MA-004-477-1

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahlab Blida

N° D'ordre :.....

Faculté des sciences

Département d'informatique

Mémoire Présenté par :

MESSIKA Oussama

En vue d'obtenir le diplôme de master

Domaine : Mathématique et informatique

Filière : Informatique

Spécialité : Informatique

Option : Génie des systèmes informatiques

Thème

Application mobile pour l'extraction des itemsets fréquents

Soutenu le :

M.	Kaneh	Président
M.	Gesserm	Examinateur
Mme	ZAHRA FATMA ZOHRA	Promotrice

Promotion : 2016 / 2017

MA-004-477-1

Remerciements

Tout d'abord, je tiens à remercier " DIEU" le tout puissant de m'avoir donné la santé, le courage et la volonté de réaliser ce travail.

Nos remerciements les plus sincères, accompagnés de toute notre gratitude vont tout d'abord à notre promotrice Mlle. F/Z Zahra pour nous avoir proposé ce sujet et pour ses précieux conseils et de nous avoir dirigé durant notre projet, et surtout pour la confiance qu'il nous a accordé pour la réalisation de ce projet.

Nous remercions tous les enseignants de la faculté des sciences de BLIDA et surtout ceux du département informatique.

Nous remercions les membres de jury pour nous avoir fait l'honneur de juger notre travail.

Nous remercions également toute personne ayant contribué à notre éducation et notre formation.

Enfin, nos remerciements vont à toute personne ayant contribué, de près ou de loin, à l'aboutissement de ce travail.

Dédicaces

Je dédie ce travail ;

A ma famille pour leurs soutiens et pour leur encouragements et précieux conseils durant toute ma vie.

A ma mère et mon père pour leurs amours inconditionnels et leurs présences à mes côtés dans les moments difficiles.

A mes chers amis Riad, Anouar et Djaber pour leurs encouragements.

A mes amis Raouf, Othemene, Walid, Farouk , Ibtissem et Khalida pour leurs encouragements.

A tous mes enseignants pour le savoir et les connaissances qu'ils m'ont inculqué.

A tous mes amis que j'aime et qui m'aiment.

Résume

RESUME

La disponibilité de la puissance de calcul dans les appareils mobiles est un outil clé pour Mobile Data Mining (MDM) dans les locaux de l'utilisateur. Alternativement, les contraintes de ressources comme la réduction de la bande passante, la bande passante étroite et les petits écrans sont difficiles à adopter lors de l'adoption de MDM. À le temps actuelle, MDM est basé sur des algorithmes légers qui sont adaptatifs dans des environnements à contraintes de ressources, mais une étude pour évaluer la performance des algorithmes généraux manque encore dans la littérature. À cette fin, nous avons étudié trois algorithmes d'extraction de motifs fréquents (FPM), les avons déployés dans des appareils mobiles pour évaluer la faisabilité et souligné les défis associés. Les expériences ont été effectuées sur des ensembles de données réelles et synthétiques strictement dans un appareil mobile basé sur Android.

Mots Clés : Data Mining Mobile, EPM et mobile

ABSTRACT

The availability of computational power in mobile devices is key-enabler for Mobile Data Mining (MDM) at user-premises. Alternately, resource constraints like limited energy, narrow bandwidth, and small screens challenge in adoption of MDM. Currently, MDM is based on light-weight algorithms that are adaptive in resource constrained environments but a study to evaluate the performance of general algorithms still lacks in the literature. To this end, we have studied three Frequent Pattern Mining (FPM) algorithms, deployed them in mobile devices to evaluate the feasibility, and highlighted the associated challenges. The experiments were performed on real and synthetic data sets strictly in android-based mobile device.

Keys words: Mobile Data Mining (MDM), FPM and Mobile.

ملخص

توافر القدرة الحاسوبية في الأجهزة النقالة هو مفتاح تمكين لاستخراج البيانات الأجهزة النقالة في مكان العمل. بالتناوب، قيود الموارد مثل الطاقة المحدودة، عرض النطاق الترددي الضيق، والشاشات الصغيرة تحديا في اعتماد استخراج البيانات الأجهزة النقالة. حاليا، يستند استخراج البيانات الأجهزة النقالة خوارزميات خفيفة الوزن التي تتكيف في بيئات محدودة الموارد ولكن دراسة لتقييم أداء الخوارزميات العامة لا تزال في بداياتها نظرا لقلة المعطيات والدراسات

Résume

القليلة عنها. ولهذه الغاية، درسنا ثلاثة خوارزميات نمطية متكررة، ونشرتها في الأجهزة النقالة لتقييم جدوى، وأبرز التحديات المرتبطة بها. أجريت التجارب على مجموعات البيانات الحقيقية والاصطناعية بدقة في الجهاز المحمول

الكلمات المفتاحية: استخراج البيانات الأجهزة النقالة، استخراج أنماط متكررة

Sommaire

Sommaire

Sommaire

Introduction Générale

1. Contexte

Grace à la technologie moderne et les nouvelles inventions les smartphones sont devenus une chose indispensable dans notre vie quotidienne. Selon le principe d'un ordinateur, il peut exécuter divers logiciels/applications grâce à un système d'exploitation spécialement conçu pour mobiles, et donc peut concurrencer les normes pc au niveau de la performance.

La puissance de calcul croissante des dispositifs mobiles a ouvert de nouveaux axes de recherche en Data Mining. En effet, La mobilité et la portabilité de ces dispositifs imposent implicitement des contraintes de calcul en raison de limitations en énergie, bande passante, affichage, puissance de calcul, et capacité de stockage. Néanmoins, de nombreux algorithmes de data mining ont été exploité avec succès sur les smartphones, on les adapte à la contrainte de calcul et de ressources limitées avec les meilleures performances.

Plusieurs travaux ont été proposés dans ce sens, ce qui a engendré un nouveau champ de recherche en Data Mining appelé Mobile Frequent pattern Mining (MFPM). Actuellement, MFDM est basé sur des algorithmes légers qui sont adaptatifs dans des environnements à ressources limitées.

1. Problématique

Dans ce travail, on s'intéresse à une des techniques du Data Mining qui est l'extraction de motifs fréquents. Cette technique s'appuie sur des principes relativement simples. Son objectif est de trouver les motifs qui apparaissent fréquemment dans un ensemble de données. Or, les algorithmes classiques d'extraction de motifs ne s'adaptent pas aux environnements mobiles.

2. Objectifs

Le travail proposé s'intègre dans le cadre d'extraction des motifs fréquents en utilisant des dispositifs mobiles appelé Mobile Pattern Mining. Plus précisément, l'objectif de ce travail est la conception et la réalisation d'une application mobile pour l'extraction des itemsets fréquents à partir de données incertaines. La méthode utilisée est une méthode approchée qui se base sur les métas heuristiques.

Comme c'est possible de personnaliser son smartphone en y installant des applications additionnelles dans notre travail nous allons exploiter cet axe ou nous allons faire une Application mobile pour l'extraction des itemsets fréquents, qui peut être utile pour l'utilisateur.

4. Organisation du mémoire

Le mémoire se réparti en cinq chapitres.

Chapitre 1 : « extraction des motifs fréquents à partir des données incertaines »

Ce chapitre est consacré à étudié les différentes méthodes d'extraction des motifs fréquents à partir des données incertaines et les types de support.

Chapitre 2 : « Mobile Frequent pattern Mining»

Dans ce chapitre nous allons présenter les travaux reliés a ce domaine

Chapitre 3 : « Méta-heuristiques et Algorithmes évolutionnaire »

Dans ce chapitre nous détaillerons les différentes méthodes Meta heuristique et nous ferons une comparaison entre eux

Chapitre 4 : « Approche proposée »

Ce chapitre, sera réservé pour notre propres méthodes que nous allons utiliser dans notre application

Chapitre 5 : « Test et Validation »

Le dernier chapitre sera pour valider les méthodes adoptées, et on terminera par une conclusion générale.

Chapitre 01 :

Extraction des itemsets fréquents à partir des données incertaines

Chapitre 01 : Extraction des Itemsets fréquents à partir des données incertaines

1. Introduction

L'information et la donnée, Ces deux notions au cœur de la théorie de l'information se recoupent et les spécialistes du domaine ne les définissent pas de la même manière. Mais dans notre étude, on retiendra les définitions suivantes : Les données sont des faits ou statistiques pouvant être quantifiées, comptés, mesurés et stockés. Une information est définie comme étant l'ensemble de données fourni avec le contexte nécessaire pour la prise de décision.

Vu l'habilite et le comportement naturel de l'être humain et sa capacité de prendre des décisions en présence d'une grande masse d'informations imparfaites (incomplètes, manquantes, incertaines ou imprécises. . .), alors pour cela il viendra le rôle des systèmes décisionnels autrement dit l'automatisation de traitement de ce type d'information à fait l'objet plusieurs recherches dont l'objectif était proposé des modèles et méthodes pour représenter et manipuler l'information incomplète et incertaine pour faire face à des situations réelles.

2. L'information incertaine

Parfois notre connaissance du réel (précis ou d'imprécis) ne peut pas être énoncée avec confiance ou garantie absolue. L'information énoncée avec l'incertitude (probabilité ou possibilité) n'est pas incorrecte et ne compromet pas l'intégrité de l'information. Alors Une valeur est dite incertaine si la validité de la donnée reste douteuse au champ de validation indiquée dans l'information.

3. Base de données probabilistes

Une base de données d'information probabiliste est un espace de probabilité finie dont les résultats sont toutes des instances de base de données possibles compatibles avec un schéma donné. Cela peut être représenté comme la paire (X, p), où X est un ensemble fini d'instances possibles de base de données cohérentes avec un schéma donné, et p(I) est la probabilité associée à toute instance $I \in X$. Nous notons que puisque p(*) représente le vecteur de probabilité sur Toutes les instances de X[1], nous avons

$$\sum_{I \in X} p(I) = 1 \qquad (1)$$

3.1. Terminologie

Dans une base de données probabiliste, chaque élément de données - relation, pair et valeur qu'un attribut peut prendre est associé à une probabilité $\in (0,1)$, avec 0 représentant que les données sont certainement incorrectes et 1 représentant que c'est certainement correct.

3.2. Mondes possibles

Une base de données probabiliste pourrait exister dans plusieurs états. Par exemple, si nous sommes incertains quant à l'existence d'un pair dans la base de données, la base de données pourrait être dans deux états différents par rapport à ce pair.

Le premier état : contient le pair, tandis que le second ne le fait pas. De même, si un attribut peut prendre l'une des valeurs x, y ou z, la base de données peut être dans trois états différents par rapport à cet attribut. Chacun de ces états s'appelle un monde possible.

Considérons la base de données suivante dans le tableau 1.1 :

Tableau 1.1 : une base de données incomplet.

A	B
a1	b1
a2	b2
a3	{ b3, b3', b3' '}

Ici {b3, b3 ', b3' '} indique que l'attribut peut prendre l'une des valeurs b3, b3' ou b3 " Chacun de ces états s'appelle un monde possible.

Supposons que nous sommes incertains sur le premier pair, certain sur le deuxième pair et incertain sur la valeur de l'attribut B dans le troisième pair. Ensuite, l'état actuel de la base de données peut ou non contenir le premier pair (selon qu'il est correct ou non). De même, la valeur de l'attribut B peut être b3, b 'ou b3". Par conséquent, les mondes possibles correspondant à la base de données sont dans le tableau 1.2 :

Tableau 1.2 : base de données des mondes possible.

Monde 1		Monde 2		Monde 3		Monde 4		Monde 5		Monde 6	
A	B	A	B	A	B	A	B	A	B	A	B
a1	b1	a1	b1	a1	b1	a2	b2	a2	b2	a2	b2
a2	b2	a2	b2	a2	b2	a3	b3	a3	b3′	a3	b3″
a3	b3	a3	b3′	a3	b3″						

3.3. Types d'incertitude

Il existe essentiellement deux types d'incertitudes qui pourraient exister dans une base de données probabilistes, comme décrit dans le tableau ci-dessous (tableau 1.3) :

Tableau 1.3 : Types incertitude.

Incertitude au Niveau pair	Incertitude au niveau de l'attribut
Ici, nous ne sommes pas sûrs si un pair est correct ou non, c'est-à-dire s'il doit exister dans la base de données ou non.	Ici, nous ne sommes pas sûrs des valeurs qu'un attribut d'un pair peut prendre, c'est-à-dire qu'il pourrait prendre l'une des différentes valeurs possibles.
Correspondant à chaque pair incertain, il existe deux mondes possibles : celui qui comprend le pair alors que l'autre ne le fait pas.	Correspondant à chaque attribut incertain qui peut prendre l'une des valeurs a1, ..., an, il existe n mondes possibles.
L'incertitude au niveau des tubercules peut être considérée comme une variable aléatoire booléenne associée à chaque pair incertain	L'incertitude au niveau des attributs peut être considérée comme une variable aléatoire associée à chaque attribut incertain qui peut prendre les valeurs a1, ..., an.

4. Extraction des Itemsets frequents

L'extraction des Itemsets fréquents est une étape intermédiaire pour découvrir des règles d'association et dans certains domaines d'application, le résultat de cette technique est directement utilisé et interprété ou bien, il sera utilisé comme une entré pour d'autre méthodes de découverte de connaissance.

L'idée principale du Data Mining est La recherche des régularités dans les bases de données est. Ces régularités s'expriment sous différentes formes. Avant de profonde, on donne un exemple très connu, par exemple dans l'analyse du panier d'achats de consommateurs, l'extraction des itemsets consiste à mettre en évidence les cooccurrences entres les produits achetés c.-à-d. Déterminer les produits (les items) qui sont « souvent » achetés simultanément. On parle alors d'itemsets fréquents.

Par exemple, en analysant les tickets de caisse d'un supermarché, on pourrait produire des itemsets (un ensemble d'items) du type « le pain et le lait sont présents dans 10% des

caddies » **[02] [03] [04].** Le fichier comporte 10 observations (transactions) et 4 items (voir Tableau 1.4).

