République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique

Option: *Modèle Intelligent et Décision (MID)*

Thème

Extraction des Itemsets fréquents à base d'Answer Set Programming (ASP)

Réalisé par :

- **Ziram Yasmine**
- **Imakhlef Macilia**

Présenté le 26 Juin 2023 devant le jury composé de

- *Mr. HADJILA FETHALLAH* *(Président)*
- *Mr. BELABED AMINE* *(Encadreur)*
- *Mr. SMAHI MOHAMED ISMAIL* *(Examinateur)*

Année universitaire : 2022-2023

REMERCIEMENT

À l'issue de cette étude, nous souhaitons exprimer notre sincère gratitude et reconnaissance envers toutes les personnes qui ont contribué, de près ou de loin, à la réalisation de ce mémoire.

Tout d'abord, nous tenons à remercier chaleureusement notre encadreur, M. BELABED Amine, pour son assistance, sa disponibilité, son suivi constant et ses précieux conseils qui ont grandement contribué à la réussite de ce travail.

Nous tenons à exprimer toute notre gratitude à tous les membres de jury, pour avoir bien voulu juger notre travail.

Nous souhaitons également exprimer notre reconnaissance envers l'ensemble du personnel du département informatique et tous les professeurs qui nous ont accompagnés tout au long de nos études.

DÉDICACE

Avant toute chose, je tiens à exprimer ma gratitude envers Allah, le Tout puissant et Miséricordieux, qui m'a accordé la force et la patience nécessaires pour accomplir ce modeste travail.

C'est avec un grand plaisir que je dédie ce modeste travail, en expriment ma profonde gratitude à tous mes parent et proches.

À ma mère, qui n'a épargné aucun effort pour me rendre heureuse.

À mon père, qui est le fondement de ma vie, ma réussite et toute mon admiration.

À mon époux, qui n'a cessé de me conseiller, d'encourager et de soutenir tout au long de mes études.

À mon fils Haytham, qui sait toujours comment apporter la joie et bonheur à mon cœur.

À ma sœur et mon frère, que Dieu les protège et leur offre chance et bonheur.

Ainsi qu'à mon binôme, Imakhlef Macilia, pour notre collaboration fructueuse et notre soutien mutuel.

Enfin, cette dédicace est une humble reconnaissance envers tous ceux qui ont contribué de près ou de loin à mon parcours universitaire. Votre soutien inconditionnel, vos encouragements et vos sacrifices ont été les piliers de ma réussite.

Que cette modeste réalisation soit le témoignage de ma reconnaissance éternelle envers ceux qui m'ont accompagné tout au long de cette aventure académique.

Ziram Yasmine

DÉDICACE

C'est avec une immense gratitude que je dédie ce mémoire à vous tous. Votre soutien, votre amour inconditionnel et vos encouragements constants ont été les piliers qui ont rendu cette réalisation possible.

À ma mère, qui m'a inculqué des valeurs de persévérance et de détermination, je vous remercie pour votre amour inébranlable et votre soutien indéfectible. Votre présence réconfortante a été mon ancre dans les moments de doute et d'incertitude.

À mon père, dont la sagesse et l'engagement m'ont inspiré chaque jour, je vous suis reconnaissant pour votre soutien inépuisable et vos précieux conseils. Votre présence encourageante a été un moteur de motivation tout au long de ce parcours.

À mes trois sœurs, qui ont partagé cette aventure avec moi, je vous remercie pour votre amour inconditionnel, vos encouragements incessants et votre soutien mutuel. Votre présence joyeuse et vos encouragements constants ont allégé le fardeau des défis rencontrés.

À mon frère et sa femme, je vous exprime ma profonde reconnaissance. Votre présence solidaire et vos encouragements ont été une source de force et de motivation inestimable.

À Yasmine, qui a partagé les hauts et les bas de ce voyage académique, je vous remercie pour votre amitié, votre collaboration et votre persévérance. Votre soutien a été précieux pour surmonter les obstacles et atteindre nos objectifs communs.

À ma chère amie Nairouz, qui a été une épaule sur laquelle m'appuyer, une source d'inspiration et un soutien inconditionnel, je vous remercie pour votre présence constante et vos mots d'encouragement qui m'ont porté à travers les moments les plus difficiles.

Ce mémoire est le fruit de nos efforts collectifs et de votre soutien indéfectible. Votre présence dans ma vie a été un cadeau inestimable et cette dédicace est une modeste expression de ma gratitude éternelle.

Avec tout mon amour et ma reconnaissance,

Imakhlaf Macilia

Résumé

Ce mémoire porte sur l'extraction des itemsets fréquents en utilisant l'Answer Set Programming (ASP). L'implémentation des codes d'extraction des itemsets fréquents fermés et maximaux est réalisée en utilisant Clingo. Les expérimentations menées pour comparer les performances de l'approche ASP avec les algorithmes de l'outil SPMF ont montré l'efficacité de notre encodage ASP dans l'extraction des itemsets fréquents, surtout pour les bases de données de petite taille.

Mots-clés : itemsets fréquents, Answer Set Programming, extraction de données, Clingo, performances.

Abstract

This work presents an approach for extracting frequent itemsets using Answer Set Programming (ASP). The proposed ASP encoding for extracting frequent closed and maximal itemsets is implemented using Clingo. The experiments conducted to compare the performance of the ASP approach with the algorithms of the SPMF tool have shown the efficiency of our ASP encoding in the extraction of frequent itemsets, particularly for small databases.

Keywords: frequent itemsets, Answer Set Programming, data mining, Clingo, performance.

ملخص

تتناول هذه المذكرة استخراج مجموعات العناصر المتكررة باستخدام برمجة مجموعة الإجابات (ASP). تم تنفيذ تطبيق أكواد استخراج مجموعات العناصر المتكررة المغلقة والقصوى باستخدام Clingo. أظهرت التجارب التي أجريت لمقارنة أداء نهج ASP مع خوارزميات أداة SPMF كفاءة ترميزنا ASP في استخراج مجموعات العناصر المتكررة، خاصةً لقواعد البيانات الصغيرة.

الكلمات الرئيسية: المجموعات العنصرية المتكررة، برمجة مجموعات الإجابة، استخراج البيانات، Clingo، الأداء.

Table des matières

La liste des figures

La liste des tableaux

La liste des listings

Introduction générale

L'extraction des Itemsets fréquents est une tâche essentielle dans l'analyse de données et la découverte de connaissances dans divers domaines. Ces Itemsets fréquents sont des ensembles d'éléments qui se produisent fréquemment dans un ensemble de données. Leur identification permet de mettre en évidence des associations et des tendances significatives, ouvrant ainsi la voie à des décisions éclairées et à des stratégies efficaces. Cependant, la recherche d'Itemsets fréquents dans de grandes bases de données représente un défi en termes de complexité et de performance.

C'est dans ce contexte que l'Answer Set Programming (ASP), une approche déclarative pour la résolution de problèmes combinatoires complexes, offre des perspectives intéressantes. L'ASP permet de spécifier un problème à l'aide de règles logiques et de calculer les ensembles de réponses, fournissant ainsi une solution détaillée au problème posé. L'utilisation de l'ASP pour l'extraction des Itemsets fréquents présente de nombreux avantages, notamment la facilité de modélisation des contraintes et la capacité à traiter des ensembles de données de grande taille.

Le présent mémoire de fin d'études se concentre sur l'extraction des Itemsets fréquents à base d'Answer Set Programming. Dans le premier chapitre, nous explorerons en détail les Itemsets fréquents, en mettant l'accent sur leur définition, leur importance et leurs utilisations dans différents domaines. Cette étape initiale permettra de bien comprendre les enjeux de l'extraction des Itemsets fréquents et de situer notre approche dans le contexte plus large de la fouille de données.

Le deuxième chapitre sera consacré à une introduction approfondie à l'Answer Set Programming. Nous aborderons les bases de l'ASP, y compris sa syntaxe et sa sémantique. Nous expliquerons également les étapes de résolution d'un problème avec l'ASP, en mettant en évidence les concepts clés tels que les programmes ASP, les règles de programmation et les ensembles de réponses. Cette compréhension approfondie de l'ASP jettera les bases nécessaires pour aborder l'extraction des Itemsets fréquents à travers cette approche.

Le troisième chapitre sera dédié à l'implémentation concrète des codes d'extraction des Itemsets fréquents fermés et maximaux en utilisant l'outil Clingo pour la programmation ASP. Nous détaillerons les fondements théoriques de ces concepts d'Itemsets fermés et maximaux, en soulignant leurs avantages et leurs utilisations. Ainsi, nous présenterons le processus d'implémentation avec Clingo, décrivant en détail le codage des règles ASP pour l'extraction

des Itemsets fréquents, fermés et maximaux. Enfin, nous discuterons des expérimentations réalisées avec des jeux de données réels et des résultats obtenus, en comparant les performances de notre approche ASP avec l'outil SPMF.

En conclusion, ce mémoire vise à démontrer l'efficacité de l'Answer Set Programming pour l'extraction des Itemsets fréquents. Nous mettrons en évidence les avantages de cette approche déclarative en termes de modélisation, de flexibilité et de capacité à traiter des ensembles de données complexes. De plus, nous discuterons des perspectives futures de recherche dans le domaine de l'ASP et des Itemsets fréquents, tout en examinant les défis et les limites existants.

Chapitre I Itemsets Fréquents

I.1 Introduction

Au cours de la dernière décennie, l'activité humaine liée aux applications informatiques a connu un développement exponentiel, entraînant une demande croissante en puissance de calcul, capacité de stockage et rapidité de transmission des réseaux. Cette évolution a conduit à la collecte massive et automatisée de données provenant de divers domaines, ce qui a entraîné une augmentation considérable du volume de données numériques conservées chaque année.

Dans ce contexte, l'idée du Data-mining ou de la fouille de données a émergé, visant à exploiter ces vastes ensembles de données au-delà de leur utilisation traditionnelle. Grâce à l'utilisation d'outils d'analyse avancés, le Data mining permet aux utilisateurs de découvrir des informations jusque-là inconnues ou implicites dans leurs propres données. Ces informations sont extraites sous forme de règles, modèles ou concepts, révélant ainsi des connaissances cachées.

Grâce à l'amélioration des performances des systèmes informatiques et à la maturité des techniques d'apprentissage automatique, le Data mining trouve de nombreuses applications dans divers domaines tels que la distribution, les télécommunications, les institutions financières, la recherche médicale et scientifique, la planification urbaine, la météorologie, l'agriculture et l'industrie.

Ce chapitre présente une vue d'ensemble d'un sous domaine du Data mining, c'est l'extraction des item-sets fréquents. Nous présentons les notions de base de ce domaine, toute en mettant en évidence certaines applications importantes dans différents domaines. Nous abordons également les algorithmes d'extraction des itemsets fréquents, et en présentant les tâches et techniques associées. En conclusion, nous soulignons les points clés abordés.

I.2 Notions de bases

I.2.1 Définition

L'extraction des itemsets fréquents (FIM) est une tâche essentielle dans l'analyse des données, visant à extraire les ensembles d'éléments qui se produisent fréquemment dans un ensemble de données, dépassant un seuil d'occurrence spécifié. Initialement introduite au début des années 1990 pour l'analyse des paniers d'achat afin d'identifier les articles qui cooccurrent fréquemment, l'extraction des itemsets fréquents était initialement appelée "l'extraction des grands ensembles d'articles". Cette section présente des définitions formelles liées à cette tâche, explore divers domaines d'application et aborde les tâches connexes.

La tâche de l'extraction des itemsets fréquents, proposée par (Agrawal, Rakesh, Tomasz Imieliński, and Arun Swami, 1993, June), consiste à extraire tout itemset qui se produit fréquemment dans les données, en veillant à ce que sa fréquence d'occurrence dépasse un seuil prédéfini. Cette tâche trouve des applications dans des domaines variés, chacun ayant ses propres exigences et défis. De plus, cette section aborde plusieurs tâches connexes et met en évidence leur importance (Jose Maria Luna, Philippe Fournier-Viger, Sebastian Ventura).

I.2.1.1 Base de données transactionnelle

Une base de données transactionnelle est représentée par un triplé (O, P, R), où O est un ensemble fini d'objets, P est un ensemble fini d'éléments ou d'items, et R est une relation binaire entre ces deux ensembles. La base de données D= (O, P, R) représente notre espace de travail. (Claude Issa Nombré, Konan Brou, Kouadio Kimou, 2016)

Voici un exemple d'une base de données transactionnelle représentée par le triplé (O, P, R) :

O : {Client1, Client2, Client3} (ensemble fini d'objets représentant les clients)

P : {Lait, Pain, Bure} (ensemble fini d'éléments ou d'items représentant les produits)

R : {(Client1, Lait), (Client2, Pain), (Client3, Bure)} (relation binaire entre les clients et les produits).

Dans cet exemple, la base de données transactionnelle D représente un espace de travail lié aux transactions entre les clients et les produits. Chaque élément de l'ensemble O (clients) est associé à un élément de l'ensemble P (produits) par la relation R. Par exemple, (Client1, Lait) indique que le "Client1" a effectué une transaction avec le " Lait ". De même, (Client2, Pain) représente une autre transaction entre le "Client2" et le " Pain".

I.2.1.2 Transaction

Une transaction est un ensemble de lignes contenant les occurrences de la base de données D. Soit :

$$T = \{T1, T2, T3, \ldots, Tn\}, \text{ où } Ti \subseteq D.$$

Dans l'exemple courant du panier de la ménagère, les transactions correspondent aux tickets de caisse, c'est-à-dire aux achats effectués par les clients. (Claude Issa Nombré, Konan Brou, Kouadio Kimou, 2016)

I.2.1.3 Item

Un item représente une occurrence de la base de données D. Il est symbolisé par la variable Xi. (Claude Issa Nombré, Konan Brou, Kouadio Kimou, 2016)

I.2.1.4 Itemset

Un itemset est un ensemble d'items. Par exemple, un singleton $\{X1\}$, et une paire $\{X1, X2\}$, sont des itemsets. Un itemset de taille k est noté k-itemset. (Claude Issa Nombré, Konan Brou, Kouadio Kimou, 2016)

I.2.1.5 Support

La fréquence relative à laquelle cet itemset apparaît dans un ensemble de données. Il est calculé en divisant le nombre de transactions contenant l'itemset par le nombre total de transactions dans le jeu de données. Le support est souvent utilisé comme seuil pour déterminer les itemsets fréquents. (Claude Issa Nombré, Konan Brou, Kouadio Kimou, 2016)

Dans la Figure 1 suivante, en fixant le support minimum à 2 (ou 20% en relatif).

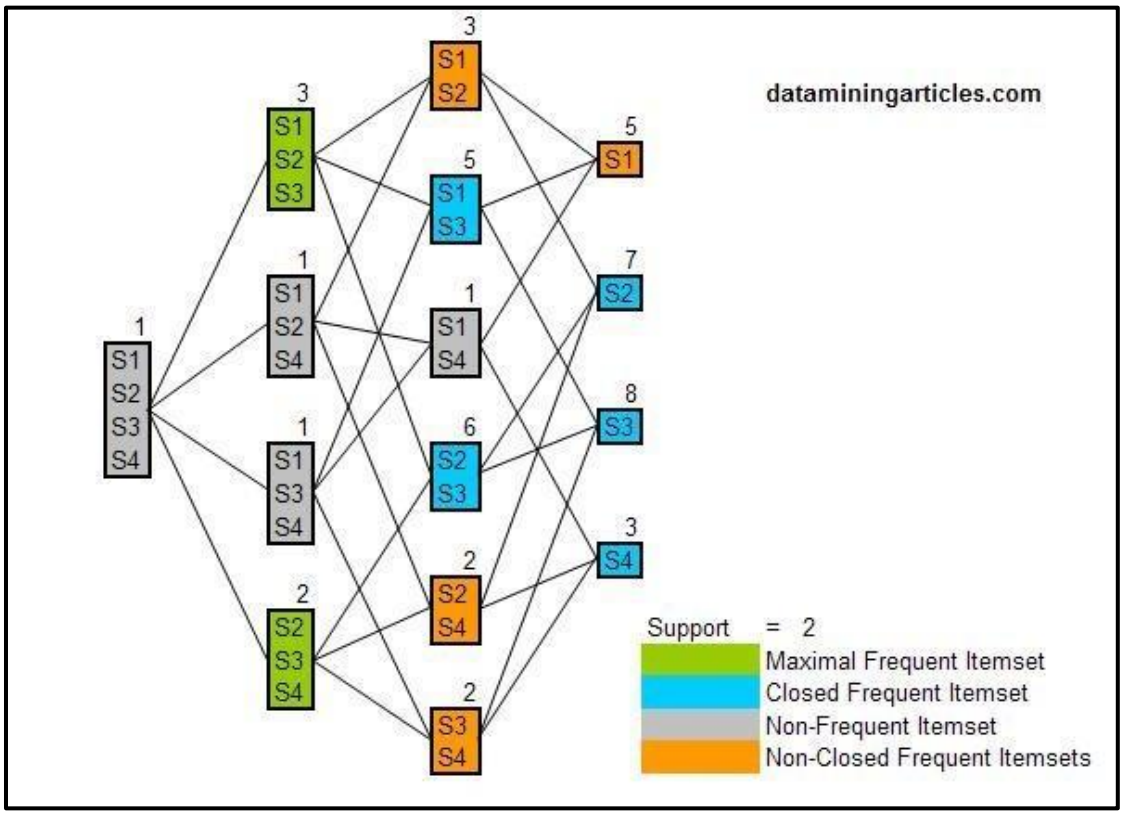

Figure 1 : Extraction des itemsets fréquent (TANAGRA, 2020)

I.2.1.6 Règle d'association

Une technique de data mining utilisée pour rechercher des relations cachées entre les attributs. Une règle d'association est un schéma qui indique que lorsque X se produit, Y se produit avec une certaine probabilité. (Sharma, N. and Om, H, 2014)

I.2.1.7 Confiance

La confiance est le pourcentage de fois où une règle (Xi, Xj) est vérifiée. Elle est calculée comme suit : (Claude Issa Nombré, Konan Brou, Kouadio Kimou, 2016)

$$Confiance(Xi \rightarrow Xj) = \frac{freq(Xi \cup Xj)}{freq(Xi)}$$

I.2.1.8 Superset

Un superset est un itemset qui est défini par rapport à un autre itemset. Par exemple, {S1, S2, S3} est un superset de {S1, S2}. (Claude Issa Nombré, Konan Brou, Kouadio Kimou, 2016)

I.2.1.9 Itemset fréquent

Un itemset fréquent est un itemset dont le support est supérieur ou égal à minsup (support minimal en dessous duquel l'itemset est considéré comme non fréquent). Si un itemset n'est pas fréquent, alors tous ses supersets ne le seront pas non plus. Si un superset est fréquent, alors tous ses sousitemsets sont également fréquents (propriété anti-monotone). (Claude Issa Nombré, Konan Brou, Kouadio Kimou, 2016)

Nous observons dans la Figure 1 les itemsets fréquents sont toutes les combinaisons non-grisées.

