objet à classer, la liste des K voisins les plus proches parmi les objets déjà classés. Ensuite, l'objet est assigné à la classe majoritaire parmi ces K voisins identifiés. Pour utiliser cette méthode, il est nécessaire de choisir le nombre K et la mesure de similarité pour comparer le nouvel objet avec ceux déjà classés.

Habituellement, la distance euclidienne est employée comme mesure pour cette comparaison, comme mentionné dans [6].

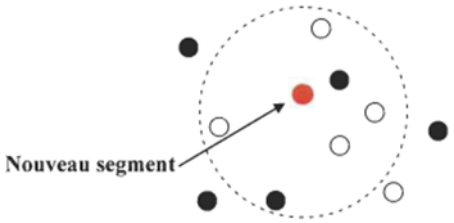

Figure 2.3: Exemple de K-NN [2].

Soit l'exemple de la Figure 2.3 où on doit classer le nouveau segment S (en rouge). Si on choisit K = 1, S sera classé cercle noir. Si K = 6, le même S sera classé cercle blanc. On voit donc que le choix de K est très important dans le résultat final.

La méthode K-NN est souvent performante, car elle utilise un apprentissage de type par cœur, l'un des types les plus simples où tous les exemples d'apprentissage sont conservés. Cependant, son inconvénient majeur réside dans le temps nécessaire pour effectuer les prédictions, car cela implique le calcul de la distance avec tous les exemples disponibles. Heureusement, il existe des heuristiques pour réduire le nombre d'exemples à prendre en compte, ce qui peut améliorer l'efficacité de l'algorithme. [7]

- **Naïve Bayes**

Naïve Bayes est un classificateur probabiliste largement utilisé dans la catégorisation et la classification. Par exemple, il est couramment employé pour classer de nouveaux documents en fonction de leur probabilité d'appartenance à chaque classe. Cette approche repose sur le théorème de Bayes et suppose une indépendance entre les caractéristiques des données [6] .

Soit le document d_j représenté par le vecteur $V_j = (V_{j1}..., V_{jk}, ..., d_{jn})$. Les variables aléatoires$V_{j1}, V_{jk}, ..., V_{jn}$ représentent les occurrences des mots retenues pour la

classification dans le document d_j. En utilisant le théorème de Bayes, la probabilité que le document d_j appartient à la classe c_i est :

$$P(c_i|v_i) = \frac{P(c_i)P(v_i|c_i)}{P(v_i)}$$

Itiration 2.1.

- **Arbres de décision**

Un **arbre de décision** est un outil qui aide à prendre des décisions en divisant un groupe d'individus en sous-groupes homogènes en fonction de critères discriminants spécifiques et connus. À chaque étape de cet arbre, les prédictions sont basées sur les données connues, et en réduisant progressivement la portée de la solution, on peut émettre des prédictions sur le problème à partir de ces données. Chaque nœud interne de l'arbre est associé à un critère discriminant des éléments à classer, permettant ainsi de les répartir de manière homogène entre les différents sous-groupes associés à ses enfants. Les branches reliant un nœud à ses enfants représentent les valeurs discriminantes pour les attributs de ce nœud. Enfin, les feuilles de l'arbre contiennent les prédictions pour les données à classer.

L'un des avantages de cette méthode est sa lisibilité pour les analystes, car elle permet de déterminer clairement les couples <attribut, valeur> discriminants à partir d'un grand nombre d'attributs et de valeurs. De plus, des algorithmes d'apprentissage automatique permettent de généraliser ces arbres de décision pour les rendre encore plus efficaces.

- **Réseaux de neurones**

Un **réseau de neurones** est un modèle informatique qui s'inspire du fonctionnement des neurones biologiques. Chaque neurone reçoit une somme pondérée de ses entrées, également

appelées synapses, et renvoie une valeur en fonction de sa fonction d'activation. Cette valeur peut être utilisée à la fois comme une entrée pour une nouvelle couche de neurones et comme un résultat interprété par l'utilisateur, comme une classe ou un résultat calculé.

Lors de la phase d'apprentissage, le réseau de neurones ajuste les poids associés à chaque synapse d'entrée, également appelés coefficients synaptiques. C'est un processus qui peut être long et qui doit être répété chaque fois que la structure de la base de données est modifiée..

2.5 Apprentissage non supervisé

L'apprentissage non supervisé concerne l'extraction de nouvelles informations et de modèles originaux à partir d'un ensemble de données, où aucun attribut n'est considéré comme plus important qu'un autre. Les résultats des algorithmes d'exploration de données non supervisés doivent être analysés pour déterminer s'ils sont utiles et pertinents, ou s'ils doivent être rejetés. Les objectifs de l'apprentissage non supervisé comprennent diverses tâches telles que la détection de motifs, la segmentation de données, la réduction de dimensionnalité et la visualisation de données., nous pouvons citer les suivants :

2.5.1 Classification non supervisée (clustering)

La classification non supervisée, également appelée clustering, consiste à rechercher des groupes ou des classes d'objets en se basant sur leurs caractéristiques. Ces groupes sont découverts automatiquement au cours de l'opération et ne sont pas prédéfinis. De plus, les classes doivent regrouper des objets ayant des caractéristiques similaires et séparer ceux ayant des caractéristiques différentes. Les algorithmes couramment utilisés pour effectuer cette tâche comprennent les méthodes de type nuées dynamiques telles que les centres mobiles (K-means), les méthodes de classification hiérarchiques et les cartes auto-organisatrices de Kohonen [8]. Nous présentons ici les deux premières approches :

- **Classification hiérarchique**

Ces méthodes construisent les classes graduellement sous une forme hiérarchique. Cette représentation hiérarchique s’appelle dendrogramme. Elles sont divisées en 2 sous-types :

Agglomération (Classification ascendante hiérarchique) et division (Classification descendante hiérarchique) [6].

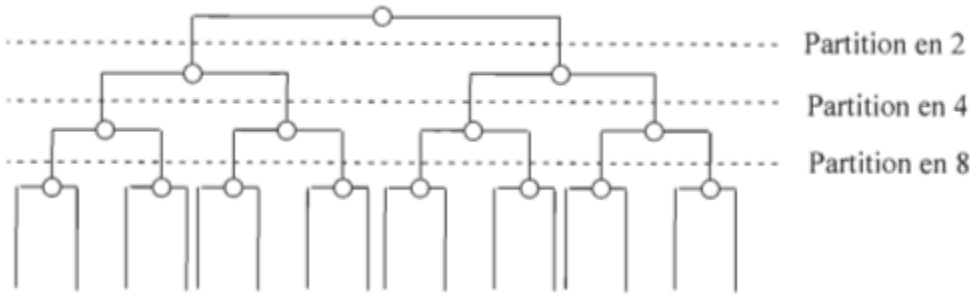

Figure 2.4: Exemple d'arbre hiérarchique [2]

- **Méthode K-Means**

L'algorithme K-means, également connu sous le nom d'algorithme des centres mobiles, est le plus célèbre et le plus largement utilisé pour le clustering. Son attrait réside dans sa simplicité de mise en œuvre, et il a été créé par McQueen en 1967. Cet algorithme divise les données en k classes, où les objets à l'intérieur de chaque classe sont aussi similaires que possible entre eux, mais aussi différents que possible des objets des autres classes. Pour commencer, l'algorithme choisit initialement les centres de classes de manière aléatoire. Le centre de gravité Bi d'une classe est calcul chaque itération comme suit :

$$g_i = \frac{1}{c_i} \sum_{j=1}^{c_i} s_j^{(i)}$$

Itiration 2.2.

2.5.2. Recherche d’associations

La fouille de motifs fréquents et de règles d'association est l'une des méthodes les plus utilisées dans le domaine du marketing et de la distribution. Elle peut être appliquée à divers secteurs d'activité pour rechercher des regroupements potentiels de produits ou de services [9], L’objectif principal de cette méthode est d'identifier des similitudes ou des associations,

notamment à travers l'analyse des paniers d'achat pour découvrir des associations entre les produits sur les tickets.

En examinant les produits achetés ensemble, cette méthode permet d'obtenir des informations sur les clients et les motivations derrière leurs achats. Elle est utile dans de nombreux secteurs, y compris la détection de fraudes en repérant des associations inhabituelles. Les règles d'association, extraites d'une base de données transactionnelle, décrivent ces associations entre éléments et permettent de mettre en évidence les produits de base et leurs compléments. Cela ouvre la voie au développement de stratégies commerciales visant à accroître les profits, telles que la promotion de ventes croisées. Ce processus est souvent désigné sous le nom de "problème de comptage d'ensembles fréquents" (FSC).

2.6 Conclusion

La fouille de données consiste à extraire des informations prédictives cachées dans de vastes bases de données. Cette technologie innovante permet aux entreprises de mettre en lumière les informations les plus cruciales de leurs entrepôts de données. Les outils d'exploration de données ont la capacité de prédire les tendances et les actions futures, ce qui permet de prendre les décisions appropriées.

Dans ce chapitre, nous avons abordé les concepts fondamentaux de la fouille de données ainsi que les principaux algorithmes de fouille de données utilisés pour des tâches telles que le

clustering, la classification et la recherche d'associations. Le chapitre suivant se concentrera spécifiquement sur la tâche de fouille de motifs pertinents à partir de bases de données

Chapitre 3 : Fouille des motifs pertinents

3.1 Introduction

Récemment, l'explosion des quantités de données disponibles dans tous les domaines a profondément modifié la façon dont nous traitons l'information, nécessitant le développement de méthodes efficaces et pertinentes pour générer des règles. Un domaine d'étude important dans ce contexte est la recherche de motifs, pour laquelle de nombreuses études ont été menées. L'extraction de motifs intéressants est une technique descriptive et l'une des plus populaires en fouille de données. Dans ce chapitre, nous explorerons les principales méthodes et algorithmes pour extraire différents types de motifs.

3.2 Fouille des motifs fréquents

La Fouille de motifs fréquents a pour objectif d'identifier des motifs qui figurent assez fréquemment dans les données. Ces motifs peuvent prendre différentes formes, comme des ensembles d'items (ou itemsets) fréquents, des sous-séquences (également appelées motifs séquentiels) fréquentes, ou d'autres types de sous-structures telles que des graphes ou des arbres. Par exemple, dans une base de transactions, un itemset réquent fait référence à une collection d'items qui apparaissent régulièrement ensemble dans plusieurs transactions [10]. Le domaine de l'extraction des motifs fréquents (FIM) est un domaine de recherche très actif où de nombreux algorithmes ont été proposés, tels que : FP-Growth, H-Mine, Apriori, FIN, etc. parmi ces algorithmes, nous allons présenter deux différents algorithmes d'extraction d'itemsets fréquents et de règles d'association : l'algorithme Apriori et l'algorithme Fp-Growth.

Les techniques d'extraction d'itemsets fréquents et de règles d'association ont été initialement introduites par R. Agrawal [11] dans le but de découvrir des tendances implicites entre les données dans des bases de données dites "transactionnelles". Plus précisément, dans une base de données contenant un ensemble d'articles, les règles d'association visent à identifier les articles ou les items qui sont fréquemment achetés ensemble.

Les règles d'association ont été explorées et appliquées dans divers contextes et domaines d'application. Leur utilisation s'étend à plusieurs domaines, tels que l'aide à la planification commerciale, le diagnostic et la recherche médicale, l'amélioration des processus de

télécommunications, l'organisation et l'accès aux sites Internet, ainsi que l'analyse de données spatiales et statistiques [12].

L'extraction des règles d'association implique l'identification des règles dont le support et la confiance sont respectivement supérieurs ou égaux à des seuils minimaux prédéfinis par l'utilisateur, appelés le seuil minimal de support (minsup) et le seuil minimal de confiance (minconf) [13].

3.2.1 L'algorithme Apriori

Apriori est un algorithme bien connu pour la recherche de règles d'association dans les bases de données transactionnelles. Il se base sur deux concepts clés, le support et la confiance, pour évaluer la pertinence des règles. Chaque règle doit avoir un support supérieur à un seuil minimal (minsup) et une confiance supérieure à un seuil minimal (minconf), définis empiriquement par l'utilisateur [13].

L'algorithme commence par identifier les produits les plus fréquents dans la base de données, en respectant les seuils de support. Ensuite, il génère des ensembles de règles candidats à partir de cette liste. Ces candidats sont testés sur la base de données pour vérifier leur occurrence, et ceux qui ne satisfont pas les critères de support et de confiance sont éliminés. Ce processus se répète en augmentant progressivement la dimension des candidats jusqu'à ce que toutes les règles pertinentes soient découvertes. Enfin, les ensembles de règles découvertes sont fusionnés [13].

La génération des candidats se fait en deux étapes :

1. La jointure
2. L'élagage

La **jointure** consiste en une jointure d'un ensemble de règles à *k-1* éléments sur lui-même qui aboutit à la génération d'un ensemble de candidats à *k* élément. Enfin, l'élagage supprime les candidats dont au moins une des sous-chaines k-1 éléments n'est pas présente dans l'ensemble des règles à *k-1* éléments. Cet algorithme est très performant, mais souffre si les ensembles d'éléments fréquents sont trop grands. De plus, scanner la base de données à la recherche d'un

motif de façon répété devient rapidement un frein aux performances sur de grosses bases de données.