Tableau 1.4 : Base de transactions. **[02]**

S1	S2	S3	S4
1	0	1	0
0	1	0	0
0	0	0	1
0	1	1	1
0	1	1	0
0	1	1	0
1	1	1	1
1	0	1	0
1	1	1	0
1	1	1	0

Voici quelques définitions pour comprendre bien les choses et prendre idée très claire sur que nous avons entre un d'expliquer :

Item : Un item correspond à un produit. Nous avons 4 items (S1, S2, S3 et S4) dans notre fichier. **[02]**

Support : Le support d'un item est égal au nombre de transactions dans lesquelles il apparaît. **[02]**

Itemset : Un itemset est un ensemble d'items. Le support d'un itemset comptabilise le nombre de transactions dans lesquelles les items apparaissent simultanément. Un itemset peut être composé d'un singleton. **[02]**

Itemset fréquent : Un itemset est dit fréquent si son support est supérieur à un seuil défini à l'avance, paramètre de l'algorithme de recherche. **[02]**

Superset : Un super set est un itemset défini par rapport à un autre itemset. **[02]**

Itemset fermé (closed itemset) : Un itemset fréquent est dit fermé si aucun de ses super sets n'a de support identique. Autrement dit, tous ses super sets ont un support strictement plus faible. **[02]**

Itemset maximal (maximal itemset) : Un itemset est dit maximal si aucun de ses super sets n'est fréquent. **[02]**

Itemset générateur (generator itemset) : Un itemset A est dit générateur s'il n'existe aucun itemset B tel que $B \subset A$ et que SUP(B) = SUP(A). Autrement dit, l'itemset est générateur si tous ses sous itemsets ont un support strictement supérieur. **[02]**

5. Les algorithmes d'extraction des itemsets fréquent à partir de données incertaines

Lorsqu'on manipule les données incertaines, on peut utiliser par exemple U-Apriori, UF- growth, UFP- growth et PUF- growth qu'ils sont les algorithmes d'extraction les plus connus où utilisent l'UF-arbre, l'UFP-arbre et le PUF-arbre respectivement. Ces arbres peuvent être larges et ainsi diminuent la performance d'extraction. Les chercheurs proposés des divers algorithmes comme U-Apriori, UH-manier, UF- growth etc. dans cette section nous allons présenter une analyse pour ces algorithmes.

Il existe deux approches des algorithmes d'extraction des itemsets fréquents à partir des données incertaines :

Tableau 1.5 : Algorithmes d'extraction des itemsets fréquents **[05]**

Algorithmes basés sur « Tester-et-générer»	Algorithmes basés sur « diviser-et-régner »
U-Apriori	UF-growth UFP- growth PUF-growth UH – mine

- **La technique « Tester-et-générer » :** les algorithmes parcourent l'espace de recherche par niveau. A chaque niveau *k*, un ensemble de candidats de taille *k* est généré. Cet ensemble de candidats est, généralement, élagué par la conjonction d'une métrique statistique (ex. le support) et des heuristiques basées essentiellement sur les propriétés structurelles des itemsets.

- **La technique « Diviser-pour-régner » :** les algorithmes essaient de diviser le contexte d'extraction en des sous-contextes et d'appliquer le processus de découverte des itemsets récursivement sur ces sous-contextes. Ce processus de découverte repose sur un élagage du contexte basé essentiellement sur l'imposition d'une métrique statistique et d'heuristiques introduites.

La technique la plus utilisé c'est la technique « **Tester-et-générer** » mais dans cette technique sur une base de données dense provoque le problème de goulot d'étranglement, à savoir la génération d'un nombre prohibitif de candidats, et sont très gourmands en ressources mémoire et aussi le temps d'exécution est un peu long.

Pour pallier à cette faiblisse de la technique précédente, des auteurs ont proposé la technique « Diviser-pour-régner » dans cette technique on va éviter le problème de goulot d'étranglement et de perde de l'espace mémoire et du temps d'exécution moins long que la technique précédente. Bref, les algorithmes qui sont implémenté dans la technique Divisé-pour-régner sont plus performant qui sont implémenté en Tester-et-générer.

5.1. Les algorithmes de type « Tester-et-générer »

5.1.1. Algorithme U-Apriori

U-Apriori est la version incertaine d'Apriori algorithme. Pour faire face au problème des itemsets fréquentes dans les données incertaines. Apriori a été modifié à l'algorithme U-Apriori. R. Agrawa été le premier qui avait proposé cette Apriori algorithme **[06].**

Étape-1: Scannez la base de données de transaction pour obtenir le support S de chaque 1-itemset, comparer avec Soutien minimum (MS), et d'obtenir un ensemble de fréquentes 1-itemsets,L1.

Etape-2: Utiliser LK-1. Rejoignez-Lc1 pour générer un ensemble de candidats k objet ensembles. Et utiliser la propriété Apriori à tailler les ensembles non fréquentés k-point de cet ensemble.

Étape-3: Scannez la base de données de transaction pour obtenir le support S de chaque k-point de l'ensemble candidat dans le dernier set, comparer S avec MS, et obtient un ensemble de fréquentes k-point ensembles, Lc.

Étape-4: Pour chaque article fréquente ensemble l, générer tous sous-ensemble non vides de l.

Étape-5: Pour toute partie non vides de l, la sortie prononcer "s =>(l-s)" si la confiance C de la règle, "S =>(l-l)".

Pseudo Code d'Algorithme U-Apriori :

Algorithme 2.1 : Algorithme U-Apriori

Entrée :context β ; seuil minimal de support minsupport ;

Sortie : ensembles F_k des k-itemsets fréquents ;

1) $F_1 \leftarrow$ {1-itemsets frequents} ;
2) Pour ($k \leftarrow 2$; $F_{k-1} \neq \emptyset$; k++) faire
3) $C_k \leftarrow$Apriori-Gen (F_{k-1}) ;
4) Pour chaque objet o faire ;
5) $C_0 \leftarrow$Subset (C_k, o);
6) Pour chaque candidat c_k , $c_{k+1} \in C_0$et c_k x $c_{k+1} \geq$ SM
7) Faire c.support++ ;
8) Fin pour
9) $F_k \leftarrow \{c \in C_k \mid c.support \geq minisupport\}$;
10) Fin pour
11) Retourner $C_k F_k$;

C_k : Ensemble de *k*-itemsets candidats (itemsets potentiellement fréquents). Chaque élément de cet ensemble possède deux champs : *itemset* et *support*.

F_k : Ensemble de *k*-itemsets fréquents. Chaque élément de cet ensemble possède deux champs : *itemset* et *support*.

5.2. Les algorithmes de type « diviser-et-régner »

5.2.1. L'algorithme UH-mine

L'algorithme UH-Mine fonctionne comme suit :

Étape -1: Elaguer la base de données initiale pour supprimer les itemsets non fréquents.

Etape -2: Diviser la base de données élaguée en égale Morceaux.

Étape -3:Utiliser l'Algorithme UH-Mine pour extraire les itemsets fréquents, UH-Mine maintient une H-Struct, qui contient des pointeurs aux éléments de transaction.

Étape -4: Jointer les résultats.

Étape -5: Scanner la base de données élaguée une autre fois pour éliminer les faux positifs et obtenir les chiffres réels.

Voilà un schéma illustratif (voir le figure 1.1) pour bien comprendre le UH-Mine

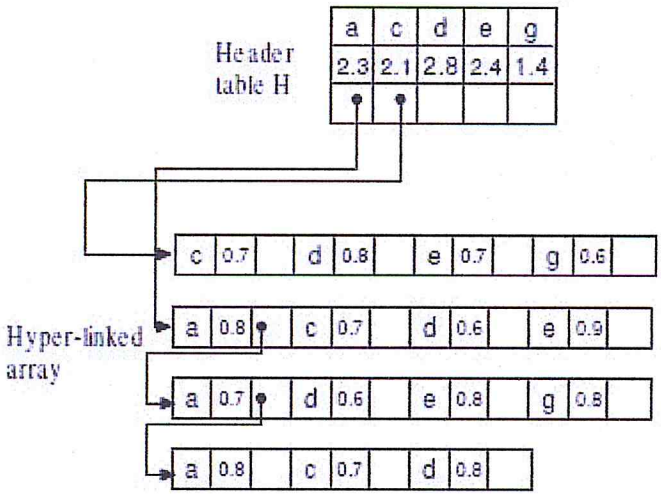


Figure 1.1 : La structure L'algorithme UH-Mine **[07]**

5.2.2. L'Algorithme UF-growth

Pour représenter les données incertaines efficacement, on a proposé l'UF-arbre qu'il est une variante de FP-arbre. Chaque nœud dans l'UF-arbre stocker un item, son support attendu et le nombre d'occurrence de tel support pour tel item.

L'algorithme proposé construit UF-arbre comme suit :

Entré : une base de données, un minimal de support minsup.

Sortie : UF-arbre, arbre des itemsets fréquents.

Etape 1 : scanner la base de données et accumuler les supports attendus de chaque item pour trouver les items fréquents où le support attendu ≥ minsup (minimal support).

Etape 2 : trier ces items fréquents en ordre décroissant des supports attendus accumulés.

Etape 3 : scanner la base de donnés une autre fois et insérer chaque transaction dans l'UF-arbre, de la même façon que dans la construction de FP-arbre, sauf pour le suivant :

Fusionner une nouvelle transaction avec le nœud fils de la racine d'UF-arbre si le même item et le même support attendu existe dans la transaction et le nœud fils.

Avec un tel processus de construction de l'arbre, UF-arbre possède une bonne propriété que le nombre d'occurrences d'un nœud est au moins la somme des chiffres d'occurrence de tous ses nœuds fils **[08].**

5.2.3. Algorithm UFP-growth

UFP- growth extend algorithm initial FP- growth. La construction de FP-arbre a besoin deux analyses de l'ensemble de données. La première analyse collecte les items fréquents et ses supports. Et dans la deuxième analyse, chaque transaction est logée dans la structure de FP-arbre. Les Itemsets fréquents sont générés de manière récursive de FP-arbre. Afin d'adapter cet algorithme :

Première analyse :

- Calculer le support attendu exacte de chaque ensemble des items par le calcul de leur support à la somme des probabilités existantes dans chaque transaction

Deuxième analyse :

- Chaque transaction est instanciée n fois, selon la probabilité existante des items dans la transaction.
- Insérer les itemsets dans la structure de FP-arbre.
- L'algorithme extrait les itemsets fréquents de la même manière que l'algorithme FP-growth.

5.2.4. L'Algorithme PUF-growth

Pour réduire la taille de l'UF-arbre et UFP-arbre, Leung a proposé la structure d'arbre des itemsets fréquents incertains préfixe capé (PUF-arbre), dans lequel les informations importantes sur les données incertaines est capturé afin qu'on extrait les itemsets incertains à partir de l'arbre. Le PUF-arbre est construit en tenant compte de la limite supérieure de la valeur de probabilité existante pour chaque item quand générer un k-itemset (k>1).

La limite supérieure d'un item x_r dans une transaction t_j est l'item plafond (préfixe) de x_r dans t_j, tel que défini ci-dessous,

Définition : L'item plafond (préfixe) Icap (x_r, t_j) d'un item x_r dans une transaction $t_j = x_1, ..., x_r,..., X_h\}$, où 1 <r <h, est défini comme un produit de P (x_r, t_j) et la plus haute valeur de probabilité existante M des items de x_1 à x_{r-1} dans tj (ie, dans le préfixe propre de x_r dans t_j):

$$I^{c} \quad (x_r, t_j) = \begin{cases} P(x_r, t_j) \times M & h > 1 \\ P(x_r, t_j) & i.\ h = 1 \end{cases} \quad w \qquad M = m \quad {}_{1 \le q \le r-1} P(x_q, t_j) \quad (2)$$

Le plafond de support attendu expSupcap(X) d'un motif X = {x 1, ..., x k} (où k> 1) est définie comme la somme de tous les items plafonds de x_k dans toutes les transactions qui contiennent X:

$$e \qquad {}^{c} \quad (X) = \sum_{j=1}^{n} \{(I^{c} \quad (x_k, t_j) | X C t_j)\} \qquad (3)$$

Dans des autres termes, le plafond de support attendu d'un itemset satisfait à la propriété de fermeture vers le bas(voir le tableau 1.6).

Exemple :

Tableau 1.6 : Une base de données de transactions avec minsup = 0,5.

TID	Transactions	Transactions tries
T1	{: 0.2, : 0.2, *c*: 0.7, *f* :0.8}	{:0. 2, : 0.7, *f*: 0.8}
T2	{: 0.5, : 0.9, *e*: 0.5}	{: 0.5, : 0.9, *e*: 0.5}
T3	{: 0.3, : 0.5, *e*: 0.4,}	{*a* :0.3, : 0.4, *f*: 0.5,}
T4	{: 0.9, : 0.2, *d*: 0.1}	{*a* :0.9, : 0.5, *d*: 0.1}

Considérons une base de données incertaine avec quatre transactions présentées dans la deuxième colonne du tableau 2. L'item plafond de c dans t1 peut calculer comme suit :

$I^{cap}(c,t_1) = 0.7 \times \max\{P(a, t_1), P(b, t_1)\} = 0.7 \times \max\{0.2,0.2\} = 0.7 \times 0.2 = 0.14$

$I^{cap}(f,t1) = 0.8 \times \max\{P(a, t_1), P(b, t_1), P(c, t_1)\} = 0.8 \times \max\{0.2,0.2,0.7\} = 0.8 \times 0.7 = 0.56$

Etapes de construction de PUF-arbre

- Scanner la base de données et trouver les itemsets fréquents distincts dans la base de données
- Construire un principal tableau I-liste pour stocker les itemsets fréquents dans un ordre cohérant (par exemple l'ordre canonique) pour faciliter la construction de l'arbre.
- Construire PUF-arbre avec la deuxième analyse de la base de données de la même façon que FP-arbre **[09].**

- Lors de l'insertion d'un item de transaction, d'abord calculer son item plafond puis l'insérer dans le PUF arbre selon l'ordre d'I-liste.
- Si ce nœud existe déjà dans le chemin, mettre à jour son item plafond en ajoutant l'item plafond calculé à l'item plafond existant.