I.2.1.10 Itemset fermé

Un itemset fréquent est dit fermé s'il n'a aucun superset avec un support identique. En d'autres termes, tous ses supersets ont un support strictement inférieur. (Claude Issa Nombré, Konan Brou, Kouadio Kimou, 2016)

Pour l'exemple de la Figure 1, {S1, S3} est fermé car aucun de ses supersets n'a de support égal à 5/10 : $Support$ ({$S1$, $S2$, $S3$}) = 3/10, $Support$ ({$S1$, $S3$, $S4$}) = 1/10.

I.2.1.11 Itemset maximal

Un itemset maximal est un itemset qui n'a aucun superset fréquent. (Claude Issa Nombré, Konan Brou, Kouadio Kimou, 2016)

D'après l'exemple de la Figure 1, {$S1$, $S2$, $S3$} est maximal car son supersets {$S1$, $S2$, $S3$, $S4$} (il n'y en a qu'un) n'est pas fréquent avec un support de 1/10.

I.2.1.12 Itemset générateur

Un itemset générateur est un itemset dont tous les sous itemsets ont un support strictement supérieur. (Claude Issa Nombré, Konan Brou, Kouadio Kimou, 2016)

Dans notre exemple (la Figure 1) : {S1, S2, S3} de support 4/10 n'est pas générateur puisqu'on trouve {$S1$, $S2$} avec un support identique. En revanche, {$S2$, $S4$}, $de\ support$ 2/10, est générateur car $Su(\{S2\})$ = 7/10 et $Support(\{S4\})$ = 3/10.

I.2.2 Algorithmes d'extraction des itemsets fréquents

I.2.2.1 Algorithme Apriori

Apriori est un algorithme proposé par (Rakesh Agrawal and Ramakrishnan Srikant., September 1994.) qui extrait des ensembles d'éléments fréquents pour générer des règles d'association booléennes. Il utilise une technique de recherche itérative de niveau en niveau pour découvrir les ensembles d'éléments (k + 1) à partir des ensembles d'éléments k. Un échantillon de données transactionnelles, composé d'articles achetés lors de différentes transactions, est présenté dans la Figure 2. Tout d'abord, la base de données est analysée pour identifier tous les ensembles d'éléments fréquents en comptant chacun d'entre eux et en capturant ceux qui satisfont le seuil de support minimal. L'identification de chaque ensemble d'éléments fréquent nécessite de parcourir l'ensemble de la base de données jusqu'à ce qu'aucun ensemble d'éléments k fréquent supplémentaire ne puisse être identifié. Selon la Figure 3, le seuil de support minimal utilisé est de 2. Par conséquent, seuls les enregistrements qui satisfont un comptage de support minimal de 2 seront inclus dans le cycle suivant du traitement de l'algorithme. (Chin-Hoong Chee, Jafreezal Jaafar, Izzatdin Abdul Aziz, Mohd Hilmi Hasan, 2018)

TID	List of item_IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Figure 2 : Échantillon de données transactionnelles (Chin-Hoong Chee, Jafreezal Jaafar, Izzatdin Abdul Aziz, Mohd Hilmi Hasan, 2018)

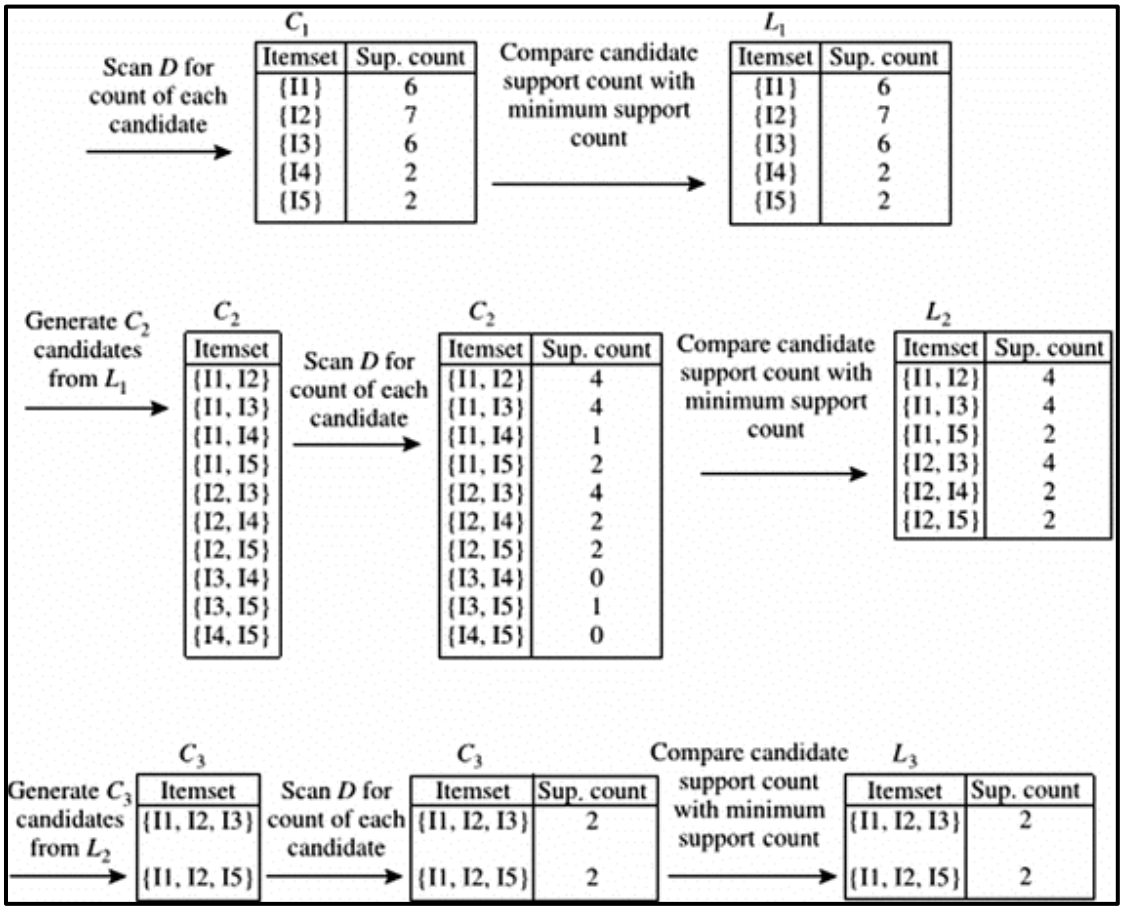

Figure 3 : Génération d'ensembles d'articles candidats et d'ensembles d'articles fréquents. (Chin-Hoong Chee, Jafreezal Jaafar, Izzatdin Abdul Aziz, Mohd Hilmi Hasan, 2018)

Dans de nombreux cas, l'algorithme Apriori permet de réduire considérablement la taille des ensembles d'articles candidats et offre un gain de performance significatif. Cependant, il présente toujours deux limitations critiques. Premièrement, un grand nombre d'ensembles d'articles candidats peuvent encore devoir être générés si le nombre total d'ensembles d'articles fréquents de taille k augmente. Ensuite, il est nécessaire de scanner à plusieurs reprises l'ensemble de la base de données et de vérifier un ensemble énorme d'articles candidats à l'aide de la technique de correspondance de motifs.

Dans le cadre de cette étude, nous présentons l'algorithme Apriori dans la figure 4 : (Raj, 2020)

Algorithm: *Apriori algorithm*

Input: D: Input Dataset
$minSup$: minimum support threshold

Output: All 2 to k-frequent itemsets

1. L_1= {1-frequent itemset} // found separately
2. ***for*** ($k = 2$; $L_{k-1} \neq \varphi$; k++)
3. $C_k = apriori\_gen(L_{k-1})$ // finds k-candidate itemsets by joining and pruning L_{k-1} with itself
4. ***for each*** transaction t in D
5. $C_t = subset\ (C_k, t)$ // finds candidate itemsets in t
6. ***for each*** c in C_t
7. c.count++
8. ***end for each***
9. ***end for each***
10. $L_k = \{c \in C_k \mid c.\text{count} \geq minSup\}$
11. ***end for***
12. Return $\cup_k L_k$

Figure 4 : Algorithme Apriori (Raj, 2020)

I.2.2.2 Algorithme FP-Growth

L'algorithme FP-Growth est une méthode alternative pour trouver des ensembles d'articles fréquents sans utiliser la génération de candidats, proposé par (Han J. P., 2004), ce qui améliore les performances. Pour ce faire, il utilise une stratégie de diviser pour régner. C'est l'utilisation d'une structure de données spéciale appelée arbre de motifs fréquents (FP-tree), qui conserve les informations d'association des ensembles d'articles.

Cet algorithme fonctionne de la manière suivante : (Jaiswal)

Premièrement, il compresse la base de données d'entrée en créant une instance de FP-tree pour représenter les articles fréquents.

Après cette première étape, il divise la base de données compressée en un ensemble de bases de données conditionnelles, chacune associée à un motif fréquent.

Enfin, chaque base de données est explorée séparément.

En utilisant cette stratégie, l'algorithme FP-Growth réduit les coûts de recherche en recherchant de manière récursive des motifs courts, puis en les concaténant pour former des motifs fréquents plus longs.

Dans de grandes bases de données, il est impossible de conserver l'arbre FP en mémoire principale. Une stratégie pour faire face à ce problème consiste à partitionner la base de données en un ensemble de bases de données plus petites (appelées bases de données projetées) et à construire ensuite un FP-tree à partir de chacune de ces bases de données plus petites.

Voici l'algorithme FP-Growth : (Konya, Türkiye, November 2020)

Procedure $FPgrowth^*(T)$

Input: A conditional FP-tree T

Output: The complete set of all FI's corresponding to T.

Method:

1. **if** T only contains a single branch B
2. **for each** subset Y of the set of items in B
3. output itemset $Y \cup T.base$ with count = smallest count of nodes in Y;
4. **else for each** i in $T.header$ **do begin**
5. output $Y = T.base \cup \{i\}$ with $i.count$;
6. **if** $T.FP\text{-}array$ is defined
7. construct a new header table for Y's FP-tree from $T.FP\text{-}array$
8. **else** construct a new header table from T;
9. construct Y's conditional FP-tree T_Y and possibly its FP-array A_Y;
10. **if** $T_Y \neq \emptyset$
11. call $FPgrowth^*(T_Y)$;
12. **end**

Figure 5 : Algorithme FP-Growth (Konya, Türkiye, November 2020)

Exemple : Ce jeu de données contient deux attributs et cinq instances. Le premier attribut est l'identifiant de transaction et le deuxième attribut représente essentiellement l'ensemble d'articles. (Amir, 2019.)

Table 1 : La base de données

Transaction ID	Itemset
01	f,a,c,d,g,i,m,p
02	a,b,c,f,l,m,o
03	b,f,h,j,o
04	b,c,k,s,p
05	a,f,c,e,i,p,m,n

Résolvons cela étape par étape.

- Étape 1 : Trouver le support minimum de chaque élément.

Table 2 : Support minimum de chaque élément

Item	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**s**
Support	3	3	4	1	1	4	1	1	1	1	1	2	3	1	2	3	1

Support minimum = 3, supprimé les éléments du tableau dont le support est inférieur à 3.

Table 3 : Les item dont le support est supérieur à 3

Item	Support
a	3
b	3
c	4
f	4
m	3
p	3

- Étape 2 : Classer les éléments fréquents par ordre décroissant

f = 4, c = 4, a = 3, b = 3, m = 3, p = 3

Table 4 : Classement des éléments fréquents par ordre décroissant

Transaction ID	Item	Ordonnée les items fréquent
1	f,a,c,d,g,i,m,p	f,a,c,m,p
2	a,b,c,f,l,m,o	a,b,c,f,m
3	b,f,h,j,o	b,f,
4	b,c,k,s,p	b,c,p
5	a,f,c,e,i,p,m,n	a,f,c,p,m

- Étape 3 : Arbre FP

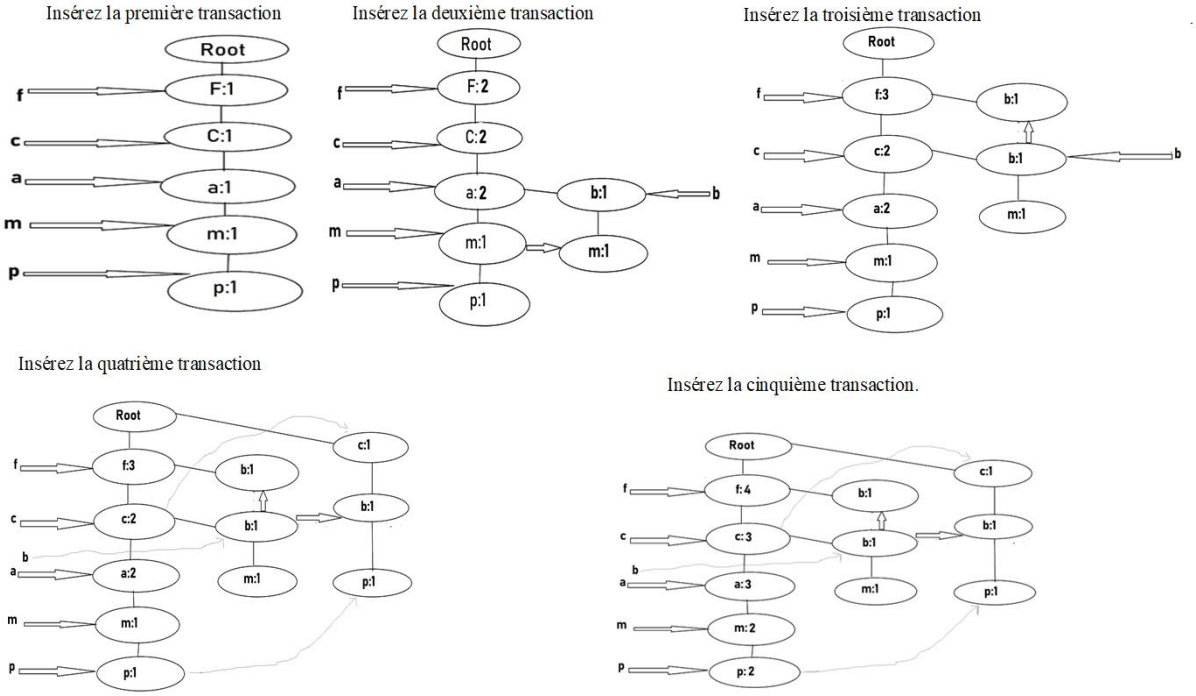

Figure 6 : Insertion des transactions (Amir, 2019.)

- Étape 4 : Extraction des motifs fréquents à partir de l'arbre FP

Table 5 : Extraction des itemsets fréquents de l'arbre FP

Item	Conditional pattern base	Conditional FP tree
p	{fcam : 2} {cb : 1}	(c : 3) / p
m	{fca : 2} {fcab : 1}	(f : 3, c : 3, a :3) / m
b	{fca : 2} {c : 1} {f : 1}	Vide
a	{fc : 3}	(f : 3, c : 3) / a
c	{f : 3}	(f : 3) / c
f	vide	vide

I.2.2.3 Analyse comparative des algorithmes Apriori, FP-Growth

Table 6 : Différence entre Apriori et FP-Growth (Verma, 2021)

Apriori	FP Growth
Apriori génère des motifs fréquents en créant des ensembles d'éléments à partir de combinaisons telles qu'un ensemble d'éléments simples, un ensemble d'éléments doubles et un ensemble d'éléments triples.	FP-Growth génère un arbre FP (FP-Tree) pour créer des motifs fréquents.
Apriori utilise la génération de candidats où les sous-ensembles fréquents sont étendus d'un élément à la fois.	FP-Growth génère un arbre FP conditionnel pour chaque élément dans les données.
Étant donné qu'Apriori analyse la base de données à chaque étape, cela devient chronophage pour des données où le nombre d'éléments est important.	L'arbre FP (FP-Tree) nécessite seulement un balayage de la base de données dans ses premières étapes, ce qui le rend moins chronophage.
Une version convertie de la base de données est sauvegardée en mémoire.	Un ensemble d'arbres FP conditionnels pour chaque élément est enregistré dans la mémoire
Il utilise une recherche en largeur (breadth-first search).	Il utilise une recherche en profondeur d'abord. (depth-first search)

I.2.3 Algorithme CHARM pour l'extraction des itemsets fréquents fermés

Charm est un algorithme d'exploration de motifs fréquents dans les bases de données transactionnelles introduit par (Zaki, M. J., & Hsiao, C.-J., 2002). Il utilise une représentation verticale de la base de données, stockant avec chaque itemset une liste des numéros de transactions qui le supportent, appelée "tid list". Cette représentation permet une recherche efficace des motifs fréquents. L'algorithme Charm se distingue par son utilisation des "diffsets", qui sont des listes des numéros de transactions qui supportent le père de l'itemset mais ne supportent pas l'itemset lui-même. Cette approche permet de réduire la taille des listes et d'optimiser les opérations de différence. En utilisant ces structures de données optimisées, Charm explore l'espace des itemsets de manière efficace, en appliquant des stratégies d'élagage pour éviter d'explorer les itemsets qui ne sont pas susceptibles de devenir fréquents. Cela permet de réduire le temps de calcul et de découvrir rapidement les motifs fréquents dans la base de données. En résumé, Charm est un algorithme d'exploration de motifs fréquents qui utilise une représentation verticale et des structures de données optimisées pour accélérer la recherche des itemsets fréquents dans les bases de données transactionnelles. (Jeudy, 2002)

```
CHARM (D, min_sup):
1.  [P] = {X_i × t(X_i) : X_i ∈ I ∧ σ(X_i) ≥ min_sup}
2.  CHARM-EXTEND ([P], C = ∅)
3.  return C //all closed sets

CHARM-EXTEND ([P], C):
4.  for each X_i × t(X_i) in [P]
5.      [P_i] = ∅ and X = X_i
6.      for each X_j × t(X_j) in [P], with X_j ≥_f X_i
7.          X = X ∪ X_j and Y = t(X_i) ∩ t(X_j)
8.          CHARM-PROPERTY([P], [P_i])
9.      if ([P_i] ≠ ∅) then CHARM-EXTEND ([P_i], C)
10.     delete [P_i]
11.     C = C ∪ X //if X is not subsumed

CHARM-PROPERTY ([P], [P_i]):
12.  if (σ(X) ≥ minsup) then
13.      if t(X_i) = t(X_j) then //Property 1
14.          Remove X_j from [P]
15.          Replace all X_i with X
16.      else if t(X_i) ⊂ t(X_j) then //Property 2
17.          Replace all X_i with X
18.      else if t(X_i) ⊃ t(X_j) then //Property 3
19.          Remove X_j from [P]
20.          Add X × Y to [P_i] //use ordering f
21.      else if t(X_i) ≠ t(X_j) then //Property 4
22.          Add X × Y to [P_i] //use ordering f
```

Figure 7 : Algorithme CHARM (Mohammed J. Zaki, 2014)

Voici un exemple concret :

Soit un sup_min = 2.

