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

MEMOIRE
Présenté en vue de l'obtention du diplôme
Master en informatique
Spécialité : Ingénierie de l'informatique décisionnelle

THEME

Découverte de règles de fouille de motifs dans les bases de données transactionnelles

Présenté par :

Miloudi Fatima Zohra

Hammadi Moncef

Soutenu publiquement le : 22/06/2024

Devant le jury composé de :

Présidente : Dr. Boutouhami Sara (MCB à l'université de BBA)

Examinateur : Dr. Benmessahel Ilyes (MCB à l'université de BBA)

Encadreur : Dr. Nouioua Mourad (MCB à l'université de BBA)

2023/2024

Dédicace

Moncef :

À mes chers parents, frères et sœurs, binôme et encadrant,

Je tiens à exprimer ma profonde gratitude envers chacun d'entre vous pour votre soutien inestimable tout au long de mon parcours académique. Votre encouragement, vos conseils et votre présence ont été essentiels pour mon succès.

À mes parents : Votre amour inconditionnel et votre sacrifice ont été ma source d'inspiration. Votre confiance en moi m'a poussé à donner le meilleur de moi-même.

À mes frères et sœurs : Vous avez été mes compagnons de route, mes alliés dans les moments difficiles et mes partenaires de célébration dans les moments de joie. Votre fierté à mon égard est ma plus grande récompense.

À ma binôme : Notre collaboration a été un voyage passionnant. Ensemble, nous avons surmonté des défis, partagé des idées et créé quelque chose de spécial. Merci d'avoir été à mes côtés.

À mon encadrant : Votre expertise, vos conseils éclairés et votre patience ont été précieux. Vous m'avez guidé vers l'excellence et m'avez encouragé à repousser mes limites.

Ce mémoire est le fruit de notre travail collectif, de nos discussions animées et de nos efforts conjoints. Puissiez-vous tous partager ma fierté et ma joie en le découvrant.

Cordialement.

Dédicace

Je dédie ce travail

À mes êtres les plus chers dans ma vie, et spécialement à ma chère tante. Je lui suis reconnaissante pour sa patience infinie, son amour inconditionnel, et son soutien constant et ses encouragements jusqu'à la fin de mes études. Elle a été un pilier essentiel et un soutien vital dans mon parcours éducatif, et je lui suis profondément reconnaissante pour tout.

À mon cher père, que Dieu lui accorde Sa miséricorde, toi qui es l'étoile qui illumine le chemin de ma vie. Avec fierté et gratitude, je te présente cette réalisation et je tiens à exprimer ma profonde gratitude pour tout ce que tu as fait pour moi. Ta mémoire reste vivante dans mon cœur et continue à m'inspirer à chaque pas que je fais. Merci pour tout.

Sans oublier mon encadreur (Nouioua Mourad), je vous suis très reconnaissante pour votre aide précieuse, votre soutien inestimable

Avec mes sincères salutations et toute ma reconnaissance,

Zahra

Remerciement

Nous exprimons notre gratitude envers Dieu pour nous avoir accordé la possibilité d'apprendre et de réaliser cette tâche.

Merci à notre superviseur. Le Dr Nouioua Mourad pour son soutien, sa disponibilité et les conseils précieux qu'il nous a prodigués. Nous exprimons notre gratitude envers les membres du jury pour avoir accepté d'évaluer notre travail.

Nous souhaitons exprimer notre gratitude envers tous les enseignants qui ont joué un rôle dans notre formation et dans la qualité de l'éducation qu'ils nous ont offerte.

Il est important de ne pas oublier nos parents pour leur soutien et leur patience, nos familles et amis qui nous ont soutenus et encouragés, ainsi que toutes les personnes qui nous ont apporté leur aide, que ce soit de près ou de loin. Je vous remercie à tous.

Résumé

Un problème majeur dans le domaine de la fouille de données est la recherche des motifs à haute utilité (High Utility Pattern Mining - HUPM), qui cherche à trouver des combinaisons d'articles ayant un impact significatif sur une mesure spécifique, comme les ventes, les profits ou la satisfaction des clients. En raison de la croissance du volume de données dans le domaine du Big Data, il est primordial de concevoir des algorithmes performants pour extraire rapidement ces ensembles d'éléments à grande valeur.

Dans notre étude, nous abordons le sujet de la recherche des motifs à haute utilité dans les bases de données transactionnelles réelles. L'objectif est de découvrir des motifs très utiles dans ces bases. L'utilité d'un item dans la base représente son importance par rapport aux autres items ; elle peut souvent être associée au prix de l'item, mais peut également être définie par d'autres critères.

Deux algorithmes ont été testés et appliqués sur deux bases réelles : la première issue d'une pharmacie et la seconde contenant les achats réalisés dans une boutique des fruits. Cela permet d'extraire deux formes différentes de motifs à haute utilité : les itemsets à haute utilité et les règles d'association à haute utilité.

Mots-clé : Haute utilité, fouille de motifs, fouille d'itemsets à haute utilité, Sequential Pattern Mining Framework (SPMF), Transaction Weighted Utilization (TWU), règles d'association à haute utilité, fouille d'itemsets fréquents

Abstract

A major problem in data mining is *High Utility Pattern Mining (HUPM)*, which seeks to find combinations of items that have a significant impact on a specific metric, such as sales, profits or customer satisfaction. Due to the growth in the volume of data in the field of Big Data, it is essential to design efficient algorithms to quickly extract these sets of high-value elements.

In our study, we address the topic of finding high utility patterns in real transactional databases. The objective is to discover very useful patterns in these bases. The utility of an item in the database represents its importance in relation to other items ; it can often be associated with the price of the item, but can also be defined by other criteria.

Two algorithms were tested and applied on two real bases : the first from a pharmacy and the second containing purchases made in a fruit shop. This allows to extract two different forms of high-utility patterns : *High Utility Itemsets (HUIs)* and *High Utility Association Rules (HARs)*.
Keywords : High Utility, Pattern Mining, High Utility Itemsets Mining, Sequential Pattern Mining Framework (SPMF), Transaction Weighted Utilization, High Utility Association Rules, Frequent Itemset Mining

ملخص

تمثل المشكلة الرئيسية في استخراج البيانات في تعدين أنماط المنفعة العالية ، الذي يسعى إلى العثور على مجموعات من العناصر التي لها تأثير كبير على مقياس معين، مثل المبيعات أو الأرباح أو رضا العملاء. نظرًا للنمو في حجم البيانات في في مجال البيانات الضخمة، من الضروري تصميم خوارزميات فعالة لاستخراج هذه المجموعات من العناصر عالية القيمة بسرعة.

في دراستنا، نتناول موضوع إيجاد أنماط فائدة عالية في قواعد بيانات المعاملات الحقيقية. الهدف هو اكتشاف أنماط مفيدة جدًا في هذه القواعد. تمثل فائدة عنصر ما في قاعدة البيانات أهميته بالنسبة للعناصر الأخرى؛ وغالبًا ما يمكن ربطه بسعر السلعة، ولكن يمكن أيضًا تحديده بمعايير أخرى.

تم اختبار خوارزميتين وتطبيقهما على قاعدتين حقيقيتين: الأولى من صيدلية والثانية تحتوي على مشتريات تمت في متجر فواكه. يسمح ذلك باستخراج شكلين مختلفين من أنماط المنفعة العالية: مجموعات عناصر المنفعة العالية وقواعد اقتران المنفعة العالية.

كلمات مفتاحية : أنماط ذات المنفعة العالية، تعدين الأنماط، تعدين مجموعات العناصر عالية المنفعة، إطار تعدين الأنماط التسلسلية ، الاستخدام المرجح للمعاملات، قواعد اقتران المنفعة العالية، تعدين مجموعة العناصر المتكررة

Table des Matières

Liste des Abréviations :

- FHM : Faster High-Utility Itemset Mining Algorithm
- FIM : Frequent Itemset Mining
- HUIM : High-Utility Itemset Mining
- HUI-Miner : High Utility Itemset-Miner
- SPMF : Sequential Pattern Mining Framework
- TWU : Transaction Weighted Utilization
- UP-Growth : Utility Pattern Growth Algorithm
- HGB : High Utility Generic Basis
- HGB-ALL : High Utility Pattern Mining ALL
- KDD : Knowledge Discovery in Databases
- ECD : Extraction de Connaissances à partir de Données/ Knowledge Extraction
- EUCS : Estimated Utility Co-Occurrence Structure.
- HUCI-Miner : High Utility Closed Itemsets Miner

Table des figures

Liste des tableaux

Chapitre 1

Introduction Générale

Le fouille de données, également connu sous le nom de data mining, consiste à examiner de grandes quantités de données et de big data sous différents angles pour identifier les relations entre ces données et les convertir en informations utilisables [1]. La fouille de motifs, également connu sous le nom de « Pattern Mining », représente une discipline du Data Mining dédiée à l'identification de modèles répétitifs ou de « modèles » intégrés dans les données. Ces modèles identifiés peuvent prendre de nombreuses formes : ensemble d'itemss (itemsets), règles d'associations, séquences, etc.

Le probléme d'itemsets fréquents, en anglais Frequent Itemsets Mining Problem (FIM), consiste à extraire l'ensembles d'articles (items) fréquemment achetés dans une base transactionnelle. FIM est un problème classique dans le domaine de l'exploration de modèles. L'objectif principal est de révéler des groupes d'articles dans la base de données de transactions que les clients achètent régulièrement en même temps, c'est-à-dire des ensembles d'articles. Ces éléments, qui apparaissent fréquemment dans les transactions, aident à décrypter les habitudes de dépenses. Avoir ces connaisances est essentiel car elles peuvent mettre en évidence des tendances utiles pour guider les décisions stratégiques.

1.1 Motivations et objectifs

Bien que FIM soit utile, elle repose sur l'hypothèse que les motifs fréquents sont tous intéressants. Mais cette hypothèse n'est pas valable pour de nombreuses applications. Par exemple, dans une base de données transactionnelle, le motif *{milk ; pain}* peut être très fréquent, mais

peut être sans intérêt car il peut générer un faible profit malgré sa fréquence haute. En d'autre part, plusieurs motifs tels que {caviar et Yellow Gold Tea} peuvent ne pas être fréquents mais peuvent générer un profit plus élevé. Par conséquent, pour trouver des motifs intéressants dans les données, d'autres aspects peuvent être pris en compte tels que le profit ou l'utilité [2].

Pour remédier à cette limitation, un domaine de recherche émergent est la découverte de motifs à haute utilité dans les bases de données transactionnelles, en anglais, High Utility Itemset Mining (HUIM).

L'utilié d'un item dans la base représente son imprortance par rapport aux autres items, elle représente fréquement le prix de l'item mais l'utilité peut être représentée par d'autre critères. Par exemple, pour étudier les habitudes d'achat dans une base de données de transactions clients, l'utilité d'un modèle (un ensemble d'articles) peut être mesurée en termes de profit qu'il génère, tandis que pour analyser les données de parcours, l'utilité peut représenter le temps passé sur les pages Web.

Depuis son apparition, la découverte de motifs à haute utilité (HUIM) a trouvé de nombreuses applications dans divers domaines. De plus, plusieurs extensions ont été proposées pour différentes variantes du problème de HUIM.

Une extension intéressante de HUIM nous permet d'obtenir les règles à haute utilité, en anglais High Utility Association Rules. Les régles à haute utilité sont des motifs plus informatifs que les itemsets à haute utilité. Ces régles sont des implications de la forme X ==> Y où X et X $\cup$ Y sont deux itemsets à haute utilités.

extraire les itemsets à haute utilité et l'algorithme HGB-ALL pour extraire les règles d'association à haute utilité. Ces algorithmes sont appliqués sur deux bases de données réelles : une base de données de pharmacie et une base de données d'un magasin de détail américain axé sur la vente de fruits.

Notre objectif est double : d'un côté, nous allons étudier en détail les deux algorithmes FHM et HGB-ALL et comparer ces deux algorithmes selon plusieurs critères d'évaluation tels que le temps d'exécution et l'utilisation de la mémoire. D'un autre côté, nous nous intéressons à extraire des motifs de ces bases de données et à leur donner des interprétations, ce qui est intéressant pour ensuite prendre de bonnes décisions.

1.2 Organisation du mémoire

Le reste de ce mémoire est organisé de la façon suivante :

- Chapitre 02 est consacré à la présentation du data mining et du pattern mining. Dans ce chapitre, nous présentons les différents concepts liés au data mining et au pattern mining.
- Dans le Chapitre 03, nous présentons en détail le problème de la fouille de motifs à haute utilité (HUIM), puis nous définissons les deux types d'apprentissage (supervisé et non supervisé), avant de présenter les deux algorithmes, FHM et HGB-ALL, que nous avons utilisés dans notre projet.
- Chapitre 04 montre le côté pratique de notre travail : les résultats de l'application des deux algorithmes FHM et HGB-ALL y sont présentés et analysés. De plus, ce chapitre présente et interprète quelques exemples de motifs extraits de l'application de ces deux algorithmes sur des bases de données réelles.
- En fin, nous concluons notre mémoire en dressant un bilan de cette étude. De plus, nous discutons les orientations futures de ce travail.

Chapitre 2

Fouille de Données (Data Mining) et Fouille des Motifs (Pattern Mining)

2.1 Introduction

Le problème classique dans le domaine de la fouille de motifs est l'extraction d'ensembles d'articles (itemsets) fréquemment achetés, également appelé le problème des itemsets fréquents, en anglais Frequent Itemsets Mining (FIM). Le but principal est d'identifier les ensembles d'articles dans une base de données transactionnelle que les clients achètent fréquemment en même temps. Avoir cette connaissance nous permet de comprendre les habitudes d'achat des clients. Cette compréhension facilite la prise de décisions stratégiques, telles que les décisions de faire des promotions sur des produits particuliers [3].

Au cours de ce chapitre, nous examinons le sujet du "Data Mining". Nous commençons par présenter le processus d'extraction des connaissances, ainsi que ses diverses tâches. Nous présentons également les techniques de data mining que l'on peut diviser en deux catégories : les techniques prédictives et les techniques descriptives. Enfin, ce chapitre se termine par une présentation des domaines d'application du data mining et du pattern mining.

2.2 Processus d'extraction de connaissances (PEC)

2.2.1 Définition 1

Le Processus d'Extraction de Connaissances à partir des Données (PEC), en anglais Knowledge Discovery in Databases (KDD), consiste à tirer des informations nouvelles, intelligibles et potentiellement utiles à partir de faits dissimulés au sein de grandes quantités de données [4].

2.2.2 Définition 2

Le Processus d'Extraction de Connaissances à partir des Données (PEC) est un processus interactif et itératif entre l'homme et la machine, qui comprend plusieurs étapes telles que la compréhension du domaine d'application, la préparation des données, la collecte de données et l'analyse des résultats. Ce processus est en partie automatisé, où l'interaction entre les individus est essentielle. Il offre une fonctionnalité itérative, permettant à l'utilisateur de revenir à n'importe quelle étape afin d'ajuster ou d'enrichir les données, d'intégrer d'autres données, d'affiner l'algorithme de fouille ou d'améliorer la présentation des résultats [5].

2.3 Les étapes du processus KDD

Le processus de découverte des connaissances à partir des données comprend généralement les étapes suivantes :

- Compréhension du domaine d'application.
- Création de l'ensemble de données cibles.
- Nettoyage et préparation des données.
- Réduction et projection des données.
- Choix des fonctions de data mining.
- Exploration de données (Data Mining).
- Interprétation et évaluation.
- Déploiement.

2.4 Data mining

2.4.1 Définition

Le Data Mining représente une avancée majeure dans l'ingénierie de la connaissance, offrant la possibilité de trouver de nouvelles corrélations, tendances et modèles au sein de vastes quantités de données [1].

Aujourd'hui, les méthodes de data mining démontrent leur efficacité. La preuve en est l'utilisation étendue de ces techniques dans divers secteurs tels que la santé, l'éducation et le marketing, entre autres [1].

2.4.2 Techniques de data mining

Les principaux algorithmes de data mining sont répartis en deux catégories de techniques : les techniques prédictives et les techniques descriptives.

En fait, il n'existe pas de techniques de fouille plus efficaces. Il sera nécessaire de trouver des solutions en fonction des besoins identifiés et des caractéristiques connues des outils.,Ainsi, la sélection de la méthode adéquate dépend de divers éléments tels que :

- La nature et la disponibilité des données.
- Le contexte de l'entreprise.
- Les diverses responsabilités.
- L'objectif du modèle élaboré.
- La difficulté de l'élaboration du modèle.
- La difficulté d'utilisation du modèle et ses résultats.
- Déploiement.

2.4.3 Techniques prédictives

Les techniques prédictives offrent la possibilité d'extrapoler et de prédire de nouvelles informations et connaissances à partir des données initiales. Il existe principalement deux méthodes de prédiction : la classification et la régression.

2.4.3.1 La classification

La classification consiste à attribuer des étiquettes de classe à des instances ou des ensembles de données en se basant sur des caractéristiques acquises à partir d'un ensemble d'entraînement [1].

Dans le processus de classification, la population est minimisée en enregistrant chaque élément ou enregistrement dans une classe préétablie en utilisant un modèle obtenu par apprentissage à partir d'exemples pré-classés [1].

2.4.3.2 La régression

La regression est employée pour anticiper la valeur d'une variable en fonction des valeurs d'autres variables, en supposant une relation linéaire ou non-linéaire entre elles. Cette méthode est couramment utilisée pour prédire et étudier les causes [6]. Par exemple, il est possible d'évaluer le coût d'une maison en prenant en compte sa superficie, le nombre d'étages, son emplacement, etc [6].

2.4.4 Techniques descriptives

les techniques descriptives sont utilisées pour résumer et présenter les caractéristiques principales d'un ensemble de données, sans nécessairement chercher à prédire des valeurs futures. Elles sont essentielles pour explorer et comprendre les données avant d'appliquer des méthodes plus avancées de Data Mining [1].

2.4.4.1 Fouille des motifs (Pattern mining)

L'objectif du pattern mining est de repérer des motifs récurrents, des associations, des corrélations ou des structures qui se manifestent de façon récurrente dans un ensemble de données. Notre étude concerne le pattern mining, d'où la suite de ce chapitre contient une étude détaillée sur le pattern mining[7].

2.4.4.2 Clustering

Le clustering consiste à regrouper des données similaires en fonction de leurs similarités ou distances. Cette approche permet d'identifier des groupes homogènes au sein des données, facilitant ainsi la compréhension des relations entre les différentes instances[7].

Exemples d'utilisation du clustering :

Segmentation de clients : Regrouper des clients similaires en fonction de leurs comportements d'achat ou de leurs préférences.

Classification de documents : Organiser des articles, des courriels ou d'autres textes en fonction de leur contenu similaire

Reconnaissance de motifs dans les images : Identifier des motifs visuels similaires dans des images.

2.4.4.3 Les règles d'association

La découverte de relations intéressantes et fréquentes entre les éléments d'un ensemble de données est l'objectif des règles d'association. Elles offrent la possibilité de repérer des ensembles d'éléments qui se produisent fréquemment ensemble.

Exemple concret : Supposons que nous disposions de données sur les achats effectués dans une épicerie. La combinaison "lait et céréales" est intéressante si nous établissons un seuil de support de 10%, car elle se rencontre dans 30% des situations. Il serait bénéfique d'adopter cette règle d'association afin d'optimiser les promotions ou de structurer les rayons de l'épicerie [8].

2.4.5 Fouille des motifs (Pattern mining)

La fouille de motifs (ou pattern mining) est une activité particulière du data mining qui cherche à repérer des motifs fréquents, des règles d'association ou des séquences importantes dans un ensemble de données. Cette tâche est fréquemment utilisée pour identifier des schémas, des informations captivantes ou des tendances dissimulées qui peuvent être utilisées pour prendre des décisions éclairées. Un problème fondamental dans le domaine de la fouille de motifs est la fouille des itemsets fréquents, également appelée Frequent Itemset Mining (FIM).

2.4.5.1 Fouille des itemsets fréquents

La fouille des itemsets fréquents, frequent itemsets mining (FIM), est une méthode analytique qui cherche à repérer les combinaisons d'items (itemsets) qui se retrouvent souvent ensemble dans des transactions, au-delà d'un seuil de support défini. Cette méthode joue un

rôle essentiel dans la création de règles d'association, qui mettent en évidence des relations intéressantes entre ces itemsets dans des grandes bases de données [6].

2.4.5.2 Concepts fondamentaux

Afin de mieux comprendre le FIM, nous allons définir quelques termes essentiels :

2.4.5.3 Item

Un item est un élément séparé d'un ensemble de données, pris isolément en compte. Il incarne une mesure ou un événement dans une base de données transactionnelle [1].

2.4.5.4 Itemset

Un itemset, également connu sous le nom d'ensemble d'items, est une série d'articles (items) qui se retrouvent ensemble dans une transaction d'une base de données transactionnelle.

Pour l'extraction d'itemsets fréquents, un itemset est qualifié de "fréquent" lorsqu'il se retrouve dans un nombre de transactions supérieur ou égal à un seuil minimum spécifié, appelé support [1].

2.4.5.5 Itemset fréquent

Un itemset est qualifié de fréquent si et seulement si son support dépasse ou égale un support minimum pré-défini par l'utilisateur [1].