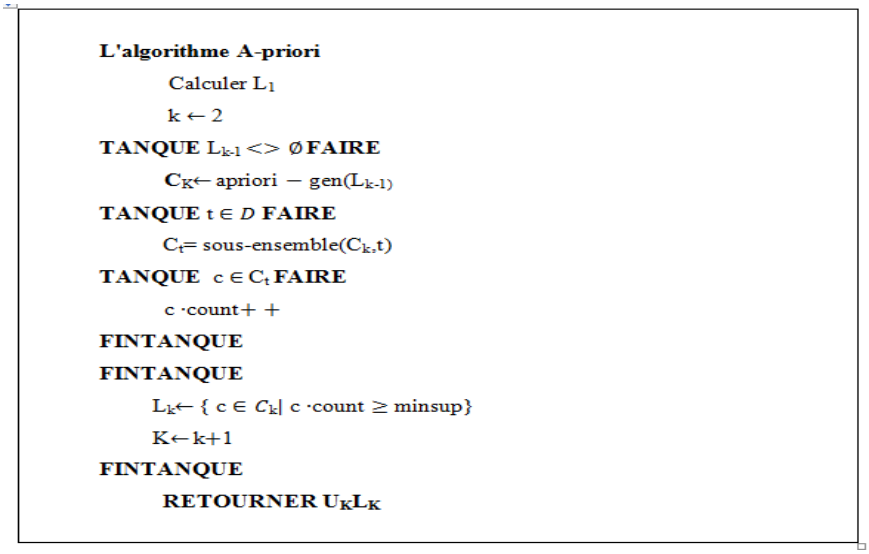

Figure 3.1: Algorithme Apriori. [13]

On notera que la donnée C_k (ainsi que L_k) est un ensemble d'enregistrements contenant deux champs :

- **Itemset:** ce champ contient le sous ensemble d'items.
- **Count :** ce champ contient la fréquence de cet ensemble dans la base de transactions.

Exemple. La figure 3.2. illuse un exemple de fonctionnement de l'algorithme Apriori.

L'inconvénient majeur de cet algorithme est le nombre considérable de l'accès à la base de données D. Une amélioration qui consiste à intégrer les identificateurs des transactions est proposée dans la méthode suivante.

3.2.2 L'algorithme FP-Growth :(Frequent-Pattern Growth)

FP-Growth représente une avancée significative par rapport aux autres algorithmes de recherche de règles d'association, en particulier par rapport à Apriori [14]. L'algorithme utilise une structure de données compacte appelée arbre de motifs fréquents (Fréquent-Pattern tree), qui résout efficacement le problème de la fouille de motifs fréquents dans de grandes bases de données transactionnelles.

Chapitre 3 : Fouille des motifs pertinents

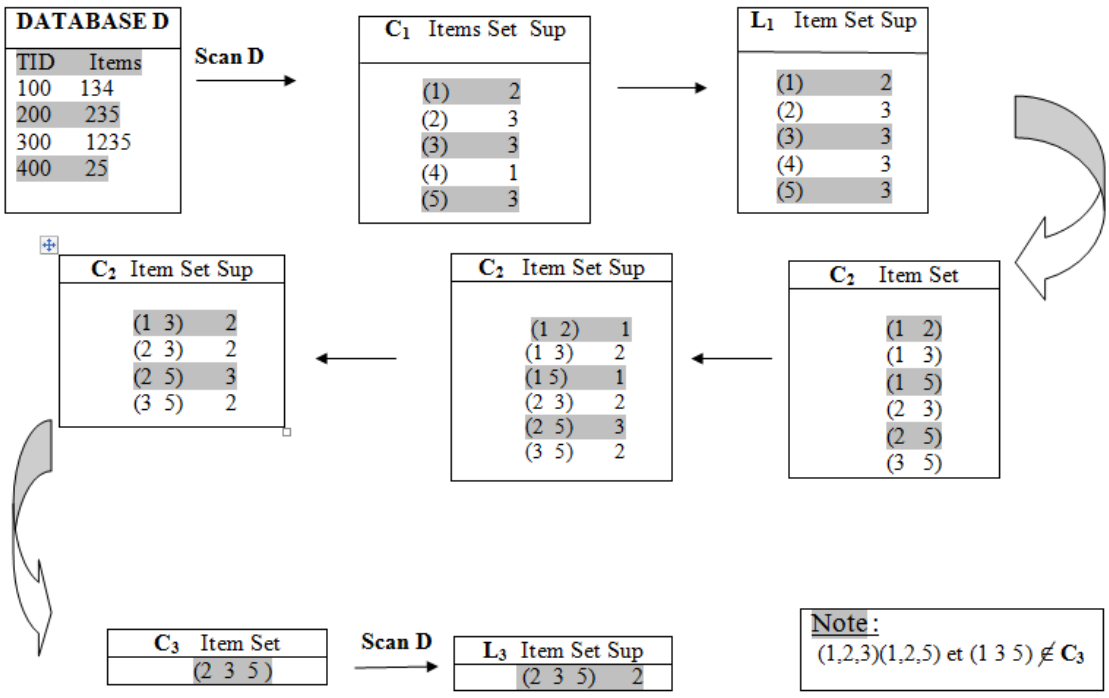

Figure 3.2: Exemple d'application de l'algorithme Apriori [13].

En stockant tous les éléments fréquents de la base de transactions dans une structure compacte, FP-Growth élimine la nécessité de parcourir répétitivement la base de transactions. De plus, en triant les éléments dans cette structure, la recherche des motifs est accélérée.

Un arbre FP est constitué d'une racine nulle et d'une série de nœuds, chacun étant préfixé par un élément représenté. Chaque nœud contient le nom de l'élément, le nombre d'occurrences de transactions où cet élément apparaît jusqu'à ce nœud, ainsi qu'un lien inter-nœuds vers d'autres occurrences du même élément dans d'autres séquences de transactions. Une table d'en-tête pointe sur la première occurrence de chaque élément.

L'avantage de cette représentation des données est qu'il suffit de suivre les liens inter-nœuds pour découvrir toutes les associations fréquentes où l'élément fréquent apparaît [14].

Chapitre 3 : Fouille des motifs pertinents

3.3 Autres types de fouille de motifs pertinents

Lorsqu'on explore des données qui ne sont pas structurées de manière transactionnelle ou lorsque l'on prend en compte d'autres critères de pertinence des motifs en dehors de la fréquence, cela nous conduit à examiner d'autres variantes d'algorithmes pour la recherche de motifs et de règles d'association. Dans ce qui suit nous résumons ces principales variantes :

3.3.1 Fouille des motifs périodiques

La fouille des motifs périodiques vise à détecter les motifs qui se répètent à intervalles réguliers dans les transactions. En général, les algorithmes d'extraction de motifs périodiques écartent un motif, le considérant comme non périodique s'il dépasse un seuil de périodicité maximum défini par l'utilisateur. Voici deux algorithmes de recherche de motifs périodiques :

- **L'algorithme PFP**

PFPM (periodicf requent patterns with Novell periodicity measures) est un algorithme permettant de découvrir les motifs périodiques fréquents dans une base de données transactionnelle. Il a été proposé par Fournier-Viger et al. (2016). PFPM peut découvrir des motifs qui apparaissent périodiquement dans une séquence de transactions. L'exploration périodique de modèles a de nombreuses applications telles que la découverte du comportement périodique des clients, et l'identification d'événements récurrents.

- **L'algorithme PHM**

PHM (periodic High utility itemsets mining) est un algorithme permettant de découvrir des ensembles d'éléments périodiques à haute utilité dans une séquence de transactions (une base de données transactionnelle) contenant des informations sur l'utilité des éléments. Il a été proposé par Fournier-Viger et al (2016). PHM peut découvrir des motifs qui apparaissent périodiquement dans une séquence de transactions et qui génèrent un profit élevé (ont une grande utilité). D'après les résultats expérimentaux, on constate que l'algorithme PHM est efficace et peut filtrer une grande quantité d'éléments atypiques pour ne révéler que les groupes d'éléments périodiques d'intérêt élevé souhaités [15].

3.3.2 Fouille des motifs à haute utilité

L'identification de motifs de haute utilité vise à mettre en avant les motifs qui ont une valeur élevée (prix, profit, etc.) plutôt que ceux qui ont une fréquence élevée. L'extraction des itemsets à haute utilité (HUIM) à partir de bases de données constitue un domaine émergent dans le domaine de l'exploration de données, axé sur la découverte des motifs dont l'utilité dépasse un seuil minimum défini par l'utilisateur [8]. Cependant, cette tâche reste chronophage en termes de calcul, tant en temps d'exécution qu'en consommation de mémoire. Par conséquent, concevoir des algorithmes plus efficaces pour cette tâche représente un défi majeur. Nous présentons ici deux algorithmes récents dans ce domaine : EFIM et Up Growth.

- **L'algorithme EFIM**

L'algorithme EFIM (Efficient High-utility Itemset Mining) est un algorithme monophasé, qui introduit plusieurs idées novatrices et utilise une approche de croissance par modèle pour trouver les motifs à haute utilité. Le processus principal est de trouver les éléments candidats en balayant l'ensemble de données, puis génère itérativement de nouveaux éléments candidats en balayant l'ensemble de données local de chaque candidat.

- **L'algorithme UP-Growth**

UP-Growth (UPG) ou (an efficient algorithme for High utility Itemset mining) est l'un des algorithmes efficaces pour générer des motifs à haute utilité fonction de la construction d'un arbre UP global. Le cadre d'UP-Tree suit deux étapes [7] :

- ✓ Construire l'UP-Tree.
- ✓ Générer des PHUI à partir d'UP-Tree.

3.3.3 Fouille des motifs séquentiels

L'exploration de modèles séquentiels (SPM) consiste à extraire des modèles séquentiels particuliers dont le support dépasse un seuil de support minimum prédéfini. Cette exploration permet également d'extraire des séquences à partir de bases de données séquentielles, reflétant ainsi les comportements les plus courants, qui peuvent être interprétés comme une connaissance du domaine à diverses fins. Pour réduire un grand nombre de séquences aux motifs de séquences

les plus pertinents et répondre aux différents besoins des utilisateurs, plusieurs algorithmes ont été proposés, parmi lesquels figurent l'algorithme GSP et l'algorithme SPAM.

- **L'algorithme GSP**

GSP (Généralisation and performance impovement) est l'un des premiers algorithmes de découverte de motifs séquentiels dans les bases de séquences. Il a été proposé dans Sikrant et al. (1992). Il utilise une approche de type Apriori pour découvrir des modèles séquentiels. GSP et défini par deux paramètres : min − gap et max − gap. Le paramètre min − gap spécifie la borne inférieure de temps écoulé requis pour que les transactions d'une séquence soient considérées comme valides, alors que le paramètre max−gap spécifie la borne supérieure de temps écoulé.

L'algorithme GSP se distingue de l'algorithme Apriori sur deux aspects : la génération des candidats et le calcul du support.

- **L'algorithme SPAM**

SPAM (Sequential pattern mining using a bitmap représentation) est un algorithme pour découvrir des modèles séquentiels fréquents dans une base de séquences. Il a été proposé dans Ayres (2002). Cet algorithme est la première stratégie de recherche pour l'extraction de modèles séquentiels [16].

SPAM utilise deux techniques d'élagage :

1- Elagage en S-step

2- Elagage en I-step

Ces techniques sont basées sur l'heuristique Apriori pour minimiser la taille des éléments candidats [17].

3.3.4 Fouille d'épisodes

La découverte d'épisodes est un cadre largement utilisé dans les tâches d'exploration de données et trouve de nombreuses applications dans le monde réel. Un épisode se compose d'un ensemble d'événements partiellement ordonnés, où chaque événement est associé à un type

spécifique. On peut aussi considérer un épisode comme une sous-séquence d'événements complexes.

- **L'algorithme EMMA**

EMMA (Episode Mining using Memory Anchor) est un algorithme qui extrait l'ensemble complet d'épisodes fréquents à partir d'une séquence complexe. Dans les deux premières phases, EMMA extrait un ensemble de motifs fréquents représentant des 1-up let épisodes, associe un identifiant id à chaque 1-up let épisode, puis encode la séquence avec ces extractions de règles d'épisodes minimales dans des séquences complexes id. Les épisodes sont construits de façon incrémentale pendant une troisième phase en concaténant des ids [18].

EMMA comporte trois phases résumées dans les sous-sections suivantes [18] :

1- Extraction de motifs fréquents.

2- Encodage de la séquence de données.

3- Extraction des épisodes fréquents.

- **L'algorithme TKE**

TKE (top-k Episodes) (Fournier-Viger et al., 2020) est un algorithme permettant de découvrir les épisodes les plus fréquents dans une séquence d'événements complexe. En termes simples, cet algorithme trouve les k sous séquences qui apparaissent le plus souvent dans une longue séquence d'événements, où certains événements peuvent être simultanés, et k est un paramètre défini par l'utilisateur.