Table 7 : Exemple d'une base de données transactionnelle

TID	Itemset
01	abcde
02	cd
03	acde
04	bcd
05	ad
06	bcde

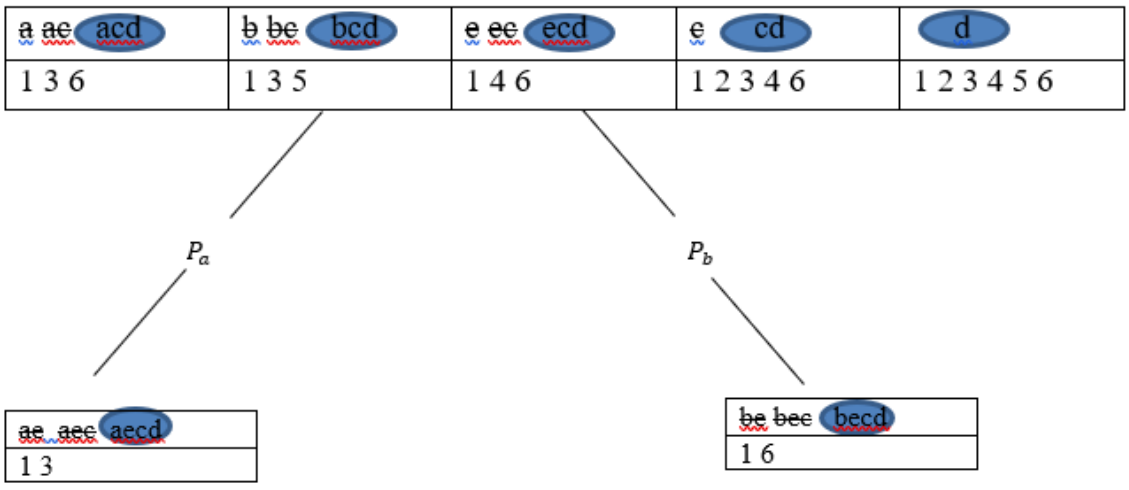

Figure 8 : Exemple d'extraction des itemsets fréquents fermés (CHARM)

I.2.4 Algorithme Gen-MAX pour l'extraction des itemsets fréquents maximaux

GenMax utilise une recherche par retour en arrière pour énumérer le Itemset Fréquemment Maximale (MFI) par (Karam Gouda & Mohammed J. Zaki , 2005). Nous décrivons d'abord le paradigme du retour en arrière dans le contexte de l'énumération de tous les motifs fréquents. Nous modifierons ensuite cette procédure pour énumérer le MFI.

Les algorithmes de retour en arrière sont utiles pour de nombreux problèmes combinatoires où la solution peut être représentée comme un ensemble $I = \{i_o, i_1, \ldots\}$,où chaquei_i est choisi parmi un ensemble fini possible, P_j. Initialement, I est vide ; il est étendu un élément à la fois, à mesure que l'espace de recherche est parcouru. La longueur de I est la même que la profondeur du nœud correspondant dans l'arbre de recherche. Étant donné une solution partielle de longueur l, $I_l = \{i_0, i_1, \ldots, i_{l-1}\}$, les valeurs possibles pour le prochain élément il proviennent d'un sous-ensemble $C_l \subseteq P_l$ appelé ensemble de combinaison. Si $y \epsilon C_l - P_l$, alors les nœuds du sous-arbre dont le nœud racine est $I_l = \{i_0, i_1, \ldots, i_{l-1}, y\}$ ne seront pas pris en compte par l'algorithme de retour en arrière. Étant donné que de tels sous-arbres ont été élagués de l'espace de recherche d'origine, la détermination de C_l est également appelée élagage. (Karam Gouda & Mohammed J. Zaki , 2005)

```
// Initial Call: M ← ∅, P ← {⟨i, t(i)⟩ | i ∈ I, sup(i) ≥ minsup}
GENMAX (P, minsup, M):
1  Y ← ⋃ X_i
2  if ∃Z ∈ M, such that Y ⊆ Z then
3      return // prune entire branch
4  foreach ⟨X_i, t(X_i)⟩ ∈ P do
5      P_i ← ∅
6      foreach ⟨X_j, t(X_j)⟩ ∈ P, with j > i do
7          X_ij ← X_i ∪ X_j
8          t(X_ij) = t(X_i) ∩ t(X_j)
9          if sup(X_ij) ≥ minsup then P_i ← P_i ∪ {⟨X_ij, t(X_ij)⟩}
10     if P_i ≠ ∅ then GENMAX (P_i, minsup, M)
11     else if ∄Z ∈ M, X_i ⊆ Z then
12         M = M ∪ X_i   // add X_i to maximal set
```

Figure 9 : Algorithme GENMAX (Karam Gouda & Mohammed J. Zaki , 2005)

De l'exemple précèdent (Tableau 4) appliquons l'algorithme de GenMax pour extraire les itemset fréquent maximum pour un sup_min = 3 :

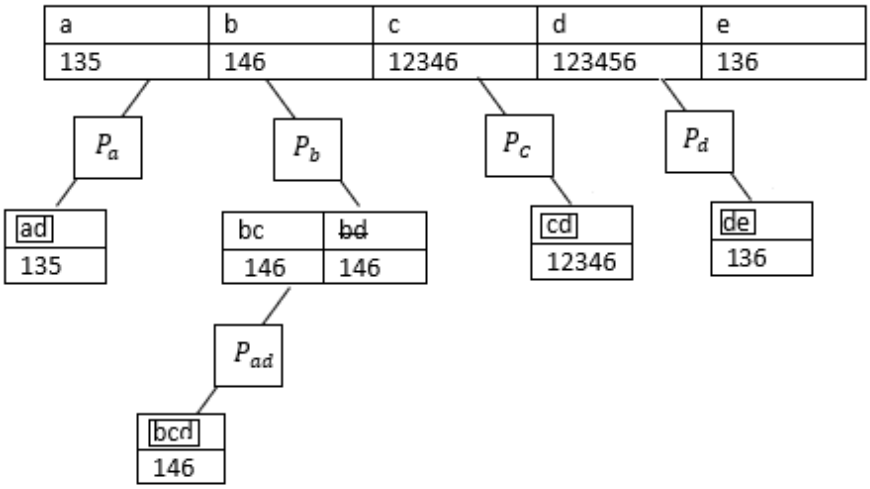

Figure 10 : L'extraction des itemsets fréquents maximaux (Gen-MAX)

I.3 L'utilisation des itemsets fréquents dans la pratique

I.3.1 Market Basket Analysis (Analyse des paniers d'achats)

Cette application vise à identifier les associations fréquentes entre les produits achetés par les clients. Les titres possibles pour cette application pourraient être : "Analyse des combinaisons fréquentes de produits dans les transactions commerciales" ou "Détection des relations entre les articles achetés ensemble". (Cavique, 2007)

I.3.2 DNA Sequence Analysis (Analyse des séquences d'ADN)

Cette application concerne l'identification de motifs fréquents dans les séquences génétiques. Les titres possibles pour cette application pourraient être : "Recherche de motifs récurrents dans les séquences d'ADN" ou "Identification de régions conservées dans les génomes". (Front. Bioeng. Biotechnol., 2020)

I.3.3 Network Traffic Analysis (Analyse du trafic réseau)

Cette application concerne l'analyse des motifs de communication entre les utilisateurs dans les réseaux de télécommunications. Les titres possibles pour cette application pourraient être : "Détection de schémas de communication suspects dans les journaux d'appels" ou

"Analyse des relations d'appel entre les utilisateurs de téléphonie mobile". (Politi L, Codish S, Sagy I and Fink L. , 2020)

I.3.4 Medical Data Analysis (Analyse des données médicales)

Cette application vise à découvrir des associations fréquentes entre les symptômes, les maladies et les traitements dans les dossiers médicaux. Les titres possibles pour cette application pourraient être : "Identification des relations de comorbidité entre les maladies" ou "Analyse des combinaisons fréquentes de symptômes dans les enregistrements médicaux". (Zhang, Wenjing; Ma, Donglai; Yao, Wei., May 2014)

I.4 Perspectives l'extraction des itemsets fréquents à base de ASP

L’utilisation de l'Answer Set Programming (ASP) pour l'extraction d'itemsets fréquents se révèle être une approche prometteuse et efficace. L'ASP présente des avantages significatifs, notamment sa richesse et son expressivité en termes de langage de modélisation. Cela permet de décrire des contraintes complexes et de représenter des conditions spécifiques pour extraire des itemsets fréquents en fonction de critères précis. De plus, l'utilisation de l'ASP évite la nécessité de concevoir des algorithmes spécifiques pour des tâches de fouille d'itemsets particuliers, ce qui facilite le processus de résolution.

L'approche basée sur l'ASP pour l'extraction d'itemsets fréquents présente également des perspectives intéressantes pour la recherche future. Il existe des opportunités d'amélioration de cette approche, notamment en termes de scalabilité, d'optimisation des performances et d'extension du modèle pour traiter des aspects spécifiques des problèmes de fouille d'itemsets. Dans le prochain chapitre, nous approfondirons davantage l'extraction d'itemsets fréquents en utilisant l'ASP. Nous examinerons des techniques spécifiques, des modèles de contraintes avancés et des cas d'application réels. De plus, nous explorerons les extensions de l'ASP pour aborder des problèmes plus complexes de la fouille d'itemsets.

En somme, l'utilisation de l'ASP pour l'extraction d'itemsets fréquents offre des perspectives intéressantes pour améliorer l'efficacité et la précision de la fouille de données. L'ASP se révèle être un outil puissant pour modéliser et résoudre des problèmes complexes liés à la fréquence des itemsets. Avec une compréhension approfondie de cette approche, nous pourrons explorer de nouvelles possibilités et contribuer à l'avancement de la fouille d'itemsets fréquents.

I.5 Conclusion

En conclusion, ce premier chapitre a présenté les notions fondamentales concernant les itemsets fréquents. Nous avons commencé par définir ce qu'est un itemset fréquent, mettant en évidence son importance dans l'analyse de données. Ensuite, nous avons exploré les différents algorithmes utilisés pour extraire ces itemsets fréquents, en mettant particulièrement l'accent sur les algorithmes CHARM et Gen-MAX.

Nous avons également examiné les diverses applications pratiques des itemsets fréquents, telles que l'Analyse des paniers d'achats, l'Analyse des séquences d'ADN, l'Analyse du trafic réseau et l'Analyse des données médicales. Ces domaines d'application démontrent l'importance et la pertinence des itemsets fréquents dans différents contextes.

Enfin, nous avons abordé les perspectives prometteuses de l'extraction des itemsets fréquents basée sur la Programmation par Ensembles de Réponses (ASP), ouvrant ainsi de nouvelles possibilités pour des méthodes d'extraction plus efficaces et flexibles.

Ce chapitre introductif a posé les bases essentielles pour la compréhension et l'exploration ultérieure des itemsets fréquents. Dans les chapitres suivants, nous approfondirons davantage ces concepts et examinerons des techniques avancées ainsi que leurs applications spécifiques.

En somme, ce chapitre a offert un aperçu complet des itemsets fréquents, de leurs algorithmes d'extraction, de leurs applications pratiques et des perspectives d'évolution. Il constitue une base solide pour la suite de notre étude et pour une compréhension approfondie de ce domaine passionnant de l'analyse de données.

Chapitre II L’answer Set Programming

II.1 Introduction

L'Answer Set Programming (ASP) est un paradigme de programmation déclarative basé sur la logique formelle pour la résolution de problèmes complexes. L'ASP a été introduit à la fin des années 1990. Il permet de modéliser et de résoudre des problèmes combinatoires difficiles. En utilisant une approche déclarative plutôt que procédurale (Vladimir, 2008).

L'ASP est fortement enraciné dans le domaine de la représentation des connaissances et du raisonnement, et là-dedans dans la programmation logique (Eiter, 2009). Il est issu de l'interaction de 2 axes de recherche : Sémantique de la programmation logique, et Application des solveurs SAT aux problèmes de recherche (McIlraith, 2006).

L'ASP est particulièrement adapté à la résolution de problèmes NP-complets (Baral, 2001), tels que la planification, la recherche de satisfaction de contraintes, la planification de tâches, la vérification de modèle, la planification de ressources, etc. Il offre un moyen puissant et expressif pour spécifier les contraintes et les relations entre les données du problème.

Une caractéristique clé de l'ASP est l'utilisation de la notion d'ensemble de réponses (answer set), qui représente un ensemble cohérent de propositions logiques qui satisfont les contraintes du problème. Les ensembles de réponses sont déterminés par la résolution d'un ensemble de règles logiques, où les règles peuvent être utilisées pour déduire de nouvelles propositions ou pour restreindre les ensembles de réponses possibles.

L'ASP offre plusieurs avantages, tels que la capacité à modéliser des problèmes complexes de manière naturelle et concise, la possibilité de gérer des problèmes avec des contraintes temporelles et de ressources, et la capacité à effectuer des raisonnements non monotones. Il est utilisé dans divers domaines tels que l'intelligence artificielle, la représentation des connaissances, la recherche opérationnelle, la bioinformatique, et bien d'autres.

Dans ce chapitre nous présentons une vue générale sur le paradigme de programmation ASP : sa syntaxe, sa sémantique, le processus de résolution d'un problème avec ASP, ainsi que quelques exemples de domaines de l'utilisation de l'ASP.

II.2 Notions de base de l'ASP

L'Answer Set Programming (ASP) est un langage de programmation déclaratif permettant de résoudre des problèmes de satisfiabilité en logique propositionnelle ou en logique de premier ordre. Dans cette partie, nous allons détailler la syntaxe et la sémantique des programmes ASP, y compris les différentes formes de règles, les variables, les fonctions, les

ensembles de réponses, etc. nous allons également expliquer comment les règles sont interprétées et comment les ensembles de réponses sont calculés.

II.2.1 Syntaxe de l'ASP

La syntaxe d'un programme ASP est définie par un ensemble de règles qui peuvent être de types les règles de base, les règles de contraintes etc…

II.2.1.1 Une règle normale (normal rule)

Objectif : définir les relations logiques entre les prédicats

Syntaxe : Une règle normale est de la forme

$$a_0 \leftarrow a_1,\ldots,a_m,\neg a_{m+1},\ldots,\neg a_n$$

où $0 \leq m \leq n$ et chaque ai est un atome pour $0 \leq i \leq n$

Exemple :

```
parent(X, Y) :- father(X, Y).
parent(X, Y) :- mother(X, Y).
```

Dans cet exemple, nous avons deux règles normales qui définissent la relation de parenté. La première règle indique que si X est le père de Y (représenté par le prédicat "father(X, Y)"), alors X est parent de Y. La deuxième règle indique que si X est la mère de Y (représenté par le prédicat "mother(X, Y)"), alors X est parent de Y.

Ces règles normales établissent une relation logique entre les prédicats "father", "mother" et "parent" et définissent les conditions pour que la relation de parenté soit vraie. Elles permettent de déduire la relation de parenté à partir des informations sur les pères et les mères.

II.2.1.2 Contrainte d'intégrité (integrity constraint)

Objectif : Éliminer les candidats de solution indésirables

Syntaxe : Une contrainte d'intégrité est de la forme

$$\leftarrow a_1,\ldots, a_m,\ldots,\neg a_{m+1},\ldots,\neg a_n$$

Où $0 \leq m \leq n$ et chaque a_i est un atome pour $1 < i < n$(Gebser M. K., Answer set solving in practice, 2012)

Example:

```
:- a, not b, c.
```

Cette règle peut être lue comme suit : "il est interdit que 'a' soit vrai et que 'non b' et 'c' soient vrais". Dans cette règle, a, b et c sont des atomes.

II.2.1.3 Variables et fonctions

L'ASP permet également l'utilisation de variables et de fonctions (ou prédicats) . Les variables sont définies par un nom qui commence par une majuscule, comme dans l'exemple suivant :

```
personne(X) :- homme(X), not femme(X).
```

Dans cette règle, X est une variable qui peut prendre n'importe quelle valeur. Les fonctions sont définies par un nom qui commence par une minuscule, comme dans l'exemple suivant :

```
parent(john, marie).
parent(john, bob).
parent(jane, marie).
parent(jane, bob).

grandparent(X, Y) :- parent(X, Z), parent(Z, Y).
```

Dans cette règle, `grandparent` est une fonction qui prend deux arguments, X et Y. Elle est définie en utilisant la fonction parent.