2.4.6 Domaines d'applications de data mining et pattern mining

2.4.6.1 Data mining

Data mining est exploité dans plusieurs domaines. Le tableau suivant présente quelques champs d'application du data mining :

2.4.6.2 Pattern mining

Le tableau 2.2 illustre comment le pattern mining peut être utilisé pour découvrir des motifs et des tendances significatives qui aident à la prise de décision dans divers secteurs :

TABLE 2.1 – Les domaines d'application de data-mining.

Secteurs d'activité	Exemples d'application
Business Intelligence	Aide à la prise de décision stratégique en entreprise
Santé	Amélioration de la qualité des soins et gestion des données de santé
E-commerce	Analyse des comportements des consommateurs pour ajuster les stratégies de vente
Finance	Evaluation des risques et détection de de la fraude
Télécommunication	Optimisation des réseaux et services personnalisés

TABLE 2.2 – Les domaines d'application de pattern mining.

Secteurs d'activité	Exemples d'application
Commerce électronique	Analyse des séquences d'achats pour recommander des produits
Santé publique	Identification de motifs dans les données génétiques pour la recherche médicale
Finance	Détection de comportement frauduleux dans les transactions
Marketing	Segmentation de la clientèle basée sue des comportements d'achats communs
Bio-informatique	Recherche de motifs dans les séquences d'ADN pour des études génétiques

2.5 Conclusion

En résumé, data mining et pattern mining sont des domaines actives de l'analyse de données qui ont profondément transformé la façon dont les organisations prennent des décisions et élaborent leur stratégie. Grâce à ces méthodes prédictives et descriptives, data mining offre aux entreprises la possibilité de convertir d'énormes quantités de données brutes en informations exploitables, ce qui leur confère un avantage concurrentiel considérable dans différents domaines d'activité.

Plus précisément, pattern mining se démarque par sa capacité à repérer des motifs dissimulés et des associations importantes dans les données, ce qui est crucial pour appréhender les comportements des consommateurs, anticiper les tendances du marché et améliorer les services de santé publique. Lorsqu'elles sont correctement mises en œuvre, ces méthodes peuvent conduire à des découvertes novatrices et à des progrès.

Chapitre 3

Fouille des Motifs à Haute Utilité (High Utility Pattern Mining)

3.1 Introduction

Au cours de ce chapitre, nous examinerons de manière approfondie le problème de l'extraction des itemsets à haute utilité. Nous aborderons d'abord les contraintes et les limitations liées à l'extraction fréquente d'itemsets. Ensuite, nous soulignerons l'importance de prendre en compte l'utilité des items.

En sachant comment extraire des itemsets à haute utilité, les chercheurs et les professionnels peuvent utiliser cette compétence pour identifier des motifs importants dans les données. Ces motifs peuvent être exploités pour améliorer les processus de prise de décision et obtenir un avantage concurrentiel dans diverses applications. L'objectif de ce chapitre est de fournir une vision approfondie de la fouille des itemsets à haute utilité, de ses méthodes, de ses indicateurs d'évaluation, de son analyse de performance et de ses éventuels avantages.

3.2 Fouille des itemsets fréquents ou *frequent itemset mining (FIM)*

Parmi les outils les plus couramment utilisés en data mining, on retrouve les techniques d'extraction des itemsets fréquents et les règles d'association. Elles facilitent la mise en évi-

dence de règles compréhensibles et exploitables dans un vaste ensemble de données. Ces méthodes sont fréquemment appliquées aux bases de données transactionnelles, qui sont des ensembles de transactions où chaque transaction contient un ensemble d'items. Les bases de données transactionnelles peuvent être créées dans différents domaines tels que le marketing, l'aide au diagnostic médical, les télécommunications, l'analyse de données spatiales, et bien d'autres encore [2].

3.2.1 Concepts fondamentaux

Avant de donner la définition formelle du problème de l'extraction des itemsets fréquents (FIM), nous exposerons les concepts fondamentaux de la FIM.

Définition 1 (Base de données transactionnelle) : Les transactions d'une base de données transactionnelle sont représentées par D = { T1, T2, . . ., Tn }, où n, est le nombre de transactions de la base. Chacune de ces transactions possède un identifiant TID et comprend un ensemble d'éléments [2].

Un exemple d'une base de données transactionnelle D est illustré dans le Tableau 3-1. La base de données comprend 5 transactions. Par exemple, la première transaction comprend cinq éléments : a, b, c, d et e.

TABLE 3.1 – *Un exemple d'une base de données transactionnelle.*

TID	Transactions
T0	a, b, c, d, e
T1	b, c, d, e
T2	a, c, d
T3	a, c, e
T4	b, c, e

Définition 2 (Support d'un itemset) : Le support d'un itemset X est la proportion de transactions de D contenant X est appelée Supp(X). Étant donné une base de données transactionnelle D et un itemset X, le support de l' itemset X dans la base D est calculé comme suit :

$$\text{Supp}(X) = \frac{\text{Freq}(X)}{|D|}$$

où Freq(X) correspond au nombre de transactions dans D qui incluent l'itemset X.

Exemple :

- Le support de l'itemset $\{ad\}$ dans la base D est $\text{Supp}(\{ad\}) = \frac{\text{Fréq}(\{ad\})}{|D|} = \frac{2}{5}$.

Definition 3 (Les Itemsets Fréquents) : Étant donné une base de données transactionnelle D et un seuil de support $MinSup$, un itemset X est un itemset fréquent si son support $Supp(X)$ est au moins égal au seuil $MinSup$. Formellement, si $Supp(X) \geq MinSup$. Sinon, X est un itemset non-fréquent.

3.2.2 Définition formelle de FIM

Frequent Itemset Mining (FIM) a pour objectif de repérer des ensembles d'articles qui se retrouvent souvent ensemble dans une base de données transactionnelle [2].

Plus précisement, étant donné une base de données transactionnelle D et le seuil de support $MinSup$ défini par l'utilisateur, FIM consiste à découvrir tous les itemsets fréquents dans D.

3.2.3 Limites de frequent itemset mining (FIM)

Bien que la recherche d'itemsets fréquents (FIM) ait plusieurs applications pour l'analyse de paniers d'achat (market basket analysis), elle présente certaines limitations et ne peut pas couvrir tous les aspects de certains problèmes en pratique [2] .Plus précisement, deux limitations importantes ont été identifiées :

(1) Un souci important de FIM réside dans le fait qu'il estime que tous les éléments ont la même importance (utilité), peu importe s'ils sont très utiles ou non.

Prenons l'exemple concret de l'itemset pain, lait qui est souvent acheté par les clients, mais qui n'est pas intéressant pour un commerçant car il ne génère pas de revenus considérables (utilité)

(2) Un autre souci de FIM réside dans l'absence de prise en compte des quantités d'éléments.

Par exemple, si un client fait l'acquisition de cinq pains ou de dix pains, il est considéré comme étant identique. Donc, l'exploration de motifs fréquents (mining de motifs fréquents) peut identifier de nombreux motifs fréquents qui ne présentent pas d'intérêt.

En plus des problèmes mentionnés ci-dessus, la définition du seuil de support $MinSup$ représente l'un des défis majeurs. Il est courant que les utilisateurs rencontrent des difficultés pour

établir un seuil de support minimum adéquat afin d'obtenir des résultats satisfaisants, en particulier lorsqu'ils manquent de connaissances pertinentes. Un autre problème survient lorsque le volume de données augmente, dans ce cas, l'extraction d'ensembles d'articles fréquents demande de plus en plus de temps et de mémoire [9].

Pour surmonter ces limitations, une extension plus complexe de la recherche d'itemsets fréquents, appelée High Utility Itemset Mining (HUIM), a été développée [2, 10].

3.3 Fouille des motifs à haute utilité ou *high utility itemset Mining (HUIM)*

Ces dernières années, motivée par la nécessité d'analyser des données plus complexes et de trouver des modèles plus utiles, la fouille des motifs à haute utilité est devenue une tâche clé dans le domaine de la fouille des motifs. L'objectif est de trouver des modèles qui ont une grande importance mesurée par une fonction d'utilité numérique. Cette utilité peut être utilisée pour mesurer la fréquence d'occurrence, mais aussi d'autres critères plus intéressants. Par exemple, pour étudier les habitudes d'achat dans une base de données transactionnelle, l'utilité d'un motif peut être mesurée en termes de profit qu'il génère. Tandis que pour analyser les données de flux (click stream data), l'utilité peut représenter le temps passé sur les différentes pages Web, etc [11, 12] .

3.3.1 Concepts fondamentaux

HUIM a été développée pour trouver l'ensemble d'itemsets pouvant générer un profit élevé dans une base de données transactionelle quantitative [13] . Dans ce qui suit, nous allons voire les concepts fondamentaux liés au probléme de HUIM ainsi que sa définition formelle.

Definition 4 (Base de données transactionnelle quantitative) : Contrairement au FIM, HUIM est appliqué aux base de données quantitatives où chaque item a une quantité d'achat (utilité interne). De plus, nous devons également définir la table d'utilité qui contient le profit (utilité externe) de chaque item de la base de données. L'objectif principal de HUIM est de trouver tous les itemsets qui ont des utilités supérieur ou égale au seuil d'utilité, $MinUtil$ prédéfinie par l'utilisateur.

Une base de données transactionnelle quantitative contient des informations supplémentaires, telles que les quantités des items dans les transactions et les profits des items qui indiquent l'importance relative de chaque item [14].

Les Tableaux 3.2 et 3.3 illustrent une base de données quantitative qui inclut les quantités externes des items. Par exemple, la transaction $T3$ révèle qu'un client a acheté 2 unités de l'item a, 2 unités de l'item c et une seule unité de l'item e [14].

TABLE 3.2 – *Un exemple d'une base de données transactionnelle quantitative.*

TID	Transactions
T0	(a, 1), (b, 5), (c, 1), (d, 3), (e, 1)
T1	(b, 4), (c, 3), (d, 3), (e, 1)
T2	(a, 1), (c, 1), (d, 1)
T3	(a, 2), (c, 6), (e, 2)
T4	(b, 2), (c, 2), (e, 1)

TABLE 3.3 – *Utilités des items*

item	Utilité externe
a	5
b	2
c	1
d	2
e	3

Definition 5 (Utilité d'un item i dans une transaction T_c) : L'utilté d' un item i dans une transaction T_c est représentée par $u(i,T_c)$ et définie par $p(i) \times q(i,T_c)$. L'utilité externe de l'item i est $p(i)$ et la quantité de i dans la transaction T_c est $q(i,T_c)$[14].

Exemple :

- L'utilité de l'item a dans la transaction $T3$ est :

$$u(a,T3) = 5 \times 2 = 10.$$

Definition 6 (Utilité d'un itemset X dans une transaction T_c : L'utilité d'un itemset X dans une transaction T_c est la somme des utilités des items qui le composent dans cette trasaction. Elle est notée par $u(X,Tc)$ et définie comme :

$$u(X,T_c) = \sum_{i \in X} u(i,T_c) \quad \text{si} \quad X \subseteq T_c$$

. **Exemple :**

- L'utilité de l'itemset $\{ac\}$ dans la transaction $T3$ est :

$$u(ac,T3) = u(a,T3) + u(c,T3) = 5 \times 2 + 1 \times 6 = 16.$$

Definition 7 (Utilité d'un itemset X dans la base) : L'utilité d'un itemset X dans une base de données D est la somme des utilités de ce itemset dans toutes les transactions qui contiennent X. Elle est représentée par $u(X)$ et est définie comme

$$u(X) = \sum_{T_c \in g(X)} u(X,T_c)$$

où $g(X)$ est l'ensemble des transactions contenant X.

Le profit obtenu lors de la vente de l'item X dans la base de données transactionnelle quantitative D est représenté par cela [7].

Definition 8 (Itemset à haute utilité ou High utility itemset (HUI)) : Un itemset X est considéré comme un itemset à haute utilité si son utilité $u(X)$ est supérieure ou égale à un seuil d'utilité $MinUtil$ prédéfini par l'utilisateur. Autrement, X est un itemset à faible utilité [2].

3.3.2 Définition formelle du problème de HUIM

HUIM consiste à découvrir toutes les combinaisons des items (itemsets) ayant des utilités non inférieures au seuil d'utilité minimum ($MinUtil$) défini par l'utilisateur. L'ensemble de ces itemsets est appelé HUI (High Utility Itemset). Formellement, $HUI = \{X | u(X) \geq MinUtil\}$.

3.4 Les défis de HUIM

le problème du HUIM est très difficile pour les raisons suivantes [14, 2] :

1. La première raison est que le nombre d'ensembles d'articles à considérer peut être très élevé pour identifier ceux qui ont une grande utilité. En d'autre termes, l'espace de recherche dans HUIM est exponentiel. Si une base de données contient m éléments distincts, il y a $2^m - 1$ ensembles d'articles (itemsets) possibles. Une approche naïve pour résoudre le problème HUIM consiste à calculer l'utilité de tous les itemsets possibles

en parcourant la base de données, puis on conserve ceux qui ont une utilité élevée. Bien que cette méthode donne le résultat correct, elle est inefficace. En effet, le nombre d'itemsets possibles peut être extrêmement élevé. Par exemple, si un magasin de détail possède 10 000 articles ($m = 10000$), il faudrait calculer l'utilité de $2^{10000} - 1$ itemset, ce qui est impraticable.

2. Les ensembles d'articles à haute utilité sont souvent dispersés dans l'espace de recherche. Par conséquent, un algorithme doit examiner de nombreux ensembles avant de trouver ceux qui ont réellement une haute utilité. Cette difficulté existe parce que, contrairement au problème du FIM où le support d'un ensemble est toujours supérieur ou égal à celui de l'un de ses sur-ensembles, dans le cas du HUIM, l'utilité d'un ensemble d'articles peut être supérieure, inférieure ou égale à celle de n'importe lequel de ses sur-ensembles ou sous-ensembles. En d'autres termes, l'utilité n'est ni monotone ni anti-monotone.

   La mesure d'utilité ne présente ni monotonie ni anti-monotone. Puisque X et Y sont des itemsets, $X \subset Y$, la relation entre les utilités de X et Y est soit : $u(X) < u(Y)$, $u(X) > u(Y)$ ou $u(X) = u(Y)$.

3.5 Solutions proposées pour HUIM

Différentes méthodes ont été suggérées pour extraire les itemsets à haute utilité, telles que UMining, IHUP, UP-Growth, HUP-Growth, MU Growth, HUI-Miner, FHM, ULBMiner, HUI-Miner [2, 15]. Ces algorithmes utilisent la même entrée (une base de données transactionnelle quantitative ainsi que la valeur minimale d'utilité) et produisent la même sortie (l'ensemble de tous les itemsets à haute utilité).

L'espace de recherche de la recherche d'itemsets à haute utilité (HUIM) est représenté sous forme de graphe. Pour explorer ce graphe, les algorithmes proposés utilisent soit une stratégie de recherche en profondeur (DFS), soit une stratégie de recherche en largeur (BFS). L'exploration de l'espace de recherche n'est pas réalisée de manière exhaustive mais en utilisant des stratégies d'élagage (*pruning strategies*) [7]. Grâce aux stratégies d'élagage, les algorithmes peuvent explorer intelligemment l'espace de recherche en éliminant autant que possible les itemsets qui ne sont probablement pas des itemsets à haute utilité, ainsi que leurs sur-ensembles.

Les premiers algorithmes développés avaient pour objectif de réaliser deux phases afin d'extraire l'ensemble de tous les itemsets à haute utilité dans la base de données. Par la suite, d'autres algorithmes plus efficaces ont été créés, reposant sur une seule étape seulement [7].

La conception d'un algorithme pour le HUIM doit prendre en compte tous les aspects essentiels suivants [7] :

1. Si la recherche en profondeur ou en largeur est utilisée par l'algorithme ?
2. est ce que l'algorithme est basé sur deux phases ou une seule phase.
3. Les méthodes d'élagage qui sont employées afin de trouver des items à haute utilité.
4. Comment procèdent-ils à la création ou à la définition des prochains itemsets à explorer dans l'espace de recherche ?

Ces choix influence d'une façon directe la performance des algorithms en fonction de : temps d'exécution, l'utilisation de la mémoire et la facilité avec laquelle ils peuvent être mis en place et étendu pour d'autres tâches d'exploration de données.

Maintenant, nous allons exposer les deux catégories d'algorithmes de HUIM : les algorithmes basé sur deux phases et les algorithmes basé sur une seule phase [10].

3.6 Algorithmes basés sur deux phases (Two-phase based Algorithms)

Comme mentionné dans leurs noms, ces algorithmes effectuent deux phases afin de repérer les itemsets à haute utilité. La première phase est généralement employée pour créer un ensemble d'itemsets candidats à haute utilité en surestimant leurs valeurs d'utilité.

Dans la deuxième phase, les algorithmes procèdent à une analyse de la base de données afin de déterminer l'utilité exacte des candidats et de filtrer les ensembles d'éléments à faible utilité.

Plusieurs algorithmes ont été créés en se basant sur cette idée, comme les algorithmes Two-Phase, IHUP et UPGrowth [2, 15].

Afin de relever les défis de la recherche d'itemsets à haute utilité (HUIM) et de réduire l'es-

pace de recherche, les algorithmes à deux phases utilisent la mesure d'utilité pondérée par les transactions, également connue sous le nom de *Transaction-Weighted Utilization* (TWU), qui représente une borne supérieure de la mesure d'utilité [16]. La TWU permet de minimiser l'espace de recherche de manière sûre et d'ignorer les itemsets à faible utilité tout en ne manquant aucun ensemble d'éléments à haute utilité. La mesure TWU est utilisée par les algorithmes basés sur deux phases dans leurs stratégies d'élagage afin d'éliminer les itemsets non promoteurs (itemsets à faible utilité).

Etant donnée une base de données transactionelle quantitative D, pour pouvoir calculer la mesure TWU des itemsets de D, il est nécessaire d'abord de commencer par le calcule des utilités des transactions de D :

Definition 9 (Utilité d'une transaction) : L'utilité d'une trancation T_c de la base D est la somme des utilités de tous les items inclus dans T_c. l'utilité d'une transaction T_c est représentée par $u(T_c)$ et calculée comme suit : $TU(T_c) = \sum_{x \in T} u(x, T_c)$.[16].

Exemple :

On prend toujours la base représentée par les Tableaux 3.2 et 3.3, $TWU(T3) = u(a,T3) + u(c,T3) + u(e,T3) = (2 \times 5) + (6 \times 1) + (2 \times 3) = 22$.

Definition 10 (TWU d'un itemset) : TWU d'un itemset X est la somme des utilités des transactions qui contiennent X. En d'autres termes, $TWU(X) = \sum_T tu(T_c) \quad \forall c \in g(X)$.

Exemple :

L'itemset bc apparaît dans trois transactions $T0$, $T1$ et $T4$. Donc, $TWU(bc) = u(T0) + u(T1) + u(T4) = 25 + 20 + 9 = 54$.

En se basant sur la mesure TWU, la propriété suivante est utilisée par les algorithmes à deux phases pour élaguer l'espace de recherche :

Propriété 1 (Elagage de l'espace de recherche à l'aide de la borne superieure TWU) : Pour tout itemset X, si $TWU(X) < MinUtil$, cela signifie que X est un itemset à faible utilité, ainsi que tous ses sur-ensembles. Par conséquent, on peut ignorer cet itemset X ainsi que tous ses sur-ensembles.

3.6.1 Fonctionnement des algorithmes à deux phases

Les algorithmes à deux phases fonctionnent de la façon suivante [16] :

- Phase 1 : Dans cette phase, ces algorithmes calculent d'abord le TWU des itemsets dans l'espace de recherche. Pour un itemset X :
  - Si $TWU(X) < MinUtil$, alors X et tous ses supersets ne peuvent pas être des itemsets à haute utilité. Ils peuvent donc être éliminés de l'espace de recherche sans qu'il soit nécessaire de calculer leurs TWU.
  - Sinon, X et ses supersets peuvent être sélectionnés comme itemsets à haute utilité. Ainsi, X est conservé et sauvegardé en mémoire en tant qu'itemset candidat à haute utilité et ses supersets seront explorés.
- Phase 2 : On calcule l'utilité exacte de chaque itemset candidat X trouvé lors de la phase 1 en parcourant la base de données. Si $u(X)$ est supérieur à $MinUtil$, alors X sera conservé car X est un itemset à haute utilité.

3.6.2 Avantages des processus à deux phases

- Ces algorithmes surestiment certains itemsets à faible utilité durant la première phase, mais ils ne sous-estiment jamais les itemsets à haute utilité. Par conséquent, ces algorithmes garantissent l'obtention de tous les itemsets à haute utilité dans la base [16].
- Les algorithmes à deux étapes ont la capacité de repérer tous les itemsets essentiels tout en minimisant l'espace de recherche afin d'améliorer leurs résultats [16].

3.6.3 Inconvénients des algorithmes à deux phases

- Les algorithmes basés sur deux phases peuvent entraîner des dépenses en temps de calcul et en ressources, notamment pour les ensembles de données volumineux ou les problèmes de grande taille.
- Ces algorithmes explorent l'espace de recherche sous la forme d'un graphe sans utiliser la base de données. Par conséquent, cette exploration peut créer des itemsets candidats qui ne sont même pas présents dans la base de données [14].
- Ces algorithmes effectuent des scans répétitifs de la base de données afin de déterminer les TWU et les utilités précises des itemsets. Les scans répétitifs de la base entraînent une durée très longue de ces algorithmes [7].