6. Comparaison

Nous allons faire une petite synthèse comparative en ce qui concerne les avantages et les inconvénients de quelques algorithme cités au paravent dans ce chapitre.

Les algorithmes basés sur FP-growth tel que UF-growth, UFP-growth et PUF-growth possède un très grand avantage en ce qui concerne le balayage de la base de données, car ils utilisant la structure d'arbre qui est très efficace ou il scanne la base de données dans deux itérations seulement pour extraire les itemsets fréquents. Cependant l'inconvénient de ces algorithmes est qu'ils contiennent un chemin d'arbre distinct pour chaque élément distinct est représenté par la paire (item, probabilité).

L'algorithme U-Apriori basé sur Apriori cet algorithme permet de réduire la taille de l'ensemble des candidats, cependant l'inconvénient de cet algorithme est le scanne de base de données plusieurs fois.

7. Support

Si on peut classifier les algorithmes d'extraction des itemsets fréquents à partir des données incertaines selon les mesures pour calculer les itemsets fréquents, il y a deux types des algorithmes :

1. Les premiers utilisent *l'Expected support.*
2. Les deuxièmes utilisent le *Probabiliste Support*

7.1. Expected support

L'itemset X est un fréquent si son Expected support supérieur ou égale le seuil minsup.

$$e \qquad (X) \geq m \qquad (4)$$

Définition :

Expected support d'un itemests X dans la base de données incertaine est la somme de produit probabilité P (X, t j) de X dans la transaction t j sur tous les n transactions dans la base de données. **[10]**

Donc l'expression de la fonction Expected

Support est présenté comme suit :

$$expSup(X) = \sum_{j=1}^{n} P(X, t_j) = \sum_{j=1}^{n} \left(\prod_{x \in X} P(x, t_j) \right) \qquad (5)$$

Lorsque des éléments x ∈ X dans chaque transaction t j sont indépendants.

7.2. Probabiliste Support

Un itemset X est un fréquent probabiliste si son existence dans une transaction *minsup* est supérieure ou égale le seuil d'utilisateur spécifique *minprob*

$$\boldsymbol{P(sup(X) \geq minsup) \geq minprob} \qquad (6)$$

Dans les bases de données de transactions incertaines, le support d'un ou plusieurs éléments ne peut pas être représenté par une valeur unique, mais plutôt, doit être représenté par une distribution de probabilité discrète. **[11]**

Soit T est la base de données de transaction et W est l'ensemble des éléments possibles de T, le $P_i(X)$ est la probabilité de support d'un itemset X tel que X a le soutien i.

$$P_i(X) = \sum_{w_j \in W, (S(X, w_j) = i)} P(w_j) \qquad (7)$$

Où S (X;wj) est le support de X dans un élément wj.

Le support probabiliste d'un itemset X dans une base de données de transaction incertaine T est donné par les probabilités de support de X (P_i (X)) pour toutes les valeurs possibles de soutien i ∈{0,….., |T|}.

$$\sum_{0 \leq i \leq |T|} P_i(X) = 1 \qquad (8)$$

Soit I l'ensemble de tous éléments possibles et T représente la base de données incertaines, où une transaction tj∈ T est un ensemble des éléments incertains, à savoir, tj⊆ I. Contrairement à la base de données précise traditionnelle, chaque item xi ∈tj est associée à une probabilité existentielle P (xi, tj) ∈ (0,1], ce qui dénote la probabilité que xi est réellement présent dans tj. Pour un itemset X ⊆tj, basée sur l'hypothèse commune que les éléments de X

sont indépendante la probabilité existentielle P (X ⊆tj) =∏ x∈XP (x, tj). le soutien attendu (expected support) de X est alors la somme de la probabilité existentielle sur toutes les transactions.

$$expSup\ (X) = \sum tj \in TP\ (X \subseteq tj) \qquad (9)$$

La probabilité fréquente P(X) de X peut être calculé par sommer Pi(X) pour tout i ≥ minsup :

$$P(X) = \sum_{i=m}^{|T|} P_i(X) \qquad (1\ )$$

Où Pi (X) enregistre la probabilité de X qui se produit exactement dans une transaction:

$$P_i(X) = \sum_{\substack{A \subseteq T \\ |A| = i}} \left(\prod_{t \in A} P(X \subseteq t) \prod_{t \in T - A} (1 - P(X \subseteq t)) \right) \qquad (11)$$

Si P (X) ≥ minprob, alors X est un itemset fréquent probabiliste.

On donne (i) une base de données de transaction incertaine T, (ii) un seuil minimum de support minsup et (iii) un seuil minimum de la probabilité fréquente min prob, le problème d'extraction de Itemsets fréquents probabilistes est de trouver tous et seuls les Itemsets fréquents probabilistes ; à savoir, trouver tous les X tel que P (X) ≥ minprob.

8. Conclusion

Dans ce chapitre, l'étude globale des algorithmes classiques d'extraction d'itemsets fréquents de données incertaines est faite et a identifié des nombreuses forces et des faiblesses de chacun, tel que leurs complexités algorithmiques qui est d'ordre NP-complet ce qui signifie on doit balayer tout l'espace de recherche.

Cette comparaison peut aussi tomber dans diverses questions d'optimisation qui conduiront à une meilleure performance. L'efficacité des algorithmes d'extraction est plus obstacle, mais encore il est nécessaire de développer des méthodes pour obtenir d'excellents résultats. Pour cela nous devons utiliser les techniques d'optimisations (méta-heuristique) pour l'extraction des itemsets fréquents.

Chapitre 02 :

Mobile frequent pattern Mining

1. Introduction

Récemment, il y a quelques systèmes génériques d'exploration de données sont introduits pour utiliser les Smartphones comme plate-forme de calcul. Il s'agit notamment de Pocket Data Mining (PDM), Open Mobile Miner (OMM) et Mobile WEKA, parmi les plus populaires. Ces outils prennent en charge une variété d'algorithmes de classification et de clustering, mais la recherche sur les algorithmes d'exploration de itemsets fréquents (Mobile Fréquent pattern Mining MFPM) est encore à ses débuts.

Ce chapitre contribue en étudiant la faisabilité d'exploiter les algorithmes FPM dans les appareils mobiles et les travaux reliés de comprendre les besoins fondamentaux et les problèmes liés aux algorithmes étudiés.

2. Les Travaux Reliés au FPM dans le mobile

L'extraction de itemsets fréquents varie de modèles de base aux modèles multi-niveaux et multidimensionnels aux modèles étendus pour les ensembles et flux de données. Or, l'étude approfondie de la littérature montre que la recherche dans le contexte d'extraction de itemsets fréquents en utilisant des appareils mobiles est encore au début.

Selon les recherches et l'étude de la littérature dédiée à cet axe de recherche qu'on a fait, on a constaté que quelques travaux intéressants ont été proposés récemment. Les deux premiers travaux appliquent l'extraction de itemsets fréquents dans le domaine de commerce mobile et la reconnaissance d'activité. Le troisième travail compare les performances de quelques algorithmes d'extraction des itemsets fréquents dans des environnements desktop et mobiles.

Le travail proposé par Lu, E-C et al [12] présente un Framework appelé Personale Mobile Commerce Pattern (PMCP-Mine) utilisé pour découvrir les modèles de magasinage personnel des utilisateurs mobiles dans les environnements m-commerce. PMCP-Mine lance d'abord les transactions mobiles fréquentes à partir des données d'achat locales de l'utilisateur, puis met à jour la base de données des transactions locales en supprimant les transactions peu fréquentes.

PMCP-Mine prévoit de nouveaux modèles de transaction sur la base de la nouvelle base de données de transactions locales. L'analyse de performance de PMCP-Mine montre que le temps d'exécution est incrémental avec la diminution de la valeur de seuil de support.

Le travail proposé par Wang, L et al [13] a présenté une technique d'exploration de données basée sur Emerging Patterns (EP) dans un système de reconnaissance d'activité complexe qui fonctionne à deux niveaux. Dans la première couche, les données sont traitées aux nœuds BSN puis transmises à un appareil mobile pour un traitement ultérieur. Au niveau des nœuds, des algorithmes légers sont utilisés pour les reconnaissances gestuelles. De plus, des algorithmes de reconnaissance en temps réel basés sur des itemsets sont utilisés dans un appareil portable central. EP représente un ensemble des items fréquents dans une classe et les rares dans une autre classe.

La notion derrière la technique basée sur EP est que les instances contenant des articles EP appartiennent très probablement à la classe EP correspondante. L'analyse de complexité de l'algorithme proposé montre que la complexité temporelle des éléments EP correspondants avec des éléments stockés dans la classe est O ((m.l + k) .n) où n est la longueur du vecteur d'entrée, k représente l'activité totale, m Indique le nombre de EP et l montre le nombre moyen d'articles dans un EP. D'autre part, la complexité de l'espace est (m.l) pour contenir les EP. L'analyse de performance rapporte que la précision de reconnaissance moyenne est de 82,87%, le délai moyen de reconnaissance est de 5,7 périodes de détection et l'utilité moyenne est de 0,81 sur (0-1). En dépit de la haute détection de miss et du taux de détection fausse, l'algorithme proposé fonctionne mieux que les algorithmes basés sur une seule couche et HMM.

Des analyses de faisabilité réalisées sur six algorithmes basiques d'extraction des itemsets fréquents sont présentées par Mohamed,H et al [14]. Ces algorithmes comprennent Apriori et AprioriTid proposés par Agrawal, R et al [15], FP-Growth proposée par Han, J et al [16], Relim [17], Eclat proposée par Zaki,M.J [18], et dEclat proposée par Zaki,M.J et al [19]. Les auteurs ont testé les six différents algorithmes sur trois différents types de base de données au but de connaitre la fiabilité de ses algorithmes dans des environnements mobiles.

A. Apriori et AprioriTid

Les algorithmes Apriori et AprioriTid sont utilisés pour l'extraction des règles d'association qui est un processus en deux étapes :

1) la recherche de tous les itemsets, Appelé grands itemsets, ayant minsup

2) utilisant des grands Itemsets pour la génération de règles d'association. Le processus est : ∀ non vide subset de grand Itemset a, rechercher les subsets non-vides d'un a et ∀ subset non vide b, en sortie (règle): b →(a-b) si le ratio support (a) / support (b) est au moins minconf. Apriori et AprioriTid sont multi-passe Algorithmes où les items candidats satisfaisant le minimum sont trouvés Initialement et les itemsets candidats sont générés de manière itérative distraitement des items candidats. Cette procédure itérative se poursuit Jusqu'à ce qu'il n'y ait pas des grands itemsets trouvés. Compte tenu du subset d'un grand itemset d'éléments est également grand, aide à joindre de grand a itemset avec les éléments k-1 et la suppression de ces subsets sans grands itemsets Et formant de grand itemsets avec k-items.

Alternativement, AprioriTid est différent d'Apriori en accéder à la base de données pour une fois en comptant le support minimum Après la première passe. Les items d'items candidats sont encodés après la première Passer pour réduire la taille des données et les efforts de lecture plus tard Passent.

L'algorithme Apriori fonctionne en comptant les éléments pour déterminer Grand 1-itemsets. Par la suite, il fonctionne en deux phases. Pour exemple pour le passe k, la fonction Apriori-gen est utilisée pour générer Les objets d'objets candidats Ck, qui fonctionnent sur l'entrée de (k-1) th passe En utilisant un grand ensemble d'éléments de k-1, c'est-à-dire Lk-1. Ensuite, le support de Ck est Compté après une analyse de base de données. Ici Ck d'une transaction est Devait être déterminé efficacement pour le comptage rapide. Un sous-ensemble La fonction basée sur hash-tree est utilisé pour stocker Ck. Où feuille Les nœuds de l'arbre contiennent la liste des articles et les magasins de nœuds internes Tables de hachage. Le coût de garder l'arbre entier dans la mémoire est Réduit en comptant Ck + 1 au kth passe. Cette stratégie fonctionne quand le coût de la base de données de balayage est plus que le coût de la conservation Mémoire et comptage de Ck + 1 supplémentaires Ck + 1 candidats.

De même, AprioriTid génère également $\overline{Ck}$ en utilisant Apriori-gen fonction avant que le passage commence, mais le support est compté sans accès à la base de données. L'ensemble $\overline{Ck}$ contient les éléments de La forme <Tid, [4]> où chaque Ik représente un grand k-itemet avec identificateur de transaction (Tid).

B. FP-Growth

Le coût de calcul croissant de la génération de candidats Dans de gros objets, particulièrement avec de longs modèles, Développement des algorithmes FP-Growth. L'algorithme est basé Sur l'arbre à itemset fréquent (FP) pour stocker des informations importantes dans Forme comprimée, à propos de itemsets fréquents. FP-Growth L'algorithme fonctionne efficacement en raison de la stratégie en trois étapes :

1) La base de données est compressée et stockée dans un arbre FP pour éviter les multiples scans de base de données au passage postérieur

2) une croissance fréquente est utilisée pour réduire le coût de la génération de candidats dans Des modèles longs

3) une approche diviser-n-conquérir est utilisée pour limiter l'espace de recherche et réduire le coût du niveau recherche utilisée dans les algorithmes Apriori.

Le résultat était après l'analyse détaillée de trois types de base de données, que les téléphones mobiles pourraient être utilisés comme plate-forme d'exploration de données après avoir réglé certains paramètres de base. Les paramètres pourraient être en cours d'analyse de données fragmentées, En initiant l'approche périgordienne de l'exploration de données ou en utilisant des informations contextuelles pour utiliser les ressources disponibles pour le téléphone portable.