3.4 Problème de PPDM (Privacy Preservation Data Mining)

L'exploration de données vise à extraire des informations précieuses à partir de diverses sources, tandis que la protection de la vie privée dans ce domaine cherche à prévenir les fuites ou la perte de données sensibles. L'exploration de données préservant la confidentialité (PPDM) représente une nouvelle orientation de recherche dans le domaine de l'exploration de données et des bases de données statistiques. Son objectif est d'analyser les implications de la confidentialité

des données sur les algorithmes d'exploration de données. Les préoccupations principales de l'exploration de données préservant la confidentialité sont :

- Premièrement, les données brutes sensibles, telles que les identifiants, les noms et les adresses, doivent être altérées ou retirées de la base de données d'origine. Cela permet d'éviter que le destinataire des données ne puisse compromettre la vie privée d'autrui [19].
- Deuxièmement, il est également crucial d'exclure les connaissances sensibles qui peuvent être extraites d'une base de données à l'aide d'algorithmes d'exploration de données, car elles pourraient compromettre la confidentialité des données. L'objectif principal de l'exploration de données préservant la vie privée est de développer des algorithmes capables de modifier les données d'origine d'une manière qui garantit la confidentialité des données et des connaissances, même après le processus d'extraction. [19]

3.4.1 Dimensions de PPDM

Différentes techniques sont utilisées dans le cadre du PPDM. Elles peuvent être classées en fonction des six dimensions suivantes : [19]

La première dimension concerne différents scénarios d'exploration de données pour la protection de la vie privée, qui sont généralement répartis en deux catégories couramment utilisées. Premièrement, les organisations publient des ensembles de données pour l'exploration de données et autorisent un accès illimité à ces données. Dans ce cas, la modification des données est utilisée pour garantir la protection de la vie privée. Dans le second cas, les organisations ne rendent pas leurs ensembles de données publics, mais autorisent toujours les tâches d'exploration de données. La technologie de cryptage est principalement utilisée pour assurer la confidentialité.

La deuxième dimension concerne les différentes tâches d'exploration de données, influencées par la présence de divers modèles dans l'ensemble de données. Ces tâches incluent la classification, la recherche de règles d'association, l'analyse des valeurs aberrantes, le regroupement et l'analyse des tendances. Une exigence essentielle des techniques de préservation

de la vie privée est de maintenir la qualité des données pour prendre en charge toutes ces tâches potentielles d'exploration de données et d'analyse statistique.

La troisième dimension concerne la distribution des données. Certaines méthodes ont été développées pour des données centralisées, tandis que d'autres impliquent des scénarios de données distribuées. Ces scénarios de données distribuées peuvent également être classés en distribution horizontale et distribution verticale des données. La distribution horizontale se réfère à la situation où différents enregistrements de la base de données sont situés à des emplacements différents, tandis que la distribution verticale des données se réfère à la situation où toutes les valeurs des différents attributs sont situées à des endroits différents.

La quatrième dimension aborde différents types de données, principalement divisés en trois catégories : numériques, catégorielles et booléennes. Les données booléennes sont un cas particulier de données catégorielles qui prennent deux valeurs possibles, 0 et 1. Contrairement aux données catégorielles, les données numériques ont un ordre naturel inhérent. Cette distinction fondamentale entre les valeurs catégorielles et numériques nécessite des approches différentes pour la préservation de la vie privée.

La cinquième dimension concerne les différentes définitions de la vie privée en fonction du contexte. La définition de la vie privée varie selon le contexte. Dans certains cas, les valeurs des données individuelles sont considérées comme privées, tandis que dans d'autres cas, certaines règles de groupe, d'association ou de classification sont considérées comme privées. Généralement, il existe deux catégories de protection de la vie privée : la protection individuelle et la protection collective, qui visent respectivement à préserver la vie privée des individus et celle des groupes ou des ensembles de données.

La sixième dimension implique différentes méthodes de protection : Pour protéger la vie privée dans l'exploration de données, diverses méthodes sont utilisées, telles que la modification des données et le calcul multipartite sécurisé. En général, la modification des données est utilisée pour altérer les valeurs originales de la base à divulguer, assurant ainsi un niveau élevé de confidentialité. Il est essentiel que les techniques de modification des données soient alignées sur

les politiques de confidentialité adoptées par l'organisation. Ces méthodes de modification incluent l'interruption des données, l'échange de données, l'agrégation et la suppression.

3.4.2 Différentes techniques de PPDM

Différents types de techniques de PPDM sont utilisés. Elles sont principalement classées dans les catégories suivantes :

- Anonymisation.
- Réponse aléatoire.
- Condensation.
- Perturbation.
- Cryptographie et cryptographie à courbe elliptique.

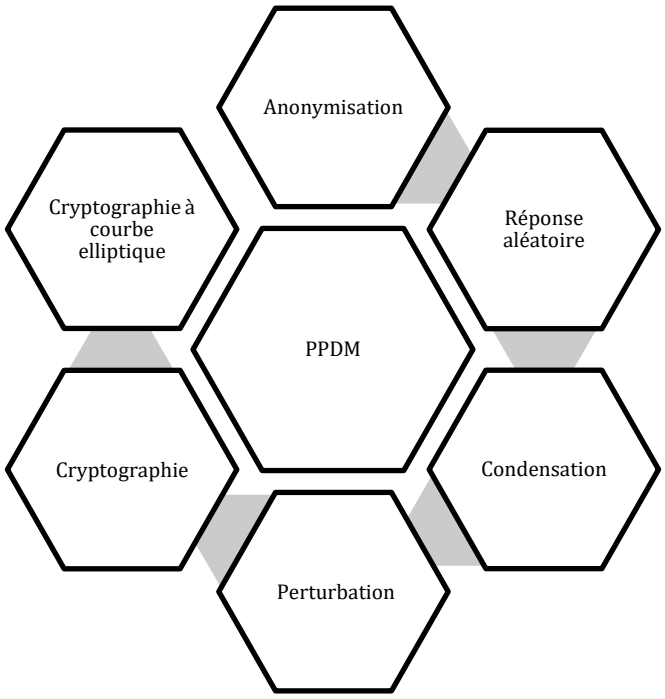

Figure3.3 : Les techniques de PPDM. [19]

Chapitre 3 : Fouille des motifs pertinents

3.4.3 L’approche que nous suivons pour le PPDM

Dans ce travail, nous visons à protéger les informations sensibles fournies en entrée lors du processus de fouille d'itemsets fréquents à partir d'une base de données transactionnelle. Plus précisément, notre objectif est d'effectuer des changements minimaux dans la base de données, tels que la suppression de transactions ou d'items, afin de réduire les supports des itemsets sensibles jusqu'à ce qu'ils ne soient plus considérés comme fréquents. Ainsi, ces itemsets sensibles sont en quelque sorte cachés car ils ne sont plus identifiés comme itemsets fréquents. La méthode proposée doit minimiser le nombre d'itemsets sensibles qui ne seront pas cachés.

En conséquence, il est possible que certains itemsets fréquents et non sensibles voient leur support diminuer en dessous du seuil minimal de fréquence, les rendant ainsi non fréquents. Ces itemsets, qui devraient rester fréquents car ne posant aucun problème de sensibilité, seront donc cachés par erreur. La méthode proposée doit donc minimiser le nombre d'itemsets non sensibles qui seront cachés.

En dernier lieu, nous préférons toujours une méthode qui atteint son objectif avec le moins de changements possibles afin de ne pas altérer profondément le contenu sémantique de la base de données. On est donc face à un problème d’optimisation multi-objectif où le but est de minimiser trois fonctions objectives (ou side effects) :

- Le nombre d’itemsets fréquents et sensitive que l’algorithme n’arrive pas à cacher (ils restent fréquents). Ce nombre est noté : *Hidding Failure.*
- Le nombre d’itemsets fréquents et non sensitive que l’algorithme va rendre non fréquent et donc va les cacher par erreur. Ce nombre est noté : *Missing Cost.*
- La quantité totale de changement effectuée sur la base d’origine. Cette quantité peut être par exemple mesurée par le nombre total d’items supprimés de la base de données.

Il y a plusieurs types d’algorithmes qui peuvent apporter des solutions à ce problème précis. On peut distinguer :

- Les algorithmes exacts qui visent à trouver des solutions exactes au problème.

Chapitre 3 : Fouille des motifs pertinents

- Les algorithmes approchés qui visent à apporter des bonnes solutions approximatives qui ne soient pas forcément les solutions optimales. On distingue ici les algorithmes basés sur des métaheuristiques (algorithmes génétiques, optimisation par essaims de particules, colonies de fourmis, etc.) et les algorithmes basés sur des heuristiques qui sont des techniques plus simples que les métaheuristiques.

C’est l’approche heuristique qui nous intéresse dans ce travail et ce sera l’objet du chapitre suivant.

3.5 Conclusion

En résumé, la fouille de motifs est une méthode d'analyse de données essentielle qui permet de découvrir des motifs fréquents au sein d'un ensemble de données. Elle trouve des applications dans de nombreux domaines pour détecter des tendances et des modèles cachés, malgré des défis tels que le traitement de vastes ensembles de données et la préservation de la confidentialité.

Cependant, avec les progrès technologiques et les outils d'analyse de données de plus en plus sophistiqués, la fouille de motifs demeure une technique puissante pour extraire des informations précieuses des données. De même, la préservation de la vie privée dans l'exploration de données est un domaine complexe nécessitant une approche équilibrée pour maximiser les avantages tout en protégeant les droits fondamentaux des individus. Les outils et techniques avancés peuvent aider à protéger les données tout en permettant leur analyse.

Le problème de la préservation de la vie privée dans l'exploration de données est un défi d'optimisation difficile, exigeant des solutions efficaces. Dans le prochain chapitre, nous examinerons deux solutions basées sur des heuristiques pour résoudre ce problème.

.

Chapitre 4 : Heuristiques existantes pour résoudre le problème du PPDM

4.1 Introduction

Le problème de la préservation de la vie privée dans la fouille de données (PPDM) est crucial étant donné la prolifération des données électroniques. Il devient de plus en plus urgent de trouver des solutions efficaces qui permettent de bénéficier des techniques modernes de fouille de données tout en garantissant la confidentialité des informations sensibles.

Dans notre étude, nous nous concentrons sur le problème PPDM dans le contexte de la fouille d'itemsets fréquents à partir de bases transactions. L'objectif est de développer des méthodes permettant de masquer les itemsets sensibles en effectuant des modifications minimales, telles que la suppression de transactions ou d'items. Ce processus de modification de la base de données pour masquer les itemsets sensibles est appelé nettoyage de données.

4.2 Approches heuristiques de nettoyage des données

Comme évoqué dans le chapitre précédent, le problème de nettoyage des données est difficile et l'une des pistes possibles pour le résoudre est d'utiliser des méthodes heuristiques. Nous présentons ici deux approches heuristiques existantes.

La première, appelée *approche agrégée (Aggregate approach)*, consiste à tenter de réduire le support des itemsets sensibles en dessous du seuil minimum (*minsup)* en supprimant certaines transactions de la base de données tandis que le nombre d'itemsets non sensibles avec un support inférieur au seuil (*minsup)* dans la base de données doit être minimisé.

La deuxième approche, appelée *approche désagrégée (Disaggregate approach)*, consiste à modifier certaines transactions en supprimant certains items de ces transactions afin que, dans la base de données nettoyée, chaque itemset sensible ait un support inférieur au seuil (*minsup)* et que le nombre d'éléments non sensibles qui ont un support inférieur au seuil soit minimisé [01].

Chapitre 4 : Heuristiques existantes pour résoudre le problème du PPDM

4.2.1 Approche "Agrégée"

Cette méthode est utilisée de manière itérative et progressive jusqu'à ce que le support de chaque itemset sensibles soit inférieur au seuil (minsup) et que le nombre d'itemsets non sensibles avec un support inférieur au seuil (minsup) soit réduit. La transaction qui sera supprimée de la base de données est déterminée en sélectionnant les transactions qui contiennent le nombre maximum d'éléments sensibles par rapport au nombre d'éléments non sensibles présents dans la transaction ce qui conduit à la diminution du support d'un nombre maximum d'itemsets sensibles et la diminution du support d'un nombre minimum d'itemsets non sensibles. La démarche est constituée des deux principales étapes suivantes:

Données en entrée

- *D* : Base de données originale (frequent, non fréquent, sensible, non sensible)
- *D'*: Base de données modifiée, initialement identique à la base de données originale
- *K* : Liste de toutes les transactions sensibles qui contiennent des itemsets sensibles
- *minsup* représente le seil minimuùm de support
- *F* est l'ensemble des itemsets sensibles et non sensibles, $F=S \cup N$. Rappelons que *F* est l'ensemble de tous les itemsets fréquents, *S* est l'ensemble des itemsets fréquents sensibles et *N* est l'ensemble des itemsets fréquents non sensibles. Initialement, on a : $Supp(j) \geq minsup$, $\forall j \in F$.

Chapitre 4 : Heuristiques existantes pour résoudre le problème du PPDM

***Etape 01** : Initialisation*

```
D' = D ;
S1=S;
Sup_S1=Sup_S;
Ns1=Ns ;
Sup_Ns=Sup_Ns1 ;
Z= [] ;
```

Algorithme 4.1. Partie initialisation de l'heuristique Agrégée.