3.7 Algorithmes basés sur une phase

Par la suite, afin de contourner les contraintes des algorithmes basés sur deux étapes, une deuxième catégorie d'algorithmes a vu le jour avec l'émergence de l'algorithme HUI-Miner qui ne requiert qu'une seule étape pour découvrir HUIM [17].

Une structure appelée *utility-list* est utilisée par HUI-Miner pour stocker à la fois des informations d'utilité sur les itemsets et des informations heuristiques afin de réduire l'espace de recherche et d'éviter la génération coûteuse de nombreux itemsets candidats, comme le font les algorithmes de deux phases[17].

Les algorithmes basés sur une seule phase sont plus efficaces que les algorithmes de deux phases en raison de l'introduction de nouvelles bornes supérieures (upper-bounds) sur l'utilité. Ces bornes supérieures sont basées sur l'utilité précise des itemsets, ce qui permet d'élaguer une plus grande partie de l'espace de recherche par rapport à la mesure TWU.

Ces bornes supérieures comprennent les utilités restantes (utilité restante) ainsi que des mesures plus récentes comme l'utilité locale (utilité locale) et l'utilité de sous-arbre (utilité de sous-arbre) [2].

L'*utility-list* structure est une façon pour représenter les itemsets. Cette représentation permet de calculer rapidement l'utilité des différents itemsets sans besoin de scanner la base de données. De plus, *utility-lists* de itemsets larges peuvent être rapidement créées en joignant des *utility-lists* des itemsets courts [7].

Definition 11 (Utility-list d'un itemset X) : Etant donnée une base de données quantitative D, un ensemble d'items $I = \{i_1, i_2, \ldots, i_n\}$ et une relation d'ordre total $\succ$ prédéfinie sur l'ensemble des items I. L'utilité-liste de X, dénotés par $ul(X)$, est un ensemble de tuples de la forme (Tid, iutil, rutil). Chaque tuple représente une transaction contennat l'itemset X. Plus précisement, Tid est l'identifiant de la transaction qui contient X. $iutil$ correspond à l'utilité de X dans la transaction Tid, tandis que la fonction $rutil$ correspond à l'utilité restante définie par [17] :

$$rutil(X) = \sum_{i \in T} u(i, \text{Tid})$$

L'utilité exacte d'un itemset X peut être facilement déterminée à l'aide de sa *utility-list* sans besoin de scannner la base. Plus précisément, l'utilité de l'itemset X est obtenue en sommant

toutes les valeurs *iutil* présentes dans sa *utility-list*.

Exemple :

Prenons toujours la base transactionelle présentée dans les tableaux 3.2 et 3.3, Les utilité-lists des itemsets {a}, {d} et {a,d} sont illustrées par la Figure 3.1.

The utility-list of $\{a\}$

tid	iutil	rutil
T0	5	20
T2	5	3
T3	10	12

The utility-list of $\{d\}$

tid	iutil	rutil
T0	6	3
T1	6	3
T2	2	0

The utility-list of $\{a,d\}$

tid	iutil	rutil
T0	11	3
T2	7	0

FIGURE 3.1 – *Les utility-lists des itemsets {a}, {d} et {ad}.*

L'utilité exacte de l'itemset {ad} est $u\{ad\} = 11+7 = 18$.

De plus, il est possible d'utiliser les *utility-lists* pour élaguer l'espace de recherche en utilisant une nouvelle borne supérieure appelée la borne supérieure de l'utilité restante, en anglais, *remaining utility upper bound.* Cette borne supérieure est plus plus éfficace que la borne TWU.

Definition 12 (*Remaining utility upper-bound (reu)* : Etant donné un itemset X avec sa *utility-list* correspondante $ul(X)$, Remaining utility upper-bound (reu) de X, denotée par $ru(X)$, est la somme de toutes les $iutil$ et $rutil$ de X. Formellement :

$$ru(X) = \sum_{e \in ul(X)} (e.iutil + e.rutil).$$

Exemple :

La Figure 2.1 montre que l'utilité restante de l'itemset {ad} est :

$$ru(\{ad\}) = 11+7+3+0 = 21.$$

Propriété 2 (Élagage de l'espace de recherche à l'aide d'une liste d'utilitaires en utilisant la borne supérieure remaining utility) :

Etant donné un itemset X avec sa *utility-list* $ul(X)$, si la somme des valeurs $iutil$ et $rutil$ dans $ul(X)$ est inférieure au seuil d'utilité minimum ($MinUtil$), X et toutes ses extensions (supersets) sont des itemsets de faible utilité.

Exemple :

La Figure 2.1 montre que $ru(\{ad\}) = 11+7+3+0 = 21$. En choisissant 25 comme valeur de seuil d'utilité, $MinUtil = 25$, l'utilisateur éliminera l'itemset {ad} avec tous ses sur-ensembles car ils ont sertainement une utilité inférieur à $MinUtil = 25$.

3.8 L'algorithme FHM (Faster High-Utility Itemset Mining Algorithm)

L'algorithme FHM, qui est l'acronyme de *Faster High-Utility Itemset Mining Algorithm*, est l'un des algorithmes les plus populaires de HUIM [10]. FHM est basé sur une seul phase et utilise la structure *utility-list* pour extraire les itemsets a haute utilité. Il a été démontré que FHM a une vitesse de 7 fois supérieure à celle d'HUI-Miner [11].

Algorithme 1 prsenté le processus de la fouile des itemsets à haute utilité avec l'algorithme FHM. FHM necessite deux entrée : (1) Une base de données transactionnelle quantitative D et (2) Le seuil minimal d'utilité ($MinUtil$) spécifié par l'utilisateur sous forme d'une valeur ou percentage. FHM opte pour une approche de recherche en profondeur (DFS) afin d'explorer l'espace de recherche et retourner tous les itemsets à haute utilité dans D [10]. Notez que, durant le processs de l'exploration des itemsets, trois stratégies d'élagage sont adopter pour accélérer l'exploration et éviter le traitement des itemsets non promoteurs. En d'autre termes, les itemsets à faible utilité.

Algorithm 1 *The FHM algorithm.*

Input : D : a transaction database, $MinUtil$: a user-specified threshold
Output : the set of high-utility itemsets

1. Scan D to calculate the TWU of single items ;
2. $I^* \leftarrow$ each item i such that TWU(i) $\geq$ MinUtil ;
3. Let $\succ$ be the total order of TWU ascending values on I^* ;
4. Scan D to build the *utility-list* of each item $i \in I^*$ and build the EUCS ;
5. Search($\emptyset$, I^*, MinUtil, EUCS) ;

3.8.1 Les étapes principales de FHM

Nous allons maintenant expliquer les étapes de FHM :

Etape1 : Calculer la mesure TWU des items initiaux et eliminer les items non promoteurs

Initialement, FHM calcule la mesure TWU de chaque item dans la base de données (line 1). La mesure TWU d'un item est la somme des transactions contenant cet item. FHM applique ensuite la première stratégie d'élagage et élimine tous les items ayant une TWU inférieure à $MinUtil$ et conserver seulemnt les items promoteurs (line 2).

Etape 2 : Construire les *Utility-lists* des itemset promoteurs et construire la structure EUCS

FHM scane la base de données pour la deuxième fois pour construire les *utility-lists* des itemsets initiaux qui ont été conservé dans la premiére étape. De plus, FHM construire la structure de données EUCS (Estimated Utility Co-Occurrence Structure).

Defintion 13 (La structure *EUCS (Estimated Utility Co-Occurrence Structure)*) :

Etant donné une base de données quantitative D, EUCS est une structure de données composée d'un ensemble de triplets de la forme (a, b, c) sachant que : a et b sont deux items de la base D et c est la valeur de TWU de l'itemset {ab}. Formellement :

$$EUCS = (a,b,c = TWU(ab))/aetb \in I^* etc \in R^+$$

EUCS contient tous les co-occurrences des items de la base D avec les valeurs TWU correspondantes. Cette structure sera ensuite exploitée par FHM pour élaguer l'espace de recherche d'une façon éfficace.

La figures ci-dessous présentent l'Utilité Totale (TU) et les valeurs de l'Utilité de Transaction Pondérée (TWU) ainsi que la structure de la Co-occurrence d'Utilité Estimée (EUCS), qui correspondent à la base de données illustrée dans les tableaux 3.2 et 3.3.

TID	TU
T_0	25
T_1	20
T_2	08
T_3	22
T_4	09

Items	TWU
a	55
b	54
c	84
d	53
e	76

Items	a	b	c	d
b	25			
c	55	54		
d	33	45	53	
e	47	54	76	45

FIGURE 3.2 – *Utilités des transactions (à gauche), valeurs TWU (au centre) et structure EUCS (à droite).*

Etape 3 : Effectuer la procédure de recherche récursive

Une fois que l'EUCS est construit, on entame une exploration approfondie des itemsets en utilisant la procédure de recherche récursive (ligne 5). La Figure 3.3 illustre la procédure de recherche récursive. L'objectif de cet algorithme consiste à représenter l'ensemble des itemsets possibles sous forme d'un graphe, puis à effectuer une recherche par profondeur afin de l'explorer.

L'algorithme commence des itemsets de taille 1 et explore les itemsets de plus grande taille en effectuant le processus de construction (Voir line 10 de Figure 3.3).

La Figure 3.4 présente le processus de construction. La construction des *utility-lists* d'itemsets plus grands nécessite de combiner les *utility-lists* des petits itemsets. La réalisation de cette construction ne nécessite pas de scanner la base de données.

Algorithm 2: The *Search* procedure

input : P: an itemset, *ExtensionsOfP*: a set of extensions of P, the *minutil* threshold, the $EUCS$ structure
output: the set of high-utility itemsets

```
1  foreach itemset Px ∈ ExtensionsOfP do
2      if SUM(Px.utilitylist.iutils) ≥ minutil then
3          output Px;
4      end
5      if SUM(Px.utilitylist.iutils)+SUM(Px.utilitylist.rutils) ≥ minutil then
6          ExtensionsOfPx ← ∅;
7          foreach itemset Py ∈ ExtensionsOfP such that y ≻ x do
8              if ∃(x, y, c) ∈ EUCS such that c ≥ minutil) then
9                  Pxy ← Px ∪ Py;
10                 Pxy.utilitylist ← Construct (P, Px, Py);
11                 ExtensionsOfPx ← ExtensionsOfPx ∪ Pxy;
12             end
13         end
14         Search (Px, ExtensionsOfPx, minutil);
15     end
16 end
```

FIGURE 3.3 – *La procédure de recherche récursive de FHM.*

Un exemple de construction de l'*utility-list* de l'itemset ad à partir des itemsets a et d est présenté la Figure 3.5.

Il est important de noter que FHM ne procède pas de manière naïve au processus de construction, mais vérifie d'abord s'il est nécessaire de le faire ou non. Cette vérification est effectuée en utilisant deux stratégies d'élagage afin d'éviter de créer des *utility-lists* pour des itemsets qui ne sont certainement pas promoteurs.

FHM adopte deux méthodes d'élagage : (1) l'élagage par l'utilité restante *(remaining utility)* et (2) l'élagage par co-occurrence d'utilité (co-occurrence-based pruning). Nous avons déjà donné une explication approfondie de l'utilité restante ; le lecteur peut consulter *Définition 12* et *Propriété 2* pour obtenir plus de détails.

Nous allons maintenant expliquer l'élagage par co-occurrence, qui est basé sur la structure EUCS présentée précédemment. L'élagage par co-occurrence, également connu sous le nom co-occurrence based pruning, repose sur la structure EUCS et il est défini par la propriété suivante :

Propriété 4 (Élagage de l'espace de recherche à l'aide l'co-occurrence d'utilité) : Soit une base de données transactionnelle *D* avec sa structure EUCS correspondante, et deux items

Algorithm 3: The Construct procedure

input : P: an itemset, Px: the extension of P with an item x, Py: the extension of P with an item y
output: the utility-list of Pxy

```
1  UtilityListOfPxy ← ∅;
2  foreach tuple ex ∈ Px.utilitylist do
3      if ∃ey ∈ Py.utilitylist and ex.tid = exy.tid then
4          if P.utilitylist ≠ ∅ then
5              Search element e ∈ P.utilitylist such that e.tid = ex.tid.;
6              exy ← (ex.tid, ex.iutil + ey.iutil − e.iutil, ey.rutil);
7          end
8          else
9              exy ← (ex.tid, ex.iutil + ey.iutil, ey.rutil);
10         end
11         UtilityListOfPxy ← UtilityListOfPxy ∪ {exy};
12     end
13 end
14 return UtilityListPxy;
```

FIGURE 3.4 – *L'algorithme de construction de FHM.*

The utility-list of $\{a\}$

tid	iutil	rutil
T0	5	20
T2	5	3
T3	10	12

The utility-list of $\{d\}$

tid	iutil	rutil
T0	6	3
T1	6	3
T2	2	0

The utility-list of $\{a,d\}$

tid	iutil	rutil
T0	11	3
T2	7	0

FIGURE 3.5 – *Un exemple de construction de l'itemset ad à partir des items a et d.*

x et y de D. On extrait le triple (a,b,c) de la structure EUCS, sachant que $a = x$ et $b = y$. Si la valeur de c est inférieure au seuil $MinUtil$, alors l'itemset xy ainsi que tous ses sur-ensembles sont des itemsets non promoteurs (itemsets à faible utilité). Par conséquent, on peut les éliminer.

Etape 4 : Retourner l'ensemble de tous les itemsets à haute utilité

Durant la procédure de recherche récursive, l'algorithme vérifie l'utilité exacte des itemsets courants et conserve ceux qui ont une haute utilité. Une fois que la procédure récursive est terminée, FHM retourne à l'utilisateur l'ensemble de tous les itemsets à haute utilité.

3.9 Les algorithmes HGB & HGB-All

HGB et HGB-All sont deux algorithmes proposés pour l'extraction de règles d'association à haute utilité [18]. HGB est l'acronyme de *High Utility Generic Basis*. HGB-All est conçu pour trouver toutes les règles d'association à haute utilité, tandis que l'algorithme HGB trouve uniquement les règles d'association non redondantes, qui constituent un sous-ensemble de toutes les règles.[18]

Les techniques d'exploration de règles d'association sont étudiées de manière exhaustive par les chercheurs pour découvrir les relations entre les ensembles d'éléments à l'aide de mesures statistiques de support et de confiance. Cependant, ils ne fournissent aucune implication sémantique entre les ensembles d'éléments, telle qu'une relation par rapport à l'utilité. Pour fournir une relation sémantique entre les ensembles d'éléments, nous avons utilisé la mesure de confiance d'utilité pour exploiter les règles d'association dans l'extraction d'ensembles d'éléments à haute utilité [18].

Avant de présenter les algorithmes permettant d'obtenir des règles à haute utilité, nous devons d'abord expliquer ce qu'est une règle à haute utilité et comment elle est déterminée avec la mesure *utility-confidence.*

Définition 14 (Les régles d'association) : Une régle d'association est une implication de la forme : $R : X -> Y$, où $X, Y \in I$ et I est l'ensemble des items. X est connu comme antécédent du règle, Y est appelé conséquent de la règle, et $X \cap Y$.

Définition 15 (la mesure *utility-confidence* d'une régle) : l'*utility-confidence* d'une régle R est notée $uconf(R)$ et définie par :

$$uconf(R) = \frac{luv(X, X \cup Y)}{u(X)}$$

où $luv(X, X \cup Y)$ est la valeur d'utilité locale de X dans l'itemset $X \cup Y$ et définie par la somme des valeurs d'utilité X dans toutes les transactions contenant $X \cup Y$.

L'*utility-confidence* de la règle $R : X -> Y$ reflète la part d'utilité de itemset X à l'utilité de $X \cup Y$. Cela signifie que la fraction de l'utilité de X est couverte par $X \cup Y$. Elle est similaire à la mesure de support-confiance classique, sauf qu'elle est basée sur la valeur d'utilité d'itemsets.

3.9.1 Définition formelle des régles d'association à haute utilité, High Utility Association Rules Mining (HUARM)

On dit qu'une régle de la forme $X->Y$ est une régle d'association à haute utilité si les trois conditions suivante sont vérifiées :

1. L'antécédent X est un itemset à haute utilité.
2. $X \cup Y$ est un itemset à haute utilité.
3. L'*utility-confidence* de la règle $X->Y$ n'est pas inférieure au seuil de confiance minimum $MinConf$. Notez que, l'*utility-confidence* est définie comme la somme des valeurs d'utilité des éléments de X dans les transactions contenant $X \cup Y$, divisée par l'utilité de X.

3.9.2 Les étapes principales de l'algorithme HGB & HGB-All

Le processus général pour obtenir les règles d'association à haute utilité est présenté dans l'Algorithme 2. L'utilisateur peut sélectionner soit d'avoir toutes les règles à haute utilité dans la base (ALL-HAR), soit d'obtenir uniquement les règles non redondantes (NHAR).

Algorithm 2 *The overall process of generation of utility based non-redundant association rules.*

Input : D : a transaction database, *MinUtil* : a user-specified utility threshold, *MinConf* : a user-specified utility confidence,
Output : the set of high-utility association rules

1. Construct the initial *utility-lists* of items ;
2. Discover the set of HUIs ;
3. Discover the set of High utility closed itemsets (HUCI) and generators ;
4. Generate the set of non high utility association rules (NHARs) ;
5. **if** *The user looks for all HARs* **then**
6. Generate all HARs form NHARs
   end
7. Output the desired rules

HGB ou HGB_All prend en entrée une base de données transactionnelle avec des informations d'utilité, un seuil d'utilité minimum $MinUtil$ (un entier positif) et un seuil de confiance minimum $MinConf$ (une valeur double représentant un pourcentage). Voici les étapes à suivre pour obtenir les règles à haute utilité :

Étape 1 : Construire les *Utility-lists* des itemsets initiaux

Comme pour l'algorithme FHM, il est d'abord nécessaire de construire les *utility-lists* des items initiaux.

Étape 2 : Extraire les itemsets à haute utilité

La deuxième étape consiste à extraire tous les itemsets à haute utilité en utilisant un algorithme similaire à l'algorithme FHM appelé FHIM [18]. FHIM contient des stratégies d'élagage supplémentaires pour accélérer l'exploration de l'espace de recherche.

Étape 3 : Extraire les itemsets à haute utilité fermés (*Closed high utility itemsets (CHUIs)*)

La troisième étape consiste à extraire tous les itemsets à haute utilité fermés en utilisant l'algorithm HUCI-Miner, qui représentent une des représentations concises des patterns. Un itemset à haute utilité est appelé fermé s'il n'a pas de sur-ensembles apparaissant dans les mêmes transactions (ayant le même support), c'est-à-dire :

$$CHUIs = X | X \in HUIs \; et \; il \; n'existe \; pas \; Y \; \in HUIs \; tels \; que \; X \subset Y \; et \; Supp(X) = Supp(Y)$$

Notez que les représentations concises sont utilisées pour obtenir un ensemble de patterns minimal mais représentatif et éviter d'obtenir un grand nombre d'itemsets, car cela compliquerait l'utilisation et l'interprétation d'un grand nombre de patterns. Dans ce contexte, HGB et HGB-All utilisent les itemsets fermés (closed itemsets) pour obtenir par la suite un ensemble de règles sans redondance.

Étape 4 : Extraire les règles non redondantes à haute utilité

À partir de l'ensemble des itemsets à haute utilité fermés, l'algorithme HGB va découvrir les règles à haute utilité non redondantes. En général, une règle d'association non redondante est une règle avec un antécédent minimal et un conséquent maximal. Dans le cadre de l'*utility-confidence*, une règle non redondante est appelée une règle générique de haute utilité *(high utility generic rule, HGR).* L'ensemble de toutes les règles non redondantes avec haute utilité est appelé *NHARs (Non-redundant High Utility Association Rules).*

Étape 5 : Extraire toutes les règles à haute utilité

Cette étape est optionnelle, elle est effectuée si l'utilisateur souhaite obtenir toutes les règles à haute utilité (*all high utility association rules, All-HARs*).

3.10 Comparaison entre l'algorithme FHM et lgorithme HGB-ALL

L'algorithme HGB-ALL (High Utility Association Rules Mining with All Items)a été suggéré par Sahoo et ses collègues (2015) afin d'extraire des règles d'association à grande utilité. Cet algorithme de post-traitement exploite les résultats de l'algorithme HUCI-Miner afin d'extraire ces règles.

En revanche, l'algorithme FHM (Fast High Utility Itemset Mining) est un algorithme qui permet d'extraire des ensembles d'éléments à grande utilité. Il emploie des méthodes sophistiquées comme la Structure de Co-Occurrence d'Utilité Estimée (EUCS) et la liste d'utilité afin d'optimiser l'extraction des ensembles d'éléments à grand usage [19].