C. Relim

L'algorithme Relim (élimination récursive) a été inspiré par FP-Growth mais fonctionne sans arbres préfixés. Ici, les éléments sont comptés et comparés à la valeur minimale donnée par l'utilisateur au début, puis disposés dans un ordre croissant pour un traitement rapide. Il est noté que le noyau de Relim est une fonction récursive qui a moins de coût de calcul que les algorithmes Eclat [19] et Apriori. La fonction récursive fonctionne en éliminant les éléments peu fréquents des transactions et en sélectionnant les transactions contenant les items les moins fréquents.

Par la suite, les items les moins fréquents sont supprimés des transactions sélectionnées et la procédure commence à nouveau jusqu'à ce qu'il ne reste que de grands itemsets fréquents.

D. Eclat et dEclat

La limitation des analyses multiples de bases de données et la complexité de Structures de données internes, comme les arbres en Apriori et FP-Growth, conduit au développement d'algorithmes Eclat / dEclat. Les algorithmes fonctionnent avec moins de balayages de base de données et utilisent efficacement Technique pour explorer l'espace de recherche.

Les algorithmes Eclat / dEclat sont basés sur la liste vertical-tid Format de base de données où chaque élément est associé à son Conclure des transactions. Cette stratégie d'inscription a aidé à Énumérant tous les items fréquents. En plus de cela, un treillis Et l'approche basée sur le réseau a été utilisée pour décomposer la (lattice) ensembles des items et parcourir dans (sub-lattices) subsets, entièrement dans mémoire principale. En outre, Eclat / dEclat a utilisé l'arbre ascendant traversal comme stratégie de recherche pour les énumérations de sub-lattices.

L'adoption de schémas de recherche efficaces a permis de Consomme moins de ressources informatiques et des analyses de base de données minimales Par rapport à Apriori, donc éclat / dEclat gagner de manière significative reconnaissance. En outre, le schéma Tid-list est également très utile pour réduire les écarts de données et conserver une évolutivité linéaire dans base de données en termes de compte de transaction. La différence entre Eclat et dEclat sont leur manipulation avec de grandes listes Tid verticales tout en générant des résultats intermédiaires. Ici, dEclat utilisé diffsets qui ne comptent que des différences d'ID de transaction (Tids) de modèles de candidats. Par conséquent, les petits subsets font l'intersection très rapide et améliore la performance globale de l'algorithme comme par rapport à Eclat. Cette approche de stockage des différences dans Tidsets plutôt que de stocker des classes entières d'itemsets et leurs Tidlists a permis de contrer le problème d'évolutivité associé à Éclat.

Pour l'analyse de données fragmentées, la modélisation à base de fenêtre coulissante pourrait être utilisée pour exploiter les données dans des fenêtres temporelles petites et gérables afin que les données maximales puissent être exploitées en temps réel. En outre, les tâches d'exploration de données pourraient être lancées après une période de temps fixe afin de restreindre la taille des ensembles de données. De même, des informations contextuelles sur l'état de la batterie et les informations sur le temps de repos peuvent également être adoptées pour planifier les tâches d'exploration de données. Nous espérons que ces recommandations peuvent être utiles dans de nouvelles recherches sur l'exploration de données mobiles.

Le travail présenté par Lin,Yi et al [20] propose evergreen FP-tree (EvFPTree), une technologie durable, efficace et non volatile (NVM) de la structure de données l'arbre connue (arbre FP). Dans EvFP-tree, ils ont proposé un compteur paresseux avec une stratégie de construction en deux phases pour réduire les écritures NVM afin d'améliorer les performances et l'efficacité énergétique dans la construction d'un arbre FP. Une telle idée d'exploiter l'information en vrac d'entrées multiples d'un algorithme hors ligne pourrait également être étendue pour l'optimisation de la performance d'autres algorithmes en ligne. De plus, ils ont proposé un schéma d'encodage à modulation minimale (MBA) pour égaliser le port de différents bits dans le même compteur de support de nœuds d'arbres FP pour prolonger la durée de vie NVM. Le schéma de MBA pourrait effectivement accompagner les techniques de nivellement d'usure existantes qui égalisent les accès à travers différents mots NVM. Comme cela a été vérifié par les résultats expérimentaux, EvFP-tree surpasse l'algorithme classique FPgrowth de 40,28% en moyenne en termes de performance minière et étend la durée de vie NVM de 87,20% en moyenne.

Et EvFP-tree réduit la consommation d'énergie de 50,30% en moyenne. À l'avenir, l'extension des stratégies proposées à la mémoire flash pour permettre des données performantes reste une perspective.

Selon Srinivasan Vijay et al **[21]** le nouveau système Mobile Miner pour l'extraction phénomènes fréquentes de cooccurrence sur le téléphone indiquant les événements contextuels qui se produisent souvent ensemble.

En utilisant des données de contexte longitudinales collectées auprès de 106 utilisateurs de plus de 1 à 3 mois, ils ont montré que Mobile Miner génère efficacement des modèles en utilisant des ressources téléphoniques limitées, ce qui permet une amélioration de la performance de 15 fois par rapport à l'algorithme minier Apriori et génère des modèles globaux fréquents en 16 minutes et une application détaillée Itemsets d'utilisation en 21 secondes. Ils ont trouvé des modèles de comportement intéressants pour les utilisateurs individuels et les utilisateurs, allant des modèles d'appel pour placer les modèles de visite.

Enfin, ils ont montré comment leurs modèles de cooccurrence peuvent être utilisés pour améliorer l'interface utilisateur du téléphone pour lancer des applications ou appeler des contacts, en prévoyant la prochaine application ou le contact invoqué en fonction des modèles Mobile Miner ; Ils ont conclu avec un sondage auprès des utilisateurs qui a montré la promesse de leur service de prévision de l'application.

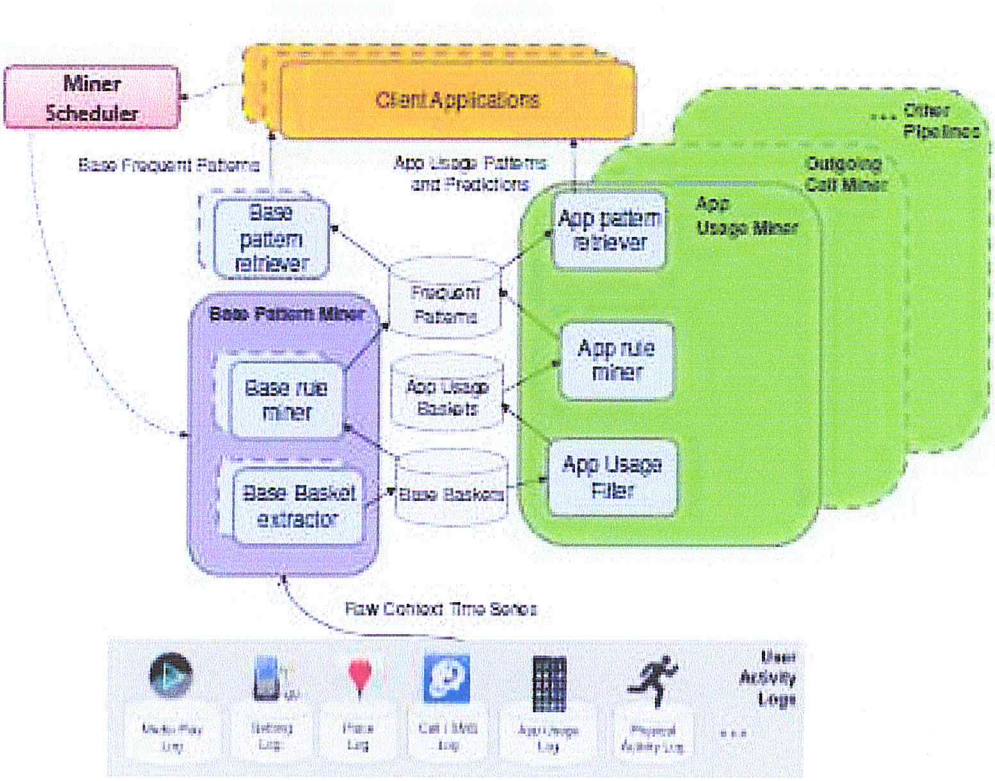


Figure 2.1 : Architecture de système de Mobile Miner. **[21]**

Les auteurs du travail présenté par Evangeline et al [22] ont créé un Framework s'appelle le commerce mobile, est utilisé pour l'exploitation minière et la prédiction du comportement des utilisateurs mobiles tels que leurs mouvements et leurs transactions d'achat. Il est recommandé de recommander les magasins et les objets à l'utilisateur inconnu. Lorsque les utilisateurs mobiles se déplacent entre les magasins, le système prédit que les éléments fréquemment déplacés pour l'utilisateur en fonction de leur sélection, qui comprend les informations mobiles telles que l'identification des utilisateurs, les magasins et les articles achetés, sont stockés dans la base de données des transactions mobiles. En ce sens, un nouvel algorithme à base d'arbre systolique est efficace pour exploiter les ensembles d'objets fréquents. Il produit de meilleurs résultats dans les performances que les autres algorithmes.

3. Conclusion

Après avoir fait un balayage dans la littérature, on a remarqué que les travaux qui existent n'atteignent pas tous l'objectif attendu. Cependant les travails proposés traitent l'étude de faisabilité seulement.

Chapitre 03 :

Meta-heuristiques et Algorithmes évolutionnaires

1. Introduction

L'optimisation combinatoire est un outil indispensable combinant diverses techniques de la mathématique discrète et de l'informatique afin de résoudre des problèmes d'optimisation combinatoire de la vie réelle. Un problème d'optimisation combinatoire consiste à trouver la meilleure solution dans un ensemble discret de solutions appelé ensemble des solutions réalisables. En général, cet ensemble est fini mais de cardinalité très grande.

Il s'agit, en général, de maximiser (problème de maximisation) ou de minimiser (problème de minimisation) une fonction objective sous certaines contraintes. Le but est de trouver une solution optimale dans un temps d'exécution raisonnable.

Dans ce chapitre nous présentons certaines méthodes parmi les plus représentatives, développées pour résoudre les problèmes d'optimisation combinatoire.

2. Problème du L'optimisation combinatoire

2.1.Notion

On qualifie généralement de « combinatoires » les problèmes dont la résolution se ramène à l'examen d'un nombre fini de combinaisons (appelée aussi solutions). Bien souvent cette résolution se heurte à une explosion du nombre de combinaisons à explorer.

En mathématique, l'optimisation combinatoire recouvre toutes les méthodes qui permettent de déterminer l'optimum d'une fonction avec ou sans contraintes

2.2. La théorie de la complexité

La théorie de la complexité consiste à estimer la difficulté ou la complexité d'une solution algorithmique d'un problème posé de façon mathématique. Elle se concentre sur les problèmes de décision qui posent la question de l'existence d'une solution comme le problème de satisfiabilité booléenne.

La théorie de la complexité repose sur la notion de classes de complexité qui permettent de classer les problèmes en fonction de la complexité des algorithmes utilisés pour les résoudre. Parmi les classes les plus courantes, on distingue : la classe P (Polynomial time) qui englobe les problèmes pour lesquels il existe un algorithme déterministe de résolution en temps polynomial, et la classe NP (Non deterministic Polynomial time) qui contient des problèmes de décision pour lesquels la réponse oui peut être décidée par un algorithme non-déterministe en un temps polynomial par rapport à la taille de l'instance. Les problèmes NP-complets sont définis comme Un problème de décision π est NP-complet s'il satisfait les deux conditions suivantes : $\pi \in$ NP, et tout problème NP se réduit à π en temps polynomial.

2.3. Les Problèmes d'optimisation combinatoire

Un Problème d'Optimisation Combinatoire (POC) peut être définie comme suit :

- Un ensemble de combinaisons ou de solutions X.
- Un ensemble de contraintes C (éventuellement vide).
- Un sous-ensemble S de X représentant les solutions admissibles (ou réalisables) qui vérifient les contraintes C.
- Une fonction de coût f (fonctions objectives) qui assigne à chaque solution $s \in S$ une valeur f (s).
- Il s'agit de trouver une solution optimale (ou optimum global) $s^* \in S$ qui optimise (minimise ou maximise) la fonction de coût f.

$$f(s^*) \leq f(s_i), \forall s_i \in S \qquad (12)$$

Outre la classification selon le nombre de critère à optimiser, les méthodes de l'optimisation combinatoire peuvent être classées aussi en méthodes heuristiques et méthodes exactes (voir figure 3.1).

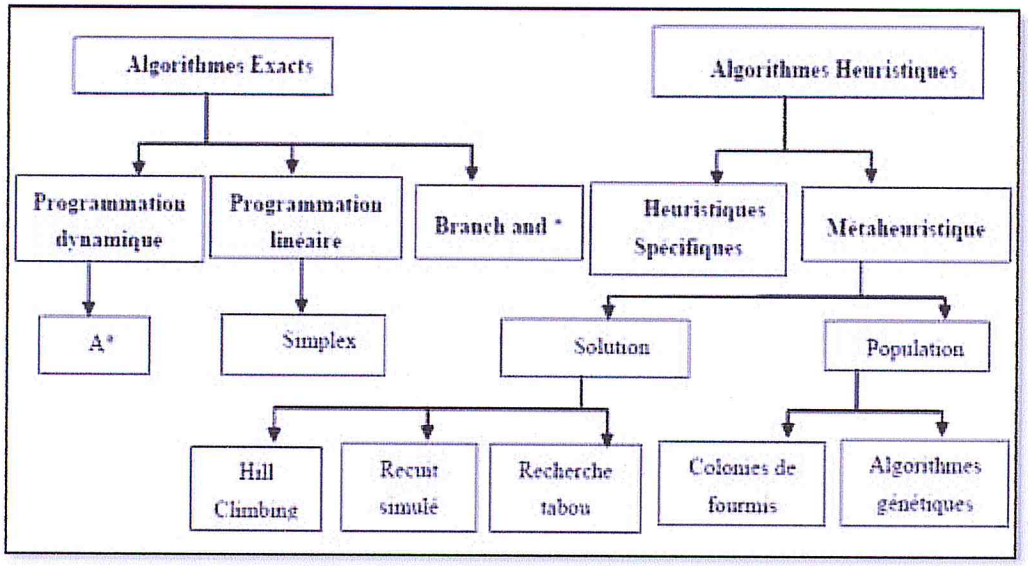


Figure 3.1 : classification des méthodes d'optimisations combinatoires.