Etape 02 :

Tanque : Il y a encore des éléments sensibles qui ont un support ≥minsup (c'est-à-dire $S1 \neq \emptyset$)
faites ce qui suit :

1. *Pour chaque transaction k dans K, faire :*

    - Déterminer le nombre *(b)* d'itemsets sensibles dans *S1* (c'est-à-dire avec un *support* ≥*minsup*) qui sont contenus dans *k*.
    - Déterminer le nombre *(a)* d'itemsets non sensibles dans *NS1* (c'est-à-dire avec un *support* ≥*minsup*) qui sont contenus dans *k*.
    - Déterminer le rapport f_k comme suit :

$$f_k = {}^{b}/_{a} \quad si\ a > 0 \quad sinon \quad f_k = b$$

Chapitre 4 : Heuristiques existantes pour résoudre le problème du PPDM

```
Tanque (S1) ± 0
  maxe=0;
   b=0;
    a=0;
   Pour i=1 jusqu'a (D) Faire :
       b=0;
        a=0;
      Pour j=1 jusqu'a (S1) Faire :
        vec=[S1{i}];
        zz= (vec ismembe D(j).sequenc);
         Si (zz==1) Alors
           Si(sup_s1(i)/(T) >= minsup)
             b=b+1;
           finSi
        finSi
      finPour
      Ecrire ('ok b');

      Pour u=1 jusqu'a (Ns1) Faire :
         vec1=[NS1{u}];
         aa= (vec1 ismember D(j). sequenc);

         Si (aa==1) Alors
            Si(sup_Ns1(u)/numel(D') >= minsup)
              a=a+1;
            FinSi
         FinSi
      FinPour
       Ecrire ('ok a') ;
```

```
        Si a>0 Alors
         fk=b/a;
         Sinon
           fk=b;
        FinSi
      Si fk > maxe Alors
         maxe = fk;
      FinSi
   FinPour
   Ecrire ('ok') ;
```

Algorithme 4.2. Boucle "While" de l'approche agrégée.

2. *Sélectionner la transaction $k_1 \in K$ telle que $f_{k1} = max \{f_k : k \in DC\}$.*

```
Fmxe=maxe ;
K1= indx ;
```

Algorithme 4.3. Partie Sélection.

3. *Mise à jour*

- Retirer la transaction k_1 de K et de D.

```
value=D(k1). index;
   Si k1==value Alors
      D(value). sequenc= [];
      D(value). index= [];
   FinSi
```

Algorithme 4.4. Mise à jour.

Chapitre 4 : Heuristiques existantes pour résoudre le problème du PPDM

- Réduire de *1* le support de chaque itemset $x \in S \cup NS$ qui est supporté par k_1 ; c'est-à-dire, *support (x)*$\leftarrow$*support(x) – 1, pour tout* $x \in S \cup NS$ *et* $x \subseteq k_1$.

```
function [sup_s1] = get_new_Sup2(setS, sup_s1, D, k1)
    Pour q1=1jusqu'a (setS) Faire
     Si (setS{q1} IsContainedIn, D{k1})
        sup_s1(q1) =sup_s1(q1) -1;
     FinSi
    FinPour
```

Algorithme 4.5. Mise à jour du support.

- Retirer chaque itemset sensible de *S* si son support est <*minsup*.
- Retirer chaque itemset non sensible de *NS* si son support <*minsup*.

4.2.2 Approche "Désagrégé"

Cette approche modifie les transactions sensibles dans la base de données individuellement, au lieu de les supprimer comme dans l'approche agrégée, en supprimant certains items jusqu'à ce que le support de chaque itemsets tombe en dessous du seuil (*minsup)* tandis que le nombre d'itemsets non sensibles dont les supports tombent en dessous du seuil est minimisé. L'approche désagrégée est similaire dans sa structure à l'approche agrégée. À chaque itération de l'approche désagrégée, une transaction sensible et un item de la transaction sont identifiés de sorte que la suppression de l'item de la transaction entraîne la maximisation du nombre d'itemsets sensibles et la minimisation du nombre d'itemsets non sensibles. La démarche se déroule en deux principales étapes :

Chapitre 4 : Heuristiques existantes pour résoudre le problème du PPDM

Données en entrée

- *D* : La base de données assainie est égale à l'original base de données *T*.
- *K* : défini toutes les transactions sensibles qui soutiennent les ensembles d'éléments sensibles.
- Pour chaque item *x* dans chaque transaction sensible *K*, ensemble $\bar{t}_{xk}=t_{xk}$ (est utilisé pour garder une trace des éléments encore inclus dans chaque transaction sensible au cours du processus d'assainissement) :

$$D'_{xk} = \begin{cases} 1 & si\ x\ \in\ I\ est\ inclus\ dans\ la\ transaction\ K\ et D' \\ 0\ sinon & \end{cases}$$

- *minsup* : le support minimal de tous itemsets sensibles et non sensibles, $j \in F=S \cup N$ (Rappelons que *F* est l'ensemble de tous les itemsets fréquents, *S1* est l'ensemble des itemsets sensibles et *NS1* est l'ensemble des itemsets non sensibles, et qu'initialement $minsup \geq s\ \forall j \in F$).
- *S1* : L'ensembles des itemsets sensibles dans *TT*, avec : *support* ≥ *minsup* est égal à l'ensemble de tous les itemsets sensibles *S*.
- *NS1* : L'ensembles des itemsets non sensibles dans *TT*, avec : *support* ≥ *minsup* est égal à l'ensemble de tous les items non sensibles *NS*.

***Etape 01* : Initialisation**

```
D' = D ;
S1=S;
Sup_S1=Sup_S;
Ns1=Ns ;
Sup_Ns=Sup_Ns1 ;
Z= [] ;
```

Algorithme 4.6: Partie initialisation de l'heuristique Désagrégée.

Etape 02

Tant que : il existe toujours des itemsets sensibles avec *support* ≥ *minsup* (c'est-à-dire $S1 \neq \emptyset$) ***faites ce qui suit*** :

1. *Pour chaque transaction k dans K (c'est-à-dire t_{xk}= 1), faire :*

   - Déterminer le nombre (b_{xk}) d'itemsets sensibles *S1*dont le support diminuera de *1* si l'item *x* est retiré de la transaction *K*.

     Déterminer le nombre (a_{xk}) d'itemsets non sensibles dans *NS1*dont le support diminuera de *1* si l'item x est retiré de la transaction *K*.

   - Déterminer le rapport f_{xk} comme suit : $f_{xk} = {}^{b_{xk}}/_{a_{xk}}$ *si* $a_{xk} > 0$ *sinon* $f_{xk} = b_{xk}$.

```
TanqQue S1 ≠ ∅
fxkmx = 0;
  Pour j = 1 jusqu'à (D) Faire
     X = D(j). Sequenc;
    Pour x = 1 jusqu'a de X
        bxk = 0;
      Pour s = 1 jusqu'à (S1) Faire
        Si (S1{s} ismember D(j). Sequenc) Alors
          Si X(x) ismember S1{s}alors,
            bxk = bxk +1;
          FinSi
       FinSi
     FinPour
```

Chapitre 4 : Heuristiques existantes pour résoudre le problème du PPDM

```
   FinPour
axk = 0;
      Pour ns = 1 jusqu'à (NS1) Faire
         Si (NS1{ns} ismember D(j). Sequenc) Alors
                              Si X(x) ismemberNS1{ns} Alors
            axk = axk +1;
          FinSi
        FinSi
    FinPour
  Si axk ≠ 0 Alors
     fxk = bxk / axk;
  Sinon
      fxk = bxk;
FinSi
    Si fxk>fxkmx Alors
         fxkmx = fxk;
        indx = D(j). index;
    FinSi
 FinPour
```

Algorithme 4.7: Boucle "While" de l'heuristique Désagrégé

2. *Sélection d'un item x_m dans la transaction $k \in K$, telle que $fxkmx=max\{ f_{xk} : x \in I, k \in K$ telle que $t_{xk}= 1\}$.*
3. *Mise à jour :*
    - Retirer l'item *xm* de la transaction *K*.

Chapitre 4 : Heuristiques existantes pour résoudre le problème du PPDM

```
seq = D(kmx). sequenc;
posi = find (seq == xm);
D(kmx). sequenc(posi) = 0;
```

Algorithme 4.8: Mise à jour.

- Réduire de 1 le support de chaque ensemble d'éléments $j \in S \cup N$ qui est supporté par K ; c'est-à-dire, $minsup \leftarrow minsup - 1$.
- Retirer chaque ensemble sensible de *S1*si son $support < minsup$.
- Retirer chaque ensemble non sensible de *NS1* si son $support < minsup$.

4.3 Exemple illustratif

L'application des deux approches d'assainissement est démontrée à l'aide d'un exemple. Le tableau (4.1) présente une base de données de 20 transactions, comprenant 12 items. Les ensembles d'itemsets fréquents obtenus à un seil de support $minsup = 4$ sont répertoriés dans le tableau (4.2). Sept de ces items - *{2, 10}, {4,11}, {2, 3, 9}, {2, 3, 10}, {2, 9, 10}, {2, 3, 4, 9} et {2,3, 9, 10}* - sont les items sensibles. L'ensemble des transactions sensibles est constitué des transactions *5, 8, 9, 12, 13, 15, 16, 17 et 19*.

Tableau 4.1: Exemple de base de données transactionnelle [20]

TID	Items	TID	Items
1	1, 11, 3	11	1, 9, 12
2	9, 4, 2	12	9, 3, 2, 11, 10, 8, 4
3	6, 1, 11	13	8, 4, 11
4	4, 12, 5	14	4, 7
5	4, 9, 11, 2, 6, 3, 10	15	3, 2, 4, 7, 6, 8, 9, 1
6	6, 2, 11	16	1, 2, 10
7	10, 12, 6	17	5, 8, 6, 9, 10, 2, 1
8	10, 1, 2, 7, 8, 9, 3, 12	18	3, 9, 7
9	12, 3, 9, 4, 2, 10	19	5, 11, 4
10	11, 7, 12, 9	20	8, 9, 11

Chapitre 4 : Heuristiques existantes pour résoudre le problème du PPDM

Les deux approches assainissent la base de données en supprimant certaines de ces transactions (pour l'approche agrégée) ou en supprimant certains éléments de ces transactions (pour l'approche désagrégé).

L'application de l'approche *agrégée* permet d'assainir la base de données de l'exemple en supprimant les transactions *5, 9, 13 et 16* de la base de données. Cela permet de cacher tous les itemsets sensibles et les itemsets non sensibles suivants : - *{1, 2}, {2, 3}, {2, 4}, {2, 6}, {3, 4}, {3, 10}, {4, 9}, {9, 10}, {9, 11}, {9, 12}, {2, 3, 4}, {2, 4, 9}, {3, 4, 9} et {3, 9, 10}* -, ce qui ne laisse que 7 itemsets non sensibles - *{1, 9}, {2, 8}, {2, 9}, {3, 9}, {7, 9}, {8, 9}, {2, 8, 9}* - pouvant être extraits de la base de données nettoyée.

L'application de l'approche *Désagréger* assainit la base de données de l'exemple en supprimant les items *9,10,11*de la transaction *5*, les items *3,10* de la transaction *8* et l'item *10* de la transaction *16*. Cela permet de cacher tous les itemsets sensibles et les itemsets non sensibles suivants - *{3, 10}, {9, 10}, {9,11}, {3, 4, 9}, et {3, 9, 10}* -, ce qui permet d'extraire les *16* autres itemsets non sensibles de la base de données nettoyée.

Tableau 4.2: Itemsets fréquents non-singleton avec *minsup = 4* [20]

Itemsets	s_j	Itemsets	s_j
1, 2	4	8, 9	5
1, 9	4	9, 10	5
2, 3	5	9, 11	4
2, 4	5	9, 12	4
2, 6	4	2, 3, 4	4
2, 8	4	**2, 3, 9**	5
2, 9	7	**2, 3,10**	4
2, 10	6	2, 4, 9	5
3, 4	4	2, 8, 9	4
3, 9	6	**2, 9, 10**	5
3, 10	4	3, 4, 9	4
4, 9	5	3, 9, 10	4
4, 11	4	**2, 3, 4, 9**	4
7, 9	4	**2, 3, 9, 10**	4

Chapitre 4 : Heuristiques existantes pour résoudre le problème du PPDM

4.4 Conclusion

Dans ce chapitre, nous avons abordé l'importance du nettoyage des données en tant que processus crucial pour assurer la protection des informations stockées dans une base de données. Nous avons présenté deux méthodes heuristiques proposées dans la littérature pour ce nettoyage. La première méthode consiste à supprimer un sous-ensemble de transactions, tandis que la deuxième méthode se concentre sur la suppression des items isolés à partir de certaines transactions sélectionnées. Ces approches visent à réduire l'exposition des informations sensibles en effectuant des modifications ciblées dans la base de données, contribuant ainsi à renforcer la confidentialité des données.

Chapitre 5 : Nouvelle heuristique pour le PPDM

5.1 Introduction

Dans ce chapitre, nous abordons une nouvelle heuristique pour résoudre le problème de Privacy Preserving Data Mining (PPDM), en se concentrant spécifiquement sur le problème du Set Cover Problem (SCP). Le PPDM est un défi majeur dans le domaine de la fouille de données, où l'objectif est de concilier l'extraction d'informations utiles avec la protection de la vie privée des individus. Le SCP, quant à lui, est un problème bien connu en optimisation combinatoire, où l'objectif est de trouver le nombre minimum de sous-ensembles nécessaires pour couvrir tous les éléments d'un ensemble donné. Dans ce chapitre, nous explorons comment appliquer une approche heuristique basé sur le SCP dans le contexte du PPDM, en proposant une méthode innovante pour résoudre ce problème complexe.