II.2.2 Sémantique de l'ASP

La sémantique de l'ASP, se réfère à la signification des programmes ASP, c'est-à-dire comment les règles d'un programme sont interprétées et comment les ensembles de réponses sont calculés.

En termes plus précis, la sémantique de l'ASP se base sur la notion de "programme d'atomes", qui est un ensemble de règles logiques construit à partir d'atomes, de variables, de connecteurs logiques et d'opérateurs d'agrégation. Les règles définissent des relations entre les atomes et les variables, et les ensembles de réponses sont des ensembles d'atomes qui satisfont toutes les contraintes du programme.

La sémantique de l'ASP est définie par une procédure de calcul appelée "saturateur". Le saturateur calcule *les ensembles de réponses* en utilisant une technique de propagation de contraintes qui construit des ensembles d'atomes qui satisfont toutes les règles du programme.

- Ensembles de réponses :

Les ensembles de réponses sont des ensembles de faits qui peuvent être déduits à partir des règles du programme ASP. Les ensembles de réponses sont calculés en utilisant une technique appelée programmation par ensembles de réponses. Cette technique consiste à générer tous les ensembles de faits qui satisfont les règles du programme, en utilisant une technique de saturation (saturation technique) (Eiter, 2009).

L'interprétation des règles dans un programme ASP se fait par saturation. Cela signifie que l'on considère toutes les règles du programme comme des contraintes, et on construit des ensembles de réponses qui satisfont toutes ces contraintes.

Pour construire un ensemble de réponses, on commence par un ensemble vide de faits, et on ajoute progressivement des faits qui satisfont les règles du programme. À chaque étape, on examine les règles qui ne sont pas encore satisfaites, et on ajoute des faits qui permettent de les satisfaire.

Lorsque toutes les règles sont satisfaites, on obtient un ensemble de réponses. Cet ensemble peut contenir plusieurs ensembles de faits, car il est possible d'avoir plusieurs ensembles de faits qui satisfont les règles du programme. Cette approche est illustrée à la Figure 11, qui montre les ensembles de réponse générés par le code suivant :

```
{a}. b :- a.- not b.
```

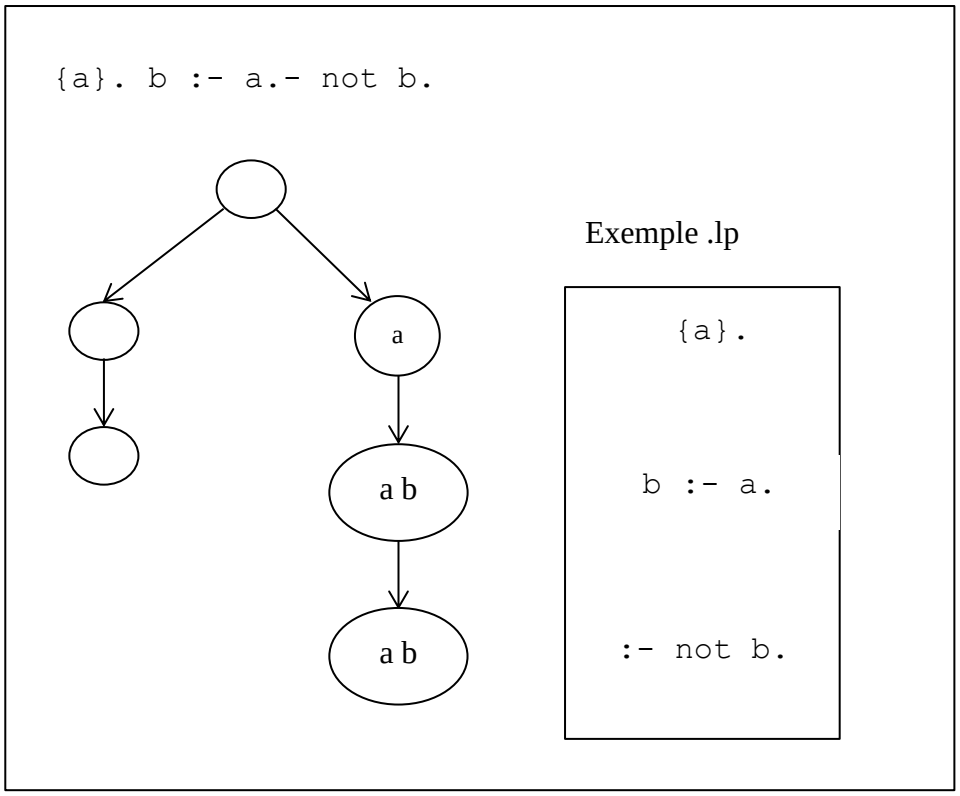

Figure 11 : les ensembles de réponse générée par le code.

- **L'explication du code précédent**

`{a}` : on peut ajouter "a" ou pas l'ajouter donc on aura deux ensembles un qui contient "a" et l'autre qui est vide.

`b :-a` : puisque nous avons "a" dans l'ensemble nous ajoutons "b", mais si on n'a pas le "a" on ne peut pas ajouter le "b" donc on obtient deux ensembles un qui contient « a b » et un ensemble vide.

`:-not b` : on élimine l'ensemble qui ne contient pas le "b".

L'ensemble de réponse de ce programme c'est « a b ».

II.3 Résolution d'un problème avec l'ASP

La résolution d'un problème avec l'ASP (Answer Set Programming) implique généralement les étapes suivantes (Figure 12) :

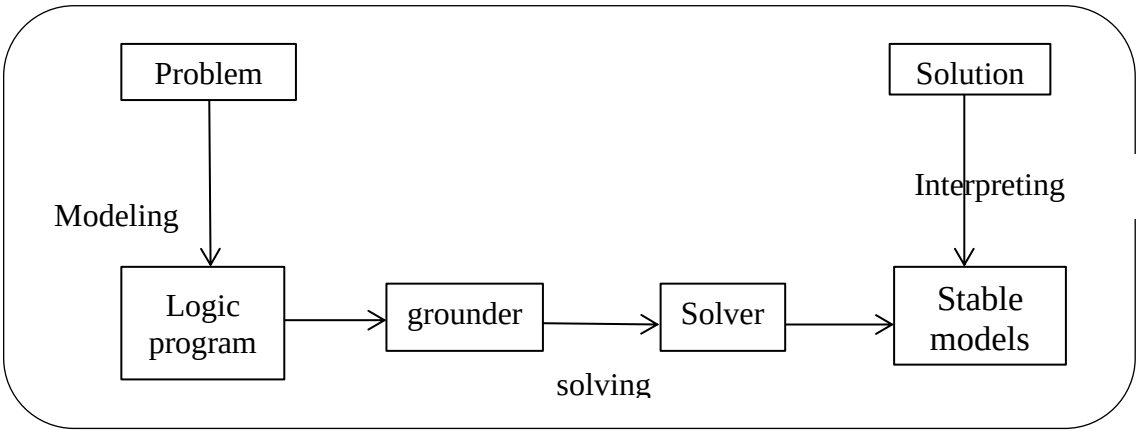

Figure 12 : Les étapes de la résolution d'un problème avec l'asp. (Gebser M. K., Answer set solving in practice, 2012)

II.3.1 Définition du problème

Identifiez clairement le problème que vous souhaitez résoudre à l'aide de l'ASP. Déterminez les objectifs, les contraintes et les variables pertinentes pour la résolution du problème.

II.3.2 Modélisation du problème

Exprimez le problème sous forme de programme ASP en utilisant des prédicats, des règles logiques et des contraintes. La modélisation doit représenter fidèlement les aspects du problème à résoudre.

II.3.3 Le grounding

Le Grounding fait référence à la première étape du processus de résolution d'un programme ASP. Elle consiste à traduire un programme ASP donné dans un langage de haut niveau en une représentation de bas niveau qui peut être traitée plus efficacement par le moteur de résolution. Cette étape comprend la transformation des règles logiques, des contraintes et des faits en une forme plus simple, appelée "grounded program". Ce dernier, représente une version détaillée et explicite des règles logiques et des faits.

Une fois que le programme a été « grounded », le moteur de résolution peut procéder à la résolution effective du programme en cherchant les ensembles de réponses qui satisfont les règles logiques et les contraintes spécifiées. La résolution implique l'application de règles de déduction, la propagation des contraintes et la recherche d'ensembles de réponses cohérents. (Kaufmann, 2016, pp. 26-27)

II.3.4 La résolution (Solver)

La résolution dans la programmation ASP consiste à trouver les ensembles de réponses qui satisfont les règles logiques et les contraintes spécifiées dans le programme. Ces ensembles de réponses représentent les solutions possibles pour un problème donné.

Le moteur de résolution de l'ASP utilise différentes techniques pour effectuer cette recherche. Il applique des règles de déduction pour inférer de nouvelles informations à partir des règles logiques et des faits du programme. Il propage également les contraintes à travers le programme pour garantir que les ensembles de réponses ne violent pas les contraintes spécifiées.

La résolution dans l'ASP implique généralement une recherche non déterministe dans l'espace des ensembles de réponses possibles. Le moteur de résolution explore différents ensembles de réponses en utilisant des stratégies de recherche telles que la recherche en profondeur (depth-first search) ou la recherche en largeur (breadth-first search). Ces stratégies déterminent l'ordre dans lequel les ensembles de réponses sont générés et examinés.

Lors de la résolution, le moteur de résolution peut également appliquer des techniques d'optimisation pour améliorer l'efficacité de la recherche. Par exemple, il peut utiliser des techniques de propagation de contraintes pour réduire l'espace de recherche en éliminant les ensembles de réponses qui ne peuvent pas satisfaire certaines contraintes.

L'objectif final de la résolution dans l'ASP est de trouver tous les ensembles de réponses cohérents, également appelés modèles stables qui représentent les solutions possibles pour le problème donné. Ces ensembles de réponses peuvent ensuite être utilisés pour l'analyse, la prise de décision ou d'autres tâches spécifiques liées au problème. (Kaufmann, 2016, p. 29)

En analysant les ensembles de réponses générés par le solveur, qui représentent les modèles stables du programme ASP. Un modèle stable est un ensemble cohérent de faits et de règles qui respecte toutes les contraintes du problème. (Vladimir, 2008)

II.3.5 Solution

Une solution d'un problème avec ASP, consiste à interpréter les modèles stables pour obtenir une solution au problème initial. C-à-dire, extraire les faits et les relations pertinents des modèles stables qui satisfont les critères de résolution et les objectifs du problème.

Il est important de noter que la qualité de la solution dépend de la précision de la modélisation initiale du problème. Une modélisation adéquate et complète garantit que les modèles stables représentent fidèlement les solutions possibles.

Pour chaque étape mentionnée ci-dessus, il existe des outils et des bibliothèques spécifiques à l'ASP qui peuvent être utilisés, tels que Clingo. Ces outils fournissent une interface pour définir, résoudre et interpréter les programmes ASP.

II.4 Exemple d'utilisation de l'ASP : Problème de coloration de graphe

Le problème de coloration de graphe consiste à assigner à chaque sommet une couleur de sorte que deux sommets adjacents n'aient pas la même couleur, tout en utilisant un nombre minimal de couleurs.

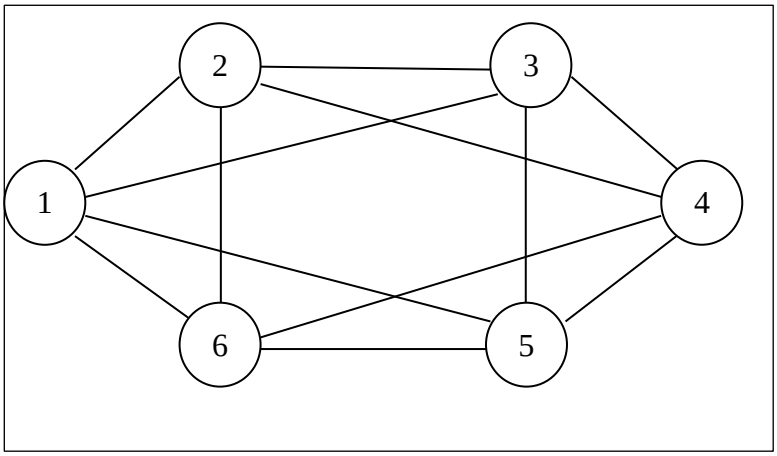

Figure 13 : Un graphe avec six nœuds.

II.4.1 Définition du problème

- Instance de problème :

Un graphe composé de nœuds et d'arêtes :

Faits formés par les prédicats node/1 et edge/2.

Faits formés par le prédicat color/1.

- Classe de problème :

Attribuez une couleur à chaque nœud de sorte qu'il n'y ait pas deux nœuds reliés par une arête ont la même couleur.

Autrement dit :

- ✓ Chaque nœud a une couleur
- ✓ Deux nœuds connectés ne doivent pas avoir la même couleur

II.4.2 Modélisation du problème

Instance de problème :

```
node(1..6).

edge(1,(2;3;5;6)).

edge(2,(1;3;4;6)).

edge(3,(1;2;4;5)).

edge(4,(2;3;5;6)).

edge(5,(1;3;4;6)).

edge(6,(1;2;4;5)).

color(r). color(b). color(g).
```

Graph.lp

- Classe de problème :

```
{ assign(N,C) : color(C) } = 1 :- node(N).

:- edge(N,M), assign(N,C), assign(M,C).
```

color.lp

II.4.3 Grounding

! gringo --text graph1.lp color1.lp

```
color(r).
color(b).
color(g).
edge(1,2). edge(1,3).edge(1,5).edge(1,6).
edge(2,1).edge(2,3).edge(2,4).edge(2,6).
edge(3,1).edge(3,2).edge(3,4).edge(3,5).
edge(4,2).edge(4,3).edge(4,5).edge(4,6).
edge(5,1).edge(5,3).edge(5,4).edge(5,6).
edge(6,1).edge(6,2).edge(6,4).edge(6,5).
node(1).node(2).node(3).node(4).node(5).node(6).
```

```
:-assign(1,r),assign(2,r).    :-assign(2,r),assign(1,r).   :-assign(3,r),assign(1,r).
:-assign(1,r),assign(3,r).    :-assign(2,r),assign(3,r).   :-assign(3,r),assign(2,r).
:-assign(1,r),assign(5,r).    :-assign(2,r),assign(4,r).   :-assign(3,r),assign(4,r).
:-assign(1,r),assign(6,r).    :-assign(2,r),assign(6,r).   :-assign(3,r),assign(5,r).
:-assign(1,b),assign(2,b).    :-assign(2,b),assign(1,b).   :-assign(3,b),assign(1,b).
:-assign(1,b),assign(3,b).    :-assign(2,b),assign(3,b).   :-assign(3,b),assign(2,b).
:-assign(1,b),assign(5,b).    :-assign(2,b),assign(4,b).   :-assign(3,b),assign(4,b).
:-assign(1,b),assign(6,b).    :-assign(2,b),assign(6,b).   :-assign(3,b),assign(5,b).
:-assign(1,g),assign(2,g).    :-assign(2,g),assign(1,g).   :-assign(3,g),assign(1,g).
:-assign(1,g),assign(3,g).    :-assign(2,g),assign(3,g).   :-assign(3,g),assign(2,g).
:-assign(1,g),assign(5,g).    :-assign(2,g),assign(4,g).   :-assign(3,g),assign(4,g).
:-assign(1,g),assign(6,g).    :-assign(2,g),assign(6,g).   :-assign(3,g),assign(5,g).

:-assign(4,r),assign(2,r).    :-assign(5,r),assign(1,r).   :-assign(6,r),assign(1,r).
:-assign(4,r),assign(3,r).    :-assign(5,r),assign(3,r).   :-assign(6,r),assign(2,r).
:-assign(4,r),assign(5,r).    :-assign(5,r),assign(4,r).   :-assign(6,r),assign(4,r).
```

```
:-assign(4,r),assign(6,r).    :-assign(5,r),assign(6,r).   :-assign(6,r),assign(5,r).
:-assign(4,b),assign(2,b).    :-assign(5,b),assign(1,b).   :-assign(6,b),assign(1,b).
:-assign(4,b),assign(3,b).    :-assign(5,b),assign(3,b).   :-assign(6,b),assign(2,b).
:-assign(4,b),assign(5,b).    :-assign(5,b),assign(4,b).   :-assign(6,b),assign(4,b).
:-assign(4,b),assign(6,b).    :-assign(5,b),assign(6,b).   :-assign(6,b),assign(5,b).
:-assign(4,g),assign(2,g).    :-assign(5,g),assign(1,g).   :-assign(6,g),assign(1,g).
:-assign(4,g),assign(3,g).    :-assign(5,g),assign(3,g).   :-assign(6,g),assign(2,g).
:-assign(4,g),assign(5,g).    :-assign(5,g),assign(4,g).   :-assign(6,g),assign(4,g).
:-assign(4,g),assign(6,g).    :-assign(5,g),assign(6,g).   :-assign(6,g),assign(5,g).
```

II.4.4 La résolution (Solver)

```
! gringo graph1.lp color1.lp | clasp 0
clasp version 3.3.9
Reading from stdin
Solving...
Answer: 1
assign(2,r) assign(5,r) assign(1,b) assign(3,g) assign(6,g) assign(4,b)
Answer: 2
assign(3,r) assign(6,r) assign(1,b) assign(2,g) assign(5,g) assign(4,b)
Answer: 3
assign(2,r) assign(5,r) assign(3,b) assign(6,b) assign(1,g) assign(4,g)
Answer: 4
assign(3,r) assign(6,r) assign(2,b) assign(5,b) assign(1,g) assign(4,g)
Answer: 5
assign(1,r) assign(2,b) assign(5,b) assign(3,g) assign(6,g) assign(4,r)
Answer: 6
assign(1,r) assign(3,b) assign(6,b) assign(2,g) assign(5,g) assign(4,r)
SATISFIABLE

Models       : 6
Calls        : 1
Time         : 0.003s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.000sle modèle stable (Stable models) :
  Answer: 5
  assign(1,r) assign(2,b) assign(5,b) assign(3,g) assign(6,g) assign(4,r)
```

II.4.5 Solution

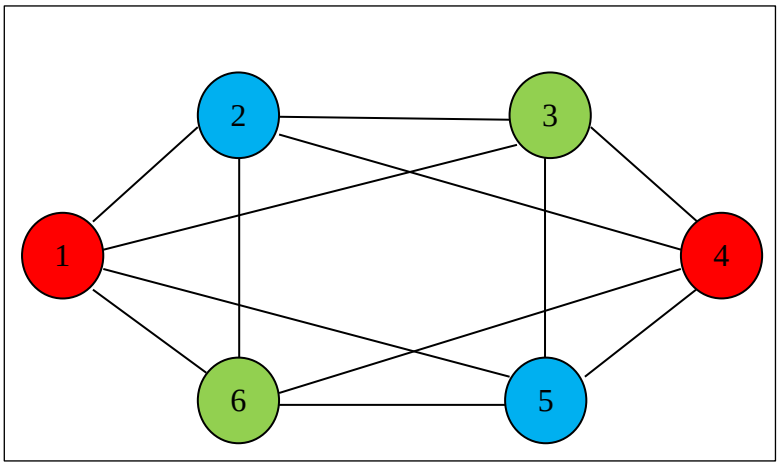

Figure 14 : La coloration de graphe.