2.4. Résolution d'un problème d'optimisation combinatoire

Résoudre un problème d'optimisation combinatoire nécessite l'étude de trois points particuliers :

- La définition de l'ensemble des solutions réalisables.
- L'expression de l'objectif à optimiser.
- Le choix de la méthode d'optimisation à utiliser.

Les deux premiers points relèvent de la modélisation du problème, le troisième de sa résolution. Afin de définir l'ensemble des solutions réalisables, il est nécessaire d'exprimer l'ensemble des contraintes du problème. Ceci ne peut être fait qu'avec une bonne connaissance du problème sous étude et de son domaine d'application.

3. Les méthodes d'optimisation combinatoire

Depuis tous ce qu'on a vus précédemment dans ce chapitre on peut classifier les méthodes d'optimisation combinatoire en deux catégories :

- Des méthodes exactes
- Des méthodes approches

3.1. Les méthodes exactes

Elles se basent généralement sur une recherche complète de l'espace des combinaisons afin de trouver une solution optimale

- Les méthodes de recherche arborescente (Branch & bound).
- La programmation linéaire.
- La programmation dynamique.

3.1.1. Méthode de branch and bound

La méthode de branch and bound (procédure par évaluation et séparation progressive) consiste à énumérer ces solutions d'une manière intelligente en ce sens que, en utilisant certaines propriétés du problème en question, cette technique arrive à éliminer des solutions partielles qui ne mènent pas à la solution que l'on recherche. De ce fait, on arrive souvent à obtenir la solution recherchée en des temps raisonnables. Bien entendu, dans le pire cas, on retombe toujours sur l'élimination explicite de toutes les solutions du problème.

Pour ce faire, cette méthode se dote d'une fonction qui permet de mettre une borne sur certaines solutions pour soit les exclure soit les maintenir comme des solutions potentielles. Bien entendu, La performance d'une méthode de branch and bound dépend, entre autres, de la qualité de cette fonction (de sa capacité d'exclure des solutions partielles tôt).

Pseudo code :

Début
Placer le nœud début de longueur *0* dans une liste.
Répéter
Si la première branche contient le nœud recherché **alors**
Fin avec succès.
Sinon
- Supprimer la branche de la liste et former des branches nouvelles en étendant la branche supprimée d'une étape.
- Calculer les coûts cumulés des branches et les ajouter dans la liste de telle sorte que la liste soit triée en ordre croissant.

Jusqu'à (liste vide ou nœud recherché trouvé)
Fin

3.1.2. Programmation Dynamique

La programmation dynamique a été appelée comme cela depuis 1940 par Richard Bellman et permet d'appréhender un problème de façon différente de celle que l'on pourrait imaginer au premier abord. Le concept de base est simple : une solution optimale est la somme de sous-problèmes résolus de façon optimale. Il faut donc diviser un problème donné en sous-problèmes et les résoudre un par un.

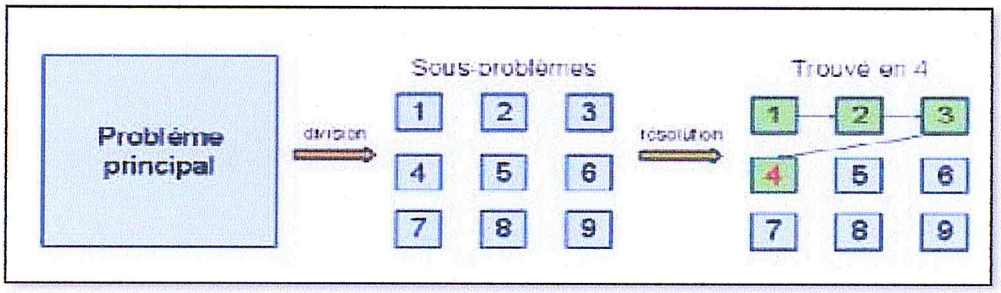


Figure 3.2 : Divisé en sous-problèmes.

- **Algorithme général de programmation dynamique**

La conception d'un algorithme de programmation dynamique peut être planifiée dans une Séquence de quatre étapes.

1. Caractériser la structure d'une solution optimale.

2. Définir récursivement la valeur d'une solution optimale.

3. Calculer la valeur d'une solution optimale en remontant progressivement jusqu'à l'énoncé du problème initial.

4. Construire une solution optimale pour les informations calculées.

3.2. Les méthodes approchées

Une méthode heuristique ou approchée est une méthode d'optimisation qui a pour but de trouver une solution réalisable de la fonction objective en un temps raisonnable, mais sans garantie d'optimalité.

L'avantage principal de ces méthodes est qu'elles peuvent s'appliquer à n'importe quelle classe de problèmes, faciles ou très difficiles, D'un autre côté les algorithmes d'optimisation tels que les algorithmes de recuit simulé.

Elles englobent deux classes : Heuristiques & Méta-heuristiques

3.2.1. Heuristique

Une méthode heuristique (ou simplement une heuristique) est une méthode qui aide à découvrir la solution d'un problème en faisant des conjectures plausibles mais faillible de ce qui est la meilleure chose à faire. **[23]**

Les heuristiques sont des critères, des méthodes ou des principes pour décider qui, parmi plusieurs d'autres plans d'action permet d'être le plus efficace pour atteindre un certain but. **[24]**

Les méthodes heuristiques sont spécifiques à un problème donné. Elles nécessitent des connaissances du domaine du problème traité. En fait, ce sont des règles empiriques qui se basent sur l'expérience et les résultats acquis afin d'améliorer les recherches futures.

3.2.2. Méta-Heuristique

Le mot Méta-Heuristique est dérivé de la composition de deux mots grecs :

- Heuristique qui vient du verbe heuriskein et qui signifie 'trouver'
- Meta qui est un suffixe signifiant 'au-delà', 'dans un niveau supérieur'.

Une ***méta-heuristique*** est un processus itératif qui subordonne et qui d'une heuristique en combinant intelligemment plusieurs concepts pour explorer et exploiter tout l'espace de recherche. Des stratégies d'apprentissage sont utilisées pour structurer l'information afin de trouver efficacement des solutions optimales ou presque optimales. **[25]**

Les méta-heuristiques qui se subdivisent en deux sous-classes :

1. Les méthodes de voisinage.
2. Les méthodes évolutives.

3.2.2.1. Les méthodes de voisinage

Ces méthodes partent d'une solution initiale (obtenue de façon exacte, ou par tirage aléatoire) et s'en éloignent progressivement, pour réaliser une trajectoire, un parcours progressif dans l'espace des solutions. Dans cette catégorie, se rangent :

- **Recuit simulie**

Est une méthode basée sur la recherche locale dans laquelle chaque mouvement est accepté s'il améliore la fonction objective. Cette méthode est inspirée du processus de recuit utilisé en métallurgie pour améliorer la qualité d'un solde en cherchant un état d'énergie minimum. Cette méthode considère une solution initiale et cherche son voisinage d'une autre façon aléatoire. [26]

Pseudo code de l'Algorithme :

Algorithme Le recuit simulé
Début
Construire une solution initiale s ;
Calculer la fitness f(s) de s ;
Initialiser une valeur de la température T ;
$s_{best} = s$;
Tant que la condition d'arrêt n'est pas satisfaite **faire**
Générer une solution s' voisine de s ;
Calculer f (s') ;
Calculer $\Delta f) = f(s') - f(s)$;
Si $\Delta f) \geq 0$ **alors** // cas de maximisation
$s_{best} = s'$
$S=s'$
Sinon Si $r < exp\frac{\Delta(f)}{T}$ **alors**
$s=s'$
Fin Si
Décroitre la température T ;
Fin Tant que
Retourner s_{best} ;
Fin

- **La recherche Tabou**

La méthode Tabou le terme de ***recherche locale*** est de plus en plus utilisée pour qualifier ces méthodes.

Le principe de cet algorithme est de sélectionné le voisinage de solution à chaque itération. La méthode autorise de remonter vers des solutions qui semblent moins intéressent mais qui ont peut-être un meilleur voisinage. Des fois ce principe engendre des phénomènes des cyclages entre deux solutions, tandis que la méthode tabou à l'interdiction de visiter une solution récemment visité. Pour cela une liste tabou contenant les attributs des dernières solutions considérée et tenue à jour. Chaque nouvelle solution considérée enlève de cette liste la solution la plus anciennement visitée. Ainsi, la recherche de solution suivante se fait dans le voisinage de la solution actuelle sans considérer appartenant à la liste Tabou. [27]

3.2.2.1. Les méthodes évolutives.

Ces méthodes se distinguent de celles déjà étudiées par le fait qu'elles opèrent sur une population de solutions, pour cela, elles sont souvent appelées des méthodes à base de population. Certaines d'entre elles ont des principes inspirés de la génétique et du comportement des insectes. La complexité de ces deux phénomènes biologiques a servi de modèle pour des algorithmes toujours plus sophistiqués ces vingt dernières années. Parmi ces méthodes on va voir plus tard sont ACO PSO, et AG.

- **Les colonies de fourmis** (Ant Colony Optimization : ACO)

ACO [28] a été inspirée des études du comportement de fourmis réelles pour résoudre naturellement des problèmes relativement complexes. Le premier algorithme de ce méta heuristique a été appliqué pour résoudre le problème du voyageur de commerce, le principe de cet algorithme est simple. Lorsqu' une fourmi k se déplace de la ville i à la ville j, elle laisse une trace sur le chemin.

De plus, elle choisit la prochaine ville à visiter à l'aide d'une probabilité basée sur un compromis entre l'intensité de la trace et la visibilité qui représente l'inverse de la distance entre i et j. (d_{ij})

L'importance relative des deux éléments est contrôlée par deux coefficients α et β. Chaque fourmi k possède une forme de mémoire $t_{i\ \ k}$, lui rappelant la liste ordonnée des villes déjà visitées afin d'obliger celle-ci à former une solution admissible.

Après un tour complet, chaque fourmi laisse une certaine quantité de phéromone qui dépend de la qualité de la solution trouvée sur l'ensemble de son parcours. L'algorithme original a été adapté à notre problème en remplaçant la ville i par la pièce i et les villes j par le routage j. Pour chaque pièce i, le choix du routage j est basé sur un compromis entre l'intensité de la trace et la visibilité η_{ij} (dépend du nombre de pièces dans la file d'entrée de la première machine de ce routage et sa charge). L'importance relative des deux éléments est toujours contrôlée par deux coefficients α et β. Si le nombre total de fourmis est m et les tailles de la station de chargement est n, un cycle est réalisé lorsque chacune des m fourmis affecte les n premières pièces de la file infini à des routages j.

Après un tour complet (l'affectation de toutes les n premières pièces de la file infini aux routages par les fourmis), chaque fourmi laisse une certaine quantité de phéromone qui dépend de la qualité de la solution trouvée (le produit de la charge de routages) sur l'ensemble des routages sélectionnés pour les pièces.

Pseudo code de l'Algorithme :

S'il y'a une place libre dans la station de chargement **alors**
  Pour $t = 1$ à t_{max}
    Pour chaque fourmi $k = 1$ à m **Choisir**
      Pour la première pièce de la file infini un routage au hasard suivant son type.
        Pour chaque pièce i contenu dans la deuxième place jusqu' au la $n^{ème}$place de la file infini. **Choisir**
          Un routage j, parmi les routages possibles selon une probabilité dépendant de l'intensité de la trace et du nombre de pièces dans la file d'entrée de la première machine de ce routage et sa charge.
        Fin Pour
      Evaluation de la fonction objective. (Produit des charges de routages)
      Déposer une $\Delta\pi_{ij}^{k}(t)$ piste sur le trajet $T^{k}(t)$ (pour chaque routage j choisi pour la pièce i par la fourmi k).
      Fin Pour.
  Évaporer les pistes et modifier les intensités.
  Fin Pour.

Fin si.

- **PSO (Particle Swarm Optimization: PSO)**

Selon D.T Phan et al [29] PSO est basée sur la métaphore des interactions et communications sociales. Au départ de l'algorithme, un essaim est réparti au hasard dans l'espace de recherche, chaque particule ayant également un.

✓ Chaque particule est capable d'évaluer la qualité de sa position et de garder en mémoire sa meilleure performance, c'est-à-dire la meilleure position qu'elle a atteinte jusqu'ici (qui peut en fait être parfois la position courante) et sa qualité (la valeur en cette position de la fonction à optimiser).

✓ Chaque particule est capable d'interroger un certain nombre de ses voisins et d'obtenir de chacune d'entre elles sa propre meilleure performance (et la qualité afférente).

✓ A chaque pas de temps, chaque particule choisit la meilleure des meilleures performances dont elle a connaissance, modifie sa vitesse en fonction de cette information et de ses propres données et se déplace en conséquence.

A chaque itération, La vélocité et la position sont mise à jour suivant deux forces best local et global. La première l'attire au best local qui est la position qui a donné le meilleur fitness pour la particule (i.e. c'est la position la plus proche de l'objectif que cette particule a pu atteindre). L'autre best est le global best, c'est la meilleure position trouvée par la particule et ses voisins B_i, et B_g sont les bests locaux et globaux. N est la taille des files d'attente.

Pseudo code d'algorithme PSO :

S'il y'a une place libre dans la station de chargement **alors**
  Initialiser la population
  Initialiser les paramètres.
  Tant que (critère d'arrêt est non atteint)
    Pour chaque particule X_i
      Calculer le produit des charges de routages de cette particule.
      Si $F(x_i) > F(Bl_i)$ **alors** :
        (Mise à jour du best local) $Bl_i = x_i$
      Fin si
      Si $F(x_i) > F(B_g)$ **alors** :
        (Mise à jour du best global)
        $B_g = x_i$
      Fin si
    Fin pour
    Pour chaque particule X_i
      $X_i(t) = c_2 \otimes F_3(c_1 \otimes F_2(W \otimes F_1(X_i(t-1)), Bl_i(t-1)), B_g(t-1))$
    Fin pour
  Fin tant que
Fin si.