5.2 Position du problème

Dans cette section, nous examinons la position du problème en utilisant l'approche du Set Cover Problem (SCP) dans le contexte du Privacy Preserving Data Mining (PPDM), avec un accent particulier sur la méthode heuristique d'agrégation. Le PPDM cherche à concilier l'exploration des données avec la confidentialité des informations sensibles. Le SCP est une problématique bien étudiée en optimisation combinatoire, visant à déterminer le nombre minimum de sous-ensembles nécessaires pour couvrir tous les éléments d'un ensemble donné. Notre approche consiste à appliquer le SCP aux transactions sensibles victimes, détectée par la même démarche de l’approche agrégée discutée dans le chapitre précédent, dans le contexte du PPDM. L’idée est que plutôt que de supprimer toutes les transactions victimes entières, ce qui peut altérer considérablement sa structure, notre méthode vise à éliminer uniquement les items sensibles de chaque transaction affectée. La sélection des items à supprimer est effectuée à l’aide d’une modélisation du problème en termes d’un SCP. Ainsi, notre objectif est de développer une nouvelle heuristique efficace qui minimise l'impact sur la base de données tout en assurant la confidentialité des données sensibles.

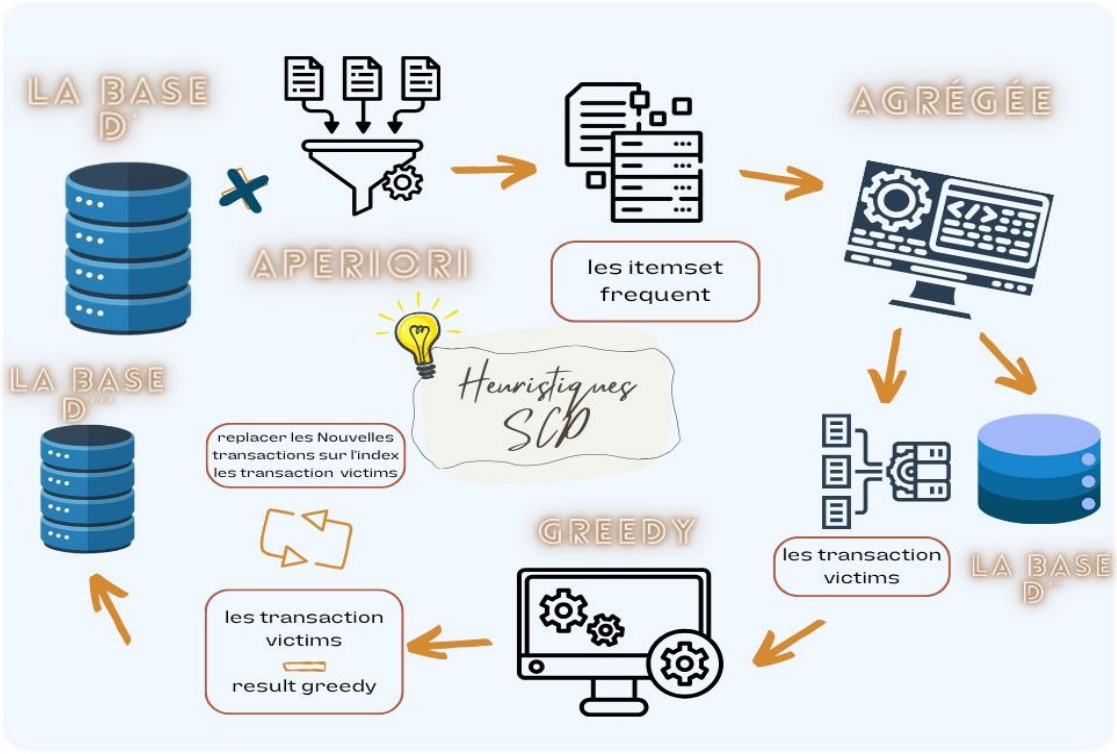

Figure 5.1 : Nouvelle heuristique à base de SCP

En général, le problème de PPDM peut être vue comme un problème d'optimisation multi-objectif, où le but est de faire un changement sur la base de données afin de cacher les itemsets sensibles tout en essayant de minimiser plusieurs objectifs :

1) **Hiding failure :** Le nombre d'itemsets fréquents et sensitive que l'algorithme n'arrive pas à cacher (ils restent fréquents).
2) **Artifical cost :** le nombre d'itemsets non fréquents qui deviennent fréquents après le processus de nettoyage.
3) **Missing Cost :** Le nombre d'itemsets fréquents et non sensitive que l'algorithme rend non fréquent et donc va les cacher par erreur.
4) **Dissimilarité :** La quantité totale de changement effectuée sur la base d'origine. Cette quantité est mesurée par le nombre total d'items supprimés de la base de données.

Chapitre 5 : Nouvelle heuristique pour le PPDM

Dans notre cas la valeur de « Haiding failure » est toujours égale à 0 car les deux heuristiques utilisées comparées assurent toutes les deux que tous les itemsets fréquents sensibles soient cachés à la fin du processus. Aussi la valeur de « Artifical cost » est toujours égale à 0 car en supprimant des transactions ou des items, le support de n'importe quel itemset ne peut que diminuer et il serait donc impossible qu'un itemset set non fréquent devienne fréquent. Nous nous limitons donc ici à minimisation des deux objectifs : « Missing Cost » et « Dissimilarité ».

5.3 Problème de couverture d'ensemble (SCP: Set Cover Problem)

Le problème SCP est un problème qui est conceptuellement très simple mais qu'il fait partie des problèmes NP-Hard. Le SCP est défini formellement comme suit :

Etant donnée un univers $U = \{u_1, u_2, \dots, u_n\}$ d'éléments et une collection S de m sous-ensembles de U : $S = \{S_1, S_2, \dots, S_m\}$ dont l'union est égale à U : $\bigcup_i S_i = U$. Le SCP consiste à trouver un nombre minimum de sous-ensemble de S qui couvrent l'univers U. Formellement, il s'agit de trouver une sous-collection $C \subseteq S$ tel que $\bigcup_{S_i \in C} S_i = U$ et $|C|$ est minimal.

Heureusement il existe un algorithme glouton simple et de complexité polynomiale qui trouve une bonne solution approximative à ce problème. Cet algorithme est itératif et à chaque itération il choisit le sous-ensemble de *S* (parmi ceux qui ne sont pas encore été utilisés) qui couvre le maximum d'éléments de *U* non encore couverts. L'algorithme continue jusqu'à ce que tous les éléments de *U* soient couverts.

Exemple. Considérons une instance du problème SCP (Voir Figure 5.2) où *U = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}* et *S = { S_1={1, 8}, S_2={2, 3, 9, 10}, S_3={4, 5, 6, 7, 11, 12, 13, 14}, S_4={1, 2, 3, 4, 5, 6, 7}, S_5={8, 9, 10, 11, 12, 13, 14}}.*

Chapitre 5 : Nouvelle heuristique pour le PPDM

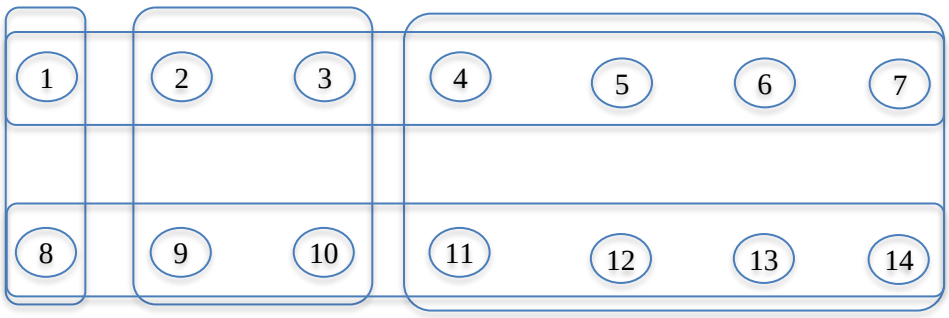

Figure 5.2 : Exemple du problème SCP

L'application de l'algorithme glouton permet de sélectionner S_3 à la première itération puis S_2 à la deuxième itération et enfin S_1 à la troisième itération. La solution retournée est donc les sous-ensembles S_1, S_2 *et* S_3 ayant un coût égal à *3*. Notons que pur cet exemple, l'algorithme glouton ne trouve pas la solution optimale qui est formée des deux sous-ensemble S_4 et S_4 et ayant un coût de *2*.

5.4 Modélisation du problème PPDM par un problème SCP

Comme évoqué plus haut, notre approche consiste à utiliser le SCP comme moyen de détecter les items à supprimer d'une transaction victime donnée afin de cacher les itemstes sensibles q'elle contient. La détection des transitions victimes se fait en suivant la même démarche utilisée par l'heuristique agrégée. Cependant, au lieu de supprimer directement ces transactions, notre méthode est de juste enlever de chacune de ces transactions un sous-ensemble minimal d'items. Le choix de ces items se fait justement en faisant appel à une instance du SCP. L'idées est que la suppression d'un item *i* d'une transaction *t*, permet de supprimer tous itemsets sensibles de cette transaction qui contiennent *i*. Le problème est donc de déterminer un ensemble minimal d'items à supprimer afin que tous itemsets sensibles de transaction soient caché. Il s'agit exactement d'un SCP en suivant la modélisation suivante :

Etant donnée une transaction victime *t* et considérons que la transaction *t* contient *n* itemsets sensibles nommées $s_1, \ldots, s_n$. Soit $It = \{i_1, \ldots, i_m\}$ l'ensemble de tous les items figurant dans au

Chapitre 5 : Nouvelle heuristique pour le PPDM

moins un itemset sensible de t. Pour chaque item $i \in It$ on définit l'ensemble Si des itemsets sensibles de *t* contenant *i* : $S_i = \{s_i | i \in s_i\}$. On définit alors une instance du SCP comme suit :

- L'univers du problème est $U = \{s_1, s_{21}, \dots, s_n\}$.
- La collection S est définit par : $S = \{S_i | i \in It\}$. Ainsi, chaque item *i* figurant dans au moins un itemset sensible donne naissance à un sous-ensemble S_i de la collection S.

Il est clair que la solution du problème consiste à trouver un nombre minimal de sous-ensembles de S qui couvrent U, autrement dit un nombre minimal d'items à supprimer de la transaction afin que celle-ci ne contienne plus d'itemset sensible.

5.5 Heuristique proposée basée sur le SCP

5.5.1. Principe

Utiliser la méthode aggrégée (voir chapitre 04) pour détecter l'ensemble des transactions victimes à modifier, ensuite sur chaque transaction détecter les items à enlever en faisant appel à une instance du SCP comme expliqué dans la Section 5.4 . La résolution du SCP qui en résulte est effectuée à l'aide de l'algorithme glouton approximatif expliqué dans la Section 5.3.

5.5.2. Algorithme

Données en entrée.

- *D* : Base de données originale (frequent, non fréquent, sensible, non sensible)
- *D'*: Base de données modifiée, initialement identique à la base de données originale
- *K* : Liste de toutes les transactions sensibles qui contiennent des itemsets sensibles
- *minsup* représente le seil minimuùm de support
- *F* est l'ensemble des itemsets sensibles et non sensibles, $F=S \cup N$. Rappelons que *F* est l'ensemble de tous les itemsets fréquents, *S* est l'ensemble des itemsets fréquents sensibles

Chapitre 5 : Nouvelle heuristique pour le PPDM

et *N* est l'ensemble des itemsets fréquents non sensibles. Initialement, on a : *Supp(j)* ≥ *minsup* , ∀j ∈ F.

Etape 01 : Initialisation.

```
D = D' ;
S1=S;
Sup_S1=Sup_S;
Ns1=Ns ;
Sup_Ns=Sup_Ns1 ;
Z= [] ;
```

Algorithme 5.1: Partie initialisation de l'heuristique à base de SCP.