3.11 Conclusion

Ce chapitre présente une analyse approfondie du domaine de la fouille de motifs. Tout d'abord, nous avons exposé le problème fondamental de ce domaine, qui consiste à fouiller les itemsets fréquents. Par la suite, nous avons fourni une explication détaillée de la recherche des itemsets à haute utilité en utilisant FHM, l'un des algorithmes les plus répandus dans cette discipline. Enfin, nous avons évoqué une extension extrêmement importante du problème de la fouille des itemsets à haute utilité (HUIM). Cette extension consiste à chercher les règles d'association à haute utilité en utilisant les algorithmes HGB et HGB_All.

Chapitre 4

Implémentation et Interprétation

4.1 Introduction

Dans le chapitre *"Implémentation et interprétation"*, nous appliquons les algorithmes présentés dans cette étude pour extraire les itemsets à haute utilité (HUIs) et les règles à haute utilité (HARs). L'application de ces algorithmes est effectuée sur deux bases de données réelles : l'une contient les ventes réalisées dans une pharmacie et l'autre contient les ventes d'une boutique de fruits. L'extraction des motifs de ces bases est suivie par une interprétation des résultats obtenus.

Nous commecons par la présenation des outils de développement utilisés dans notre étude tels que *Java*, *Python*, *NetBeans*, et la bibliothèque *SPMF* pour extraire les ensembles des patterns désirés. De plus, nous détaillons la configuration de ces outils et comment nous avons intégré les algorithmes de HUIM dans notre environnement de développement.

Pour les langages de programmation, notre choix s'est porté sur *Java* et *Python*. *Python*, connu pour sa simplicité et sa flexibilité en tant que langage interprété, il a été utilisé pour convertir notre base de données de transactions au format SPMF, profitant de ses bibliothèques riches pour le traitement et l'analyse des données. En parallèle, *Java*, réputé pour sa polyvalence et sa puissance, a été employé dans notre projet pour exploiter la bibliothèque SPMF spécialisée dans l'exploration de modèles et de bases, facilitant ainsi l'implémentation des algorithmes d'exploration de modèles.

Nous avons intégré la bibliothèque *SPMF (Sequential Pattern Mining Framework)* dans notre environnement de développement, SPMF est une bibliothèque *Java* spécialisée dans la fouille de motifs et de règles. Cette intégration a été cruciale pour la mise en œuvre des algorithmes FHM et HGB-All. En utilisant Java, nous avons pu bénéficier de la performance et de la fiabilité de SPMF, tout en ayant la flexibilité nécessaire pour manipuler et analyser les résultats obtenus.

Finalement, pour compléter ce chapitre, nous avons mené une étude comparative évaluant les performances des deux algorithmes sur différentes bases de données, notamment en termes de temps d'exécution, d'utilisation de la mémoire et de nombre de motifs découverts. De plus, nous avons tenté d'interpréter certains résultats en analysant quelques patterns extraits lors de l'application des algorithmes.

4.2 Outils de développement et langages utilisés

4.2.1 Java

Java est un langage de programmation polyvalent et puissant, lancé par *Sun Microsystems* en 1995. Il offre une portabilité, une sécurité et des performances élevées, grâce à la JVM. Sa communauté active et ses nombreuses bibliothèques en font un choix populaire pour développer différentes applications. En bref, *Java* est largement utilisé dans le monde numérique pour sa fiabilité et sa polyvalence [20].

4.2.2 Python

Python, conçu par *Guido van Rossum* et publié en 1991, est un langage de programmation populaire utilisé dans divers domaines tels que le développement web backend, l'analyse de données, l'intelligence artificielle et l'informatique scientifique. Il est prisé pour sa compatibilité multi-plateforme avec *Windows*, *Mac*, *Linux* et *Raspberry Pi*.Sa polyvalence, sa puissance et sa facilité d'utilisation en font un outil précieux pour la création d'applications Web, l'analyse de données, l'intelligence artificielle et bien plus encore [21].

4.2.3 NetBeans

NetBeans est un environnement de développement intégré (IDE) répandu pour le langage de programmation *Java*, il fournit une gamme complète d'outils et de fonctionnalités pour aider les développeurs à coder, déboguer et déployer des applications *Java*. *NetBeans* est Open source et disponible gratuitement. il prend en charge divers frameworks et technologies, tels que *JavaFX*, *Spring*, *Hibernate* et *Maven*, et il est également compatible avec les systèmes d'exploitation *Windows*, *MacOS* et *Linux* [22].

4.2.4 La bibliothèque SPMF (Sequential Pattern Mining Framework)

SPMF, une bibliothèque de data-mining open source basée sur *Java*, *SPMF* se spécialise dans les tâches de la fouille des motifs. Avec 262 algorithmes, il couvre la fouille des règles d'association, le clustering, la fouille des itemsets fréquents, des itemsets à haute utilité, etc. Sa légèreté et sa facilité d'utilisation le rendent polyvalent. Les capacités d'intégration de *SPMF* s'étendent aux projets *Java* et à d'autres langages comme *Python* et *R*. La version 2.62, publiée le 12 juin 2024, marque sa dernière mise à jour [23].

4.3 Le premier cas d'étude : La base *Pharmacie*

4.3.1 Présentation de la base *Pharmacie*

Notre premier cas d'étude est une base de données provenant d'une pharmacie. Cette ressource essentielle regroupe des informations détaillées sur les produits disponibles dans une pharmacie. Les données collectées sont organisées dans deux fichiers Excel distincts. Le premier fichier contienent les médicamenets achetées par les différents clients et le deuxième fichier contient la liste de tous les produits avec leurs informations.

Cette base contient les enregistrements réels des transactions des clients entre le 12/09/2022 et le 21/03/2023. Le nombre total de produits dans la base est 672, et le nombre total de transactions s'élève à 2720.

Les deux figures suivantes offrent un aperçu visuel des données contenues dans ces deux fichiers.

	A	B	C	D	E	F	G	H	I
1	N	Date	Paiement	RAISON	Code	Désignation	Quantite	PrixU	Montant HT
2	0001/2022	9/5/2022	Espèces	CLIENT 1	000001	AVIMEC 100ML	1,00	1 248,95	1 248,95
3	0002/2022	11/5/2022	Espèces	CLIENT 2	000018	NOBIVAC DHP 10X1 DS	10,00	654,00	6 540,00
4	0002/2022	11/5/2022	Espèces	CLIENT 2	000019	NOBIVAC L 10X1 DS	10,00	397,00	3 970,00
5	0003/2022	12/5/2022	Espèces	CLIENT 3	000111	VITAMINE AD3E ARTIMON 1L	2,00	2 300,00	4 600,00
6	0004/2022	12/5/2022	Espèces	CLIENT 1	000059	DIURIZAL PS	1,00	763,00	763,00
7	0005/2022	14/5/2022	Espèces	CLIENT 4	000057	DEXAPRO INJ 100ML	1,00	630,00	630,00
8	0005/2022	14/5/2022	Espèces	CLIENT 4	000084	MULTIVAL 250 ML	1,00	1 660,00	1 660,00
9	0005/2022	14/5/2022	Espèces	CLIENT 4	000091	OX AL 20% 250ML	2,00	1 080,00	2 160,00
10	0006/2022	14/5/2022	Espèces	CLIENT 5	000104	TOXIDREN POTS 1 KG	4,00	2 081,00	8 324,00
11	0007/2022	14/5/2022	Espèces	CLIENT 3	000073	INTROVIT A 1L	3,00	1 770,00	5 310,00
12	0007/2022	14/5/2022	Espèces	CLIENT 3	000012	ULTRAMIN 1KG	6,00	1 780,00	10 680,00
13	0008/2022	15/5/2022	Espèces	CLIENT 3	000072	HIPRAVIAR -CLON	22,00	640,00	14 080,00
14	0009/2022	15/5/2022	Espèces	CLIENT 6	000016	VETECARIOL 100ML	1,00	1 600,00	1 600,00
15	0009/2022	15/5/2022	Espèces	CLIENT 6	000013	AZIUM 50ML	2,00	1 392,00	2 784,00
16	0009/2022	15/5/2022	Espèces	CLIENT 6	000044	CALMIVET 100 ML	1,00	809,00	809,00
17	0009/2022	15/5/2022	Espèces	CLIENT 6	000110	VITAMINE C VTQ 50 ML	1,00	696,00	696,00
18	0009/2022	15/5/2022	Espèces	CLIENT 6	000050	COREBRAL ING	1,00	1 006,00	1 006,00
19	0010/2022	15/5/2022	Espèces	CLIENT 1	000068	HIPRALONA ENRO S 1L	2,00	4 500,00	9 000,00
20	0011/2022	15/5/2022	Espèces	CLIENT 2	000077	KILTIX GM	5,00	1 450,00	7 250,00

FIGURE 4.1 – *Un extrait du fichier contenant la liste des transactions de la base Pharmacie.*

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1						LISTE DES PRODUITS											
2	Arrêté au :			21/03/2023													
3		Code			Désignation			U.M			Famille	Prix Vente		Prix Achat		St.Initial	
4		1	AVIMEC 100ML				U			ANTIPARASITAIRE			1248,95		1049,54		0
5		2	AVIMEC 50ML				U			ANTIPARASITAIRE			679,25		596,33		0
6		3	AVICYCLINE 250ML				U						1452		1270,64		0
7		4	AVICORT INJ 100ML				U						655,05		550,05		0
8		5	AVITRYL 500ML				U						1991,93		1732,11		0
9		6	AVITRYL 1L				U						3466,02		3040,37		0
10		7	DOXYSTIN 500G				U						1645,87		1444,95		0
11		8	COCCIDIOPAN 200G				U						1009,86		848,62		0
12		9	COCCIDIOPAN 500G				U						2393,67		2045,87		0
13		10	COCCIDIOPAN 1KG				U						4415,13		3876,15		0
14		11	ULTRAMIN 500G				U						0		798,32		0
15		12	ULTRAMIN 1KG				U						0		1484,87		0
16		13	AZIUM 50ML				U						1345,89		1160,25		0
17		14	METHIO-B12 250ML				U						1784,13		1651,97		1
18		15	PANACURE 2.5 SU 1*2.5L				U						5684,74		5030,74		0
19		16	VETECARIOL 100ML				U						1641		8097,72		0
20		17	NOBIVAC TRICAT TRIO 10 X 1 DS				U						750,2		682		0
21		18	NOBIVAC DHP 10X1 DS				U						691,31		544,79		0
22		19	NOBIVAC L 10X1 DS				U						415,1		330,34		0
23		20	NOBIVAC SOLVANT 10X1 DS				U						0		0		0
24		21	CARNET DE VACCIN CHAT				U								0		0
25		22	CARNET DE VCCIN CHIEN				U						0		0		0
26		23	GANAMIC250ML/VITAMINEB+SELINI				U						2120		1964,48		0

FIGURE 4.2 – *Un extrait du fichier contenant la liste des produits de la base Pharmacie.*

4.3.2 Prétraitement de la base *Pharmacie*

La base de données ne peut pas être utilisée directement avec les algorithmes FHM et HGB-All de SPMF, car SPMF nécessite que la base de données soit dans un format spécifique (format SPMF). Après avoir nettoyé les données pour éliminer les doublons, gérer les valeurs

manquantes ou aberrantes, et vérifier la cohérence des informations telles que les quantités vendues et les montants de vente, nous avons implémenté le processus de conversion au format SPMF à l'aide d'un script *Python*. Voici un aperçu de ce script *Python* pour effectuer cette conversion.

```python
items = ""
profits = ""
sum_trans = 0
past_client = emv["RAISON"][0]
past_date = emv["Date"][0]

with open('data_phrmacie.txt', 'w') as file:
    for index in emv.index:
        current_client = emv["RAISON"][index]
        current_date = emv["Date"][index]
        print("--", current_client, "--", index)

        if current_client == past_client and current_date == past_date:
            item = emv["Code"][index]
            profit = emv["Montant HT"][index]  # Correction du nom de colonne
            sum_trans = sum_trans + emv["Montant HT"][index]
            items = items + str(item) + " "
            profits = profits + str(profit) + " "
            print(items)
            print(profit)
            print(sum_trans)
        else:
            with open('data_phrmacie.txt', 'a') as file:
                items = items[:-1]
                profits = profits[:-1]
                file.writelines(items + ":" + str(round(sum_trans, 2)) + ":" + profits + "\n")

            past_client = current_client
            past_date = current_date
            items = str(emv["Code"][index]) + " "
            profits = str(emv["Montant HT"][index]) + " "  # Correction du nom de colonne
            sum_trans = emv["Montant HT"][index]

        if index == emv.index.stop - 1:
            print("we are in the last")
            items = items[:-1]
            profits = profits[:-1]
            with open('data_phrmacie.txt', 'a') as file:
                file.writelines(items + ":" + str(round(sum_trans, 2)) + ":" + profits + "\n")
```

FIGURE 4.3 – *Script Python utilisé pour formater les données au format SPMF.*

Après le passage au format SPMF, la base de données est représentée sous forme de transactions où chaque transaction correspond à un achat d'un client. La Figure 4.4 donne un aperçu de la base de données après la conversion au format SPMF.

De plus, nous avons créé un script en Python afin de transformer un format SPMF contenant des valeurs décimales en un format SPMF avec des valeurs entières. Cette conversion est indispensable, car certains algorithmes de HUIM, y compris FHM, n'acceptent que les valeurs

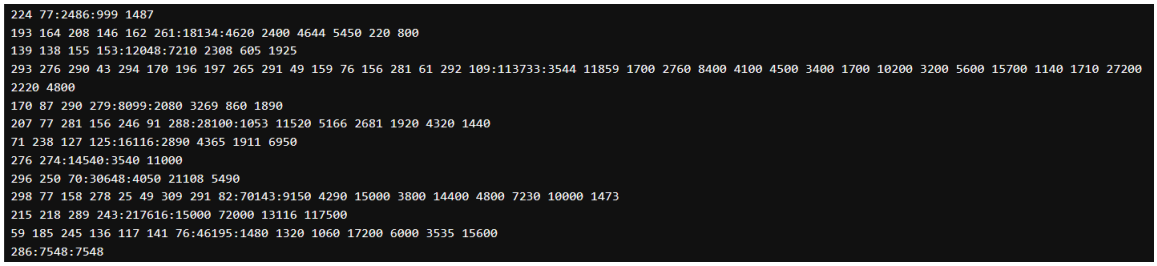


FIGURE 4.4 – *Un extrait du fichier SPMF de la base Pharmacie.*

entières. Le script *Python* permettant d'effectuer cette conversion est illustré dans la Figure 4.5.

```python
# Spécifier le chemin du fichier d'entrée (format SPMF avec des valeurs de type flottant)
input_file = r"C:/Users/pc/Desktop/pratique/phrmacie/cnvSPMF/data_phrmacie.txt"

# Spécifier le chemin du fichier de sortie (format SPMF avec des valeurs de type entier)
output_file = "data_1.txt"

# Lire le contenu du fichier d'entrée et convertir les valeurs en entiers
with open(input_file, 'r') as f_in, open(output_file, 'w') as f_out:
    for line in f_in:
        # Diviser la ligne en éléments individuels en utilisant ':' comme délimiteur
        items_with_spaces = line.strip().split(':')

        # Convertir chaque élément en valeur entière et les joindre en une chaîne de caractères
        converted_items = []
        for item_with_spaces in items_with_spaces:
            # Diviser chaque élément par les espaces
            items_without_spaces = item_with_spaces.strip().split(' ')
            # Convertir chaque élément en nombre entier et les joindre en une chaîne de caractères
            converted_items.append(' '.join(str(int(float(item))) for item in items_without_spaces))

        converted_line = ':'.join(converted_items)

        # Écrire les données converties dans le fichier de sortie
        f_out.write(converted_line + '\n')

print("Conversion réussie!")
```

FIGURE 4.5 – *Script Python pour convertir les utilités de valeurs flottantes en valeurs entières.*

La Figure 4.6 montre la forme de la base après cette deuxième conversion.

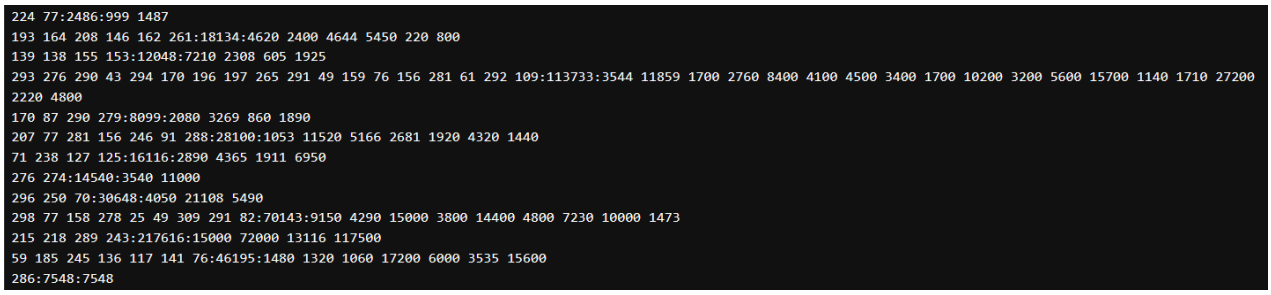


FIGURE 4.6 – *Un extrait du fichier SPMF de la base Pharmacie (avec des valeurs d'utilité de type entier).*

4.3.3 L'étape de fouille de données

A) Vous pouvez obtenir NetBeans en le téléchargeant directement depuis le site officiel de NetBeans, accessible via le lien suivant : https://netbeans.apache.org.

B) Nous avons téléchargé la bibliothèque SPMF (version du code source) à partir du lien suivant : https://www.philippe-fournier-viger.com/spmf/.

C) Après avoir initié un nouveau projet Java, nous avons intégré la bibliothèque SPMF dans le répertoire source (src) de ce projet.

4.3.4 Exécution de l'algorithme FHM

À Présent, nous allons exécuter l'algorithme FHM sur la base de données de la pharmacie, puis nous procéderons à une analyse des résultats.

La Figure 4.7 présente un exemple d'exécution de FHM avec la base Pharmacie, en utilisant la classe *MainTestFhm-Float.java* de SPMF. Le fichier de données est *pharmacie.txt*, et le seuil minimal d'utilité (*MinUtil*) est de 350 000. Les motifs trouvés par FHM sont enregistrés dans un fichier texte de résultats.

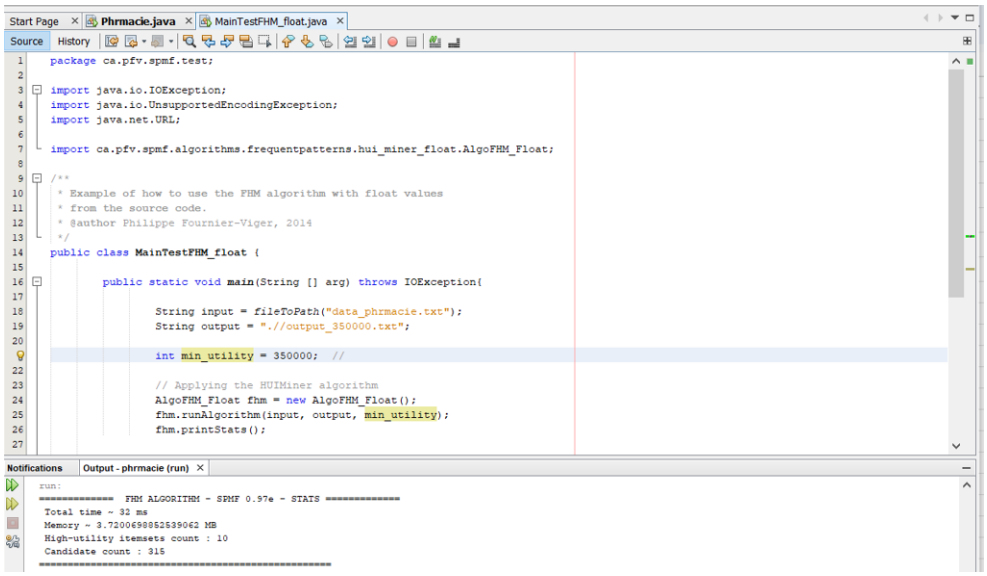


FIGURE 4.7 – *Exécution de l'algorithme FHM avec la base Pharmacie.*

4.3.4.1 Les résultats obtenus

Les résultats obtenus lors de l'application des algorithmes FHM ont été évalués selon trois critères : le temps d'exécution, l'utilisation de la mémoire et le nombre de motifs découverts.