- **Algorithme génétique AG**

Algorithme génétique [30] s'inspire des mécanismes biologiques tels que les lois de Mendel et la théorie de l'évolution. Son processus de recherche de solutions à un problème donné imite celui des êtres vivants dans leur évolution. Il combine une stratégie de "survie des plus forts" avec un échange d'information aléatoire mais structuré. Pour un problème pour lequel une solution est inconnue, un ensemble de solutions possibles est créé aléatoirement. On appelle cet ensemble la population.

Les caractéristiques (ou variables à déterminer) sont alors utilisées dans des s'séquences de gènes qui seront combinées avec d'autres gènes pour former des chromosomes et par après des individus. Chaque solution est associée à un individu, et cet individu est évalué et classifie selon sa ressemblance avec la meilleure, mais encore inconnue, solution au problème. Il peut être démontre qu'en utilisant un processus de sélection naturelle inspiré de Darwin, cette méthode convergera graduellement à une solution.

Comme dans les systèmes biologiques soumis à des contraintes, les meilleurs individus de la population sont ceux qui ont une meilleure chance de se reproduire et de transmettre une partie de leur héritage génétique à la prochaine génération. Une nouvelle population, ou génération, est alors crée en combinant les gènes des parents. On s'attend à ce que certains individus de la nouvelle génération possèdent les meilleures caractéristiques de leurs deux parents, et donc qu'ils seront meilleurs et seront une meilleure solution au problème. Le nouveau groupe (la nouvelle génération) est alors soumis aux mêmes critères de sélection, et par après génère ses propres rejetons. Ce processus est répété plusieurs fois, jusqu'à ce que tous les individus possèdent le même héritage génétique. Les membres de cette dernière génération, qui sont habituellement très différents de leurs ancêtres, possèdent de l'information génétique qui correspond à la meilleure solution au problème.

Il utilise le même vocabulaire que celui de la biologie et la génétique classique, on parle donc de : gène, chromosome, individu, population et génération.

- ✓ **Un gène :** est un ensemble de symboles représentant la valeur d'une variable. Dans la plupart des cas, un gène est représenté par un seul symbole (un bit, un entier, un réel ou un caractère).
- ✓ **Un chromosome** : est un ensemble de gènes, présentés dans un ordre donné de manière qui prend en considération les contraintes du problème à traiter. Par exemple, dans le problème du voyageur de commerce, la taille du chromosome est égale au nombre de villes à parcourir. Son contenu représente l'ordre de parcours de différentes villes. En outre, on doit veiller à ce qu'une ville (représentée par un nombre ou un caractère par exemple) ne doit pas figurer dans le chromosome plus qu'une seule fois.

✓ **Un individu :** est composé d'un ou de plusieurs chromosomes. Il représente une solution possible au problème traité.

✓ **Une population :** est représentée par un ensemble d'individus (c.-à-d. l'ensemble des solutions du problème).

✓ **Une génération :** est une succession d'itérations composées d'un ensemble d'opérations permettant le passage d'une population à une autre.

L'algorithme génétique fait évoluer une population composée d'un ensemble d'individus pendant un ensemble de génération jusqu'à ce qu'un critère d'arrêt soit vérifié. Le passage d'une population à une autre est réalisé grâce à des opérations d'évaluation, de sélection, de reproduction (croisement et mutation) et de remplacement.

Pseudo code d'Algorithme génétique :

L'algorithme génétique

Début

Initialiser les paramètres nécessaires ;

Initialiser une population de N individus ;

Evaluer les N individus ;

Tant que la condition d'arrêt n'est pas satisfaite **faire**

Utiliser l'opérateur de sélection pour sélectionner K individus ;

Appliquer l'opérateur de croisement sur les K individus avec la probabilité P_c;

Appliquer l'opérateur de mutation sur les individus avec la probabilité P_m ;

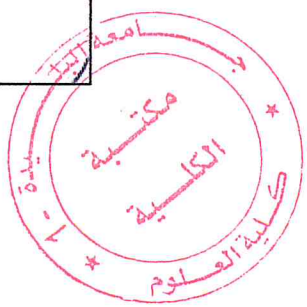
Utiliser l'opérateur d'évaluation pour évaluer les enfants obtenus ;

Utiliser l'opérateur de sélection pour remplacer quelques individus parents p des individus enfants ;

Fin Tant que

Retourner la ou les meilleures solutions ;

Fin



4. Conclusion

Nous avons parlé dans ce chapitre des méta-heuristiques et algorithmes évolutionnaires pour avoir une idée sur l'optimisation combinatoire. Nous avons cité quelques méthodes que nous utiliserons dans le prochain chapitre pour construire nos solutions proposées.

1. Introduction

Quand un nouveau problème se pose, on se retrouve devant une tâche difficile qui conduit à définir de nouvelle méthode de résolution car les techniques existantes se sont pas convenables ou dépassé. Et chaque nouvelle méthode s'inspire d'une méthode déjà existante. Dans ce chapitre, nous allons décrire les algorithmes que nous avons proposés pour l'extraction des itemsets fréquents à partir de données incertaines dans des environnements mobiles. Ces algorithmes sont des méthodes approchées, en effet ils se basent sur deux métaheuristiques (BSO pour Bee Sawrm Optimisation, BPSO pour Binary Particale Swarm Optimization) et les algorithmes génétiques.

2. L'algorithme évolutionnaire AG-UFIM basé sur les algorithmes génétiques

La figure 4.1 explique le principe de l'algorithme.

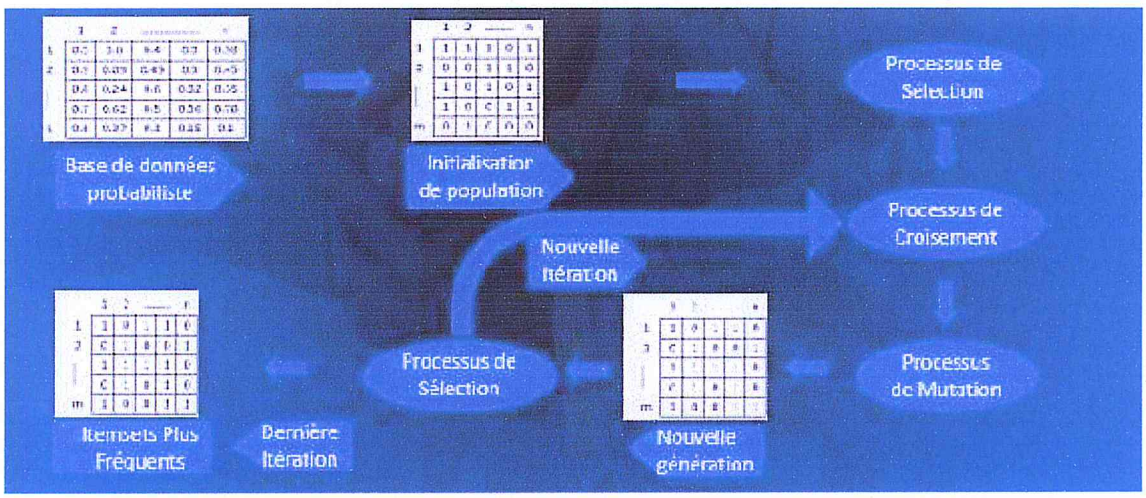


Figure 4.1 : schéma de l'algorithme AG-UFIM

Notre algorithme est basé sur les AG et procède par conséquent de la même façon. Avant de passer à l'explication de différentes étapes de l'algorithme, il faut tout d'abord expliquer l'encodage des individus qui représentent les solutions possibles.

2.1. Encodage d'individus

L'encodage binaire est utilisé dans la majorité des solutions basé sur les algorithmes génétiques. Néanmoins, d'autres types d'encodage existent aussi, tels que les arbres, les graphes, et l'encodage en utilisant des nombres entiers.

Dans notre cas l'encodage binaire d'adapte le mieux pour représenter et manipuler les solutions possibles qui sont les itemsets fréquents incertains. Chaque solution (itemset) est représentée par un vecteur S de n item où n est le nombre des items.

S [i] = 1 si item i existe dans l'itemset.

S [i] = 0 sinon.

Exemple: pour n= 10, l'itemsets suivant contient 3 items, **e, f** et **g** item.

A	b	c	d	e	f	g	h	i	j
0	0	0	0	1	1	1	0	0	0

2.2 Fonction objective

L'évaluation de la qualité des solutions générées se fait via la fonction objective. Pour calculer la fréquence des itemsets générés nous utilisons l'expected support qui est la mesure la plus utilisé pour estimer la fréquence des motifs à partir de données incertaines.

En utilisant le modèle probabiliste de l'incertitude, l'Expected support d'un itemset X dans un ensemble de données T de taille D, noté exp Sup (X, T) est calculé comme suit :

$$expSup(X,T) = \sum_{i=1}^{D}\left(\prod_{x\in X} P(x,T_i)\right) \qquad (16)$$

2.3. Opérateurs génétiques

Sélection : processus où les individus sont copiés selon la valeur de leur fonction objective *f*. On peut décrire la fonction *f* comme une mesure de profit, utilité ou qualité que l'on veut maximiser (minimiser). L'implantation de la sélection peut se faire de plusieurs façons. La plus facile est peut-être la roue de roulette biaisée, où la probabilité de reproduction d'un individu dépend de sa valeur par rapport au total des valeurs de la population.

Dans notre cas la sélection se fait de façon que nous calculons la valeur de la fonction objective d'individus *f*(individu)

- Si *f*(individu) ≥ minsup alors on sauvegarde la position d'individu dans le vecteur indiceBest et leur valeur de *f*(individu) dans tabFit.

- Sinon aller au prochain individu.

Le figure suivant explique la méthode

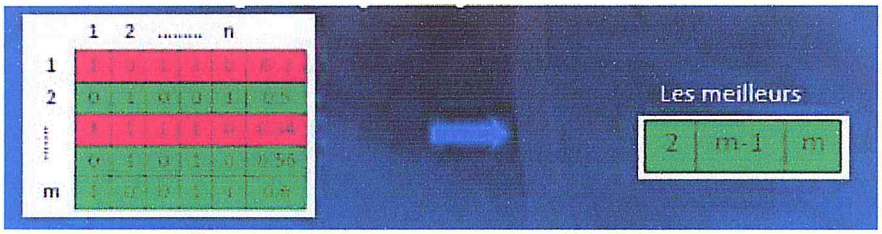


Figure 4.2 : le processus de sélection

Reproduction (Croisement): processus où de nouveaux individus sont formés à partir de parents. Ces nouveaux individus, les rejetons, sont formés en exécutant un croisement entre deux parents. On choisit une position aléatoire k entre [1, l-1] o `u l est la longueur de l'individu. Le croisement se fait en échangeant les bits de la position k + 1 à l.

Exemple : Soit $k = 4$ pour deux parents ($P1$ et $P2$) codés à 5 bits (donc $l = 5$). Les rejetons sont $O1$ et $O2$.

$$\left.\begin{array}{l} P1 = 0110|1 \\ P2 = 1100|0 \end{array}\right\} \begin{array}{l} O1 = 01100 \\ O2 = 11001 \end{array}$$

On voit bien l'échange qui s'est produit ici, le bit 5 (k + 1) passé d'un individu à l'autre, pour ainsi former deux nouveaux individus ($O1$ and $O2$).

Ce sont ces deux opérations, la sélection et la reproduction, qui sont à la base des algorithmes génétiques. Ceci peut paraitre simple à première vue, puisque aucune opération mathématique complexe n'a été exécutée. Mais on peut comparer le processus précédent à l'innovation humaine : souvent, les découvertes n'arrivent pas par chance. Elles sont le résultat d'un échange d'idées qui crée d'autres idées et finalement mènent à une solution désirée.

Dans notre méthode le croisement se fait comme suivant :

1- On prend deux individus parmi les meilleurs qu'ont été sélectionné dans l'opération Sélection

2- A partir d'entrée Taux de croisement, nous choisissons un point de croisement entre les deux individus qui ont déjà sélectionné.

Exemple :

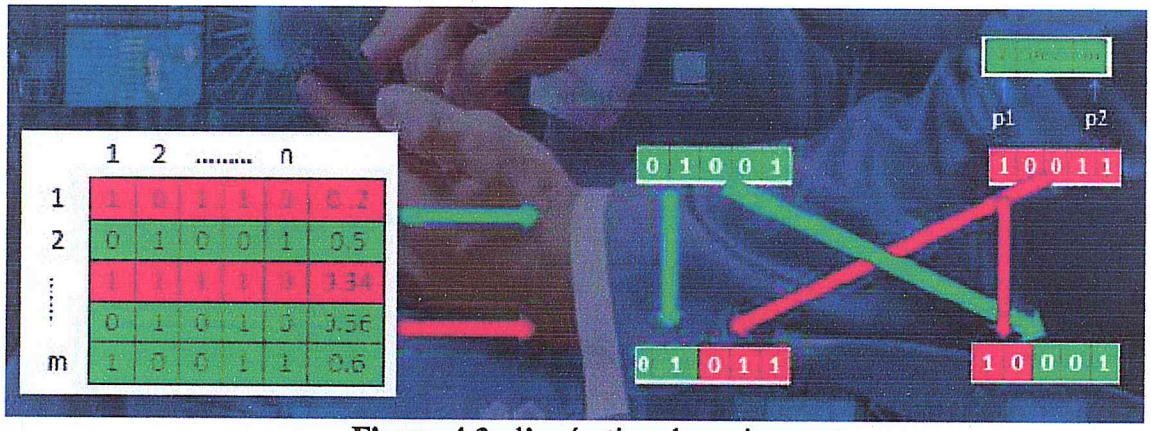


Figure 4.3 : l'opération de croisement

Mutation : processus aléatoire où un bit change de valeur. Ce processus joue un rôle secondaire dans l'algorithme génétique, mais il est quand même important. La mutation assure qu'aucun point dans l'espace de recherche n'a une probabilité nulle d'être atteint.