II.5 Techniques avancées en ASP

En Answer Set Programming (ASP), il existe plusieurs techniques avancées qui permettent de modéliser et résoudre des problèmes plus complexes. Voici quelques-unes de ces techniques couramment utilisées en ASP :

II.5.1 Les Agrégats

Objectif des agrégats est la manipulation des ensembles, Sa syntaxe est de la forme :

$S_1 < \alpha\{t_1 : L_1 ; \ldots ; t_n : L_n\} < S_2$ où

α est un nom agrégé

$t_1 : L_1 ; \ldots ; t_n : L_n$ sont des littéraux conditionnels

s1 et s2 sont des termes

Les agrégats permettent de manipuler des ensembles de manière compacte. Ils fournissent des opérations telles que la somme, le minimum, le maximum, etc., sur des ensembles de valeurs. Plus formellement, un agrégat est une fonction sur un ensemble de tuples qui sont normalement soumis à des conditions. En comparant une valeur agrégée avec des valeurs données, nous pouvons extraire une valeur de vérité de l'évaluation d'un agrégat, obtenant ainsi un atome d'agrégat. Les atomes agrégés se présentent sous deux variantes selon qu'ils apparaissent dans une tête ou un corps de règle. (Gebser M. K., 2015, pp. 30-37)

Exemple :

```
10 <= #sum {6,C:course(C); 3,S:seminar(S)g} <= 20
```

Dans cet exemple l'agrégat est utilisé pour représenter une condition qui doit être satisfaite en termes de somme avec des conditions supplémentaires sur les variables. Voici une explication détaillée de l'agrégat donné :

- Le symbole "#sum" indique qu'il s'agit d'une somme avec des conditions supplémentaires.
- "{6,C:course(C); 3,S:seminar(S)}" est un ensemble d'éléments soumis à la somme avec des conditions. Dans cet exemple, l'ensemble contient deux éléments.
- "6,C:course(C)" est le premier élément de l'ensemble. Il indique que nous voulons sélectionner 6 valeurs de la variable C qui satisfont la condition "course(C)".

- "3,S:seminar(S)" est le deuxième élément de l'ensemble. Il indique que nous voulons sélectionner 3 valeurs de la variable S qui satisfont la condition "seminar(S)".
- "10" indique que la somme totale des valeurs sélectionnées dans l'ensemble doit être supérieure ou égale à 10.
- "20" indique que la somme totale des valeurs sélectionnées dans l'ensemble doit être inférieure ou égale à 20.

II.5.2 Opérateurs de négation

La négation est utilisée pour exprimer l'absence ou l'interdiction d'une condition ou d'un fait.

La syntaxe de la négation en ASP peut varier légèrement en fonction de l'outil ou de la variante d'ASP utilisée. Voici quelques exemples courants de syntaxe pour exprimer la négation :

```
head :- not condition.
not condition.
not(condition).
```

Voici un exemple simple pour illustrer la négation en ASP :

Définissons quels cours sont agréables pour nous parce qu'ils n'ont pas de cours magistraux au début d'une journée.

```
% lecture(C,D,P): the course C has a lecture the day D at period P

lecture(asp,thu,3). lecture(asp,fri,3).

lecture(ml,wed,2). lecture(ml,thu,3). lecture(ml,fri,4).

lecture(ling,tue,1). lecture(ling,thu,2).

lecture(phil,tue,3). lecture(phil,wed,4).

% ...

% course C is early (some day)

early(C) :- lecture(C,D,1).

% course C is nice

nice(C) :- course(C), not early(C).
```

```
#show nice/1[1].
```

Les opérateurs de négation permettent de représenter la négation par rapport à des atomes ou à des ensembles de propositions. Ils permettent de modéliser des contraintes négatives, où certaines conditions ne doivent pas être satisfaites simultanément.

II.5.3 Les règles de choix (choice rule)

L'objectif est de fournir des choix sur des sous-ensembles d'atomes

La syntaxe d'une règle de choix est de la forme

$$\{ a1,\ldots,am\} \leftarrow a_{m+1} ,\ldots ,a_n,\ldots,\neg a_{n+1},\ldots,\neg a_o$$

Où $0 \leq m \leq n \leq o$ et chaque ai est un atome pour $1 \leq i \leq o$

Exemple :

```
{p(X)}.
{q(X)}.
```

Dans cet exemple, nous avons deux règles de choix, l'une pour le prédicat p(X) et l'autre pour le prédicat q(X). Ces règles indiquent que chaque prédicat peut être vrai ou faux de manière indépendante, sans imposer de contraintes supplémentaires. Cela signifie que dans l'ensemble de réponses, p(X) et q(X) peuvent être vrais ou faux de manière arbitraire.

```
{ buy( pizza ); buy( tomato ); buy( corn ) } :- at( grocery ).
```

Cette règle indique qu'il est possible d'acheter soit une pizza, soit de la tomate, soit du maïs lorsqu'on se trouve à l'épicerie.

L'expression `{ buy(pizza); buy(tomato); buy(corn) }` représente l'ensemble des choix possibles. Cela signifie que dans l'ensemble de réponses, on peut avoir une combinaison de ces choix. Par exemple, on peut avoir `buy(pizza)` et `buy(tomato)` dans un ensemble de réponses, ou seulement `buy(corn)`, ou encore toutes les options à la fois.

[1] ***Negation***, [en ligne], [https://notebooks.gesis.org/binder/jupyter/user/potassco-asp-course-notebooks-wlh59r9h/notebooks/tutorial/language/language.ipynb], (24 avril 2023).

La condition `:- at(grocery)` spécifie que cette règle de choix est applicable uniquement lorsque la condition `at(grocery)` est satisfaite, c'est-à-dire lorsque nous sommes à l'épicerie.

Ces techniques avancées en ASP offrent plusieurs avantages pour la modélisation et la résolution de problèmes complexes :

- Expressivité : Ces techniques permettent de représenter des contraintes complexes de manière concise et intuitive, ce qui facilite la modélisation des problèmes.
- Résolution efficace : Les moteurs d'inférence en ASP sont optimisés pour résoudre des problèmes combinatoires de grande taille. Les techniques avancées en ASP exploitent ces optimisations pour obtenir des solutions efficaces, même pour des problèmes complexes.
- Gestion des préférences : Les règles de choix permettent de spécifier des préférences et des priorités entre les solutions, ce qui est essentiel dans de nombreux domaines où plusieurs solutions acceptables sont possibles.

II.6 Applications de l'ASP

ASP (Answer Set Programming) a été appliqué dans diverses industries et domaines tels que la configuration, le diagnostic et la réparation, la planification et la classification.

Dans cette partie, nous ne décrirons que certains de ces Applications ASP, en particulier, dans la représentation et le raisonnement des connaissances, la robotique, la bioinformatique, ainsi que l'e-tourism.

II.6.1 ASP et représentation des connaissances

L'ASP offre un cadre expressif pour représenter les connaissances en utilisant des règles logiques et des contraintes, et il possède des caractéristiques clés qui le rendent puissant pour la représentation des connaissances.

Plus précisément, l'ASP est capable de gérer l'incertitude, les connaissances incomplètes et d'effectuer un raisonnement non monotone. Il permet de représenter des connaissances en utilisant des règles logiques sous forme de clauses de Horn étendues, et de spécifier des contraintes sur ces règles. Cette combinaison de règles et de contraintes permet de modéliser des problèmes complexes et de les résoudre en utilisant des techniques d'énumération des ensembles de réponses (Erdem E. G., 2016).

II.6.2 Applications de l'ASP à la robotique

L'ASP offre un cadre efficace pour modéliser et résoudre des problèmes complexes rencontrés dans le domaine de la robotique. En utilisant des règles logiques et des contraintes, l'ASP permet de représenter les connaissances et les contraintes spécifiques à un système robotique donné.

Dans le contexte de la robotique, l'ASP peut être utilisé pour la planification de tâches, la prise de décision et la coordination des actions d'un robot. Par exemple, en utilisant l'ASP, il est possible de modéliser des problèmes de planification robotique en spécifiant les actions disponibles, les préconditions et les effets de chaque action, ainsi que les objectifs à atteindre. L'ASP permet également de prendre en compte des contraintes telles que les ressources limitées ou les préférences spécifiques, Par exemple, (Erdem E. P., 2015) utilisent ASP pour trouver un plan global optimal pour plusieurs équipes de robots hétérogènes dans une usine cognitive pour fabriquer un nombre donné de commandes dans un temps donné (figure 15).

Figure 15 : Plusieurs robots travaillant en collaboration dans une usine cognitive (Erdem E. P., 2015)

Un autre domaine d'application de l'ASP dans la robotique concerne la résolution de problèmes de perception et d'interprétation de l'environnement. En utilisant l'ASP, il est possible de modéliser des connaissances incertaines ou incomplètes sur l'environnement du robot, ainsi que les capteurs et les actions du robot. L'ASP permet de résoudre efficacement des problèmes de perception, de fusionner des informations provenant de différentes sources et de prendre des décisions basées sur ces informations. Par exemple, (Erdem E. A., 2012) utilisent

ASP pour planifier les actions de plusieurs robots afin de ranger une maison en collaboration dans un délai donné (figure 16).

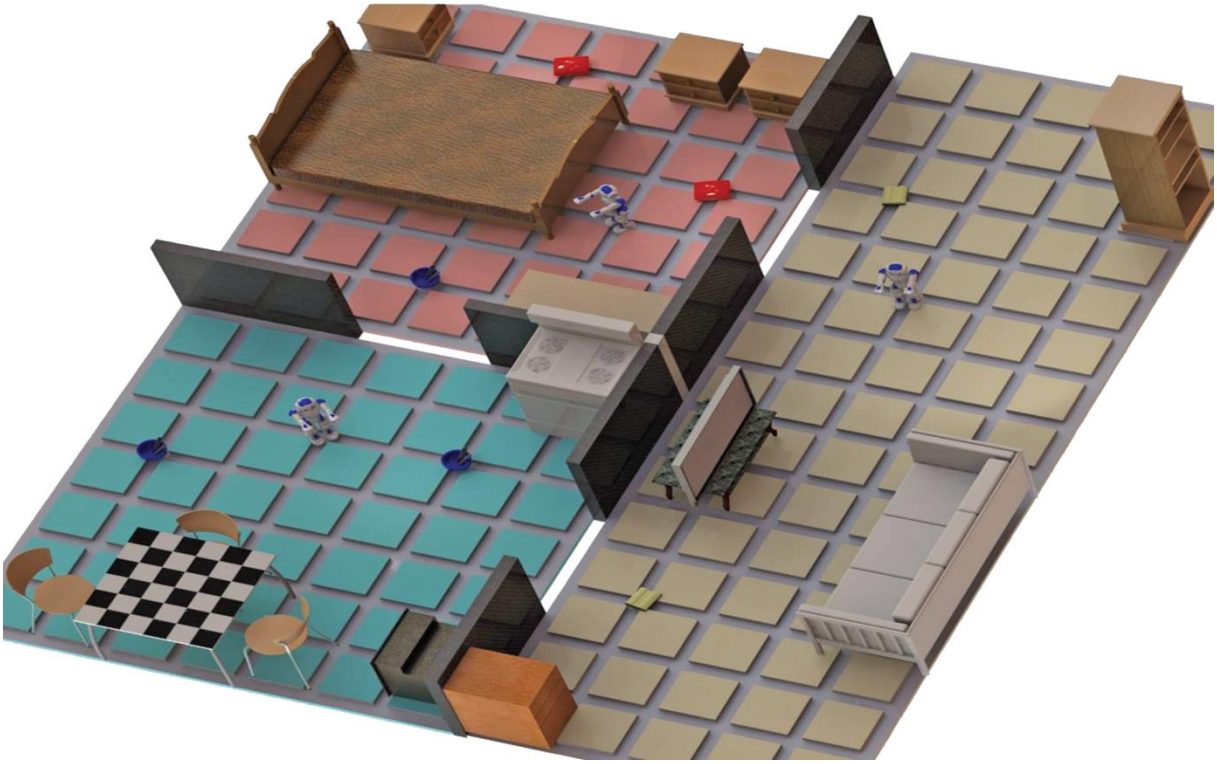

Figure 16 : Plusieurs robots rangent une maison (Erdem E. P., 2015)

De plus, l'ASP peut être utilisé pour la vérification et la validation de systèmes robotiques. En spécifiant des règles et des contraintes dans l'ASP, il est possible de vérifier si un système robotique satisfait certaines propriétés spécifiées, telles que l'absence de collisions, la sécurité ou la conformité à des réglementations spécifiques (Erdem E. G., 2016).

II.6.3 Applications de l'ASP à la biologie computationnelle et à la bioinformatique

L'ASP offre un cadre puissant pour modéliser et résoudre des problèmes complexes rencontrés dans le domaine de la biologie computationnelle et de la bioinformatique. L'ASP permet de représenter des connaissances biologiques et des contraintes spécifiques à l'aide de règles logiques et de contraintes, permettant ainsi de résoudre des problèmes de recherche et d'analyse de données biologiques.

Dans le domaine de la biologie computationnelle, l'ASP peut être utilisé pour la modélisation et la simulation de réseaux biologiques (Tran, 2009). En utilisant l'ASP, il est possible de représenter les interactions entre les composants biologiques, tels que les gènes, les protéines et les voies métaboliques, ainsi que les contraintes et les règles de régulation qui les

régissent (Dovier, 2009). L'ASP permet également de résoudre des problèmes de recherche sur les réseaux biologiques, tels que la recherche de motifs ou de sous-structures spécifiques, l'identification de régulateurs clés ou la prédiction de comportements dynamiques.

En bioinformatique, l'ASP peut être utilisé pour l'analyse et l'annotation des séquences génétiques. En utilisant l'ASP, il est possible de modéliser des contraintes et des règles logiques pour annoter les séquences génétiques avec des informations biologiques pertinentes, telles que l'identification de gènes, de motifs ou de structures protéiques. L'ASP permet également de résoudre des problèmes de recherche et de classification de séquences, tels que la recherche de similarités, la prédiction de la fonction des gènes ou l'identification de marqueurs biologiques (Gebser M. S., 2011).

II.6.4 e-tourisme

Dans cette partie nous nous décrivons une application basée sur la programmation par ensemble de réponses (ASP) dans le domaine du tourisme en ligne (e-tourisme). L'application consiste en un conseiller intelligent intégré dans un portail d'e-tourisme qui sélectionne les offres les plus prometteuses pour les clients d'une agence de voyage.

L'objectif de cette application est d'aider les employés de l'agence de voyage à trouver la meilleure solution de voyage possible en peu de temps. Elle agit comme un système médiateur qui trouve la meilleure correspondance entre les offres des tours opérateurs et les demandes des touristes. Cette application améliore l'activité de l'agence de voyage en réduisant le temps nécessaire pour identifier et vendre les offres touristiques, tout en augmentant le niveau de satisfaction des clients en leur suggérant les offres qui correspondent le mieux à leur profil.

Pour ce faire, une base de connaissances a été spécifiée en coopération avec le personnel de l'agence de voyage, modélisant les entités clés qui décrivent le processus d'organisation et de vente d'un voyage complet. L'ASP est utilisée comme moteur intelligent pour extraire des informations à partir des descriptions textuelles des offres touristiques, les classer dans une ontologie et développer plusieurs modules de recherche qui simplifient la tâche de sélection des forfaits vacances qui correspondent le mieux aux besoins des clients (Ricca F. D., 2010). À titre d'exemple, nous rapportons une version simplifiée d'un programme logique qui crée une sélection de forfaits vacances dans la figure 17.

```
%detect possible and suggested places
possiblePlace(Place) :- askFor(TripKind,_),
placeOffer(Place, TripKind).
suggestPlace(Place) :- possiblePlace(Place),
askFor(_,Period),
suggestedPeriod(Place, Period),
not badPeriod(Place, Period).
%select packages to suggest to the user
suggestOffer(O) :- touristicOffer(O, Place),
suggestPlace(Place).
```

Figure 17 : Un programme qui crée une sélection de forfaits vacances (Ricca F. D., 2010)

Le prédicat d'entrée *askFor*(*TripKind,Period*) définit le type de voyage demandé par le client et la période souhaitée. Le prédicat *touristicOffer*(*Offer, Place*) précise, pour chaque offre touristique disponible en agence de voyage, à quel lieu elle se rapporte. Les prédicats dérivés *placeOffer*(*Place, TripKind*) et *badPeriod*(*Place, Period*) définissent les lieux appropriés pour un type de voyage donné et les périodes à éviter pour un lieu donné en raison de mauvaises conditions météorologiques. Les règles logiques sélectionnent les lieux possibles correspondant aux demandes du client et suggèrent les lieux appropriés offrant le type de voyage demandé pendant la période spécifiée. Enfin, les règles sélectionnent parmi les forfaits vacances disponibles ceux qui correspondent aux critères initiaux (Erdem E. G., 2016).