L'algorithme FHM a été testé avec plusieurs valeurs de seuil *MinUtil*, variant de 170 000 à 350 000. Les résultats de l'algorithme FHM sont résumés dans le Tableau 4-1 et illustrés dans les Figures 4.8, 4.9 et 4.10. La Figure 4.10 montre que le nombre de motifs découverts

TABLE 4.1 – *Les résultats de FHM avec la base Pharmacie.*

Minutil	170000	175000	185000	200000	225000	250000	275000	300000	325000	350000
Temps (MS)	1598	1024	757	301	93	61	65	63	47	55
Mémoire (MB)	248.96	124.98	62.92	31.89	5.58	4.34	4.34	3.72	3.72	3.72
Nombre de HUIs	535899	372539	162992	28657	273	39	22	14	12	10

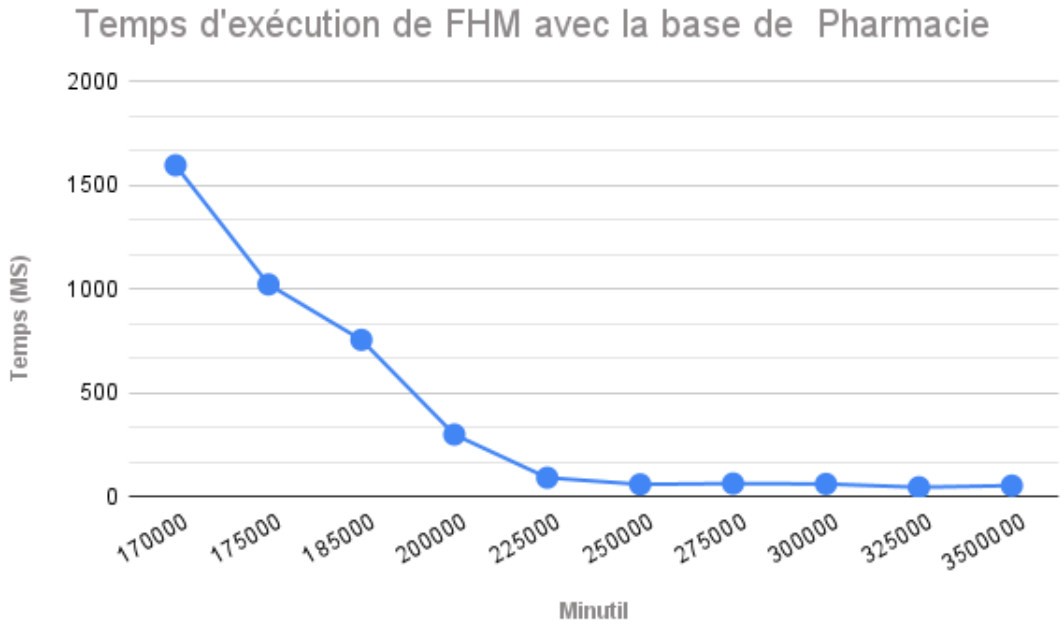


FIGURE 4.8 – Temps d'exécution de FHM avec la base Pharmacie.

par FHM diminue à mesure que la valeur du seuil *MinUtil* augmente. En effet, l'augmentation de *MinUtil* entraîne une augmentation du nombre des motifs non promoteurs (dont l'utilité est inférieure à *MinUtil*) et une diminution des itemsets à haute utilité. Ainsi, lorsque la valeur du seuil est élevée, les stratégies d'élagage éliminent davantage d'itemsets non promoteurs.

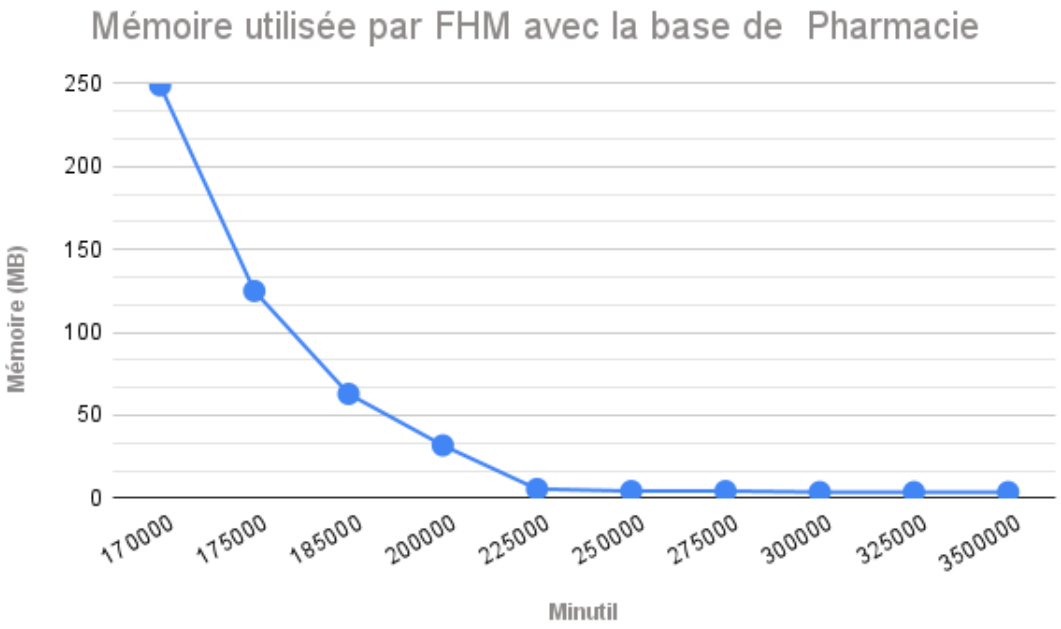


FIGURE 4.9 – Mémoire utilisée par FHM avec la base Pharmacie.

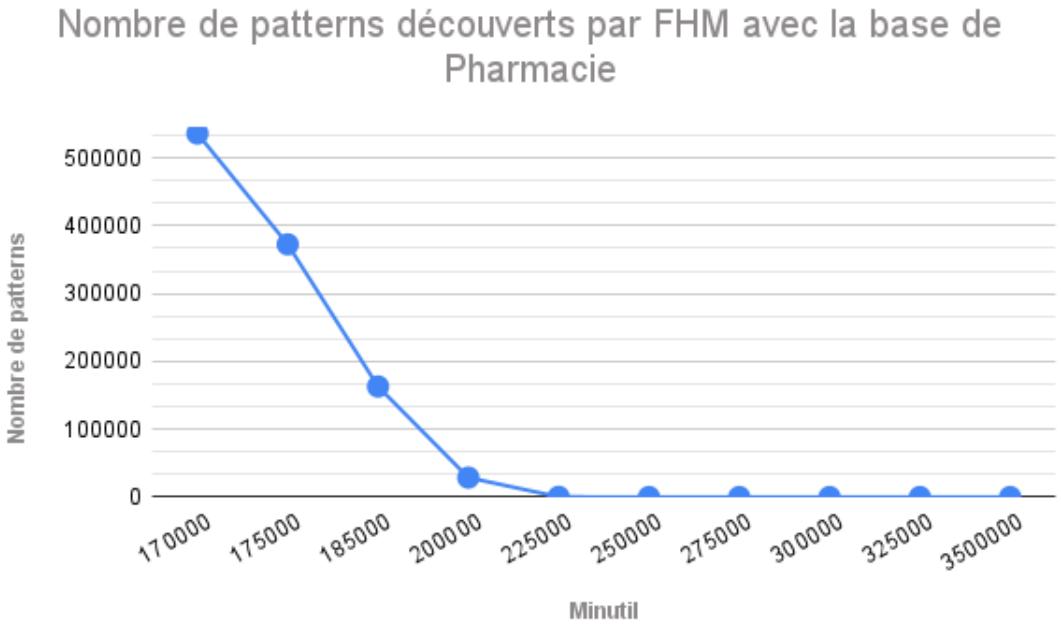


FIGURE 4.10 – *Nombre d'itemsets découverts par FHM avec la base Pharmacie.*

Par conséquent, le temps d'exécution et l'utilisation de la mémoire diminuent également avec l'augmentation de *MinUtil*, comme illustré dans les Figures 4.8 et 4.9.

4.3.4.2 Exemples de motifs découverts par FHM

La Figure 4.11 montre un extrait du fichier des résultats obtenus après l'exécution de FHM. On y constate que les motifs retournés incluent seulement les numéros des produits achetés, sans les noms correspondants. Il est donc nécessaire d'associer ces numéros à leurs noms pour chaque motif afin de les interpréter. Pour cela, un script Python a été mis au point. Il prend en entrée un fichier texte avec les motifs de haute utilité (HUI) et leurs valeurs, ainsi qu'un fichier Excel qui associe les identifiants des produits à leurs noms. Le script génère ensuite un fichier texte où chaque produit est identifié par son nom avec son utilité.

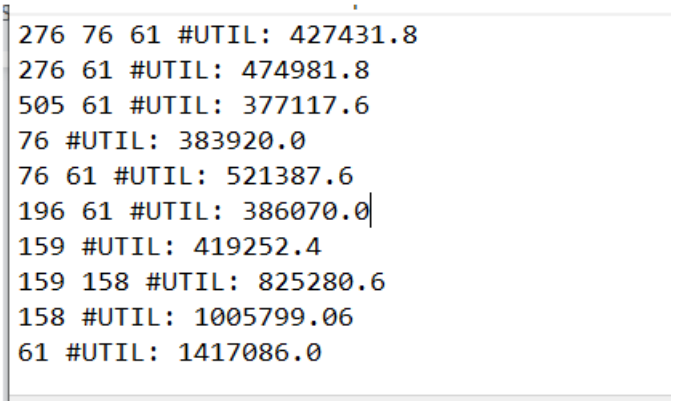


FIGURE 4.11 – *Un extrait des itemsets à haute utilité découverts par FHM dans la base Pharmacie avec MinUtil=350000.*

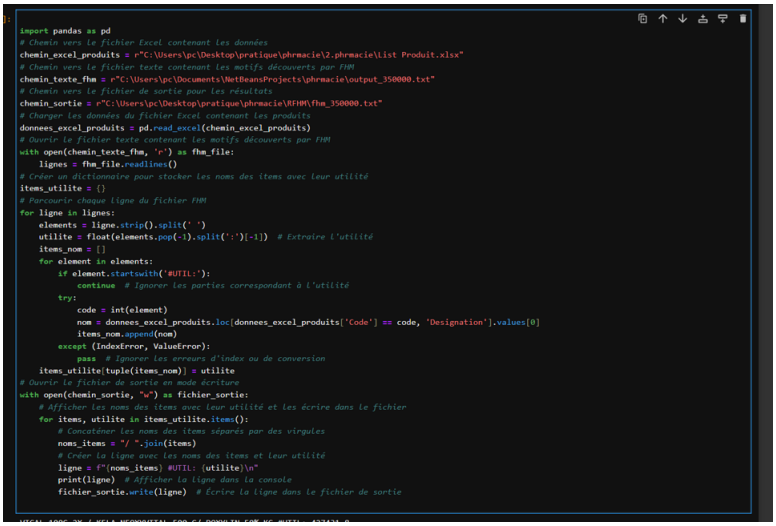
Le script Python permettant cette tache est présenté par la Figure 4.21.

a) Exemple 1 : (FHM avec MinUtil= 350000) avec interprétation

En appliquant le script présenté juste avant, nous obtenons l'ensemble des itemsets découverts par FHM dans un format plus lisible, car les noms des produits sont indiqués. Le résultat de l'application du script est montré dans la Figure 4.13.

Selon la Figure 4.13, les données indiquent que *DOXYLIN 50%* KG est le médicament ayant la plus large utilité. ce médicament est principalement utilisé pour le traitement des infections respiratoires. Il est souvent combiné avec d'autres médicaments, tels que des suppléments vitaminiques et des anti-inflammatoires, dans une approche de traitement intégrée.

Une attention particulière est accordée à l'utilisation fréquente d'anti-inflammatoires comme



```python
import pandas as pd
# Chemin vers le fichier Excel contenant les données
chemin_excel_produits = r"C:\Users\pc\Desktop\pratique\phrmacie\2.phrmacie\List Produit.xlsx"
# Chemin vers le fichier texte contenant les motifs découverts par FHM
chemin_texte_fhm = r"C:\Users\pc\Documents\NetBeansProjects\phrmacie\output_350000.txt"
# Chemin vers le fichier de sortie pour les résultats
chemin_sortie = r"C:\Users\pc\Desktop\pratique\phrmacie\RFHM\fhm_350000.txt"
# Charger les données du fichier Excel contenant les produits
donnees_excel_produits = pd.read_excel(chemin_excel_produits)
# Ouvrir le fichier texte contenant les motifs découverts par FHM
with open(chemin_texte_fhm, 'r') as fhm_file:
    lignes = fhm_file.readlines()
# Créer un dictionnaire pour stocker les noms des items avec leur utilité
items_utilite = {}
# Parcourir chaque ligne du fichier FHM
for ligne in lignes:
    elements = ligne.strip().split(' ')
    utilite = float(elements.pop(-1).split(':')[-1])  # Extraire l'utilité
    items_nom = []
    for element in elements:
        if element.startswith('#UTIL:'):
            continue  # Ignorer les parties correspondant à l'utilité
        try:
            code = int(element)
            nom = donnees_excel_produits.loc[donnees_excel_produits['Code'] == code, 'Designation'].values[0]
            items_nom.append(nom)
        except (IndexError, ValueError):
            pass  # Ignorer les erreurs d'index ou de conversion
    items_utilite[tuple(items_nom)] = utilite
# Ouvrir le fichier de sortie en mode écriture
with open(chemin_sortie, "w") as fichier_sortie:
    # Afficher les noms des items avec leur utilité et les écrire dans le fichier
    for items, utilite in items_utilite.items():
        # Concaténer les noms des items séparés par des virgules
        noms_items = " / ".join(items)
        # Créer la ligne avec les noms des items et leur utilité
        ligne = f"{noms_items} #UTIL: {utilite}\n"
        print(ligne)  # Afficher la ligne dans la console
        fichier_sortie.write(ligne)  # Écrire la ligne dans le fichier de sortie
```

FIGURE 4.12 – *Script Python pour obtenir les noms des produits de chaque itemset découvert par FHM dans la base Pharmacie.*

```
VIGAL 100G 2X / KELA NEOXYVITAL 500 G/ DOXYLIN 50% KG #UTIL: 427431.8
VIGAL 100G 2X / DOXYLIN 50% KG #UTIL: 474981.8
HIPRAVIARGUMBORO CH80/ DOXYLIN 50% KG #UTIL: 377117.6
KELA NEOXYVITAL 500 G #UTIL: 383920.0
KELA NEOXYVITAL 500 G/ DOXYLIN 50% KG #UTIL: 521387.6
ESERVIT AD3EK+B/ DOXYLIN 50% KG #UTIL: 386070.0
ALGICOX 500ML #UTIL: 419252.4
ALGICOX 500ML/ ALGICOX01L #UTIL: 825280.6
ALGICOX01L #UTIL: 1005799.06
DOXYLIN 50% KG #UTIL: 1417086.0
```

FIGURE 4.13 – *Un extrait des itemsets à haute utilité découverts par FHM dans la base Pharmacie avec MinUtil=350000 (incluant les noms des produits)*

ALGICOX, qui est largement vendu pour ses propriétés analgésiques et anti-inflammatoires.

Dans les exemples observés, plusieurs combinasons d'items (itemsets) ont été identifiées :

- La combinaison VIGAL 100G 2X / DOXYLIN 50% KG, qui cible à la fois les infections gastro-intestinales et respiratoires de manière synergique.
- L'utilisation de KELA NEOXYVITAL 500 G en association avec DOXYLIN 50% KG, ce qui semble viser à améliorer la santé globale tout en traitant les infections respiratoires.
- L'association d'ALGICOX 500ML avec ALGICOX01L, ce qui suggère une utilisation combinée d'anti-inflammatoires pour des traitements intensifs.

4.3.5 Exécution de l'algorithme HGB-All

L'algorithme HGB-All est conçue pour extraire des règles d'association à haute utilité en utilisant les résultats de l'algorithme HUCI-Miner [18].

La Figure 4.14 présente un example d'exécution de HGB-All avec la base *Pharmacie.* L'execution est effectuée en utilisant la classe principale *MainTestHGB-all* de *SPMF*. La veleur minimale de l'utilisé dans cet expmle est fixée à 260000. Après l'exécution, les règles d'association sont enregistrées dans "HGB-all-260000.txt" avec une confiance minimale de 0.5. Les statistiques de performance sont également fournies pour évaluer l'efficacité de l'algorithme dans la découverte de règles d'association à haute utilité.

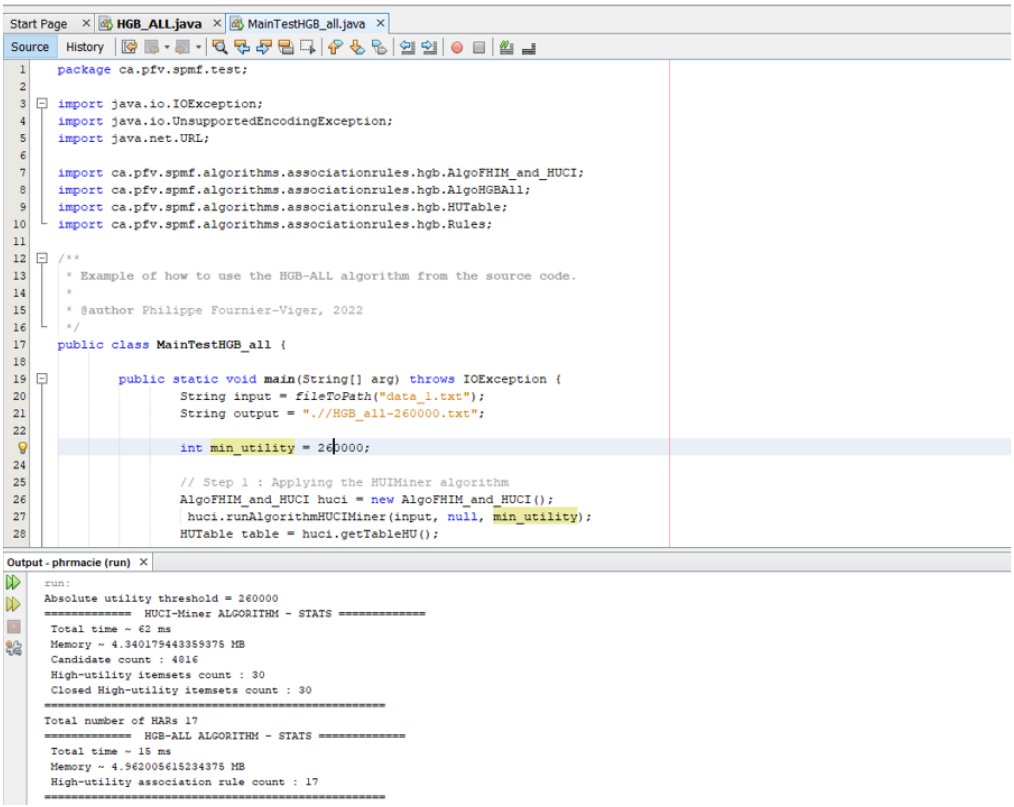


FIGURE 4.14 – *Exécution de l'algorithme HGB-All avec la base Pharmacie.*

4.3.5.1 Les résultats obtenus

Nous avons étudié l'algorithme HGB-All en évaluant ses performances selon trois critères principaux : le temps d'exécution, l'utilisation de la mémoire et le nombre de HARs (High-Utility Association Rules). Des tests ont été effectués en modifiant les valeurs *MinUtil* de 205000 à 290000, tandis que la valeur du seuil minimal de corrélation, $MinConf$, a été maintenue à 0.5. Les résultats de ces tests sont présentés dans le Tableau 4.2 et illustrés à travers les

Figures 4.15, 4.16 et 4.17.

TABLE 4.2 – *Les résultats de HGB-All avec la base Pharmacie.*

MinUtil	**205000**	**210000**	**215000**	**220000**	**235000**	**245000**	**250000**	**260000**	**280000**	**290000**
Temps (ms)	9494	824	455	47	9	9	17	15	4	4
Mémoire (MB)	141,71	44,27	11,23	17,36	5,58	4,93	4,96	4,96	4,34	4,34
Nombre de HARs	625721	166688	36545	6923	352	60	24	17	7	5

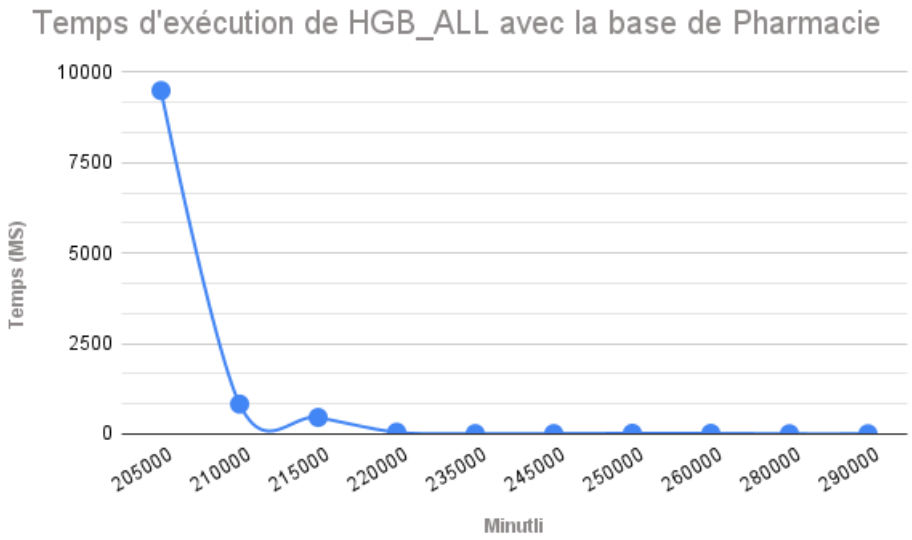


FIGURE 4.15 – *Temps d'exécution de HGB-All avec la base Pharmacie.*

L'analyse indique que l'augmentation de la valeur de *MinUtil* semble avoir un impact positif sur les performances globales de l'algorithme en termes de temps d'exécution et d'utilisation de la mémoire. Cela s'explique par le fait que l'augmentation de la valeur de *MinUtil* réduit l'espace de recherche puisque le nombre d'itemsets à faible utilité est diminué. La réduction de l'espace de recherche a une influence directe sur le temps d'exécution et l'utilisation de la mémoire. De plus, le nombre de règles d'association à haute utilité est également réduit.