La mutation est un opérateur de diversification, dans notre cas, il permet d'éviter le phénomène d'optimum local et de sauter vers des solutions pertinentes qui ne peuvent pas être atteintes par les deux opérateurs précédents.

La mutation se faite de manière suivante :

- On sélectionne un individu aléatoirement par la sélection de sa position dans vecteur de meilleure solution
- On sélectionne deux cases totalement différentes h1≠h2
- On fait le swap entre c'est deux cases

L'opération se répète jusqu'à taux Mutation fois

Le figure 4.4 démontre comment se déroule la mutation.

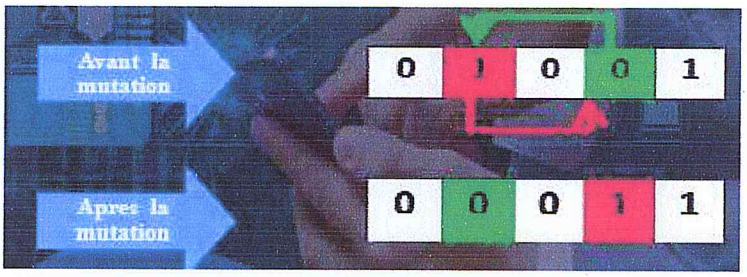


Figure 4.4 : l'opération de Mutation

Pseudo code de l'algorithme :

Algorithme 1: AG-UFIM

Input : dataSet of uncertain data, minsup, N population,CrossoverPoint, M mutation, max_generation

```
1 initialize the population
        current_generation := 1
        for i := 1 to N do
            for j := 1 to L do
            solution[i][j] := randomselect(0,1)
2 evaluate and select the best population
indBest=0
        for i := 1 to N do
            if   evaluate ( solution[i] ) ≥ minsup)
            indiceBest[indBest]=i
            indBest++
        end For
        length=indBest

3 crossover
        h1 = randomselect(length-1)
        eliminete h1 from indiceBest to make sure that is h2 ≠h1
        length--
        h2 = randomselect(length-1)
            for(int j=0;j<tauxCroisement;j++){
              popselection[i][j]=solution[indiceBest[h1]][j]; }
            for(int j=tauxCroisement;j<col;j++){
            popselection[i][j]=solution[indiceBest [h2]][j];}
4 check if popselection is Exactly the same as solution
    if returns true then go to 5
    else go to 6
5- mutation
        for i := 1 to M do
ligne= randomselect (length), col=randomselect(items), col1=randomselect(items)
    t=solution[indiceBest[ligne]][cln1];
    solution[ligne][cln1]=solution[ligne][cln2];
    solution[ligne][cln2]=t;
6 - new population
        Solution=popselection
        current_generation++
7 - termination condition
        if ( current_generation < max_generation )
        return to step 2
```

Output : N best frequent itemsets

3. L'algorithme BPSO-UFIM base sur PSO

En français Optimisation d'essaims de particules binaires L'algorithme PSO, présenté par Eberhart et Kennedy (1995), s'inspire du comportement social du flocage d'oiseaux ou du poisson scolarité. Il optimise une fonction objective en améliorant itérativement les solutions candidates.

Le figure suivant schématise le déroulement de notre algorithme :

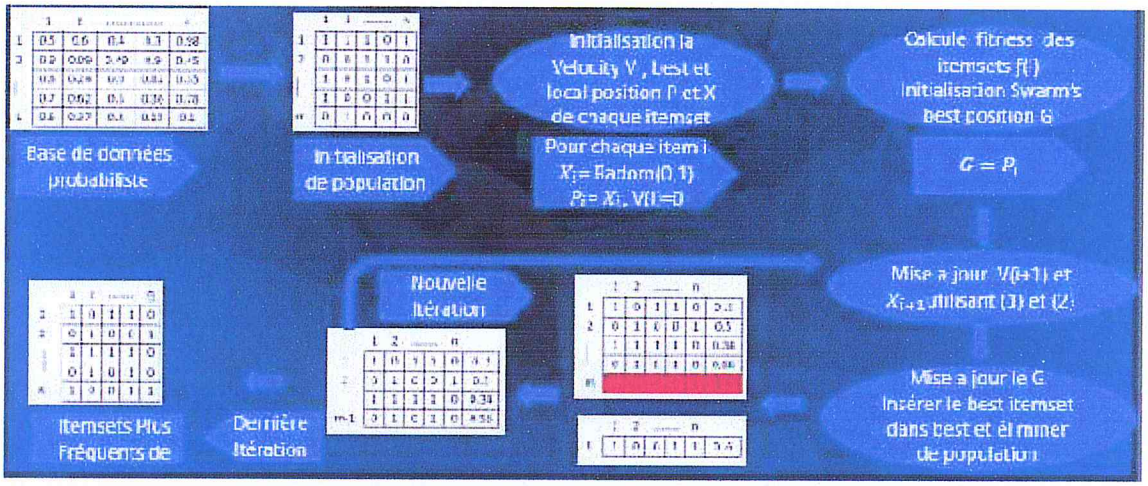


Figure 4.5 : schéma de l'algorithme BPSO-UFIM

3.1. Encodage d'individus

Chaque solution (itemset) est représentée par un vecteur S de n item où n est le numéro items.

S [i] = 1 si item I existe dans l'itemset.

S [i] = 0 sinon.

3.2. Fonction objective

Dans notre cas, nous utilisons l'expected support comme il est montré dans la formule (16).

3.3. Calcule de la vélocité

Dans PSO, chaque particule est un candidat solution au problème. Chaque particule est associée à une position et une vitesse. Chaque particule se déplace vers la position suivante en utilisant son propre expérience et expérience de la particule voisine. Les particules de quartier peuvent être les particules dans l'ensemble de l'essaim ou les particules les plus proches de la particule. Soit N le nombre de particules et 'D' la dimension de l'espace de recherche. L'algorithme PSO de base est le suivant:

Afin de résoudre des problèmes d'optimisation discrets / combinatoires. Dans la version binaire de PSO, la position de chaque particule est 0 ou 1. La valeur de Xi est

déterminée de manière probabiliste selon le taux de variation Vi. Pour la version binaire de PSO, les positions de la particule sont mises à jour comme suit :

$$V_i^{k+1} = W * V_i^k + C_i * r_1 * (P_i - X_i) + C_2 * r_2 * (G - X_i) \quad (13)$$

Si (rand () < S (V_{i+1}))

alors $X_{i+1} = 1$

sinon $X_{i+1} = 0$

tel que S(.) est la fonction sigmoid.

rand () est la fonction random uniforme entre. (0, 1)

W est facteur d'inertia et C1, C2 sont des constants positives.

Fonction Sigmoid $f(x) = {}^{1}/_{1+e^{-\lambda}}$.(14)

Pseudo code d'algorithme :

Algorithme 2: BPSO-UFIM

Input : dataSet of uncertain data, minsup,N particles, M iterarions

1. Initialize the N particles and its local position Xi with random (0,1)
2. Calculate the fitness value for all particles.
3. Initialize the particle's local best position Pi ←Xi.
4. Initialize the swarm's best position as, G←Pi, if f (G) f (Pi) For all values of i, where 'f' is the objective function to be maximized.
5. Initialize each particle's velocity vector with 0.
6. Until termination condition is met (i.e..the maximum number of iterations) repeat the following steps:

For each particle i=1…..NDo

- Pick random numbers r1, r2 from uniform distribution (0, 1).
- Update the particles velocity, positions with the following formula:

$$V_i^{K+1} = W * V_i^K + C_1 * r_1 * (P_i - X_i) + C_2 * r_2 * (G - X_i) \quad (13)$$

$$X_i^{K+1} = X_i^K + V_i^{K+1} \quad (15)$$

Where $V_i^K$is the velocity of ith particle at kth iteration and W is the inertia factor and C1, C2 are positive constants.

If (rand () < S (V_{i+1})) Then $X_{i+1} = 1$Else $X_{i+1} = 0$

- If (f(Xi) > f (Pi)) Pi←Xi
- If (f(G) >< f (Pi)) G← Pi

7. Now G holds the best position and G is reported as the optimal solution with f (G) as the fitness value. then safe the best particle and isolate it from the rest. length of the population will Decreases.
8. The PSO algorithm terminates after maximum number of iterations is reached

Output: M best particles

4. Algorithme BSO-UFIM basé sur BSO

Le processus principal de BSO est donné par le pseudo code dans Algorithme 3.

Algorithme 3: BSO-UFIM

1: Input: k _Bees, MaxIteration, Flip, minsup, Probabilistic Dataset,
2: Sref = be the initial solution generated randomly.
3: if f(Sref)≥minsup then S*=Sref; else return to 2 where f is function maximized fitness
4: TabouList = TabouList + Sref
5: while *i<MaxIteration* do
6 :-ModuloStrategy(Sref)
7: tabooSearch(BeeSpace)
TabouList = TabouList + best of dancetab
end while
8: Sref=best of best solution k bee solution
9: Output: K*MaxIteration+1 of uncertain frequent itemsets.

Pseudo code de la méthode tabooSearch avec leur schema explicatif :

Algoritme 4 : tabooSearch

For determining the initial Sref for each bee
For each i bee do
For each line j of dancetab do
C and C1 random numbers (0,items)
E=Swap(Bee_space[i][c], Bee_space[i][c1])
Dancetab[j]=E
TabooList= TabooList+f(Dancetab)
End for
End for
Sref= best of K-bee solution

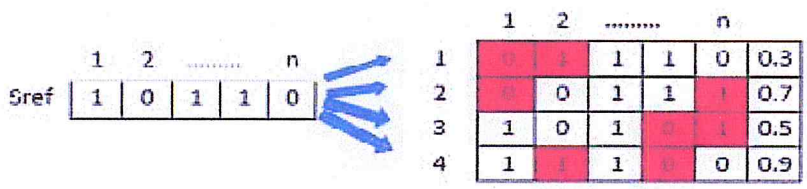


Figure 4.6 : méthode tabooSearch

Le figure 4.7 explique notre méthode proposée

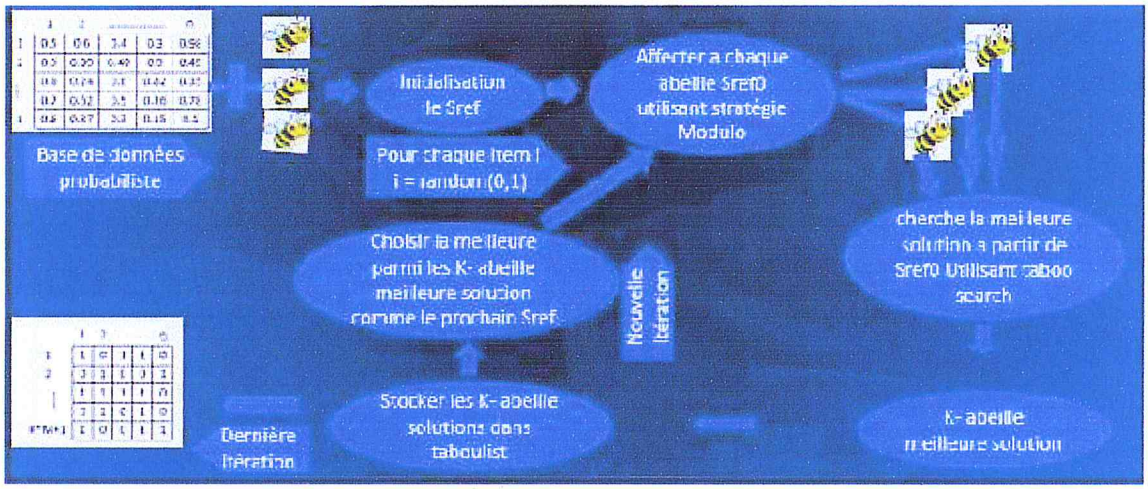


Figure 4.7 : schéma de l'algorithme BSO-UFIM

4.1.Encodage des Solutions fonction objective

Chaque solution (jeu d'éléments) est représentée par un vecteur S de n éléments, où n est le nombre d'éléments.

S [i] = 1 si l'élément i est dans le itemset.

S [i] = 0 sinon.

On compare la valeur de fréquence de Sref avec le MinSup, si ExpSup(Sref) >MinSup alors Sref est fréquente sinon le programme choisi une autre solution Sref aléatoirement.

4.2.La détermination des zones de recherche

Une solution de référence donnée Sref et une colonie d'abeilles K, les opérations de la zone de recherche détermine les espaces de recherche K, chacun étant associé à une abeille. Pour déterminer ces domaines de recherche, Il existe nombreuses stratégies différentes comme *modulo*, *next*et *syntactic*.Nous avons choisi cette parmi eux.

4.2.1. La stratégie Modulo

Dans la stratégie modulo, chaque abeille k crée sa propre zone de recherche en changeant successivement dans la solution Sref les bits k + i × Flip où i varie de 0 à n -

1 et Flip est un paramètre donné. Cette stratégie peut être utilisée si et seulement si le nombre d'abeilles est inférieur ou égal à n/Flip Si la distance entre les solutions est le nombre de bits différents, alors la distance entre les abeilles et la référence de la solution est égale à n/Flip. La figure 4.3 illustre le scénario de cette stratégie par Compte tenu de la solution de référence Sref=0110000111.

Algorithme 4 décrit plus formellement cette stratégie.