II.6.5 Configuration et reconfiguration des produits et services

(Falkner, 2018) Décrit plusieurs applications réussies de la programmation par ASP dans le domaine de la configuration de systèmes.

L'une des premières discussions sur les avantages de l'ASP pour résoudre les problèmes de configuration a été présentée sur la base de ces résultats, Variantum Oy (Tiihonen, 2003) a proposé des solutions de configuration. Par la suite, l'ASP a été appliquée avec succès à la configuration des packages Linux (Gebser M. K., 2011). Ce travail a démontré la puissance de l'ASP dans la compétition du solveur Mancoosi en 2012, où le configurateur basé sur l'ASP a remporté tous les défis.

Dans le projet RECONCILE (Aschinger, 2011), Siemens a évalué l'ASP après des tests prometteurs de divers solveurs ASP pour le diagnostic et la réparation basés sur les modèles d'instances de flux de travail défaillantes. L'ASP a été appliquée avec succès à la configuration

de certaines parties d'un système de sécurité ferroviaire de Siemens, où les configurateurs spécialisés n'ont pas réussi à générer des solutions pour certaines instances réelles difficiles. Ce problème de configuration est devenu célèbre sous le nom de "Problème des unités partenaires", car il s'agit à la fois d'un problème de configuration réellement pertinent et particulièrement difficile à résoudre pour les solveurs de problèmes généraux.

L'ASP est également utilisée pour soutenir la constitution d'équipes dans le port de Gioia Tauro (Ricca F. G., 2012), où elle est appliquée à l'assignation des employés à des tâches afin que ces tâches puissent être accomplies en respectant les réglementations du travail.

Dans le domaine de la reconfiguration, l'ASP a été appliquée avec succès au problème de réaffectation des bandes de fréquences des diffuseurs de télévision (Fréchette, 2016)**.** En utilisant une combinaison d'un solveur SAT et de clasp, l'ASP a permis d'améliorer considérablement les temps d'exécution et le nombre d'instances résolues dans le cadre d'un solveur pour le problème de reconditionnement des stations.

Ces exemples montrent l'utilisation réelle de l'Answer Set Programming dans des applications concrètes et démontrent son utilité dans des domaines variés tels que la planification, la configuration, l'optimisation, la résolution de problèmes, la biologie computationnelle et la bioinformatique.

II.7 Outils et langages de programmation ASP

Il existe plusieurs outils et langages de programmation associés à ASP qui peuvent être utilisés pour écrire, exécuter et déboguer des programmes ASP. Voici quelques-uns des principaux outils et langages de programmation ASP.

II.7.1 Smodels

Est l'un des systèmes de programmation d'ensembles de réponse les plus connus et les plus largement utilisés. Le système Smodels implémente la sémantique du modèle stable pour les programmes logiques normaux étendus par des fonctions intégrées ainsi que des contraintes de cardinalité et de poids pour les programmes à domaine restreint (Niemela, 2000).

II.7.2 DLV (Disjunctive Logic Programming with Negation as Failure)

C'est un système de résolution de programmation logique par ensembles de réponses (ASP). DLV est basé sur la sémantique des modèles stables, similaire à celle utilisée dans d'autres systèmes de résolution ASP tels que Smodels. DLV prend en charge des fonctionnalités avancées telles que la programmation arithmétique, la logique temporelle et les contraintes de

cardinalité. DLV est apprécié pour sa puissance de résolution et sa capacité à gérer des problèmes complexes (Leone N. P., 2006).

II.7.3 Clingo

C’est un système de programmation ASP open-source développé par Potassco (Potassium Answer Set Solver Suite). Il combine le solveur ASP de haut niveau "Clingo" avec le grounder "Gringo". Clingo prend en charge la spécification du problème ASP à l'aide d'un langage déclaratif appelé "ASP-Core-2". Il permet d'écrire des programmes ASP et de les exécuter en trouvant des ensembles de réponses (ensembles d'atomes qui satisfont les règles du programme). Clingo offre également des fonctionnalités de débogage, de recherche de modèles, de gestion des options de résolution et de personnalisation des stratégies de recherche (Gebser M. K., 2015).

II.7.4 ASPIDE (Answer Set Programming Integrated Development Environment)

C’est un environnement de développement intégré pour ASP. Il fournit une interface graphique conviviale pour écrire, exécuter et déboguer des programmes ASP, L'interface utilisateur d'ASPIDE est illustrée à la figure 18. ASPIDE prend en charge plusieurs solveurs ASP, y compris Clingo, et offre des fonctionnalités telles que la coloration syntaxique, l'auto-complétion, la vérification de la syntaxe, le débogage pas à pas, la visualisation des ensembles de réponses, etc. Il permet également l'intégration de scripts Python pour une personnalisation avancée (Leone N. &., 2015).

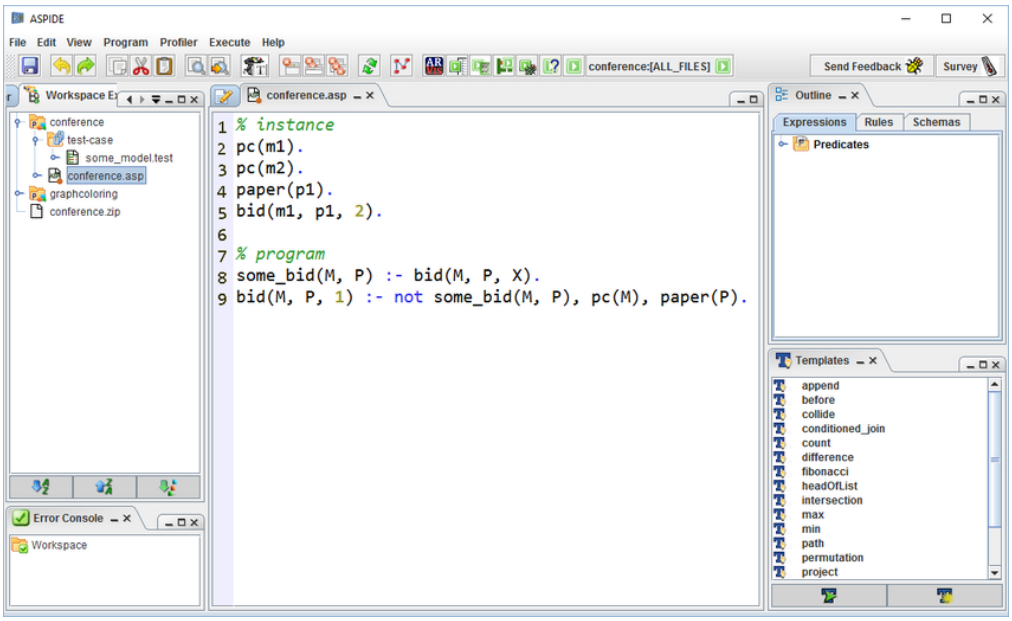

Figure 18 : L'interface utilisateur d'ASPIDE (Leone N. &., 2015)

II.8 Limitations et défis de l'ASP

Malgré les avantages de l'ASP, il présente également des limitations et des défis qui doivent être pris en compte lors de son utilisation.

II.8.1 Complexité de la résolution de problèmes

L'ASP permet de résoudre des problèmes NP-difficiles, ce qui signifie que la résolution peut devenir rapidement complexe lorsque la taille du problème augmente. La recherche d'ensembles de réponses cohérents peut nécessiter des ressources computationnelles importantes, limitant ainsi la taille des problèmes pouvant être résolus efficacement.

Par exemple Ground-and-solve paradigm, cette approche est couramment utilisée dans les systèmes ASP et divise le processus de résolution en deux étapes. Tout d'abord, le "grounder" transforme le programme d'entrée contenant des variables en un encodage propositionnel. Ensuite, les solutions pour le programme résultant sans variables sont générées par un solveur. Cependant, le grounder peut augmenter considérablement la taille du programme d'entrée, ce qui peut poser des problèmes de complexité pour les instances de problèmes volumineux (Falkner, 2018).

II.8.2 Coût de calcul

Certaines techniques avancées, en particulier celles basées sur des agrégats complexes, peuvent entraîner un coût de calcul élevé. La résolution de problèmes contenant des contraintes complexes peut nécessiter des ressources computationnelles importantes. Par exemple le problème de "grounding bottleneck", lorsque les instances de problèmes deviennent trop grandes, le processus de "grounding" peut entraîner des problèmes de taille de mémoire. Par exemple, dans le domaine de la planification, les instances de problèmes de planification industrielle peuvent atteindre des tailles considérables, ce qui les rend inaccessibles pour les approches traditionnelles de "ground-and-solve" (Calimeri, 2016).

II.9 Solution des défis de l’ASP

II.9.1 Grounding paresseux (Lazy grounding)

Cette approche permet d'éviter le problème de "grounding bottleneck" en utilisant l'interleaving (entrelacement) du processus de grounding et de recherche. Les systèmes de grounding paresseux (Weinzierl, 2017) peuvent travailler avec des programmes ASP standards et limiter leur consommation de mémoire. Cependant, ces systèmes ont souvent des temps d'exécution plus longs par rapport aux solveurs ASP traditionnels.

II.9.2 Résolution multi-shot

Cette approche permet de résoudre des programmes logiques en constante évolution, où le domaine de discours peut s'étendre progressivement jusqu'à ce qu'une solution soit trouvée. Cela est particulièrement utile dans les problèmes de configuration ou de planification où le nombre maximal d'objets ou d'étapes n'est pas prédéfini. Contrairement à l'approche "ground-and-solve", la résolution multi-shot (Gebser M. K., Multi-shot ASP solving with clingo, 2019) permet de trouver la taille du domaine au fur et à mesure de la recherche de solution, ce qui conduit à des programmes "grounded" aussi petits que nécessaire.

II.10 Conclusion

En conclusion de ce chapitre consacré à l'ASP (Answer Set Programming), nous avons examiné plusieurs éléments importants liés à cette approche de programmation logique non monotone.

Tout d'abord, nous avons étudié la syntaxe et la sémantique de l'ASP, en mettant l'accent sur les règles logiques, les clauses et les contraintes. Nous avons constaté que l'ASP permet de

représenter des connaissances sous forme de règles logiques et de définir des ensembles de réponses possibles, appelés ensembles de réponses.

Ensuite, nous avons exploré la résolution de problèmes avec l'ASP. Nous avons constaté que l'ASP offre une approche déclarative pour spécifier les problèmes, ce qui facilite leur modélisation et leur résolution. Nous avons également discuté les étapes de résolution le problème de coloration de graphe.

Nous avons également abordé les techniques avancées en ASP, telles que l'utilisation des agrégats, les opérateurs de négation, les règles de choix. Ces techniques permettent d'élargir les capacités de l'ASP et de résoudre des problèmes plus complexes.

Ensuite, nous avons examiné les différentes applications de l'ASP dans des domaines tels que la bioinformatique, la planification, la configuration de produits, e-tourisme, etc. L'ASP s'avère être une approche polyvalente et efficace pour résoudre divers problèmes du monde réel.

Nous avons également discuté des outils et des langages de programmation disponibles pour l'ASP, tels que Clingo, DLV, et Smodels. Ces outils offrent des fonctionnalités avancées pour la modélisation et la résolution de problèmes en ASP.

Enfin, nous avons pris en compte les limitations et les défis de l'ASP. Nous avons noté que la complexité de l'inférence en ASP peut devenir un défi pour les problèmes de grande taille. De plus, la modélisation des problèmes en ASP nécessite une certaine expertise et peut être sujette à des erreurs si elle n'est pas correctement réalisée.

Malgré ces limitations et défis, l'ASP se révèle être une approche prometteuse pour résoudre des problèmes complexes. Son caractère déclaratif, sa capacité à modéliser des problèmes dans différents domaines et sa disponibilité en tant qu'outils et langages de programmation en font une méthode attrayante pour les chercheurs et les praticiens.

Chapitre III Encodage ASP pour l'extraction des itemsets fréquents fermés et maximaux

III.1 Introduction

L'extraction des itemsets fréquents, fermés et maximaux est une tâche cruciale dans l'analyse de données et la fouille de données. Elle permet de découvrir des associations significatives entre les éléments d'un ensemble de données (Han J. K., 2012). Ces associations peuvent révéler des modèles intéressants, des relations cachées ou des comportements communs dans différents domaines, tels que le commerce électronique, la recommandation de produits, la bioinformatique, etc. La découverte d'itemsets fréquents, fermés et maximaux est essentielle pour prendre des décisions éclairées et effectuer des analyses approfondies.

Le présent chapitre vise à présenter en détail l'implémentation de trois codes d'extraction d'itemsets fréquents, fermés et maximaux en utilisant la programmation ASP (Answer Set Programming). Nous examinerons les fondements théoriques de ces concepts et explorerons comment Clingo, un solveur ASP puissant, peut être utilisé pour coder ces règles et extraire ces itemsets. Nous discuterons également des performances de ces approches et les comparerons à d'autres méthodes couramment utilisées telles que SPMF (Sequential Pattern Mining Framework) (Fournier-Viger, 2014).

III.2 Encodages ASP pour l'extraction des itemsets fréquents, fermés et maximaux

Dans cette partie du chapitre, nous nous concentrons sur le codage des règles ASP (Answer Set Programming) pour l'extraction des itemsets fréquents, fermés et maximaux. Nous explorons en détail les règles clés utilisées dans l'encodage, telles que la vérification du support minimum, la propriété de fermeture et la propriété de maximalité.

Nous allons maintenant fournir une explication intuitive des encodages utilisés pour les différents problèmes d'extraction d'itemsets, spécifiquement appelé ASP(1), ASP(2) et ASP(3) et présentés respectivement dans les listings 1, 2 et 3. Il est important de noter qu'il existe toujours un ensemble de réponses pour tout ensemble de données D et valeur de minsupp, car même l'ensemble vide est considéré comme un ensemble d'éléments fréquents.

Avant de commencer l'implémentation des codes, il faut d'abord définir la base de données. La base de données est définie à l'aide du prédicat `db/2`. Chaque ligne représente un enregistrement avec un identifiant de transaction (T) et un élément (I). Par exemple la base de données "splice" est utilisée pour l'analyse des séquences d'ADN. Chaque transaction représente une séquence d'ADN et les éléments dans la transaction indiquent les acides aminés présents dans cette séquence. Donc, une transaction peut être

"db(1,A).db(1,G).db(1,C).db(1,T)", indiquant la présence des acides aminés A, G, C et T (les items) dans la séquence d'ADN (transaction).

III.2.1 L'extraction d'itemsets fréquents :

Considérons d'abord le cas de l'extraction d'itemset fréquents (voir listing1). Les prédicats "`item(I)`" et "`transaction(T)`" sont définis pour représenter respectivement les items et les transactions dans la base de données `db/2`. Ils sont définis en fonction des éléments de la base de données. Par exemple, "`item(I) :- db(_,I).`" signifie que tout élément présent dans la base de données est un item. `transaction(T) :- db(T, _)` : Ce prédicat définit les identifiants uniques des transactions en se basant sur la première colonne (T) de `db/2`.
Concernant la règle `support(T):-transaction(T),#false:fitemset(I), not db(T,I).` Le prédicat `support(T)` représente le nombre de fois où une Transaction T contient un item fréquent. Ce prédicat est vrai si « T » est une transaction, et cette transaction contient un item fréquent « I ». Les détails ce cette règle est comme suit :

`transaction(T)` : Ce prédicat définit une transaction (T). Il est utilisé pour itérer sur toutes les transactions de la base de données.

`#false:fitemset(I), not db(T,I).` Si l'item « I » n'appartient pas à la liste des items fréquents alors il ne doit pas appartenir à la transaction T.

Ensuite, nous avons une règle "`{fitemset(I)}:- item(I).`" qui définit les itemsets fréquents. Cette règle utilise une contrainte de choix pour inclure tous les items de la base de données dans les itemsets.
Le prédicat "`ns(C)`" est défini pour compter le nombre de transactions soutenant chaque itemset. Il utilise une règle qui assigne à C le nombre de transactions soutenant chaque itemset.
La règle "`:- {support(T)}< minsupp.`" définit la contrainte de support minimum pour les itemsets fréquents. Cette règle indique que si le nombre de transactions soutenant un itemset est inférieur à minsupp, alors cette règle est violée.
Enfin, les directives "#show fitemset/1." et "#show ns/1." sont utilisés pour afficher les itemsets fréquents et les nombres de support respectifs.

```
1. item(I) :- db(_,I).
2. transaction(T) :- db(T,_).
3. support(T):-transaction(T),#false:fitemset(I), not db(T,I).
4. {fitemset(I)}:- item(I).
5. ns(C):-C={support(T)}.
6. :- {support(T)}< minsupp.
7. #show fitemset/1.
8. #show ns/1.
```

Listing 1: Encodage ASP(1) de l'extraction d'itemsets fréquents.

III.2.2 L'extraction d'itemsets maximaux :

Pour l'extraction des itemsets maximaux ASP (2) (Voir listing 2) on se basant sur l'encodage d'extraction d'itemsets fréquents standard, les trois premières lignes de l'encodage ASP (1) et ASP (2) sont identiques.

L'inclusion du critère de maximalité supplémentaire pour les itemsets fréquents nécessite seulement une petite modification de l'encodage'ASP (1). De ce fait, nous avons ajouté la règle " `maxitemset(I):- item(I), minsupp {support(T) : db(T,I)}.` " qui permet d'extraire les itemsets maximaux en vérifiant si chaque item de la base de données atteint ou dépasse le seuil de support spécifié. Les itemsets qui satisfont cette condition sont considérés comme maximaux et sont ajoutés à la relation `maxitemset(I).`

Cela permet d'identifier les ensembles d'items qui n'ont pas des super itemset fréquent tout en respectant la contrainte de support minimum, ce qui les rend maximaux dans le contexte donné.

```
1. item(I) :- db(_,I).
2. transaction(T) :- db(T,_).
3. support(T):-transaction(T), #false:maxitemset(I), not db(T,I).
4. maxitemset(I):- item(I), minsupp {support(T) : db(T,I)}.
5. #show maxitemset/1.
```

Listing2: Encodage ASP(2) de l'extraction d'itemsets maximaux.