4.3.5.2 Exemples de règles d'association à haute utilité découverte par HGB-All

Dans le cadre de l'analyse des règles d'association à haute utilité dans la base *Pharmacie*, nous devons interpréter les résultats fournis par l'algorithme HGB-All. Pour cet objectif, nous avons développé un script qui charge le fichier Excel contenant les désignations des produits

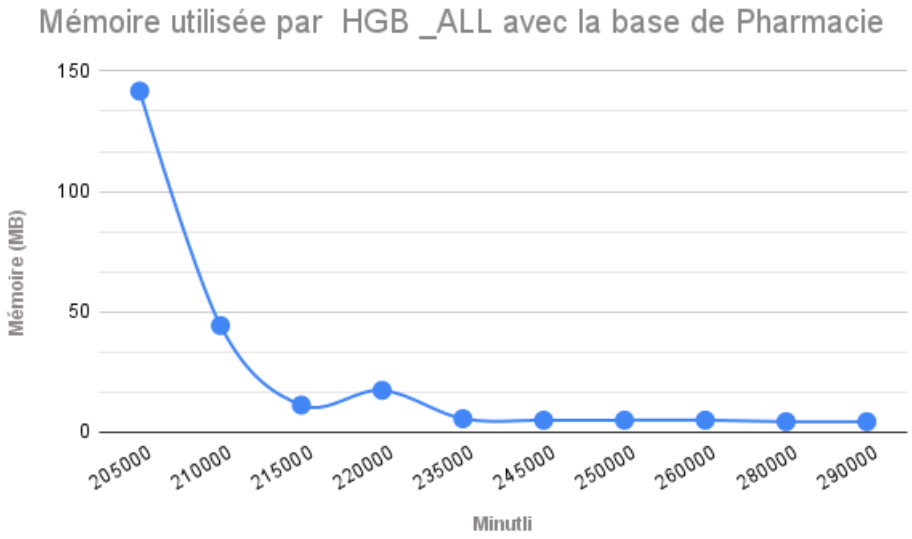


FIGURE 4.16 – *Mémoire utilisée par HGB-All avec la base Pharmacie.*

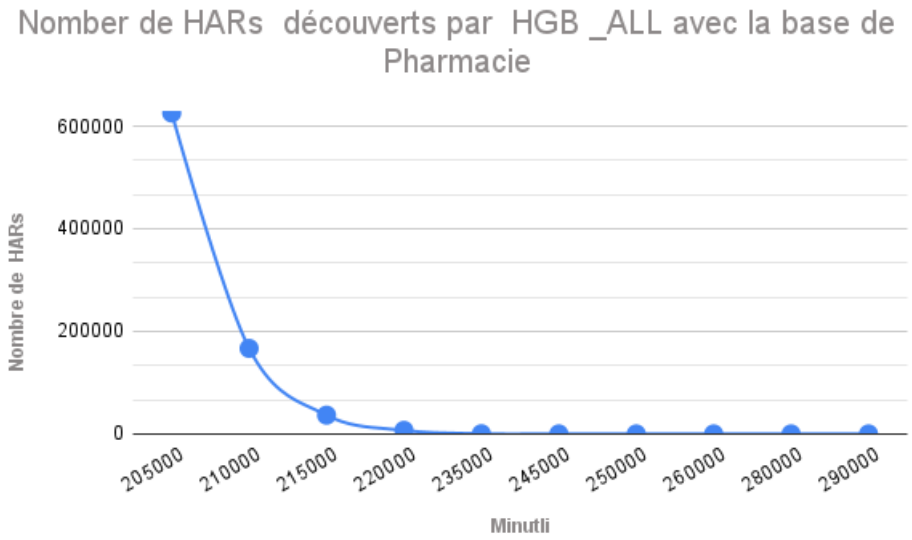
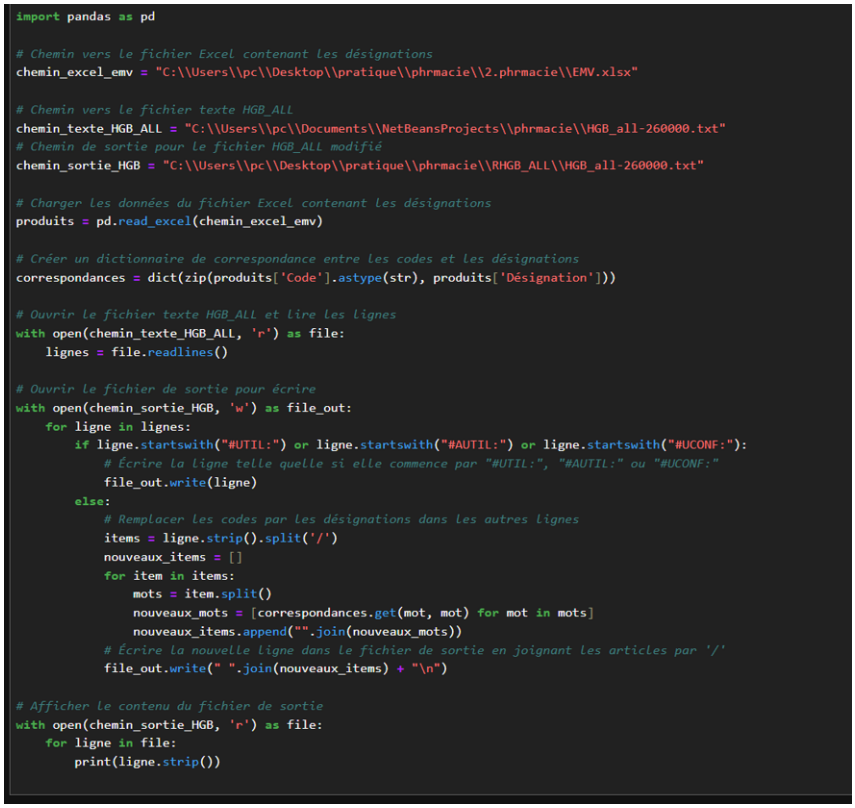


FIGURE 4.17 – *Nombre de règles d'association à haute utilité découvertes par HGB-All avec la base Pharmacie.*

ainsi que le fichier texte généré après l'exécution de HGB-All de SPMF. Le script remplace les codes des produits par leurs désignations correspondantes dans le fichier texte, puis écrit les résultats modifiés dans un nouveau fichier de sortie. Cette procédure facilite la compréhension des règles d'association découvertes en remplaçant les codes des produits par leurs noms.

La Figure 4.18 montre le script de conversion, Figure 4.19 présente un extrait des règles découvertes par HGB-All avec $MinUtil = 260000$, et Figure 4.20 présente les règles d'association découvertes avec $MinUtil = 260000$ après conversion.



```python
import pandas as pd

# Chemin vers le fichier Excel contenant les désignations
chemin_excel_emv = "C:\\Users\\pc\\Desktop\\pratique\\phrmacie\\2.phrmacie\\EMV.xlsx"

# Chemin vers le fichier texte HGB_ALL
chemin_texte_HGB_ALL = "C:\\Users\\pc\\Documents\\NetBeansProjects\\phrmacie\\HGB_all-260000.txt"
# Chemin de sortie pour le fichier HGB_ALL modifié
chemin_sortie_HGB = "C:\\Users\\pc\\Desktop\\pratique\\phrmacie\\RHGB_ALL\\HGB_all-260000.txt"

# Charger les données du fichier Excel contenant les désignations
produits = pd.read_excel(chemin_excel_emv)

# Créer un dictionnaire de correspondance entre les codes et les désignations
correspondances = dict(zip(produits['Code'].astype(str), produits['Désignation']))

# Ouvrir le fichier texte HGB_ALL et lire les lignes
with open(chemin_texte_HGB_ALL, 'r') as file:
    lignes = file.readlines()

# Ouvrir le fichier de sortie pour écrire
with open(chemin_sortie_HGB, 'w') as file_out:
    for ligne in lignes:
        if ligne.startswith("#UTIL:") or ligne.startswith("#AUTIL:") or ligne.startswith("#UCONF:"):
            # Écrire la ligne telle quelle si elle commence par "#UTIL:", "#AUTIL:" ou "#UCONF:"
            file_out.write(ligne)
        else:
            # Remplacer les codes par les désignations dans les autres lignes
            items = ligne.strip().split('/')
            nouveaux_items = []
            for item in items:
                mots = item.split()
                nouveaux_mots = [correspondances.get(mot, mot) for mot in mots]
                nouveaux_items.append("".join(nouveaux_mots))
            # Écrire la nouvelle ligne dans le fichier de sortie en joignant les articles par '/'
            file_out.write(" ".join(nouveaux_items) + "\n")

# Afficher le contenu du fichier de sortie
with open(chemin_sortie_HGB, 'r') as file:
    for ligne in file:
        print(ligne.strip())
```

FIGURE 4.18 – *Script Python pour obtenir les noms des produits de chaque régle découverte par HGB-All.*

```
159 ==> 158 #UTIL: 825280 #AUTIL: 419252 #UCONF: 0.7757148445326438
61 291 ==> 196 #UTIL: 268966 #AUTIL: 271866 #UCONF: 0.8631678841782349
61 291 ==> 25 #UTIL: 292166 #AUTIL: 271866 #UCONF: 0.8624322276415587
61 276 ==> 76 #UTIL: 427431 #AUTIL: 474981 #UCONF: 0.6714605426322316
61 505 ==> 71 #UTIL: 281401 #AUTIL: 377117 #UCONF: 0.7076504108804429
61 505 ==> 76 #UTIL: 291467 #AUTIL: 377117 #UCONF: 0.5676408117374714
61 275 ==> 159 #UTIL: 303422 #AUTIL: 325472 #UCONF: 0.7953986825287582
61 76 ==> 276 #UTIL: 427431 #AUTIL: 521387 #UCONF: 0.7360367634789513
61 196 ==> 291 #UTIL: 268966 #AUTIL: 386070 #UCONF: 0.5144144844199239
61 159 ==> 275 #UTIL: 303422 #AUTIL: 282102 #UCONF: 0.6972726177056525
61 291 ==> 25 196 #UTIL: 275166 #AUTIL: 271866 #UCONF: 0.7256001118197936
61 196 291 ==> 25 #UTIL: 275166 #AUTIL: 268966 #UCONF: 0.8442182283262568
25 61 291 ==> 196 #UTIL: 275166 #AUTIL: 292166 #UCONF: 0.8398170902842904
25 61 291 ==> 159 275 #UTIL: 266078 #AUTIL: 292166 #UCONF: 0.569149045405694
61 71 505 ==> 70 76 #UTIL: 265401 #AUTIL: 281401 #UCONF: 0.6432137767811771
61 76 505 ==> 70 71 #UTIL: 265401 #AUTIL: 291467 #UCONF: 0.805123736134794
61 159 275 ==> 25 291 #UTIL: 266078 #AUTIL: 303422 #UCONF: 0.6011825114856536
```

FIGURE 4.19 – *Un extrait des règles d'association à haute utilité découvertes par HGB-All dans la base Pharmacie avec MinUtil= 260000.*

a) Exemple 1 : (HGB-All avec MinUtil= 260000 et MinConf = 0.5) avec interprétation

Figure 4.20 présente les règles d'association à haute utilité découvertes par HGB-All avec *MinUtil* = 260000 et *MinConf=50%*. Nous voyons que, le *DOXYLIN 50% KG*, un antibiotique à large spectre largement employé, occupe une place centrale dans ces associations, soulignant

```
ALGICOX 500ML==> ALGICOX01L#UTIL:825280#AUTIL:419252#UCONF:0.7757148445326438
DOXYLIN 50% KG NEOMYCINE 50% KG PRO==> ESERVIT AD3EK+B#UTIL:268966#AUTIL:271866#UCONF:0.8631678841782349
DOXYLIN 50% KG NEOMYCINE 50% KG PRO==> ACTI TETRA B 1KG #UTIL:292166#AUTIL:271866#UCONF:0.8624322276415587
DOXYLIN 50% KG VIGAL 100G 2X ==> KELA NEOXYVITAL 500 G#UTIL:427431#AUTIL:474981#UCONF:0.6714605426322316
DOXYLIN 50% KG HIPRAVIARGUMBORO CH80==> HIPRAVIAR  SOTA#UTIL:281401#AUTIL:377117#UCONF:0.7076504108804429
DOXYLIN 50% KG HIPRAVIARGUMBORO CH80==> KELA NEOXYVITAL 500 G#UTIL:291467#AUTIL:377117#UCONF:0.5676408117374714
DOXYLIN 50% KG CEVAZURIL==> ALGICOX 500ML#UTIL:303422#AUTIL:325472#UCONF:0.7953986825287582
DOXYLIN 50% KG KELA NEOXYVITAL 500 G==> VIGAL 100G 2X #UTIL:427431#AUTIL:521387#UCONF:0.7360367634789513
DOXYLIN 50% KG ESERVIT AD3EK+B==> NEOMYCINE 50% KG PRO#UTIL:268966#AUTIL:386070#UCONF:0.5144144844199239
DOXYLIN 50% KG ALGICOX 500ML==> CEVAZURIL#UTIL:303422#AUTIL:282102#UCONF:0.6972726177056525
DOXYLIN 50% KG NEOMYCINE 50% KG PRO==> ACTI TETRA B 1KG  ESERVIT AD3EK+B#UTIL:275166#AUTIL:271866#UCONF:0.7256001118197936
DOXYLIN 50% KG ESERVIT AD3EK+B NEOMYCINE 50% KG PRO==> ACTI TETRA B 1KG #UTIL:275166#AUTIL:268966#UCONF:0.8442182283262568
ACTI TETRA B 1KG  DOXYLIN 50% KG NEOMYCINE 50% KG PRO==> ESERVIT AD3EK+B#UTIL:275166#AUTIL:292166#UCONF:0.8398170902842904
ACTI TETRA B 1KG  DOXYLIN 50% KG NEOMYCINE 50% KG PRO==> ALGICOX 500ML CEVAZURIL#UTIL:266078#AUTIL:292166#UCONF:0.569149045405694
DOXYLIN 50% KG HIPRAVIAR  SOTA HIPRAVIARGUMBORO CH80==> HIPRAVIAR  B1/H120 KELA NEOXYVITAL 500
G#UTIL:265401#AUTIL:281401#UCONF:0.6432137767811771
DOXYLIN 50% KG KELA NEOXYVITAL 500 G HIPRAVIARGUMBORO CH80==> HIPRAVIAR  B1/H120 HIPRAVIAR
SOTA#UTIL:265401#AUTIL:291467#UCONF:0.805123736134794
DOXYLIN 50% KG ALGICOX 500ML CEVAZURIL==> ACTI TETRA B 1KG  NEOMYCINE 50% KG PRO#UTIL:266078#AUTIL:303422#UCONF:0.6011825114856536
```

FIGURE 4.20 – *Un extrait des règles d'association à haute utilité découvertes par HGB-All dans la base Pharmacie avec MinUtil= 260000 (incluant les noms des produits).*

ainsi son rôle crucial dans le traitement des infections bactériennes. Nous allons essayer d'interpréter quelques régles :

- **Règle 1 :** *DOXYLIN 50% KG NEOMYCINE 50% KG PRO* → *ESERVIT AD3EK+B*
  UTIL : 268966 AUTIL : 271866 UCONF : 0.8632

  DOXYLIN 50% KG et *NEOMYCINE 50% KG PRO* sont des antibiotiques souvent utilisés ensemble pour traiter des infections bactériennes graves. La forte confiance (86.32%) indique que ces antibiotiques sont fréquemment associés à *ESERVIT AD3EK+B,* qui est un supplément vitaminé, probablement pour renforcer le système immunitaire et améliorer la réponse au traitement antibiotique.
- **Règle 2 :** *DOXYLIN 50% KG* et *NEOMYCINE 50% KG PRO* → *ACTI TETRA B 1KG*
  UTIL : 292166 AUTIL : 271866 UCONF : 0.8624

  Cette règle montre que *DOXYLIN 50% KG* et *NEOMYCINE 50% KG PRO* sont également fréquemment associés à *ACTI TETRA B 1KG*, qui est également un autre antibiotique. Cette régle suggère une stratégie de traitement combiné pour des infections nécessitant une couverture antibiotique étendue.

b) Exemple 2 : (HGB-All avec MinUtil= 215000 et MinConf = 0.5) avec interprétation

Table 4.3 présente des règles d'association de haute utilité extraites en appliquant HGB-All avec $MinUtil = 215000$ et $MinConf = 50\%$. Ces règles illustrent les associations fréquentes entre divers antibiotiques, médicaments antiparasitaires et suppléments vitaminiques.

- Une confiance élevée (au-dessus de 70 %) indique des associations très probables, sug-

gérant des pratiques courantes en traitement.

- Une confiance modérée (entre 50% et 70%) montre des associations utiles mais moins fréquentes, pouvant indiquer des traitements alternatifs ou complémentaires.
- Les médicaments comme *DOXYLIN*, *NEOMYCINE*, *ESERVIT*, et *CEVAZURIL* sont couramment utilisés en combinaison pour traiter diverses infections bactériennes et soutenir la santé des patients.

Par exemple, la régle *ACTITETRAB1KG / DOXYLIN50%KG* $\rightarrow$ *ESERVIT AD3EK+B / NEOMYCINE50% KG PRO* indique que lorsque on utilise *ACTITETRAB 1KG* et *DOXYLIN 50% KG*, il est très probable (83.42%) que *ESERVIT AD3EK+B* et *NEOMYCINE 50% KGPRO* soient également utilisés. Cela suggère un traitement des infections bactériennes avec un soutien vitaminique.

TABLE 4.3 – *Un extrait des règles d'association à haute utilité découvertes par HGB-All dans la base Pharmacie avec MinUtil= 215000 et MinConf=0.5*

High utility association rule	**Utility**	**Utility of Antecedent**	**Utility Confidence**
ACTI TETRA B 1KG / DOXYLIN 50% KG ==> ESERVIT AD3EK+B / NEOMYCINE 50% KG PRO	275166	222000	0.8342
ACTI TETRA B 1KG / DOXYLIN 50% KG ==> CEVAZURIL / NEOMYCINE 50% KG PRO	235406	222000	0.5456
ACTI TETRA B 1KG / DOXYLIN 50% KG ==> ALGICOX 500ML / CEVAZURIL	220912	222000	0.5456
ACTI TETRA B 1KG / DOXYLIN 50% KG ==> OXTRA 100ML / NEOMYCINE 50% KG PRO	222968	222000	0.6644
DOXYLIN 50% KG / HIPRAVIAR / SOTA ==> KELA NEOXYVITAL 500G / HIPRAVIARGUMBORO CH80	243401	247301	0.6490
DOXYLIN 50% KG / HIPRAVIAR / SOTA ==> HIPRAVIAR B1/H120 / KELA NEOXYVITAL 500G	244901	247301	0.6490
DOXYLIN 50% KG / ESERVIT AD3E / VIGAL 100G 2X ==> MAXIFORT	215113	237488	0.7668

c) Exemple 3 : (HGB-All avec MinUtil= 215000 et MinConf = 100%) avec interprétation

Table 4.4 présente les régle d'aasociation a haute utilité exttraite de la base de pharmacie en fixant la valeur de $MinUtil$ à 215000 et la valeur de $MinConf$ à 1 (100%). Nous remarquons que dans chaque association, les composantes *LOVIT GRANULE BX 175 GR*, *NOBILIS IB 4-*

91, *NOBILIS IB4 91 2500 DOSE* et *LOVIT GRANULE ANILYTE+CKG* sont souvent associés à d'autres produits tels que *PENIV- AL STREP RETARD 100 ML*, *TOXIDREN ST 200 G*, *MPP 12ML* et *MPP 6ML*. Nous remarquons égamlement que la valeur de confiance dans ces régles est maximale (tous égaux à 100%).

Ces résultats font référence à des groupes de produits couramment utilisés en pratique de traitement, qui peuvent être utiles pour améliorer les protocoles de traitement ou les pratiques d'approvisionnement.