Etape 02 :

Tanque : Il y a encore des éléments sensibles qui ont un support ≥ *minsup* (c'est-à-dire *S1*≠ ∅) ***faites ce qui suit*** :

1. ***Pour chaque transaction k dans K, faire :***
   - Déterminer le nombre *(b)* d'itemsets sensibles dans *S1* (c'est-à-dire avec un *support* ≥ *minsup*) qui sont contenus dans *k*.
   - Déterminer le nombre *(a)* d'itemsets non sensibles dans *NS1* (c'est-à-dire avec un *support* ≥ *minsup*) qui sont contenus dans *k*.
   - Déterminer le rapport f_k comme suit :

$$f_k = {}^{b}/_{a} \quad si\ a > 0 \quad sinon \quad f_k = b$$

Chapitre 5 : Nouvelle heuristique pour le PPDM

```
Tanque (S1) ± 0
  maxe=0;
   b=0;
    a=0;
   Pour i=1 jusqu'a (D) Faire :
       b=0;
        a=0;
      Pour j=1 jusqu'a (S1) Faire :
        vec=[S1{i}];
        zz= (vec ismembe D(j).sequenc);
         Si (zz==1) Alors
           Si(sup_s1(i)/(T) >= minsup)
             b=b+1;
           finSi
        finSi
      finPour
      Ecrire ('ok b');

      Pour u=1 jusqu'a (Ns1) Faire :
        vec1=[NS1{u}];
        aa= (vec1 ismember D(j). sequenc);
        Si (aa==1) Alors
           Si(sup_Ns1(u)/numel(D') >= minsup)
             a=a+1;
           FinSi
        FinSi
      FinPour
       Ecrire ('ok a') ;
        Si a>0 Alors
         fk=b/a;
         Sinon
           fk=b;
        FinSi
      Si fk > maxe Alors
```

Chapitre 5 : Nouvelle heuristique pour le PPDM

```
        maxe = fk;
      FinSi
    FinPour

Ecrire ('ok') ;
```

Algorithme 5.2: Boucle "While" de l'approche Agrégée

Etape 03 : ***Application de l'algorithme glouton***

1. Pour chaque transaction k_1 dans K, faire

- Construire l'univers *X* et la collection $F = \{F_{i_1}, F_{i_2}, \dots, F_{i_m}\}$ (voir Section 5.4.)
- Initialiser l'ensemble résultats (items à supprimer) R *à* ϕ
- **Tanque :** Il y a encore des éléments dans *X* non couverts (i.e., $X \neq \emptyset$) ***faites ce qui suit*** :
  - Déterminer l'ensemble F_{i_j} qui maximise le nombre $\left|F_{i_j} \cap X\right|$
  - L'indice i_j représente un item à supprimer.
  - Ajouter i_j à R : $R \leftarrow R \cup \{i_j\}$
  - Enlever les éléments couverts de X : $X \leftarrow X - F_{i_j}$
  - Enlever F_{i_j} Enlever de F :: $F \leftarrow F - \{F_{i_j}\}$
- *TF* indique la transaction filtrée (qui résulte de k1 après suppression des éléments de R)

Chapitre 5 : Nouvelle heuristique pour le PPDM

```
X= maxe ;
K1= indx ;
R=[];
Tanque (X) ± 0
  maxe=0;
   m=0;
    n=0;
   Pour i=1 jusqu'à :numel (F1{i}) Faire :
       C= numel (F1{i} intersection X);
        a=0;
      Si (C>m) Alors:
        C = M;
        n=i;
       finSi
   finPour
  R=R union F{i} ;
  TF=X.sequenc intersection R ;
```

Algorithme 5.3: Boucle "While" de l'approche à base de SCP

2. Mise à jour :

- Remplacer la transaction k_1 de *K* et de *D* par *TF*.

```
value=D(k1). index;
   Si k1==value Alors
      D(value). sequenc= TF;
      D(value). index= TF;
   FinSi
```

Algorithme 5.4: Mise à jour.

Chapitre 5 : Nouvelle heuristique pour le PPDM

- Réduire de *1* le support de chaque itemset $X \in S \cup NS$ qui est supporté par k_1 ; c'est-à-dire, *support (X)←support(x) – 1, pour tout* $X \in S \cup NS$ *et* $X \subseteq k_1$.

```
function [sup_s1] = get_new_Sup2(setS, sup_s1, D, k1)
    Pour q1=1jusqu'a (setS) Faire
      Si (setS{q1} IsContainedIn, D{k1})
         sup_s1(q1) =sup_s1(q1) -1;
      FinSi
    FinPour
```

Algorithme 5.5: Mise à jour du support.

- Retirer chaque ensemble sensible de S si son support devient est < minsup.
- Retirer chaque ensemble non sensible de *NS* si son support devient < *minsup*.

5.6 Conclusion

Dans ce chapitre, nous avons exploré une nouvelle heuristique pour le Privacy Preserving Data Mining (PPDM) en utilisant le Set Cover Problem (SCP). Le PPDM vise à extraire des informations utiles tout en protégeant la vie privée des individus, et le SCP, bien connu en optimisation combinatoire, aide à minimiser le nombre ditems à retirer pour cacher les itemsets sensibles.

Notre méthode se concentre sur la suppression des items sensibles dans les transactions plutôt que de supprimer les transactions entières. Cela préserve l'intégrité de la base de données tout en protégeant les informations privées. L'utilisation d'algorithmes heuristiques permet une gestion efficace des ressources en termes de temps et de mémoire, en appliquant des modifications ciblées et minimales.

Cette approche équilibre efficacement la protection de la confidentialité et la conservation de la qualité des données, répondant ainsi aux exigences des analyses de données sans compromettre la vie privée. En intégrant des techniques comme le SCP, notre heuristique représente une avancée significative dans le domaine du Privacy Preserving Data Mining.

Chapitre 6 : Implémentation et Résultats

6.1 Introduction

Il est important de valider notre travail avec une discussion de résultats obtenus à partir d'une étude expérimentale sur des données. C'est l'objectif de ce chapitre dans lequel nous présentons les résultats que nous avons obtenus en appliquant et en comparant les deux algorithmes discutés dans le chapitre 04 et 05 : l'algorithme " *agrégée* " et l'algorithme "*SCP*". Les expérimentations ont été réalisées sur deux bases de données transactionnelles disponibles sur le net: la base Mushroom et la base D1.

6.2 Configuration matérielle

Processor	Intel(R) Core(TM) i5-8350U CPU @ 1.70GHz 1.90 GHz
Installed RAM	8.00 GB (7.86 GB usable)
Device ID	CCB96BEA-6732-4A6B-979D-3F94D95C845C
Product ID	00330-52197-83275-AAOEM
System type	64-bit operating system, x64-based processor

Spécifications de l'appareil

Nom de l'appareil	DESKTOP-G698KH9
Processeur	AMD 3020e with Radeon Graphics 1.20 GHz
Mémoire RAM installée	8,00 Go (5,88 Go utilisable)
ID de périphérique	B51499CD-2378-4CC0-A81A-048E205260A8
ID de produit	00331-10000-00001-AA499
Type du système	Système d'exploitation 64 bits, processeur x64
Stylet et fonction tactile	La fonctionnalité d'entrée tactile ou avec un stylet n'est pas disponible sur cet écran

Figure 6.1 : Configuration matérielle

Chapitre 6 : Implémentation et Résultats

6.3 Outil logiciel : Python

Python est un langage de programmation populaire et polyvalent, apprécié pour sa simplicité, sa lisibilité et sa polyvalence. Conçu pour être facile à apprendre et à utiliser, il est largement utilisé dans divers domaines, de l'automatisation des tâches quotidiennes au développement d'applications web et à l'analyse de données avancée.

L'une des caractéristiques les plus attrayantes de Python est sa syntaxe claire et concise, qui ressemble presque à de l'anglais courant. Cela rend le code Python facile à comprendre même pour les débutants en programmation. De plus, Python dispose d'une vaste bibliothèque standard et d'une communauté active de développeurs qui fournissent des modules supplémentaires pour une gamme étendue de tâches.

Que ce soit pour les débutants ou pour les développeurs expérimentés, Python offre un environnement accueillant et puissant pour transformer vos idées en réalité.

.

6.4 Résultats et discussions

6.4.1 La Base Mushroom

Mushroom comporte un ensemble de données sur les champignons (Mushroom) fourni par l'Université de Californie, Irvine. Elle a été utilisée pour classer la toxicité des champignons qui peuvent être divisés en deux classes presque équilibrées : toxique (48%) et non toxique (52%). Cet ensemble de données comprend des descriptions hypothétiques d'échantillons de 23 espèces de champignons ramifiés dans le Field Guide to North American Mushrooms de l'Audubon Society. Famille des champignons Agaricus et Lepiota (1981). L'ensemble de données comprend 8416 observations avec 23 colonnes. Cette analyse vise à étudier les données de la colonne 2 à la

colonne 23 et à trouver la pondération de chaque attribut. L'ensemble de données est une entrée à caractère unique ayant une signification unique [21].

Tableau 6.1: Caractéristiques de la base de données Mushroom

Database name	**#of items**	**#of transactions**	**Avg.Trns.Length**	**Support**	**# Of Items**
Mushroom	119	8124	23.0	2000	6578

6.4.2 Résultats sur la base Mushroom

Nous avons appliqué nos deux heuristiques sur la base Mushroom. Dans ce qui suit nous présentons et discutons les résultats obtenus en termes de temps d'exécution, de consommation de mémoire et des objectifs à minimiser à savoir le missing cost (nombre d'itemsets non sensibles cachés par erreur) et la dissimilarité. Dans les expériences suivantes, nous avons fixé la valeur de *minsup* à 0,8 (comme la base Mushroom est une base dense, le support doit être élevé).

6.4.2.1. Temps d'exécution

Les courbes de la Figure 6.4 montrent la variation du temps d'exécution en fonction du pourcentage d'itemsets sensibles parmi les fréquents, pour l'heuristique *agrégée* et notre heuristique *à base de SCP*.

On remarque que le temps pris par l'heuristique *aggrégée* est inférieur à celui de l'heuristique SCP. Ceci est tout à fait attendu et dû au fait que l'heuristique à base de SCP reprend pratiquement la même démarche de l'heuristique agrégée jusqu'à la détection des transactions victimes. Ensuite, l'heuristique agrégée supprime simplement ces transactions alors que notre heuristique entame une nouvelle étape où un SCP est résolu pour chaque transaction afin de détecter les items à supprimer. Notre heuristique effectue donc une tâche plus fine qui nécessite

naturellement plus de temps. Ceci dit, la différence entre les deux heuristiques n'est pas très significative et elle diminue lorsque le pourcentage d'itemsets sensibles augmente.

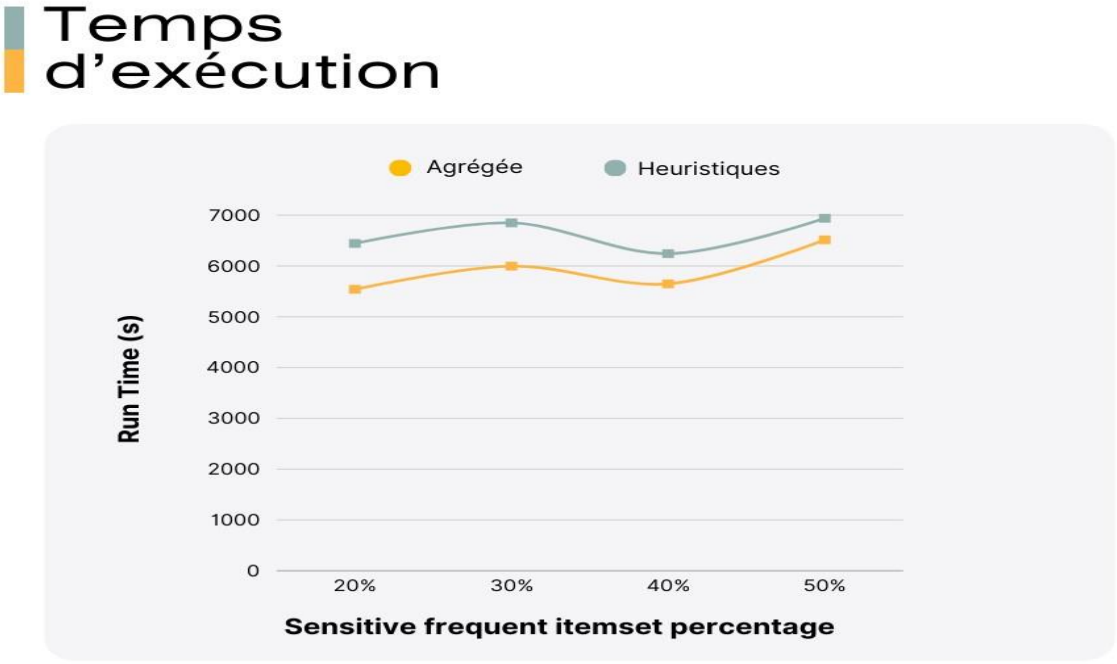

Figure 6.2: Temps d'exécution de l'heuristique agrégée et l'heuristique à base de SCP dans la base Mushroom

6.4.2.2. Consommation de mémoire

Dans la courbe de la Figure 6.5, nous montrons la variation de la moyenne de mémoire maximale consommés par l'heuristique agrégée et l'heuristique à base du SCP, en fonction du pourcentage d'itemsets sensibles parmi les fréquents.

On peut remarquer que la consommation mémoire de l'heuristique *agrégée* et de l'heuristique à base de SCP est assez stable et peu sensible aux changements de pourcentages d'ensembles d'éléments sensibles. Cependant, on note une grande différence de consommation

entre les deux méthodes, et cela est dû au fait que l'heuristique à base de SCP a besoin de plus de stockage. Cela ne pose pas de problème majeur pour l'heuristique à base de SCP car la consommation n'arrive à un stade qui pourrait gêner l'exécution. Cependant, cet aspect pourrait être amélioré en réexaminant le code et en essayant d'optimiser l'utilisation de la mémoire.