III.2.3 L'extraction d'itemsets fermés :

Pour l'extraction des itemsets fermés ASP (3). (Voir listing 3), Les trois premières lignes de cet encodage sont identiques aux trois premières lignes des encodages précédents La ligne "`:- {support(T)} < minsupp`" spécifie une contrainte de support minimum pour les itemsets fermés. Cette règle indique que si le nombre de transactions soutenant un itemset fermé est inférieur à "minsupp" (la valeur définie pour le support minimum), alors cette règle est violée.

La ligne "`minsize {closeditemset(I) : item(I)}`" fixe une taille minimale pour les itemsets fermés. Cette règle garantit que seuls les itemsets fermés ayant une taille supérieure ou égale à "`minsize`" sont pris en compte.
La ligne "`nodb(T,I) :- transaction(T), item(I), not db(T,I)`" définit le prédicat "`nodb(T,I)`" qui indique qu'un élément (I) n'est pas présent dans une transaction (T). Cette règle est utilisée pour vérifier si un élément n'appartient pas à une transaction spécifique.
La ligne "`:- item(I), not closeditemset(I), not support(T) : nodb(T,I)`" spécifie une règle qui indique que si un élément (I) n'appartient pas à un itemset fermé, et s'il n'est pas soutenu par une transaction, alors cette règle est violée. Cela permet d'assurer que tous les éléments présents dans un itemset fermé sont effectivement soutenus par au moins une transaction.
La directive "`#show closeditemset/1`" est utilisée pour afficher les itemsets fermés identifiés dans le code.

```
1. item(I) :- db(_,I).
2. transaction(T) :- db(T,_).
3. support(T):-transaction(T), #false:closeditemset(I), not db(T,I).
4. :- {support(T)} < minsupp.
5. minsize {closeditemset(I) : item(I)}.
6. nodb(T,I) :- transaction(T), item(I), not db(T,I).
7. :- item(I), not closeditemset(I), not support(T) : nodb(T,I).
8. #show closeditemset/1.
```

Listing 3: Encodage ASP(3) de l'extraction d'itemsets fermés.

III.3 Implémentation, Expérimentations et résultats

III.3.1 Description de l'environnement de développement

L'implémentation de l'encodage d'extraction des itemsets fréquents, fermés et maximaux a été réalisée en utilisant l'environnement de développement Spider d'Anaconda, associé à la version 5.6.2 de Clingo.

Spider est un IDE (Integrated Development Environment) largement utilisé dans le domaine de la science des données et du développement Python. Il offre une interface conviviale et intuitive pour écrire, exécuter et déboguer du code. L'utilisation de Spider a permis une meilleure productivité et une expérience de développement plus fluide.

Pour l'exécution des programmes ASP, la version 5.6.2 de Clingo a été installée dans l'environnement Conda. Clingo est un solveur ASP (Answer Set Programming) qui permet de résoudre des problèmes de logique et d'optimisation. En utilisant Clingo, il est possible de spécifier des contraintes et des règles logiques pour résoudre des problèmes complexes.

Dans le cadre de cette implémentation, la commande `!clingo nom_fichier --const n=K -n 0` a été utilisée pour exécuter les codes ASP. Cette commande spécifique permet de lancer Clingo. Elle définit également une constante `n` qui le support minimum et fixe la taille maximale des modèles à 0.

III.3.2 Jeu de données utilisé

Pour nos expérimentations et afin de comparer les performances de l'approche ASP avec les algorithmes implémentés dans l'outil SPMF, nous avons utilisé plusieurs jeux de données représentatifs. Les bases de données que nous avons sélectionnées sont "zoo", "splice" et "mushroom"[2].

- La base de données "zoo" : contient des informations sur différents animaux du zoo. Chaque transaction représente un animal et les éléments dans la transaction indiquent les caractéristiques de cet animal, telles que son habitat, son régime alimentaire, sa classe, etc. Par exemple, une transaction peut être "(mammal,carnivore,land)", indiquant qu'il s'agit d'un mammifère carnivore terrestre.
- La base de données "splice" est utilisée dans le domaine de la bioinformatique pour l'analyse des séquences d'ADN. Chaque transaction représente une séquence d'ADN et les éléments dans la transaction indiquent les acides aminés présents dans cette séquence. Par exemple, une transaction peut être " (A, G, C, T)", indiquant la présence des acides aminés A, G, C et T dans la séquence d'ADN.
- La base de données "mushroom" contient des informations sur différentes espèces de champignons. Chaque transaction représente un champignon et les éléments dans la transaction indiquent les caractéristiques de ce champignon, telles que sa couleur, son odeur, sa population, etc. Par exemple, une transaction peut être

[2] **Constraint Programming for Itemset Mining,**
[enligne],[https://dtai.cs.kuleuven.be/CP4IM/datasets/],(01/06/2013).

"(white,edible,abundant)", indiquant qu'il s'agit d'un champignon blanc comestible et abondant.

En détails concernant ces bases de données en termes de nombre de transactions et le nombre d'items sont présentés dans la table 8.

Table 8 : Propriétés de bases de données représentatives.

Bases de données	Transactions	Items
Zoo	101	36
Mushroom	8124	119
Splice	3190	287

En utilisant ces bases de données, nous avons pu évaluer les performances des approches ASP et les algorithmes SPMF, permettant ainsi une comparaison approfondie des résultats obtenus.

III.3.3 La validité des encodages ASP proposés

En plus des encodages ASP proposés pour l'extraction des itemsets fréquents, maximaux et close, qui sont logiquement faits et rigoureusement étudiés, nous avons testé la validité de ces encodages d'une manière expérimentale. Ainsi, nous avons utilisé les bases précédemment décrites pour comparer les itemsets extraits par nos encodages avec ceux extraits de plusieurs algorithmes implémentés dans l'outil SPMF (les algorithmes utilisés sont décrits dans la section suivante). Les comparaisons ont été faites avec différentes valeurs de support minimum. Les résultats obtenus étaient identiques à 100% pour toutes les comparaisons. Cela nous a permis de nous assurer que les encodages proposés sont corrects et valides.

III.3.4 Comparaison des performances de l'approche ASP avec les algorithmes SPMF

III.3.4.1 La bibliothèque SPMF

SPMF (Sequential Pattern Mining Framework) (Fournier-Viger, 2014) est une bibliothèque Java open-source largement utilisée pour l'extraction de motifs séquentiels et d'itemsets fréquents à partir de données transactionnelles. Elle propose une variété d'algorithmes efficaces et populaires pour l'exploration de motifs dans les bases de données.

SPMF (Fournier-Viger, 2014) met en œuvre des techniques d'extraction de motifs tels que l'algorithme Apriori, FP-Growth, Eclat, etc. Ces algorithmes utilisent des approches

différentes pour découvrir des itemsets fréquents, ce qui les rend adaptés aux différents types de jeux de données et de contraintes de performances.

SPMF offre une interface conviviale pour l'utilisation de ces algorithmes et permet de spécifier différents paramètres de configuration, tels que le seuil de fréquence, la taille maximale des itemsets, etc. Cela permet aux utilisateurs d'ajuster les paramètres en fonction de leurs besoins spécifiques.

L'un des avantages de SPMF est sa flexibilité et sa capacité à traiter des jeux de données de grande taille. Il est également bien documenté et largement utilisé dans la communauté de la fouille de données.

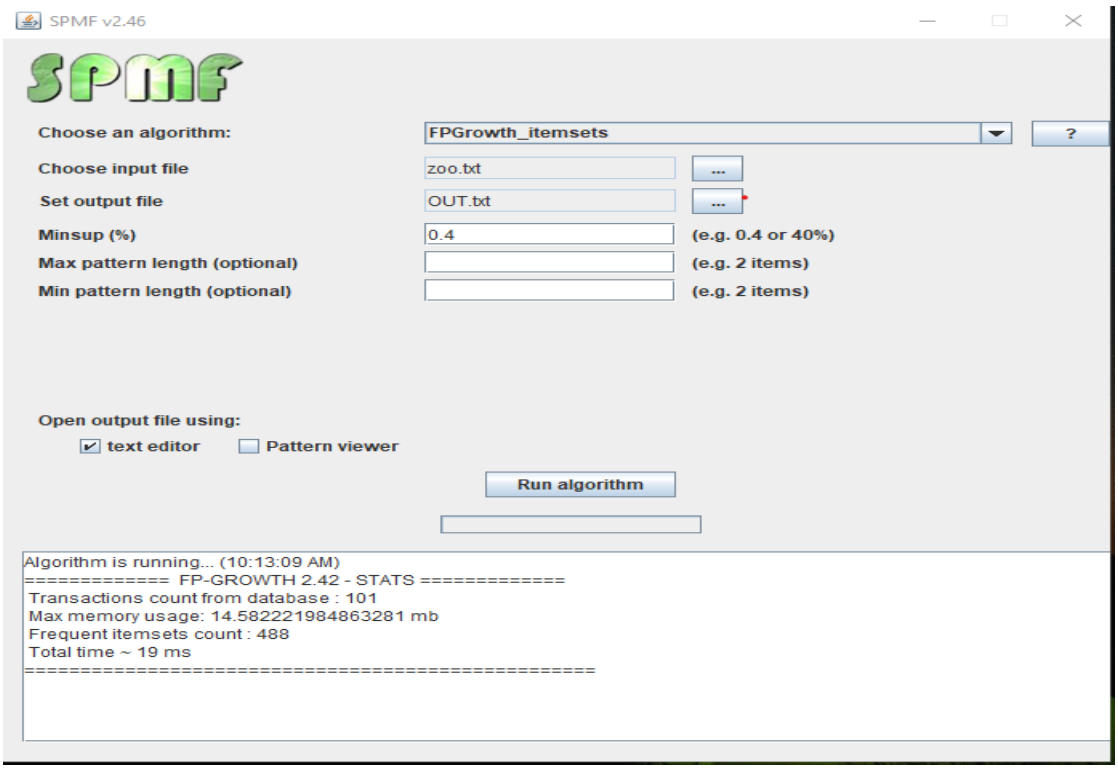

Figure 19 : Interface utilisateur de SPMF (Fournier-Viger, 2014).

III.3.4.2 Critère de comparaison

Nous avons choisi le temps d'exécution comme un critère de comparaison entre ASP et les algorithmes de SPMF. Pour faire nous avons mesuré le temp d'exécution de l'encodage et de l'algorithme sujets de comparaison à des valeurs différents du support minimum. Nous avons utilisé la bibliothèque `subprocess` pour exécuter les différentes méthodes, en leur passant les paramètres appropriés, y compris les différentes valeurs de support. Nous avons mesuré le temps d'exécution à l'aide de la bibliothèque `time` en enregistrant le moment précis avant et après chaque exécution.

III.3.5 Présentation et analyse des résultats obtenus

Dans cette étude, nous avons réalisé une évaluation préliminaire de l'efficacité des encodages ASP (1), ASP (2) et ASP (3) ainsi que des algorithmes de l'outil SPMF sur des ensembles de données du monde réel. Notre objectif était de comparer les performances des encodages ASP avec celles des algorithmes de l'outil SPMF pour l'extraction d'ensembles fréquents. Les algorithmes utilisés pour les comparaisons sont : l'algorithme FPGrowth pour l'extraction des itemsets fréquents, l'algorithme FP-Max pour l'extraction des itemsets maximaux et l'algorithme FP-Closed pour l'extraction des itemsets closed.

Pour mener nos expériences, nous avons utilisé Clingo (version 5.6.2), basé sur le solveur Clasp ASP (version 3.3.9) avec les paramètres par défaut, et SPMF (version 2.46). Nous avons évalué les performances en termes de temps d'exécution.

La comparaison entre les codes ASP et les algorithmes de l'outil SPMF sont montrés dans les figures 20,21 et 22.

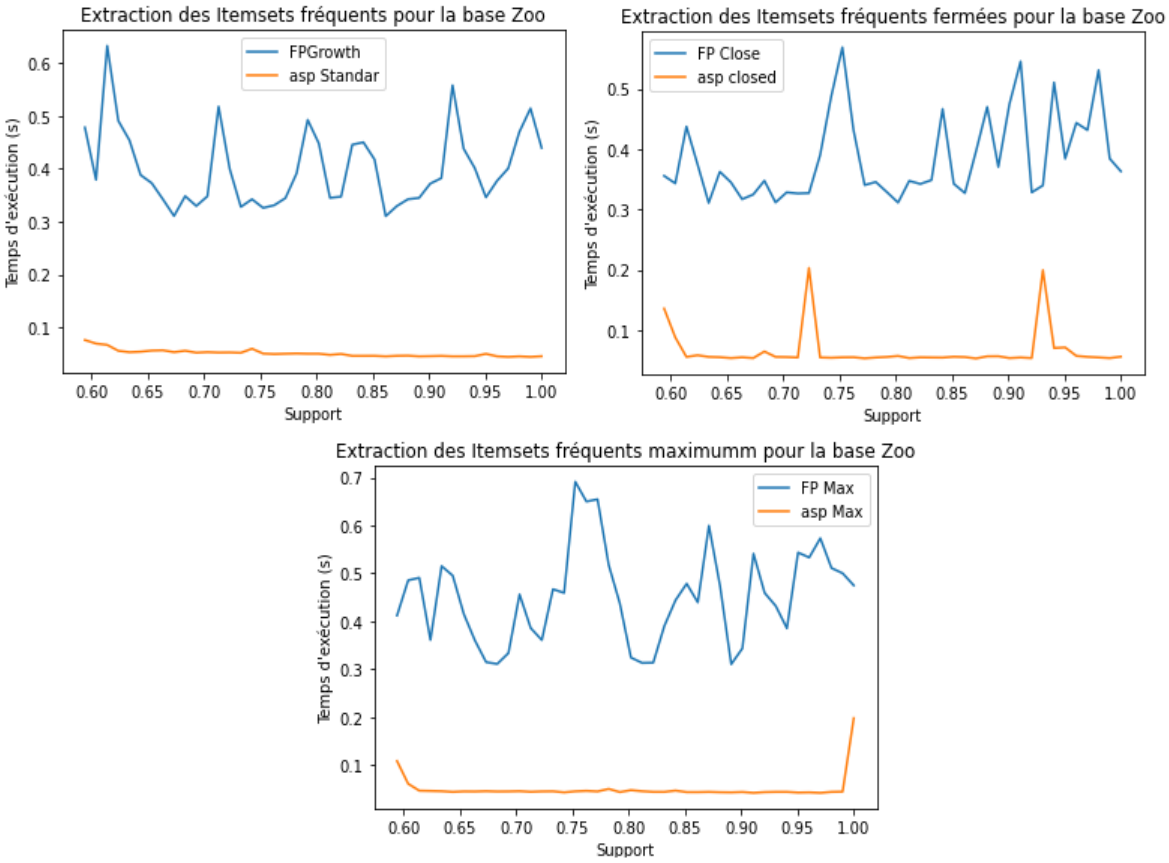

Figure 20 : Extraction des itemsets fréquents pour la base Zoo

La première expérimentation (Figure 20) concerne la base de données "Zoo". Cette base est composée de 101 transactions et 36 items. Nous avons mesuré le temps d'exécution de chaque algorithme pour extraire les itemsets fréquents, ainsi que les itemsets fréquents maximums et fermés à des valeurs différentes de support minimum.

Les résultats de cette expérimentation ont montré que l'encodage l'ASP a affiché des performances supérieures dans les trois cas d'extraction par rapport aux algorithmes FP-growth, FP-Max et FP-Close.

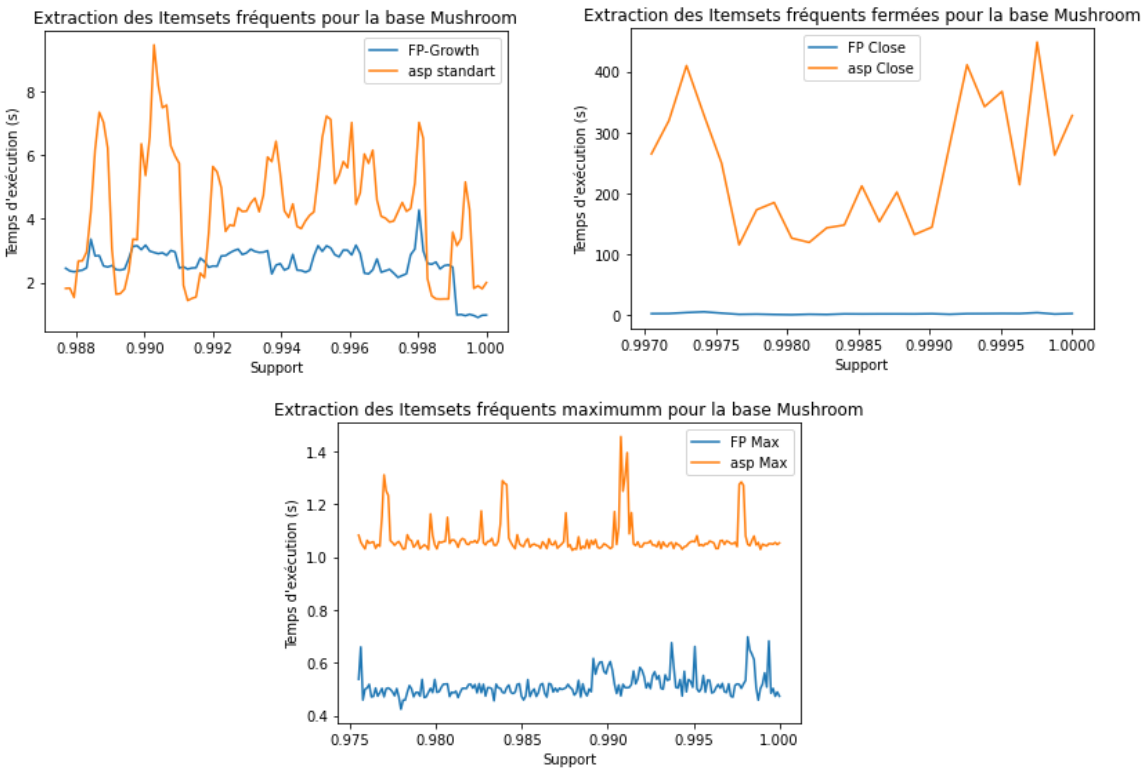

Figure 21 : Extraction des itemsets fréquents pour la base Mushroom

L'expérimentation de la figure 21 est effectuée en utilisant la base de données Mushroom, Cette base contient exactement 8124 transactions et 119.