TABLE 4.4 – *Un extrait des règles d'association à haute utilité découvertes par HGB-All dans la base Pharmacie avec MinUtil= 215000 et MinConf=1*

High utility association rule	**Utility**	**Utility of Antecedent**	**Utility Confidence**
LOVIT GRANULE BX 175 GR / NOBILIS IB 4-91 / NOBILIS IB4 91 2500 DOSE / LOVIT GRANULE ANILYTE+CKG ==> PENIV- AL STREP RETARD 100 ML / MPP 12ML / MPP 6ML	237039	225120	1.0
LOVIT GRANULE BX 175 GR / NOBILIS IB 4-91 / NOBILIS IB4 91 2500 DOSE / LOVIT GRANULE ANILYTE+CKG ==> TOXIDREN ST 200 G /MPP 12ML /MPP 6ML	232675	225120	1.0
LOVIT GRANULE BX 175 GR / NOBILIS IB 4-91 / NOBILIS IB4 91 2500 DOSE / LOVIT GRANULE ANILYTE+CKG ==> TOXIDREN ST 200 G / AD3E BCK LAMOS 01L / MPP 6ML	237216	225120	1.0
LOVIT GRANULE BX 175 GR / NOBILIS IB 4-91 / NOBILIS IB4 91 2500 DOSE / LOVIT GRANULE ANILYTE+CKG ==> AD3E BCK LAMOS 01L /PENIV- AL STREP RETARD 100 ML / MPP 12ML	241927	225120	1.0

4.4 Le deuxième cas d'étude : La base *Fruithut*

4.4.1 Présentation de la base *Fruithut*

Le deuxième cas d'étude concerne un ensemble de données de transactions de clients d'un magasin de détail aux États-Unis, spécialisé dans la vente de fruits. Nous avons téléchargé cette base de données à partir du lien suivant : https://www.philippe-fournier-viger.com/spmf/datasets/fruithut_utility.txt.

Pour cette base de données, il n'est pas nécessaire d'effectuer une phase de prétraitement car la base est déjà préparée et stockée au format SPMF. Par conséquent, nous allons directement procéder à des tests avec les algorithmes FHM et HGB-All. Les aperçus de la base de transactions et la liste des produits sont présentés respectivement par les Figures 4.21 et 4.22.

```
1063 1065 1182 5000:868:199 150 199 320
2070 2076 2080 2162:1647:250 399 299 699
1008 1017 1029 1034 1058 1112 2025 2045 2080 4033:6893:350 998 2500 699 899 300 550 149 299 149
1008 1017 1021 1076 1079 1107 1118 1167 2010 2065 2123 3003:3314:350 499 170 299 199 300 200 299 199 199 250 350
1114 2000 2087 3156 9045:2427:450 399 399 650 529
1023 1079 1113 2000 5000:1516:299 199 299 399 320
1032 1113 2079:1097:499 299 299
1178:299:299
1078 1163 2010:897:199 499 199
1049 1113 2045 4033:847:250 299 149 149
2080 3037 9181:2025:299 227 1499
1027 1112 1131 2067 2070 2080 4043 9003:3147:1200 300 299 300 250 299 300 199
1010 1029 1054 1113 2065:4348:150 2500 1200 299 199
1002 1008 1064 1081 1107 1120 1141 2010 3031 9002 9236:3374:499 350 299 299 300 290 90 199 399 99 550
```

FIGURE 4.21 – *Un extrait du ficher SPMF de la base Fruithut.*

names_fruithut - Notepad

File Edit Format View Help

```
@CONVERTED_FROM_TEXT
@ITEM=9188=Rampe Leaves Dried 50g
@ITEM=9189=Basil Seed 100g(casa casa)
@ITEM=9182=Bread Crumbs 500g
@ITEM=1220=artichokes baby
@ITEM=9183=Mangal Chicken Masala 100g
@ITEM=9180=Maldive Fish chips 100g
@ITEM=9181=Magi coconut Powder 1kg
@ITEM=9186=Pepper Kibbled 100g
@ITEM=9187=Mustard Seed Brown 250g
```

FIGURE 4.22 – *Un extrait du fichier contenant la liste des produits de la base Fruithut.*

4.4.2 Exécution de l'algorithme FHM

Dans cette section, nous testons les deux algorithmes FHM et HGB-All en suivant une méthodologie similaire à la première étude de cas. En commençant par FHM, cet algorithme a

été évalué en utilisant plusieurs valeurs de seuil *MinUtil*, variant de 100000 à 1250000.

4.4.2.1 Les résultats obtenus

Les resultats de performances de FHM en fonction de temps d'execution, utilisation de la mémoire et nombre de motifs obtenus sont présentés par la Table 4.5 et les Figures 4.23, 4.24 et 4.25.

TABLE 4.5 – *Les résultats de FHM avec la base Fruithut.*

MinUtil	**100000**	**200000**	**350000**	**450000**	**550000**	**750000**	**900000**	**1000000**	**1100000**	**1250000**
Temps (ms)	4705	3761	3028	2603	2248	2005	1930	1704	1788	1634
Mémoire (MB)	218,61	160,81	130,11	111	96,38	90,61	75,88	66,61	75,46	71,15
Nombre de high-utility itemsets	7607	2254	831	518	382	228	160	137	114	91

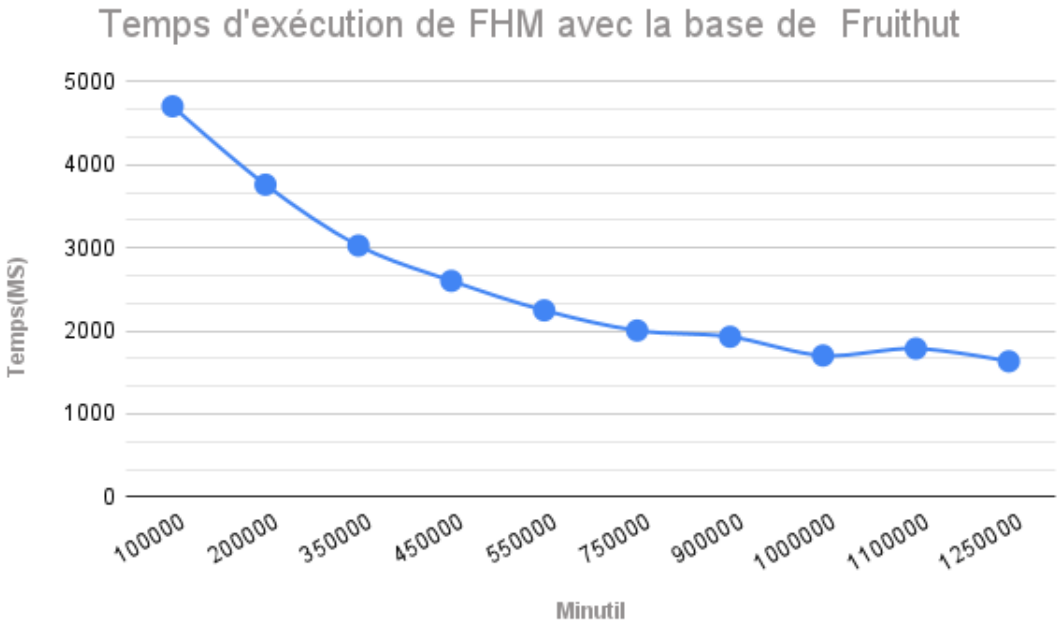


FIGURE 4.23 – *Temps d'exécution de FHM avec la base Fruithut.*

Comme pour le premier cas d'étude, on observe également une diminution du temps d'exécution avec l'augmentation des seuils d'utilité, tandis que l'utilisation de la mémoire varie sans tendance claire. Le nombre d'ensembles d'items à haute utilité diminue également avec des

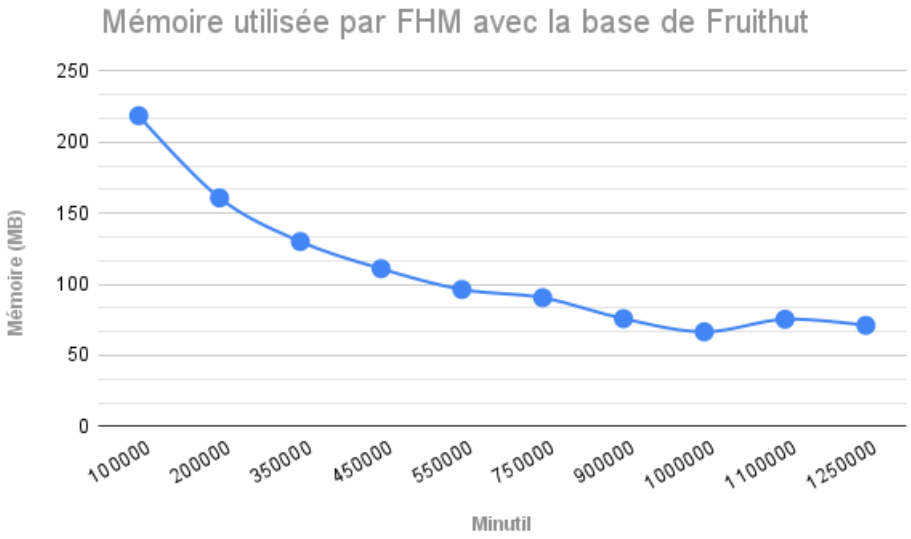


FIGURE 4.24 – *Mémoire utilisée par FHM avec la base Fruithut.*

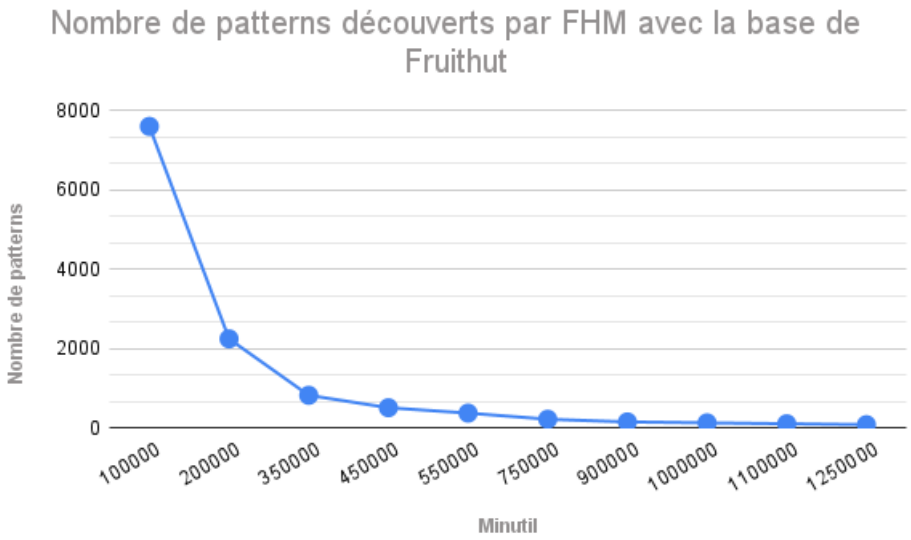


FIGURE 4.25 – *Nombre d'itemsets découverts par FHM avec la base Fruithut.*

seuils d'utilité plus élevés. Ceci est directement lié à la réduction progressive du volume de l'espace de recherche en fonction de l'augmentation du seuil minimal d'utilité *MinUtil*.

4.4.2.2 Exemples de motifs découverts par FHM

FHM permet de découvrir les itemsets à haute utilité qui sont stockés dans un fichier texte. Chaque ligne de ce fichier contient les numéros des produits avec leurs utilités. Pour pouvoir interpréter les résultats de FHM, il est nécessaire de connaître les noms des items de chaque itemset au lieu de manipuler leurs numéros. Pour cet objectif, nous avons développé un script

Python qui associe les noms aux items de chaque itemset. Le script permettant cette association est présenté dans la Figure 4.26.

```python
def load_item_names(file_path):
    item_names = {}
    with open(file_path, 'r') as f:
        for line in f:
            if line.startswith('@ITEM='):
                parts = line.strip().split('=')
                item_id = parts[1]
                item_name = parts[2]
                item_names[item_id] = item_name
    return item_names
def extract_frequent_patterns(data):
 frequent_patterns = []
    for line in data:
        parts = line.strip().split(" #UTIL: ")
        if len(parts) == 2:
            items, utility = parts
            frequent_patterns.append((items, float(utility)))
        else:
            print("Format de ligne incorrect:", line)
    return frequent_patterns
# Chemins vers les fichiers
names_file_path = "C:/Users/pc/Desktop/pratique/Fruithut/names_fruithut.txt"
utility_file_path = "C:/Users/pc/Documents/NetBeansProjects/Fruithut/output-900000.txt"
output_file_path = "C:/Users/pc/Desktop/pratique/Fruithut/fhm_convertie/output-900000.txt"
# Charger les noms des items
item_names_data = load_item_names(names_file_path)

# Lire les motifs fréquents à partir du fichier d'utilité
with open(utility_file_path, 'r') as file:
    frequent_patterns_data = file.readlines()

# Extraire les motifs fréquents avec leurs utilités
frequent_patterns = extract_frequent_patterns(frequent_patterns_data)

# Enregistrer les motifs fréquents avec leurs noms items dans un fichier
with open(output_file_path, 'w') as output_file:
    for pattern, utility in frequent_patterns:
        items = pattern.split()
        item_names = [item_names_data.get(item, item) for item in items]
        item_names_str = " ".join(item_names)
        output_file.write(f"{item_names_str} #UTIL: {utility}\n")

# Afficher les motifs fréquents avec leurs noms items
```

FIGURE 4.26 – *Script Python pour obtenir les noms des produits de chaque itemset découvert par FHM dans la base Fruithut.*

Les résultats sont ensuite enregistrés dans un nouveau fichier qui est plus lisible. Voici un aperçu des motifs à haute utilité découverts par FHM avec $MinUtil$ = 900000 dans la Figure 4.27.

a) Exemple 1 : (FHM avec $MinUtil$= 900000) avec interprétation

La figure précédente (Figure 4.27) contient un extrait des itemsets à haute utilité découverts lors de l'application de FHM à la base *FruitHut* avec $MinUtil$=900000. On voit que des produits individuels tels que *Cucumber Lebanese* et *Capsicum red* présentent des utilités très élevées, indiquant qu'ils sont souvent achetés et génèrent une valeur significative. Des combinaisons de produits comme *(Cucumber Lebanese avec Capsicum red)* et *(Capsicum red avec Banana Cavendish)* montrent également des utilités élevées, suggérant des tendances d'achat

```
Cucumber Lebanese #UTIL: 5258961.0
Cucumber Lebanese Capsicum red #UTIL: 1504984.0
Cucumber Lebanese Field Tomatoes #UTIL: 1879389.0
Cucumber Lebanese Field Tomatoes Banana Cavendish #UTIL: 974462.0
Cucumber Lebanese Banana Cavendish #UTIL: 2381463.0
Capsicum red #UTIL: 5735007.0
Capsicum red Field Tomatoes #UTIL: 1648270.0
Capsicum red Banana Cavendish #UTIL: 2530538.0
```

FIGURE 4.27 – *Un extrait des itemsets à haute utilité découverts par FHM dans la base Fruithut avec MinUtil=900000 (incluant les noms des produits).*

fréquentes et potentiellement rentables. Les motifs combinant plusieurs produits comme *(Cucumber Lebanese, Field Tomatoes, Banana Cavendish)* ont des utilités plus basses mais restent significatives, montrant des schémas d'achat moins fréquents mais encore profitables.

4.4.3 Exécution des algorithmes HGB-All

Pour l'algorithme HGB-All, les tests ont été effectués avec des valeurs de $MinUtil$ allant de 35000 à 300000, et la valeur du seuil minimal de confiance, $MinConf$, a été fixée à 0,5 (50%).

4.4.3.1 Les résultats obtenus

Les résultats obtenus avec HBG-ALL sont résumés dans la table 4.6. Les résultats du temps d'exécution, de l'utilisation de la mémoire et du nombre de motifs obtenus sont présentés respectivement dans les Figures 4.28, 4.29 et 4.30.

TABLE 4.6 – *Les résultats de HGB-All avec la base Fruithut.*

Minutil	**35000**	**50000**	**75000**	**100000**	**125000**	**150000**	**175000**	**200000**	**210000**	**300000**
Temps (ms)	82053	9098	1686	565	406	189	143	110	87	32
Mémoire (MB)	279,28	185,14	206,34	115,67	77,63	43,23	247,41	232,15	222,53	175,65
Nombre de HARs	278004	63638	6424	1581	765	447	286	187	164	51

Nous constatons que l'augmentation du seuil de $MinUtil$ améliore les performances de l'algorithme en termes de temps d'exécution et de sélection d'articles, bien que la consommation de mémoire puisse varier.

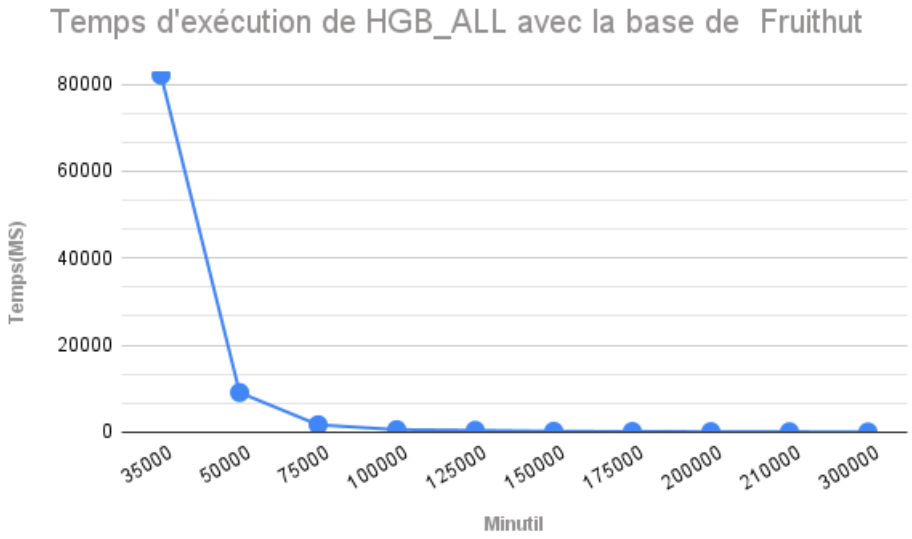


FIGURE 4.28 – *Temps d'exécution de HGB-All avec la base Fruithut.*

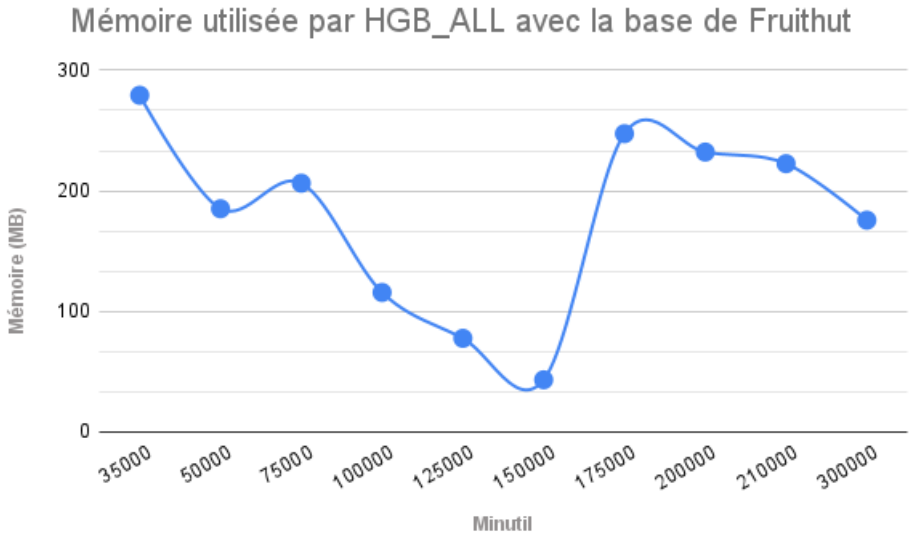


FIGURE 4.29 – *Mémoire utilisée par HGB-All avec la base Fruithut.*

4.4.3.2 Exemples de règle d'association à haute utilité découvertes par HGB-All

HGB-All retourne l'ensemble des règles à haute utilité en incluant seulement les numéros des items (Voir Figure 4.31). Pour pouvoir interpréter les règles issues de l'application de HGB-All, il est nécessaire de connaître d'abord les noms des items dans chaque règle à partir des numéros des items. Le script Python présenté dans la Figure 4.32 permet de charger les noms des items de chaque règle et d'enregistrer les règles converties dans un nouveau fichier (Voire Figure 4.33). Ce script est spécifiquement conçu pour le traitement des données de la base *Fruithut*.