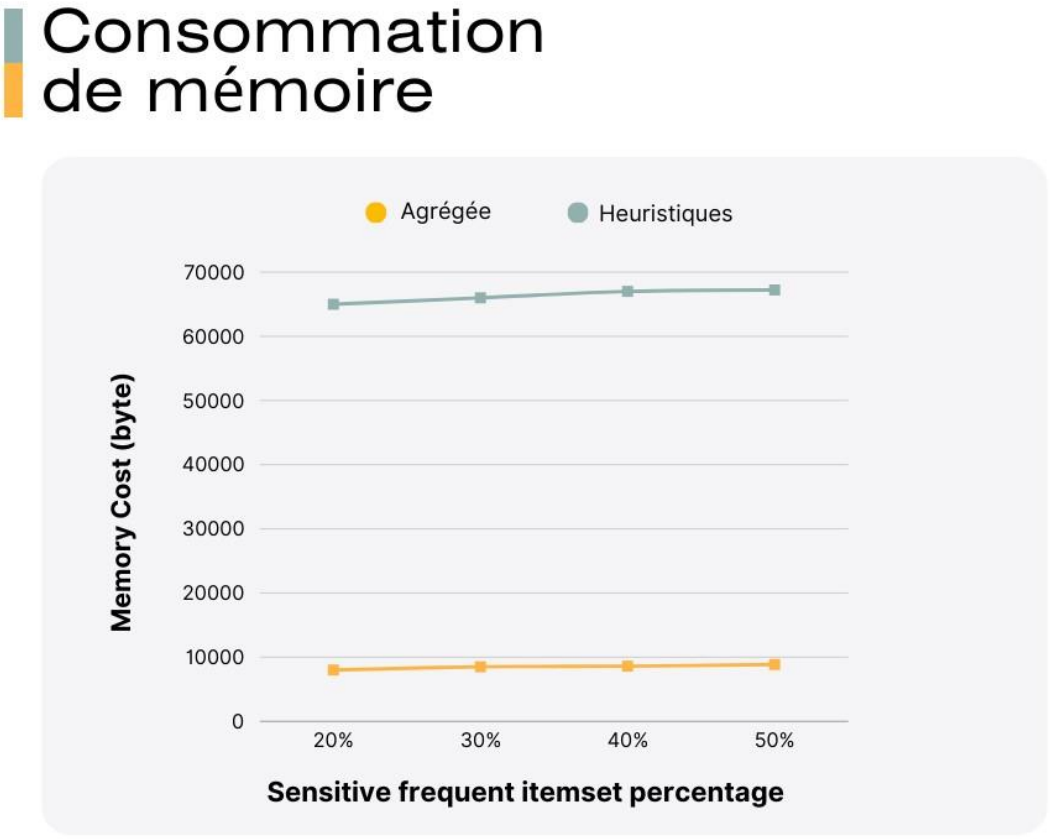

Figure 6.3: Consommation mémoire de l'heuristique agrégée et l'heuristique à base de SCP dans la base Mushroom

6.4.2.3. Fonctions objectives (Side effects)

Chapitre 6 : Implémentation et Résultats

- **Missing Cost**

  Dans l'histogramme de la Figure 6.6, on remarque que Missing Cost (coût perdu) de l'heuristique agrégée est supérieur à celui de SCP pour la plupart des pourcentages d'itemsets sensibles. Cela est dû au fait que l'heuristique agrégée supprime des transactions entières alors que l'heuristique à base de SCP supprime juste certaines items de ces transaction. En effet, la suppression d'une transaction entière veut dire la suppression sûre de tous les itemsets non sensibles de cette transaction tandis que la suppression de quelques items ne touche pas forcément tous les itemsets non sensibles.

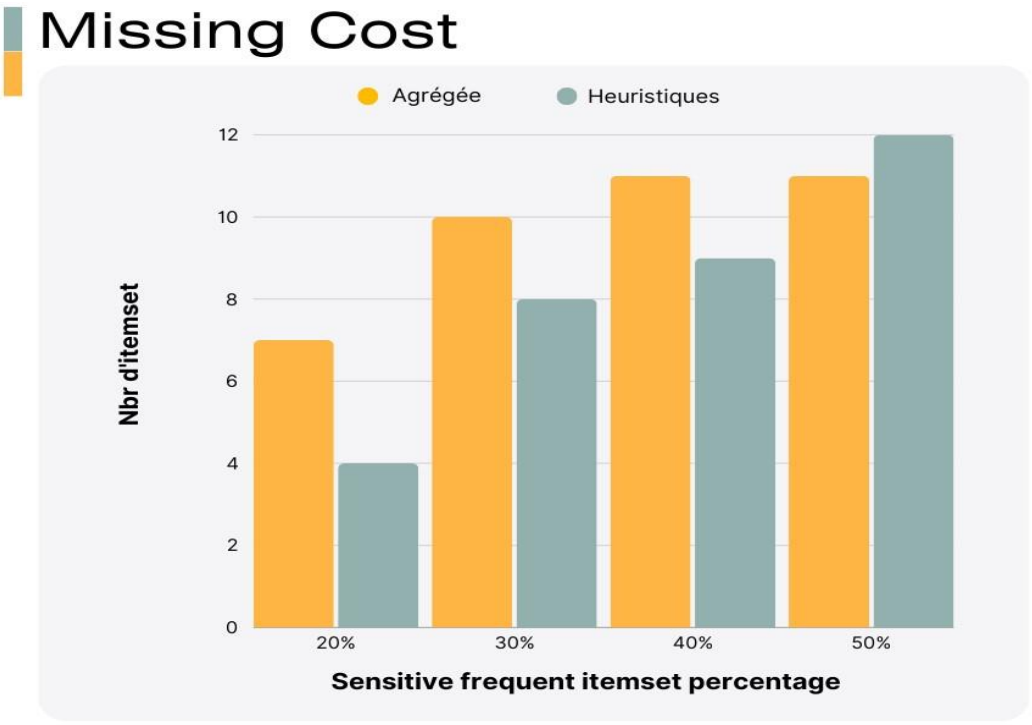

Figure 6.4 : Missing Cost de l'heuristique agrégée et l'heuristique à base de SCP dans la base Mushroom

Chapitre 6 : Implémentation et Résultats

- **Dissimilarité**

  Nous constatons qu'il existe une énorme différence entre les deux méthodes : l'heuristique à base de SCP supprime beaucoup moins d'items que l'heuristique agrégée. La raison est clair : Contrairement à l'heuristique agrégée qui supprime les transactions victimes toutes entières, l'heuristique à base de SCP ne supprime que certaines items sélectionnés de celles-ci. Cela montre que notre objectif principal de départ est bien satisfait.

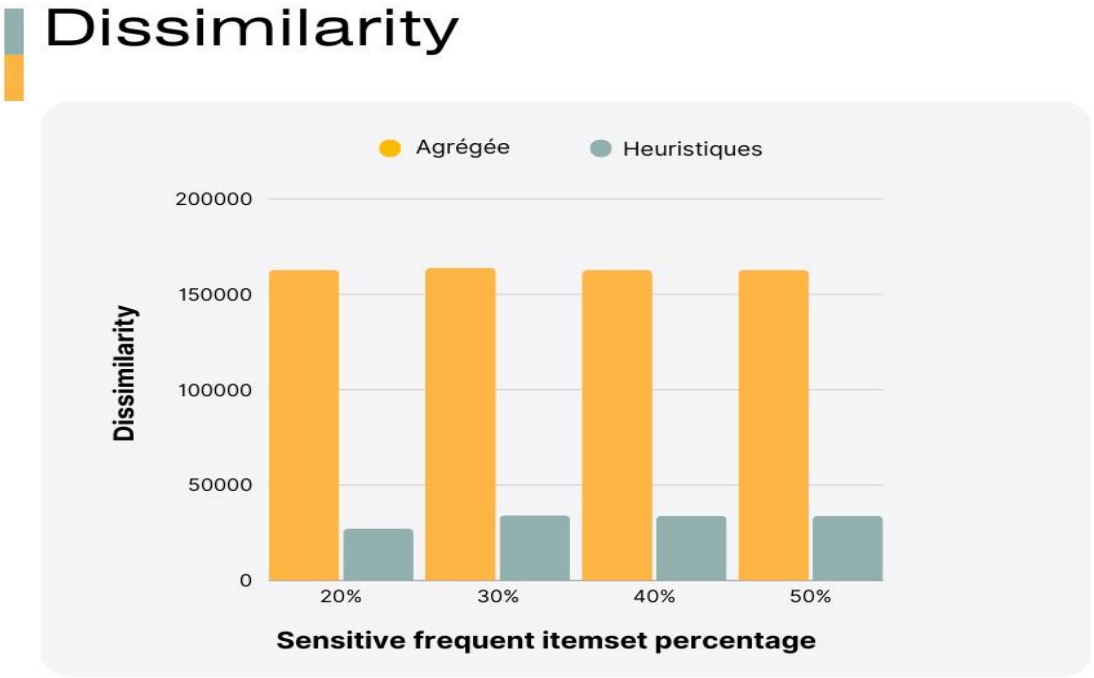

Figure 6.5 : Dissimilarité de l'heuristique *agrégée* et l'heuristique à base de *SCP* dans la base Mushroom

Chapitre 6 : Implémentation et Résultats

6.4.3. La BDT D1

Nous avons effectué les mêmes expériences décrites précédemment sur une deuxième base de données appelée D1. Dans les expériences suivantes, nous avons démontré la valeur *Minsup = 0,5*. La base du D1 est aussi une base dense ce qui explique le choix d'une valeur de support élevée.

Tableau 6.2: Caractéristiques de la base de données D1

Data base Name	**#Of Items**	**#Of Transaction**	**Avg.Trns.Length**	**Support(s)**	**# Of Items**
D1	34	3196	40.0	2500	3355

6.4.3.1. Temps d'exécution

La courbe de la Figure 6.9 montre la variation du temps d'exécution dans *agrégée* et l'heuristique à base de SCP.

Chapitre 6 : Implémentation et Résultats

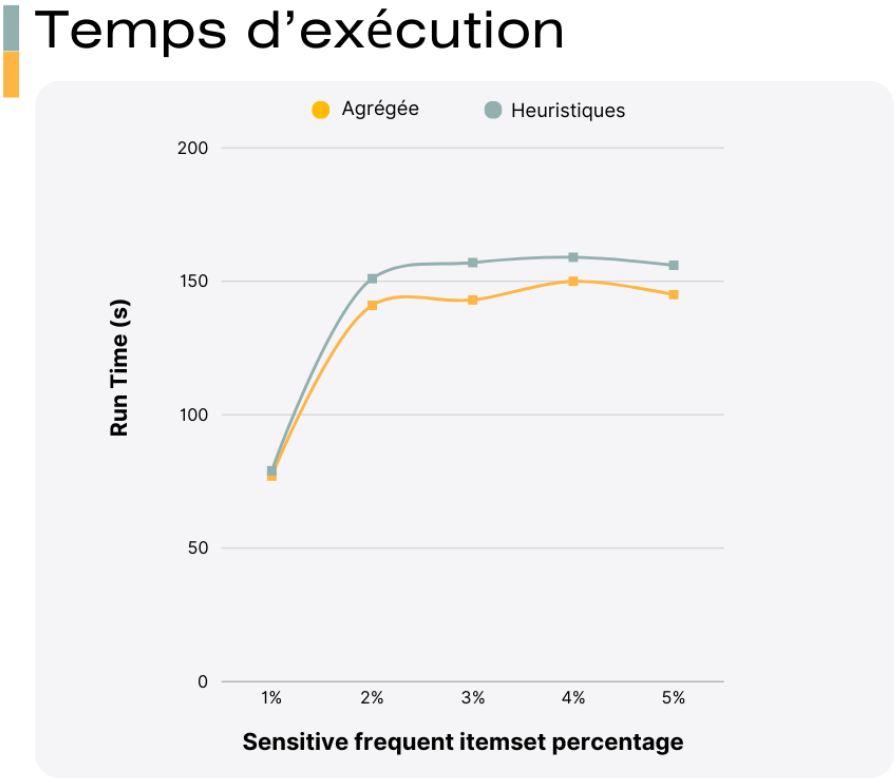

Figure 6.6: Temps d'exécution de l'heuristique agrégée et l'heuristique à base de SCP dans la base D1

À partir de cette courbe, nous pouvons observer qu'en D1, nous constatons aussi que l'heuristique agrégée est légèrement plus rapide que l'heuristique à base de SCP et que cette différence de temps d'exécution augmente à mesure que le pourcentage d'ensembles d'éléments sensibles augmente. Comme dans le cas de Mushroom, ce résultat est attendu puisque notre heuristique est plus fine que l'heuristique agrégée et donc elle consomme naturellement plus de temps.

Chapitre 6 : Implémentation et Résultats

6.4.3.2. Consommation de mémoire

La courbe de la Figure 6.10 montre la variation de la moyenne de mémoire maximale consommé par l'heuristique agrégée et l'heuristique à base de SCP.