Nous avons constaté que l'encodage ASP nécessite beaucoup plus de temp d'exécution par rapport aux algorithmes SPMF, et dans les trois cas d'extraction. Ce résultat est davantage plus apparent dans les cas d'extraction des itemsets max et les itemsets closed. Pour le cas standard (itemsets fréquent), la différence est moins importante avec quelques exceptions où l'encodage ASP a fait mieux que l'algorithme FP-Growth.

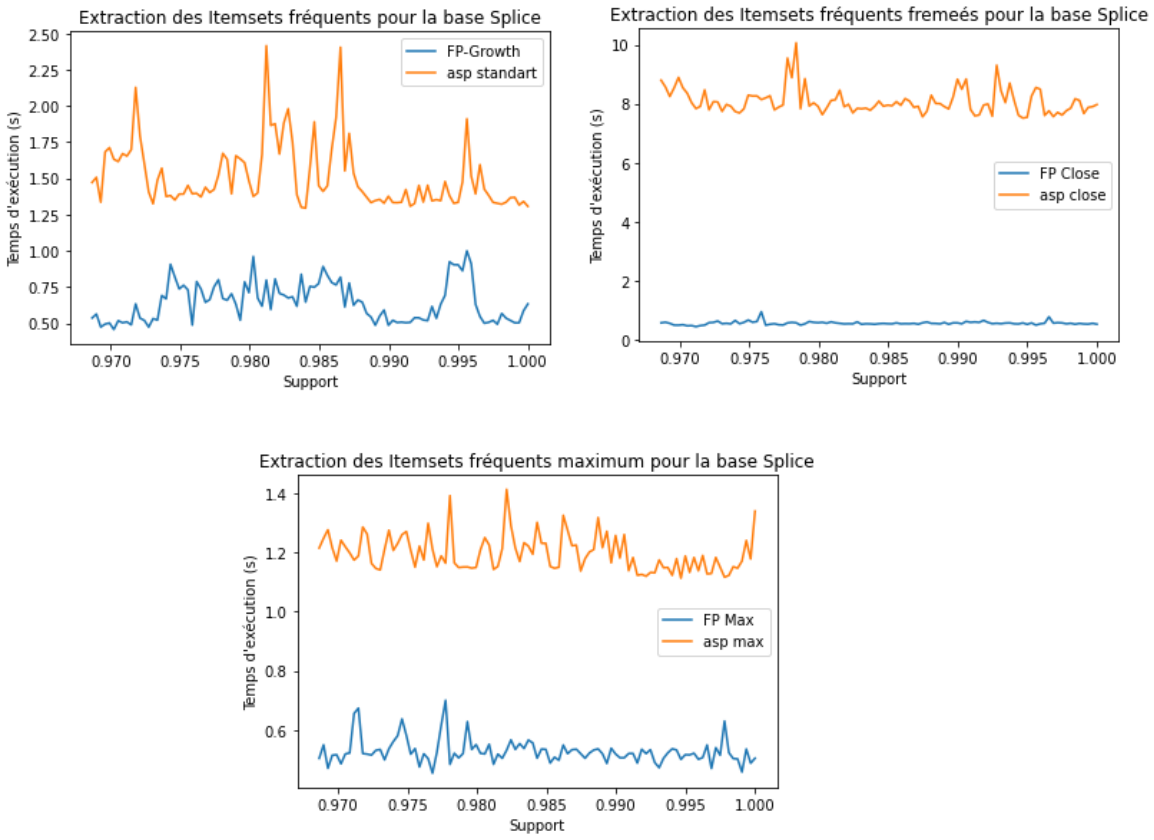

Figure 22 : Extraction des itemsets fréquents pour la base Splice

Pour la dernière expérimentation (Figure 22), nous avons effectué la comparaison en utilisant la base de données Splice. Cette base contient 3190 transactions et 287 items. Les résultats de cette expérimentation s'avèrent similaires aux résultats de l'expérimentation précédente. Du fait nous avons constaté que nos encodages ASP nécessitent beaucoup plus de temps d'exécution par rapport aux algorithmes de SPMF, et toujours avec une différence moins importante dans le cas standard c-à-dire l'extraction des itemsets fréquents.

D'une manière générale les résultats obtenus ont révélé que les encodages ASP se sont avérés particulièrement efficaces pour les petites bases de données (en termes de nombre de transactions). Par exemple, dans notre ensemble de données ZOO, nous avons observé un temps d'exécution moyen d'environ 0,1 seconde. Donc il est important de noter que ces résultats sont spécifiques à la base de données Zoo et aux paramètres choisis, tels que le support minimum 0,6. Ils peuvent varier pour d'autres bases de données. Cela suggère que les encodages ASP

peuvent être compétitifs par rapport aux algorithmes de l'outil SPMF, en particulier les algorithmes de la famille FPGrowth, et ça, pour les bases de données de petite taille.

Cependant, nous avons également constaté que les performances des encodages ASP ont diminué pour les grandes bases de données. Dans ces cas, les algorithmes de l'outil SPMF, en particulier l'algorithme FPClose, ont démontré une supériorité en termes de temps d'exécution. Cette observation soulève la question de l'évolutivité des encodages ASP pour des ensembles de données plus importants.

Dans l'ensemble, nos résultats suggèrent que les encodages ASP peuvent constituer une alternative prometteuse aux algorithmes traditionnels pour l'extraction d'ensembles fréquents, notamment pour les bases de données de petite taille. Cependant, des recherches supplémentaires sont nécessaires pour approfondir cette comparaison, en prenant en compte des critères tels que la précision des résultats, l'évolutivité et l'adaptabilité des encodages ASP à différentes configurations et tailles de bases de données.

Nous reconnaissons aussi que la réduction de l'intervalle du seuil de support minimum peut conduire à des résultats plus rapides, mais cela peut également entraîner des limitations en termes de précision et de complétude des résultats.

III.4 Conclusion

Ce chapitre a présenté une étude approfondie sur l'utilisation de la programmation ASP (Answer Set Programming) pour l'extraction des itemsets fréquents, fermés et maximaux.

Nous avons ensuite décrit l'environnement de développement utilisé, en mettant l'accent sur l'outil Clingo, un solveur basé sur Clasp ASP. Nous avons détaillé les règles ASP utilisées pour l'extraction des itemsets fréquents, fermés et maximaux, en soulignant l'approche adoptée pour chaque type d'itemset.

Pour évaluer les performances de notre approche ASP, nous avons réalisé des expérimentations en utilisant un jeu de données représentatif. Nous avons comparé les résultats obtenus avec ceux d'autres méthodes, en particulier l'outil SPMF, largement utilisé dans le domaine de l'extraction des itemsets.

Les résultats de nos expérimentations ont révélé que l'approche ASP présente des performances prometteuses pour l'extraction des itemsets fréquents, en particulier pour les bases de données de petite taille. Cependant, pour les bases de données plus importantes, les algorithmes de SPMF ont montré une supériorité en termes de temps d'exécution.

Ces résultats soulignent l'importance de prendre en compte des critères tels que la taille de la base de données et la complexité des règles lors du choix de l'approche d'extraction des

itemsets. De plus, nos expérimentations ont mis en évidence la nécessité de recherches supplémentaires pour améliorer les performances de l'approche ASP, en explorant des solveurs alternatifs et en optimisant les encodages et les stratégies de recherche des solveurs.

Conclusion Générale

Dans ce travail, nous avons présenté une vue générale sur le domaine de l'extraction des itemsets fréquents, leurs applications pratiques, ainsi que quelques algorithmes de l'état de l'art pour l'extraction des itemsets fréquents, fermés et maximaux. Le but principal de notre travail était d'appliquer les techniques de l'Answer Set Programming (ASP) dans le domaine d'extraction des itemsets fréquents. Ainsi, nous avons introduit le paradigme Answer Set Programming (ASP), en décrivant sa syntaxe, sa sémantique et ses techniques avancées telles que les agrégats, les opérateurs de négation et les règles de choix. Le mémoire met également en évidence les applications de l'ASP dans la représentation des connaissances, la robotique, la biologie computationnelle, l'e-tourisme et la configuration des produits et des services.

Parmi les points importants visés dans cette étude était d'examiner les performances des encodages ASP pour l'extraction d'ensembles fréquents dans différentes bases de données. Les résultats ont montré que les encodages ASP sont efficaces pour les bases de données de petite taille, avec des temps d'exécution compétitifs par rapport aux algorithmes traditionnels de l'outil SPMF. Cependant, pour les bases de données plus importantes, les encodages ASP ont montré une diminution des performances par rapport aux autres algorithmes.

Ces résultats soulèvent des questions sur l'évolutivité des encodages ASP pour des ensembles de données plus importants. Il est nécessaire de mener des recherches supplémentaires afin d'approfondir cette comparaison, en prenant en compte des critères tels que la précision des résultats, l'évolutivité et l'adaptabilité des encodages ASP à différentes configurations et tailles de bases de données.

Les travaux futurs pourraient se concentrer sur l'exploration de nouveaux solveurs ASP et la mise en œuvre de stratégies de recherche avancées pour améliorer les performances des encodages ASP. Une évaluation plus approfondie des avantages et des limites des encodages ASP dans le contexte de l'extraction d'ensembles fréquents permettra d'ouvrir la voie à de nouvelles avancées dans ce domaine.

Enfin, et bien que les encodages ASP montrent des performances prometteuses pour les bases de données de petite taille, il est nécessaire de continuer à approfondir cette approche et de l'adapter aux défis posés par les bases de données de plus grande envergure.

Bibliographie

Agrawal, Rakesh, Tomasz Imieliński, and Arun Swami. (1993, June). Mining association rules between sets of items in large databases. *1993 ACM SIGMOD international conference on Management of data.*

Amir, A. (2019., Feb 3). *medium*. Retrieved from The Art of Data Sciene: https://medium.com/machine-learning-researcher/association-rule-apriori-and-eclat-algorithm-4e963fa972a4

Aschinger, M. D. (2011). Optimization methods for the partner units problem. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems:8th International Conference,* (pp. 4-19). Berlin, Germany.

Baral, C. (2001). Knowledge representation, reasoning and declarative problem solving with Answer sets1. *Knowledge Creation Diffusion Utilization.*

Calimeri, F. G. (2016). Design and results of the fifth answer set programming competition. *Artificial Intelligence*, 151-181.

Cavique, L. (2007, November). A scalable algorithm for the market basket analysis. *Retailing and Consumer Services, 14*(6), 400-407.

Chin-Hoong Chee, Jafreezal Jaafar, Izzatdin Abdul Aziz, Mohd Hilmi Hasan. (2018). Algorithms for frequent itemset mining. *a literature review, 24*, march.

Claude Issa Nombré, Konan Brou, Kouadio Kimou. (2016, Oct 24). OPTIMISATION D'EXTRACTION DES KITEMSETS FREQUENTS (POUR K ≤ 2). *ALOA2i.*

Dovier, A. F. (2009). An empirical study of constraint logic programming and answer set programming solutions of combinatorial problems. *Experimental & Theoretical Artificial Intelligence, 21*(2), 79-121.

Eiter, T. I. (2009). Answer set programming: A primer. *Springer Berlin Heidelberg*, 40-110.

Erdem, E. A. (2012). Answer set programming for collaborative housekeeping robotics: representation, reasoning, and execution. *Intelligent Service Robotics, 5*, 275-291.

Erdem, E. G. (2016). Applications of answer set programming. *AI Magazine, 37*(3), 53-68.

Erdem, E. P. (2015). Integrating hybrid diagnostic reasoning in plan execution monitoring for cognitive factories with multiple robots. *In 2015 IEEE international conference on robotics and automation (ICRA)*, (pp. 2007-2013).

Falkner, A. F. (2018). Industrial applications of answer set programming. *KI-Künstliche Intelligenz, 32*, 165-176.

Fournier-Viger, P. G. (2014). Spmf: a java open-source pattern mining library. *Machine Learning Research, 15*(1), 3389-3393.

Fréchette, A. N.-B. (2016). Solving the station repacking problem. *In Proceedings of the AAAI Conference on Artificial Intelligence, 30.*

Front. Bioeng. Biotechnol. (2020, September 04). Review on the Application of Machine Learning Algorithms in the Sequence Data Mining of DNA.

Gebser, M. K. (2011). aspcud: A linux package configuration tool based on answer set programming. *In:Drescher C, Lynce I, Treinen R (eds) Proceedings second workshop on logics for component configuration*, *65*, pp. 12-25.

Gebser, M. K. (2012). Answer set solving in practice. *Morgan & Claypool Publishers, 6*, 1-238.

Gebser, M. K. (2015). *Potassco user guide.* Institute for Informatics, University of Potsdam.

Gebser, M. K. (2019). Multi-shot ASP solving with clingo. *Theory and Practice of Logic Programming, 19*(1), 27-82.

Gebser, M. S. (2011). Detecting inconsistencies in large biological networks with answer set programming. *Theory and Practice of Logic Programming, 11*(2-3), 323-360.

Han, J. K. (2012). *Data mining concepts and techniques third edition.* University of Illinois, Urbana-Champaign Micheline Kamber Jian Pei Simon Fraser University.

Han, J. P. (2004). Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. *Data Mining and Knowledge Discovery*, pp. 53–87.

Jaiswal, S. (n.d.). *fp-growth algorithm in data mining*. (javatpoint) Retrieved from https://www.javatpoint.com/fp-growth-algorithm-in-data-mining

Jeudy, B. (2002, decembre 13). OPTIMISATION DE REQUETES INDUCTIVES . *APPLICATION A L'EXTRACTION SOUS CONTRAINTES DE REGLES D'ASSOCIATION*.

Jiawei Han, Hong Cheng, Dong Xin. (2007, January 27). Frequent pattern mining: current status and future. *Data Min Knowl Disc*.

Jose Maria Luna, Philippe Fournier-Viger, Sebastian Ventura. (n.d.). Frequent Itemset Mining. *a 25 Years Review*.

Karam Gouda & Mohammed J. Zaki . (2005, July 15). Itemsets, GenMax: An Efficient Algorithm for Mining Maximal Frequent. *Data Mining and Knowledge Discovery*.

Kaufmann, B. L. (2016). Grounding and solving in answer set programming. *AI magazine, 37*(03), 25-32.

Konya, Türkiye. (November 2020). Social Campus Application with Machine Learning for Mobile Devices. *International Conference on Engineering Technologies.*

Leone, N. &. (2015). Answer set programming: A tour from the basics to advanced development tools and industrial applications. *Reasoning Web. Web Logic Rules: 11th International Summer School 2015, 11*, pp. 308-326. Berlin, Germany.

Leone, N. P. (2006). The DLV System for Knowledge Representation. *ACM Transactions on Computational Logic (TOCL), 7*, 499-562.

McIlraith, S. (2006, april 19). *Introduction to Answer Set Programming.* Department of Computer Science, University of Toronto.

Mohammed J. Zaki and Ching-Jui Hsiao. (2005). *CHARM: An Efficient Algorithm for Closed Itemset Mining.*

Mohammed J. Zaki, C.-J. H. (2014). *CHARM: An Efficient Algorithm.*

Niemela, I. S. (2000). Smodels: A System for Answer Set Programming. *In Proc. 8th International Workshop on Non-Monotonic Reasoning (NMR'2000).* Breckenridge, Colorado, USA.

Nogueira, M. B. (2001). An A-Prolog decision support system for the Space Shuttle. *In Practical Aspects of Declarative Languages: Third International Symposium(PADL)*, (pp. 169-183). Las Vegas, Nevada.

Politi L, Codish S, Sagy I and Fink L. . (2020, MAR 04). Substitution and complementarity in the use of health information exchange and electronic medical records. Journal of Information Systems , Online publication , European .

Raj, S. R. (2020). A Spark-based Apriori algorithm with reduced shuffle overhead. *Supercomputing*, 133–151.

Rakesh Agrawal and Ramakrishnan Srikant. (September 1994.). Fast algorithms for mining association rules. *VLDB* (pp. 487-499). Santiago: Chile.

Ricca, F. D. (2010). A logic-based system for e-tourism. *Fundamenta Informaticae,, 105*(1-2), 35-55.

Ricca, F. G. (2012). Team-building with answer set programming in the Gioia-Tauro seaport. *Theory and Practice of Logic Programming, 12*(3), 361-381.

Sharma, N. and Om, H. (2014). Extracting Significant Patterns for Oral Cancer Detection Using Apriori Algorithm. *Intelligent Information Management*, pp. 30-37.

tanagra. (2020, 01 20). *fr_Tanagra_Itemset_Mining*. Retrieved from eric.msh-lse.fr: https://eric.msh-lse.fr/?s=fr_Tanagra_itemset_Mining

Tiihonen, J. S. (2003). A practical tool for mass-customising configurable products. *In DS 31: Proceedings of ICED 03, the 14th International Conference on Engineering Design.* Stockholm.

Tran, N. &. (2009). Hypothesizing about signaling networks. *Journal of Applied Logic, 7*(3), 253-274.

Verma, Y. (2021, September 21). *Apriori vs FP-Growth in Market Basket Analysis – A Comparative Guide.* Retrieved from analyticsindiamag: https://analyticsindiamag.com/apriori-vs-fp-growth-in-market-basket-analysis-a-comparative-guide/

Vladimir, L. (2008). What is answer set programming. *Vladimir, L.*

Weinzierl, A. (2017). Blending lazy-grounding and CDNL search for answer-set solving. *In Logic Programming and Nonmonotonic Reasoning: 14th International Conference, LPNMR 2017* (pp. 191-204). Espoo, Finland: Springer International Publishing.

Zaki, M. J., & Hsiao, C.-J. (2002). CHARM: An Efficient Algorithm for Closed Itemset Mining. . *2002 SIAM International Conference on Data Mining*, (pp. 457–473).

Zhang, Wenjing; Ma, Donglai; Yao, Wei. (May 2014). Medical Diagnosis Data Mining Based on Improved Apriori Algorithm. *Networks*, 1339-1345.