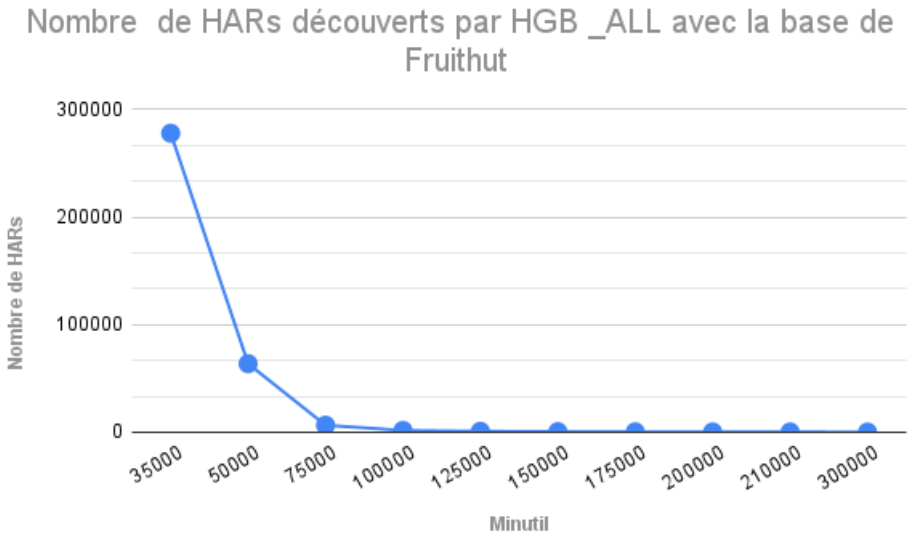


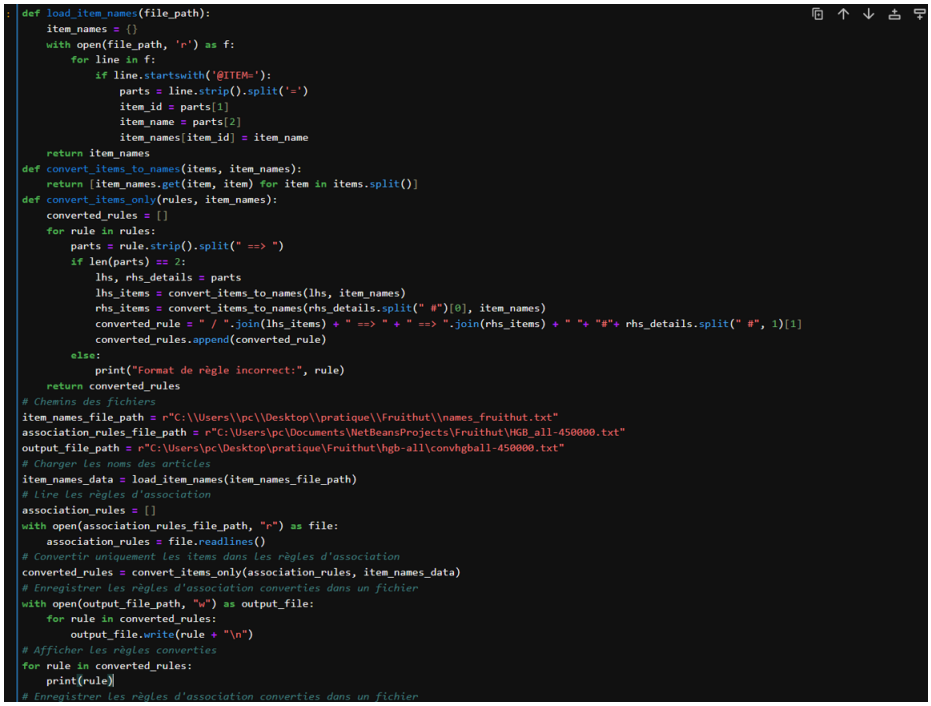
FIGURE 4.30 – *Nombre de règles à haute utilité découvertes par HGB-All avec la base Fruithut.*

```
File  Edit  Format  View  Help
1113 2003 ==> 2010 #UTIL: 640191 #AUTIL: 814177 #UCONF: 0.5905251560778553
1017 2003 ==> 2010 #UTIL: 491818 #AUTIL: 720988 #UCONF: 0.5466235221668044
2003 2079 ==> 2010 #UTIL: 490555 #AUTIL: 598299 #UCONF: 0.6166916541729135
1113 2079 ==> 2010 #UTIL: 664097 #AUTIL: 815074 #UCONF: 0.6089508437270726
1017 2079 ==> 2010 #UTIL: 487032 #AUTIL: 652364 #UCONF: 0.5967879895273192
1008 2079 ==> 2010 #UTIL: 530545 #AUTIL: 637216 #UCONF: 0.6355395972480289
1058 2079 ==> 2010 #UTIL: 469090 #AUTIL: 631645 #UCONF: 0.6358460844303366
2070 2079 ==> 2010 #UTIL: 475277 #AUTIL: 556838 #UCONF: 0.6244509175020383
1113 2070 ==> 2010 #UTIL: 593810 #AUTIL: 831789 #UCONF: 0.5222610541856167
1017 2070 ==> 2010 #UTIL: 465317 #AUTIL: 663114 #UCONF: 0.5534659198870783
1008 1039 ==> 2010 #UTIL: 478684 #AUTIL: 830250 #UCONF: 0.5008130081300813
1058 1076 ==> 2010 #UTIL: 502017 #AUTIL: 799664 #UCONF: 0.5367041657496149
1076 1113 ==> 2010 #UTIL: 599440 #AUTIL: 850057 #UCONF: 0.5265564544495251
1008 1113 ==> 2010 #UTIL: 879072 #AUTIL: 1268146 #UCONF: 0.5292111476123411
```

FIGURE 4.31 – *Un extrait des règles d'association à haute utilité découvertes par HGB-All dans la base Fruithut avec MinUtil= 450000 et MinConf=0.5.*

a) Exemple 2 : (HGB-All avec $MinUtil$= 450000 et $MinConf$=0.5 (50%)) avec interprétation.

Un extrait des règles découvertes avec $MinUtil$ = 450000 et $MinConf$ = 0,5 (50%) est présenté dans la Figure 4.33. Ces règles d'association révèlent des liens entre différents produits, indiquant des achats conjoints fréquents et retournant un haut profit. Les associations les plus remarquables impliquent les items *Field Tomatoes*, *Capsicun Red*, *Apples Pink Lady* et *Pear Packham*, qui conduisent souvent à l'achat de *Banana Cavendish*. Ces informations peuvent guider les stratégies de marketing et de gestion des stocks pour stimuler les ventes croisées et améliorer l'expérience client.



```python
def load_item_names(file_path):
    item_names = {}
    with open(file_path, 'r') as f:
        for line in f:
            if line.startswith('@ITEM='):
                parts = line.strip().split('=')
                item_id = parts[1]
                item_name = parts[2]
                item_names[item_id] = item_name
    return item_names
def convert_items_to_names(items, item_names):
    return [item_names.get(item, item) for item in items.split()]
def convert_items_only(rules, item_names):
    converted_rules = []
    for rule in rules:
        parts = rule.strip().split(" ==> ")
        if len(parts) == 2:
            lhs, rhs_details = parts
            lhs_items = convert_items_to_names(lhs, item_names)
            rhs_items = convert_items_to_names(rhs_details.split(" #")[0], item_names)
            converted_rule = " / ".join(lhs_items) + " ==> " + " ==> ".join(rhs_items) + " "+ "#"+ rhs_details.split(" #", 1)[1]
            converted_rules.append(converted_rule)
        else:
            print("Format de règle incorrect:", rule)
    return converted_rules
# Chemins des fichiers
item_names_file_path = r"C:\\Users\\pc\\Desktop\\pratique\\Fruithut\\names_fruithut.txt"
association_rules_file_path = r"C:\Users\pc\Documents\NetBeansProjects\Fruithut\HGB_all-450000.txt"
output_file_path = r"C:\Users\pc\Desktop\pratique\Fruithut\hgb-all\convhgball-450000.txt"
# Charger les noms des articles
item_names_data = load_item_names(item_names_file_path)
# lire les règles d'association
association_rules = []
with open(association_rules_file_path, "r") as file:
    association_rules = file.readlines()
# Convertir uniquement les items dans les règles d'association
converted_rules = convert_items_only(association_rules, item_names_data)
# Enregistrer les règles d'association converties dans un fichier
with open(output_file_path, "w") as output_file:
    for rule in converted_rules:
        output_file.write(rule + "\n")
# Afficher les règles converties
for rule in converted_rules:
    print(rule)
# Enregistrer les règles d'association converties dans un fichier
```

FIGURE 4.32 – *Script Python pour obtenir les noms des produits de chaque régle découverte par HGB-All dans la base Fruithut.*

```
Field Tomatoes / Apples Pink Lady ==> Banana Cavendish #UTIL: 640191 #AUTIL: 814177 #UCONF: 0.5905251560778553
Capsicum red / Apples Pink Lady ==> Banana Cavendish #UTIL: 491818 #AUTIL: 720988 #UCONF: 0.5466235221668044
Apples Pink Lady / Pear Packham ==> Banana Cavendish #UTIL: 490555 #AUTIL: 598299 #UCONF: 0.6166916541729135
Field Tomatoes / Pear Packham ==> Banana Cavendish #UTIL: 664097 #AUTIL: 815074 #UCONF: 0.6089508437270726
Capsicum red / Pear Packham ==> Banana Cavendish #UTIL: 487032 #AUTIL: 652364 #UCONF: 0.5967879895273192
Broccoli / Pear Packham ==> Banana Cavendish #UTIL: 530545 #AUTIL: 637216 #UCONF: 0.6355395972480289
Mushroom Cup / Pear Packham ==> Banana Cavendish #UTIL: 469090 #AUTIL: 631645 #UCONF: 0.6358460844303366
Mandarin Imperial / Pear Packham ==> Banana Cavendish #UTIL: 475277 #AUTIL: 556838 #UCONF: 0.6244509175020383
Field Tomatoes / Mandarin Imperial ==> Banana Cavendish #UTIL: 593810 #AUTIL: 831789 #UCONF: 0.5222610541856167
Capsicum red / Mandarin Imperial ==> Banana Cavendish #UTIL: 465317 #AUTIL: 663114 #UCONF: 0.5534659198870783
Broccoli / Ginger ==> Banana Cavendish #UTIL: 478684 #AUTIL: 830250 #UCONF: 0.5008130081300813
Mushroom Cup / Potato sweet Gold ==> Banana Cavendish #UTIL: 502017 #AUTIL: 799664 #UCONF: 0.5367041657496149
Potato sweet Gold / Field Tomatoes ==> Banana Cavendish #UTIL: 599440 #AUTIL: 850057 #UCONF: 0.5265564544495251
Broccoli / Field Tomatoes ==> Banana Cavendish #UTIL: 879072 #AUTIL: 1268146 #UCONF: 0.5292111476123411
```

FIGURE 4.33 – *Un extrait des règles à haute utilité découvertes par HGB-All dans base Fruithut avec MinUtil=450000 et MinConf = 0.5 .*

b) Exemple 3 : (HGB-All avec MinUtil= 100000 et MinConf = 0.5 (50%)) avec interprétation.

La table 4.7 contient quelque régles extraites de la base *Fruithut* en utilisant l'algorithm HGB-All avec *MinUtil*=10000 et *MinConf* = 0.5.

Les règles d'association à haute utilité présentées dans le tableau montrent des relations entre différents produits. Certaines combinaisons, telles que *Broccoli*, *Apples Red Delicious*, *Banana Cavendish* et *Watermelon Seedless*, sont associées à plusieurs autres produits tels que

TABLE 4.7 – *Un extrait des règles à haute utilité découvertes par HGB-All dans base Fruithut avec MinUtil=100000 et MinConf = 0.5*

High utility association rule	**Utility**	**Utility of Antecedent**	**Utility Confidence**
Broccoli / Apples Red Delicious / Banana Cavendish / Watermelon seedless ==> Field Tomatoes	114283	122698	0.7218
Broccoli / Apples Red Delicious / Banana Cavendish / Watermelon seedless ==> Cauliflower	113484	122698	0.7153
Broccoli / Apples Red Delicious / Banana Cavendish / Watermelon seedless ==> Sweet Corn	101873	122698	0.7442
Broccoli / Field Tomatoes / Apples Red Delicious / Watermelon seedless ==> Banana Cavendish	114283	101463	0.9538
Broccoli / Capsicum red / Apples Red Delicious / Watermelon seedless ==> Banana Cavendish	110173	102736	0.9291
Broccoli / Cauliflower / Apples Red Delicious / Banana Cavendish ==> Field Tomatoes	103785	112603	0.7332
Broccoli / Cauliflower / Apples Red Delicious / Banana Cavendish ==> Watermelon seedless	113484	112603	0.8861

Field Tomatoes, *Cauliflower* et *Sweet Corn*, avec des niveaux de confiance en l'utilité allant jusqu'à 0,7442. Les règles impliquant *Banana Cavendish* et *Watermelon Seedless* semblent particulièrement pertinentes, avec des utilités élevées et des niveaux de confiance en l'utilité dépassant 0,9. Cela indique que ces produits sont souvent achetés ensemble avec une forte certitude et un profit élevé, ce qui pourrait être utile pour mettre des stratégies de marketing.

c) Exemple 4 : (HGB-All avec $MinUtil$= 100000 et $MinConf$ = 0.5 (50%)) avec interprétation.

Un deuxième extrait avec les mêmes valeurs de $MinUtil$ et $MinConf$ est présenté dans la Table 4.8. Les règles d'association à haute utilité présentées indiquent des combinaisons de produits souvent achetés ensemble et générant un profit élevé. Par exemple, la règle *Broccoli / Capsicum red / Field Tomatoes / Apples Red Delicious* $\rightarrow$ *Banana Cavendish / Watermelon Seedless* a une confiance en l'utilité de $76,95\%$, suggérant une forte corrélation entre ces articles. De même, les règles impliquant *Broccoli*, *Field Tomatoes*, *Apples Red Delicious* et *Watermelon Seedless* associés à d'autres produits montrent des niveaux de confiance élevés,

dépassant souvent 60%.

TABLE 4.8 – *Un deuxième extrait des règles à haute utilité découvertes par HGB-All dans base Fruithut avec MinUtil=100000 et MinConf = 0.5*

Règle d'Association	**Utilité**	**Utilité de l'Antécédent**	**Confiance en l'Utilité**
Broccoli / Capsicum red / Field Tomatoes / Apples Red Delicious ==> Banana Cavendish / Watermelon seedless	100224	103958	0.7695
Broccoli / Field Tomatoes / Apples Red Delicious / Watermelon seedless ==> Cauliflower / Banana Cavendish	105384	101463	0.7157
Broccoli / Field Tomatoes / Apples Red Delicious / Watermelon seedless ==> Capsicum red / Banana Cavendish	100224	101463	0.6075
Broccoli / Capsicum red / Apples Red Delicious / Watermelon seedless ==> Field Tomatoes / Banana Cavendish	100224	102736	0.7099
Broccoli / Capsicum red / Apples Red Delicious / Watermelon seedless ==> Cauliflower / Banana Cavendish	100722	102736	0.7099
Broccoli / Cauliflower / Field Tomatoes / Watermelon seedless ==> Apples Red Delicious / Banana Cavendish	105384	109997	0.6568
Broccoli / Capsicum red / Cauliflower / Watermelon seedless ==> Apples Red Delicious / Banana Cavendish	100722	104556	0.6942

4.5 Conclusion

Dans ce chapitre, nous avons examiné les résultats obtenus suite à l'application des algorithmes FHM et HGB-All sur des bases de données réelles. Pour évaluer ces algorithmes, nous avons d'abord considéré trois critères : le temps d'exécution, l'utilisation de la mémoire et le nombre de motifs découverts. Ensuite, une analyse détaillée des motifs résultants a été effectuée, accompagnée de plusieurs exemples et de leurs interprétations.

Chapitre 5

Conclusion Générale

À travers ce mémoire, nous avons examiné en profondeur le domaine de la fouille de données, en mettant l'accent sur la fouille des motifs à haute utilité, *High Utility Pattern Mining (HUPM)*.

Les méthodes représentatives dans ce domaine ont été étudiées et les algorithmes clés ont été analysés pour extraire des ensembles de motifs significatifs.

Dans un premier temps, nous avons souligné l'importance croissante de la fouille de données dans notre société contemporaine, où la quantité de données produites quotidiennement est considérable. Grâce à la fouille de données, nous avons la possibilité de découvrir des informations cachées, des modèles et des tendances à partir de ces données massives. Ensuite, les algorithmes d'extraction des itemsets fréquents ont été examinés, tout en mettant en lumière les limites de la fouille des itemsets fréquents.

À ce stade, nous avons identifié l'importance d'extraire les motifs à haute utilité. Cette approche vise à extraire des motifs pertinents à partir de grandes quantités de données en tenant compte de l'utilité des éléments de données plutôt que de leur fréquence.

Dans la partie pratique de ce mémoire (Chapitre 4), nous avons utilisé deux algorithmes de fouille de motifs à haute utilité, FHM et HGB-ALL, pour extraire deux catégories de motifs à partir de bases de données réelles. Les motifs étudiés comprennent les itemsets à haute utilité et les règles d'association à haute utilité.

Les bases de données réelles posent souvent des défis supplémentaires par rapport aux bases de données artificielles lorsqu'elles sont utilisées avec des algorithmes d'extraction de motifs. Nous avons notamment rencontré un défi significatif lié au volume des données dans les bases de données réelles. Par exemple, une fois que la base de données de la pharmacie a été prétraitée, le nombre total de transactions atteint plus de 2720 transactions impliquant plus de 672 articles différents.

Plusieurs pistes sont envisagées pour étendre et améliorer ce travail : Nous envisageons d'explorer d'autres types de motifs, tels que les motifs séquentiels à grande utilité, et nous souhaitons également tester ces algorithmes sur des bases de données plus complexes afin d'obtenir des patterns plus intéressants.

Références

[1] D. T. Larose and C. D. Larose, *Discovering knowledge in data : an introduction to data mining*. John Wiley & Sons, 2014, vol. 4.

[2] P. Fournier-Viger, J. C.-W. Lin, B. Vo, T. T. Chi, J. Zhang, and H. B. Le, "A survey of itemset mining," *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, vol. 7, no. 4, p. e1207, 2017.

[3] M. Selosse, J. Jacques, and C. Biernacki, "Co-clustering de données textuelles et continues," in *SFdS 2018-50èmes Journées de Statistique*, 2018.

[4] M. Hlady-Rispal, "Une stratégie de recherche en gestion : l'étude de cas," *Revue française de gestion*, no. 8, pp. 251–266, 2015.

[5] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth *et al.*, "Knowledge discovery and data mining : Towards a unifying framework." in *KDD*, vol. 96, 1996, pp. 82–88.

[6] M. Cafaro and M. Pulimeno, "Frequent itemset mining," *Business and Consumer Analytics : New Ideas*, pp. 269–304, 2019.

[7] S. Makhloufi and A. Chetouh, "pfe"découverte des motifs profitables dans les base de données transactionnelles"," Master's thesis, Université de B B Arreridj, 2023.

[8] M. Dupont and J. Martin, "Machine learning et nouvelles sources de données pour le scoring de crédit," *Revue d'économie financière*, vol. XXVIII, no. 1, pp. 123–140, 2024. [Online]. Available : https://www.example.com/article123

[9] C.-H. Chee, J. Jaafar, I. A. Aziz, M. H. Hasan, and W. Yeoh, "Algorithms for frequent itemset mining : a literature review," *Artificial Intelligence Review*, vol. 52, pp. 2603–2621, 2019.

[10] P. Fournier-Viger, C.-W. Wu, S. Zida, and V. S. Tseng, “Fhm : Faster high-utility itemset mining using estimated utility co-occurrence pruning,” in *Foundations of Intelligent Systems : 21st International Symposium, ISMIS 2014, Roskilde, Denmark, June 25-27, 2014. Proceedings 21*. Springer, 2014, pp. 83–92.

[11] M. Bakour, “pfe" Étude comparative entre les algorithmes de haute utilité et les algorithmes de représentation concise.",” Master’s thesis, Université de B B Arreridj, 2023.

[12] M. Nouioua, P. Fournier-Viger, C.-W. Wu, J. C.-W. Lin, and W. Gan, “Fhuqi-miner : Fast high utility quantitative itemset mining,” *Applied Intelligence*, vol. 51, pp. 6785–6809, 2021.

[13] H. Yao, H. J. Hamilton, and C. J. Butz, “A foundational approach to mining itemset utilities from databases,” in *Proceedings of the 2004 SIAM International Conference on Data Mining*. SIAM, 2004, pp. 482–486.

[14] M. L. Merrouche and T. Miloudi, “Mémoire de fin d’étude,” Master’s thesis, Université de B B Arreridj, 2013.

[15] W. Gan, J. C.-W. Lin, P. Fournier-Viger, H.-C. Chao, V. S. Tseng, and S. Y. Philip, “A survey of utility-oriented pattern mining,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, pp. 1306–1327, 2019.

[16] Y. Liu, W.-k. Liao, and A. Choudhary, “A two-phase algorithm for fast discovery of high utility itemsets,” in *Advances in Knowledge Discovery and Data Mining : 9th Pacific-Asia Conference, PAKDD 2005, Hanoi, Vietnam, May 18-20, 2005. Proceedings 9*. Springer, 2005, pp. 689–695.

[17] M. Liu and J. Qu, “Mining high utility itemsets without candidate generation,” in *Proceedings of the 21st ACM international conference on Information and knowledge management*, 2012, pp. 55–64.

[18] J. Sahoo, A. K. Das, and A. Goswami, “An efficient approach for mining association rules from high utility itemsets,” *Expert systems with Applications*, vol. 42, no. 13, pp. 5754–5778, 2015.

[19] S. R. Sahoo, R. K. Dash, and S. C. Meher, “High utility association rule mining with all

items," *Knowledge-Based Systems*, vol. 80, pp. 1–14, 2015.

[20] Java - what is java ? En ligne. [Online]. Available : https://www.java.com/en/download/help/whatis_java.html

[21] (s.d.) Python w3schools. W3Schools Online Web Tutorials. [Online]. Available : https://www.w3schools.com/default.asp

[22] (s.d.) Netbeans. Dans Wikipédia. [Online]. Available : https://en.wikipedia.org/wiki/NetBeans

[23] P. Fournier-Viger. (s.d.) SPMF : Data mining algorithms in java. [Online]. Available : https://www.philippe-fournier-viger.com/spmf/index.php