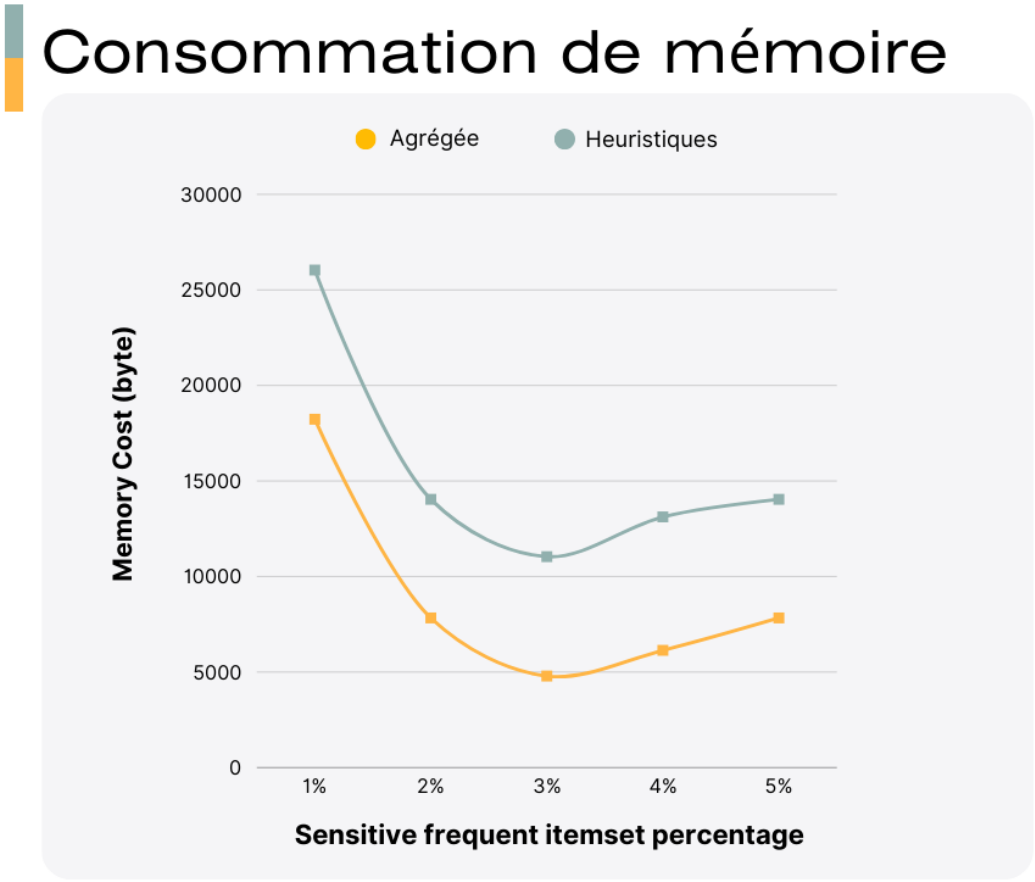

Figure 6.7: Consommation mémoire de l'heuristique agrégée et l'heuristique à base de SCP dans la base D1

On peut remarquer que les courbes de consommation de l'heuristique agrégée et de l'heuristique à base SCP suivent la même forme : elles décroissent en termes d'utilisation de la zone de stockage par rapport au pourcentage des itemset sensibles (1% à 3%) puis augmentent (3% à 5%). Cependant, il y a une différence dans la consommation dans les deux cas :

l'heuristique à base de SCP utilise plus espace mémoire que l'heuristique agrégée. La même interprétation donnée pour les résultats obtenus avec la base Mushroom restent valables ici.

6.4.3.3. Fonctions objectives (Side effects)

- **Missing Cost**

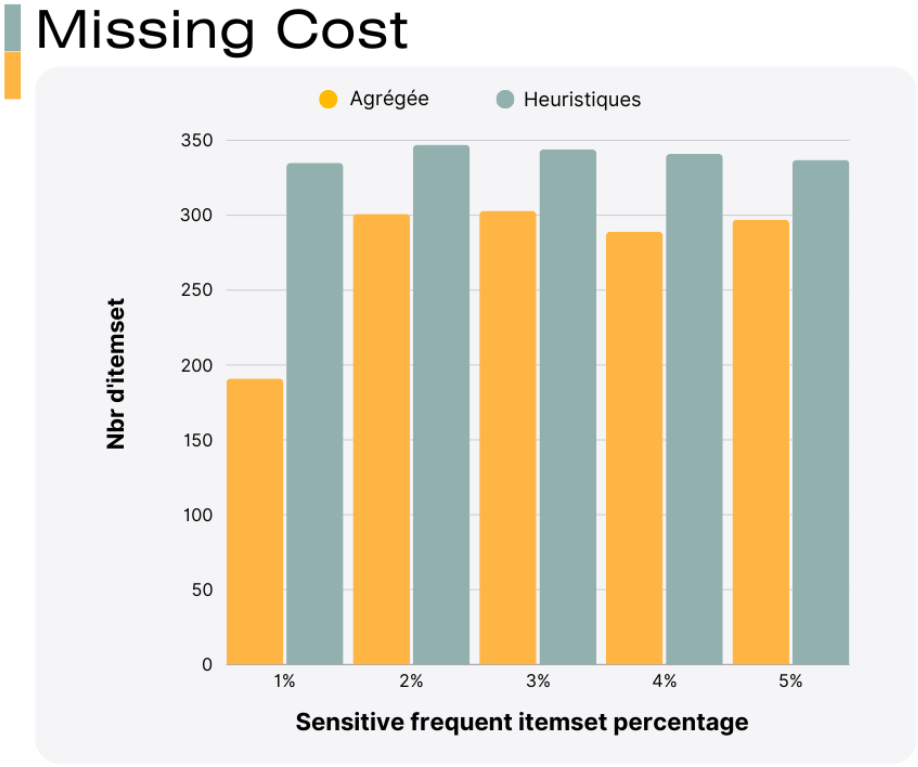

Figure 6.8 : Missing Cost de l'heuristique agrégée et de l'heuristique à base de *SCP* dans la base D1

Nous pouvons remarquer que l'heuristique à base de SCP est légèrement moins bonne que l'heuristique agrégée en termes de missing cost.

Chapitre 6 : Implémentation et Résultats

- **Dissimilarité**

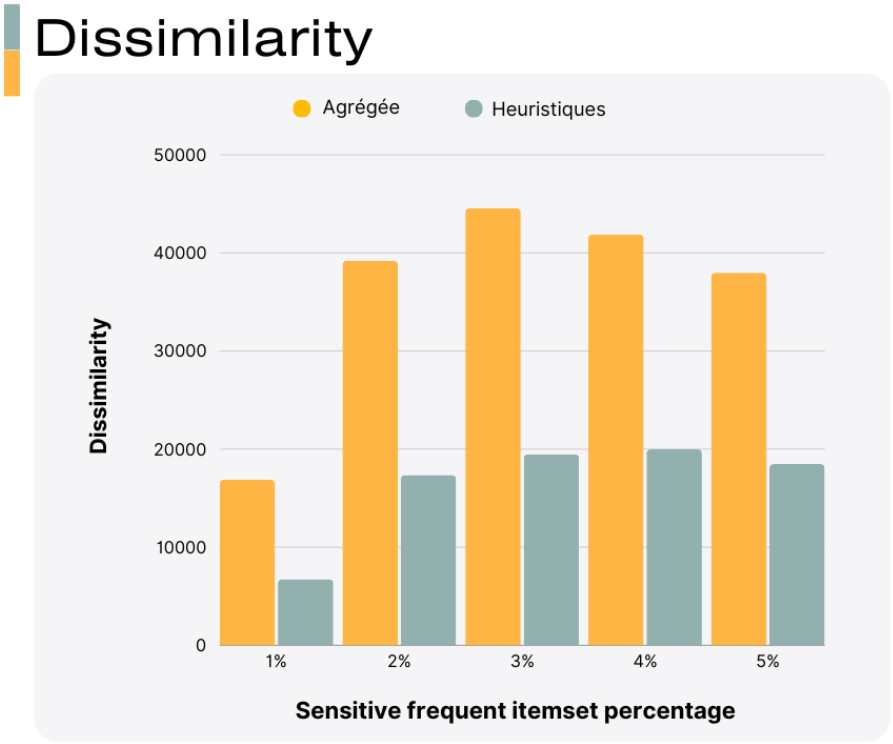

Figure 6.9 : Dissimilarité de l'heuristique agrégée et l'heuristique à base de *SCP* dans la base D1

Nous constatons là aussi qu'il existe une différence remarquable entre les deux méthodes, avec un nombre total d'items supprimés par la méthode à base de SCP nettement inférieur au nombre total d'items supprimés par la méthode agrégée.

6.4.4. Discussion générale

D'après les résultats obtenus sur les deux bases de données Mushroom et D1 nous pouvons constater globalement les points suivants :

Chapitre 6 : Implémentation et Résultats

- L’heuristique agrégée est plus rapide sur les deux bases Mushroom et D1 en raison du fait que l’heuristique à base de *SCP* intégre l’heuristique agrégée et ajoute la partie de résolution de SCP par l’algorithme glouton.
- L’heuristique à base de SCP consomme plus de mémoire que l’heuristique agrégée mais cette consommation reste raisonnable et n’arrive pas à un niveau qui gêne l’exécution de l’algorithme.
- Même si l’heuristique à base de SCP est un peu plus gourmande en temps et en espace que l’heuristique agrégée, la première est largement meilleure que la deuxième en termes de minimisation des obectifs, notamment pour le changement de la base, où notre heuristique arrive à cacher les itemsets sensible en supprimant beaucoup moins d’items que l’heuristique agrégée.

6.5 Conclusion

Notre étude comparant les heuristique "agrégée" et "à base de SCP" sur les bases Mushroom et D1 ont montré l’efficacité et l’utilité de l’exploitation du problème SCP pour un nettoyage plus fin de la base de données afin de cacher les itemsets sensibles.

Chapitre 7 : Conclusion générale

À la fin de cette recherche, nous avons découvert et mis en œuvre deux méthodes heuristiques pour le PPDM appelées respectivement, heuristique agrégée et une nouvelle heuristique à base du SCP visant à cacher les itemsets sensibles exprimant des informations sensibles ou privées que nous ne souhaitons pas divulguer lors de la fouille d'itemsets fréquents à partir d'une base de données transactionnelle.

Ce travail nous a permis d'étudier et comprendre deux méthodes avancées de la fouille de données et donc d'approfondir nos connaissances dans ce domaine. Davantage d'effort nous reste à faire pour envisager l'application de ces méthodes dans le contexte d'applications réelle. Nous pensons qu'avec l'explosion du volume de données disponibles actuellement sous forme électronique et en ligne, le besoin de protéger les informations sensibles est pressant et ce type de méthode peut être extrêmement utile pour beaucoup d'entreprises et d'organisations.

Pour cela, et à plus court termes, cette expérience nous a ouvert des perspectives pour des travaux futurs. Nous devons commencer par tester les approches sur d'autres bases de données. Après avoir compris et implémenté une nouvelle heuristique à base de SCP qui améliore l'heuristique agrégée par l'utilisation du SCP, nous voulons poursuivre la recherche en essayant de développer cette approche avec des idées nouvelles permettant de diminuer davantage les side effects tout en consommant le moins possible de temps et d'espace mémoire. Notamment, l'application de l'algorithme glouton pour résoudre le SCP s'est focalisé sur la maximisation des itemsets sensibles à cacher et n'a pas pris compte la minimisation des itemsets non sensibles à cacher. L'algorithme glouton peut être adapté pour prendre en considération ce point.

Références

[1] Ana Azevedo, Méthodologies et technologies avancées dans l'architecture de réseau, l'informatique mobile et l'analyse de données, 2019.

[2] D. Colt Méthadologie de fouille de données pour la modélisation dans les processus d'aide à la décision complexe. Lyon, avril 2002.

[3] Dominique Crié, Revue Français de Gestion, De l'extraction des connaissances au Knowledge Managment, 2003.

[4] G.ElHelou et C.Aboukhalil Data Mining(Techniques d'extraction des connaissances) UPAPII 2004.

[5] S.Tuffery Data mining et statistique décisionnelle université de Rennes 2005.

[6] AchouriAbdelghani, Mémoire d'extraction de relations d'association : Représentation graphique, 2012.

[7] B Adinarayanareddy, O SrinivasaRao, and MHM Krishna Prasad. An improved up-growth high utility itemset mining. International Journal of Computer Applications, 58(2):25–28, 2012.

[8] Cheng Wei Wu, Bai-En Shie, Vincent S Tseng, and Philip S Yu.Mining top-k high utility itemsets. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 78–86, 2012.

[9] Bouziane Abderraouf, Beghoura Mohamed Amine, Recherche des motifs fréquents périodiques à partir d'une base de données transactionnelle, univ-bba, 2022.

[10] Yacine Abboud, Fouille de motifs : entre accessibilité et robustesse. PhDthesis, Université de Lorraine, 2018.

[11] Hassane Hilali. Application de la classification textuelle pour l 'extraction des règles d 'association maximales. Université de Québec à Trois-Rivières, avril 2009.

[12] Clément Fauré. Découvertes de motifs pertinents par l'implémentation d'un réseau bayésien : application à l'industrie aéronautique. L'Institut National des Sciences appliquées de Lyon,2007.

[13] R. Agrawal, T. Imielinski, et A. Swami. Mining Association Rules between sets of Items in Large Databases. In Proceedings of the ACM SIGMOD International Conference on Management of Data, pages 207-216, Washington D.C., May 1993.

[14] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data Min. Knowl. Discov, 2004.

[15] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Quang-Huy Duong, and Thu-Lan Dam. Phm: mining periodic high-utility itemsets. In Industrial conference on data mining, pages 64–79. Springer, 2016.

[16] Jay Ayres, Jason Flannick, Johannes Gehrke, and TomiYiu.Sequential pattern mining using a bitmap representation. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 429–435, 2002.

[17] Zhenglu Yang and Masaru Kitsuregawa. Lapin-spam: An improved algorithm for mining sequential pattern. In 21st International Conference on Data Engineering Workshops (ICDEW’05), pages 1222–1222.IEEE, 2005.

[18] Lina Fahed, ArmelleBrun, and Anne Boyer. Extraction de règles d’épisodes minimales dans des séquences complexes. In EGC, pages 545–548, 2014.

[19] Dwipen Laskar, international Journal of Computer Applications Technology and Research Volume 3-Issue7, 403-408, July 2014.

[20] Ali Amiri, Department of MSIS, College of Business, Oklahoma State University, Stillwater, 2007.

[21] Yingying Wang, Jixiang Du, Hongbo Zhang, and Xiuhong Yang. Mushroom toxicity recognition based on multi grained cascade forest. Scientific Programming, 2020, 2020