Pseudo code :

Algorithme 5 :stratégie Modulo

```
1 Input: Sref, Flip, K (bees number)
2 Output: Bees_Space: Array [1...K][1...n]
3 i←0
4 while i< K do
5 copy(Sref, Bees_Space[i])
6 for j= i,j<n-Flip, j = j + Flip do
7 Change_Bit(Bees_Space[i]Up S end for
9 i←i+1
10 end while
11 return Bees_Space
```

Le figure 4.8 démontre stratégie Modulo

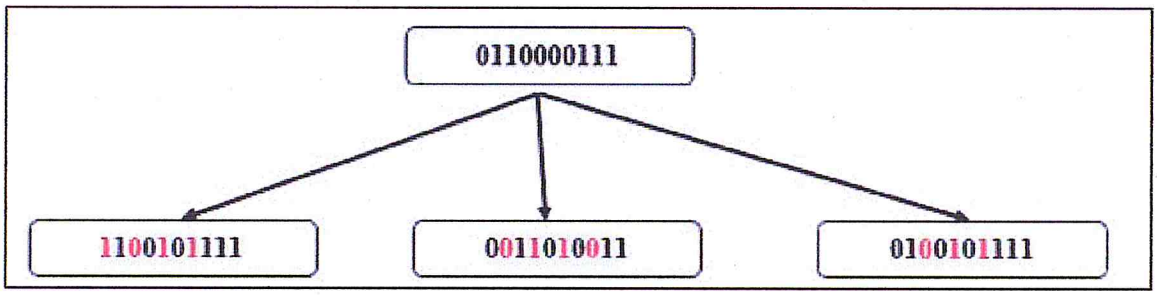


Figure 4.8: Strategies Modulo

4.3. La function Objective

Dans notre cas le BBSO utilise la meme fonction qui est l'expected support elle est reference dans la formule (16).

4.4. L'architecture d'algorithme BSO-UFIM

L'architecture présenté dans la figure (4.1) représente les différentes étapes que suit l'algorithme de colonie d'abeilles afin d'extraire les itemsets fréquents.

Les différentes étapes sont :

- **Etape1 :** entre les donnes incertaines et Sref initial, MinSup, Maxitération
- **Etape 2 :** génère zone de recherche possible.
- **Etape 3 :** orientation chaque zone avec l'abeille
- **Etape 4 :** la solution
- **Etape 5 : Test 1 :** si solution supérieure à MinSup ajuter dans table dance sinon supprime la solution.
- **Etape 6 :** tous les itemsets fréquent dans table dance.
- **Etape 7 : Test 2 :** si la meilleure solution prendre à prochaine itération comme Sref sinon nous faire une nouvelle fonction Random.

5. Conclusion

Dans ce chapitre nous avons choisi et présenté trois algorithmes différents, ces fondamentaux ainsi que ces principes de fonctionnement. Dans le prochain chapitre, on va tester leurs faisabilité et performances sur des données réelles et semi synthétique.

Chapitre 05 :

Test et validation

1. Introduction

Après avoir décrit notre solution, nous aborderons dans ce chapitre la partie test et validation de notre application. En première partie, nous présenterons l'environnement de travail et les outils de développement utilisés. Ensuite, nous présentons notre jeu d'essai. Enfin, nous présenterons notre application ainsi que son fonctionnement et les résultats.

2. Présentation de l'environnement de travail

2.1. Matériel

On a utilisé :

- Un ordinateur portable acer avec les caractéristiques suivantes :
  - 10 GO RAM
  - 512 GO DDR
  - Intel® Core(TM) i5-2467M CPU 1.6 GHz
  - Système d'exploitation : Windows 8.1 de 64-bit
- Téléphone mobile Condor P6 pro avec :
  - Dimension : 7.1 x 14.3 x 1 mm
  - Système d'exploitation (OS) : Android 5.1
  - Processeur : Quad Core 1.3 GHz
  - Taille (diagonale) : 5 pouces
  - Résolution : 1280 x 720 pixels
  - Mémoire : interne 16 GO
  - Ram : 2 GO

2.2.Outils de Développement

Java

Java est un langage de programmation et une plate-forme informatique qui ont été créés par Sun Microsystems en 1995. Beaucoup d'applications et de sites Web ne fonctionnent pas si Java n'est pas installé et leur nombre ne cesse de croître chaque jour. Java est rapide, sécurisé et fiable. Défini à l'origine comme un langage, **Java** a évolué au cours du temps pour devenir une technologie, qui intègre une bibliothèque complète pour exécuter ou développer une multitude d'applications.

La particularité principale de **Java** est que les applications écrites dans ce langage sont très facilement portables sur plusieurs systèmes d'exploitation tels que l'UNIX, Windows, Mac OS ou GNU/Linux, avec peu ou pas de modifications. C'est la plate-forme qui garantit la portabilité des applications développées en **Java**.

La technologie Java regroupe aujourd'hui :

- La plate-forme d'exécution JRE (**Java Runtime Environnement**) des programmes écrits dans ce langageEnvironnement d'exécution Java.
- la plate-forme de développement JDK (**Java Development Kit**) (Kit de développement Java).
- le greffon permettant d'exécuter des programmes spéciaux écrits dans ce langage pour les navigateurs internet.

Le tout forme un ensemble cohérent quoique modulaire, et selon vos besoins, vous installerez certains composants et pas d'autres. Dans la plupart des cas et si vous ne souhaitez pas développer en Java, l'installation de la plate-forme d'exécution des programmes JRE et le greffon pour les navigateurs internet suffisent **[17].**

Pour notre système on a utilisé la version LUNA d'éclipse comme un environnement de développement.

XML

❖ **Qu'est-ce que XML?**

• XML signifie eXtensibleMarkupLanguage.

• XML est un langage de balisage peu comme HTML.

• XML a été conçu pour décrire les données, ne pas afficher les données.

• Les balises XML ne sont pas prédéfinies, Vous devez définir vos propres balises.

• XML est conçu pour être auto-descriptif.

• XML est une recommandation du W3C.

❖ **La différence entre XML et HTML**

- XML ne remplace pas pour HTML.
- XML et HTML ont été conçus avec des objectifs différents.
- XML a été conçu pour décrire les données, en mettant l'accent sur les données.
- HTML a été conçu pour afficher des données, en mettant l'accent sur la façon dont les données regards.

HTML est sur l'affichage des informations, alors que XML est de transporter des informations.

Nous pouvons dire que le document XML ne fait rien. Il est juste de l'information enveloppée dans les tags. Quelqu'un doit écrire un morceau de logiciel pour envoyer, recevoir ou afficher.

Android Studio

Android Studio est l'environnement de développement intégré officiel (IDE) pour le développement d'applications Android, basé sur IntelliJ IDEA. En plus de l'éditeur de code et des outils développeurs puissants d'IntelliJ, Android Studio offre encore plus de fonctionnalités qui améliorent votre productivité lors de la création d'applications Android, telles que:

- Un système de construction flexible basé sur Gradle
- Un émulateur rapide et riche en fonctionnalités
- Un environnement unifié où vous pouvez développer pour tous les appareils Android

Instant Run pour pousser les modifications à votre application en cours sans créer de nouveau APK

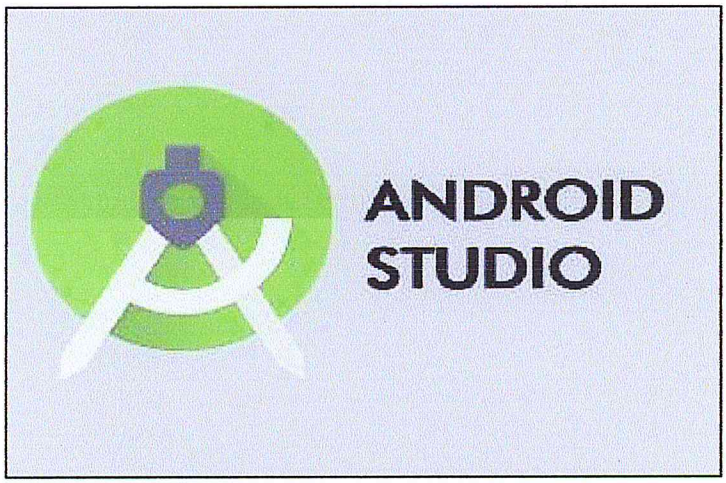


Figure 5.1: logo de Android Studio

3. Présentation des données

Les fichiers de données que nous avons utilisés sont des fichiers qui contiennent des items avec leurs probabilités d'incertitude. Ces fichiers sont générés à partir des benchmarks mais nous n'avons pas prend toute les données du fichier. A chaque fois nous prenons une partie du fichier garce a les lignes et les colonnes qui ont dans comme une entrée dans l'application.

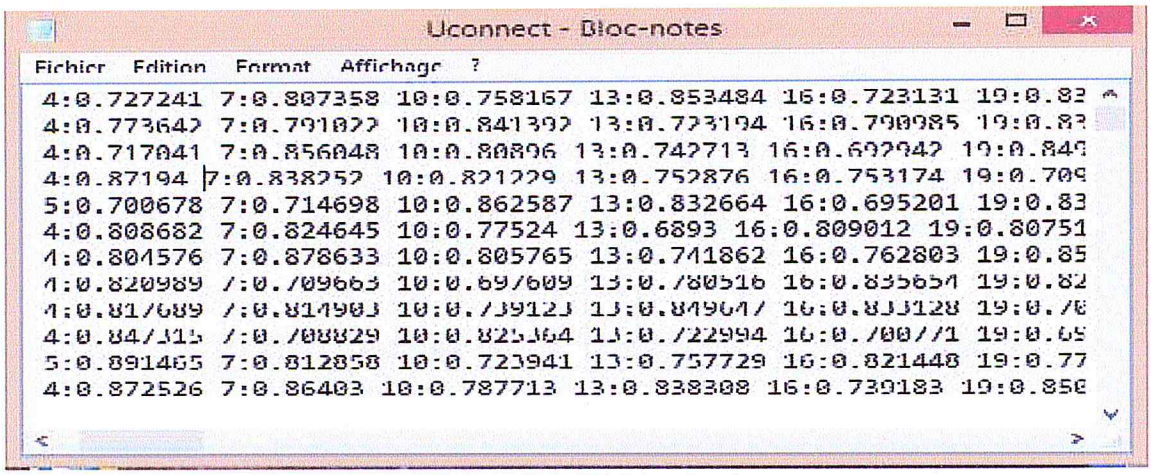


Figure 5.2: Fichier des itemsets

4. Présentation de l'interface utilisateur

Dans qui suit nous allons présenter les interfaces réalisées qui permettent à l'utilisateur d'avoir les résultats souhaités.

4.1.AG-UFIM

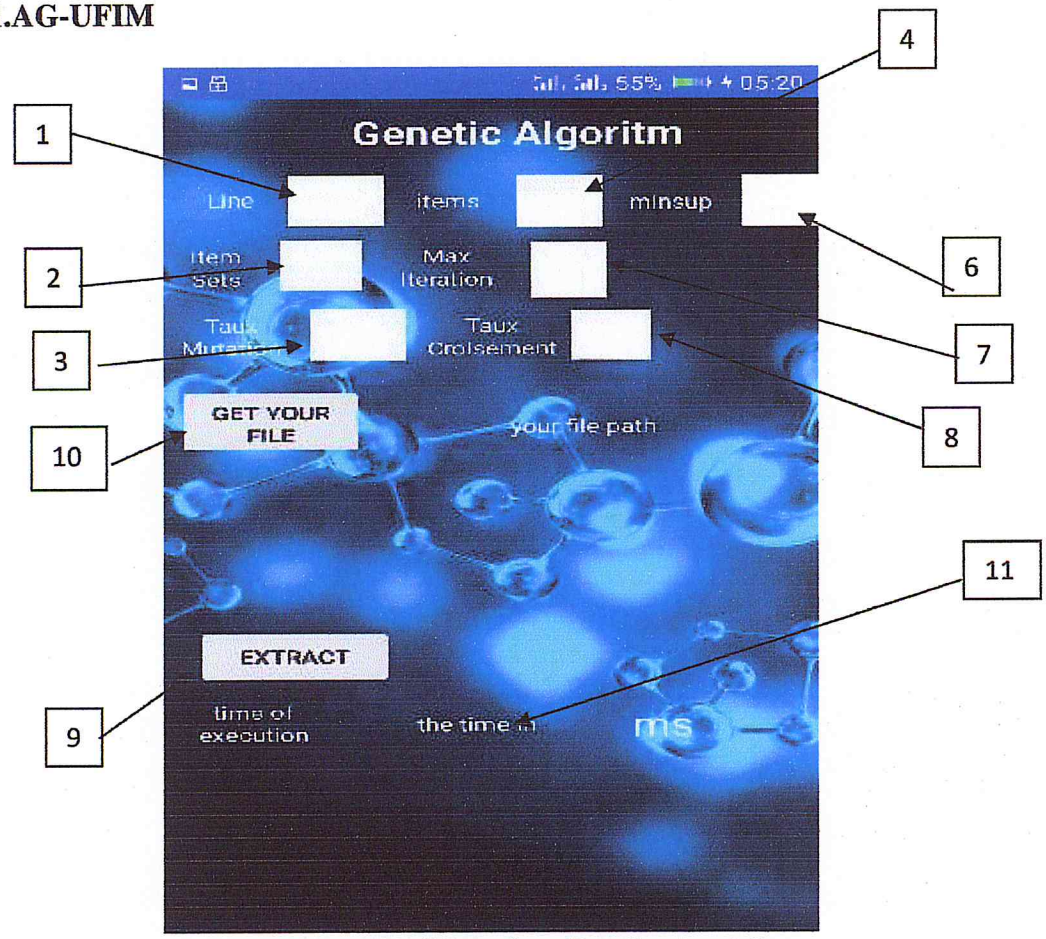


Figure 5.3: L'interface d'application AG

1 : nombre des transactions dans le dataset
2 : nombre de population
3 : taux de mutation
4 : nombre des items
6 : min sup (seuil)
7 : nombre d'itération
8 : taux de croisement
9 : bouton pour lire le fichier de datasets
10 : bouton pour extraire les items les plus fréquents
11 : le temps d'exécution

4.2.BPSO-UFIM

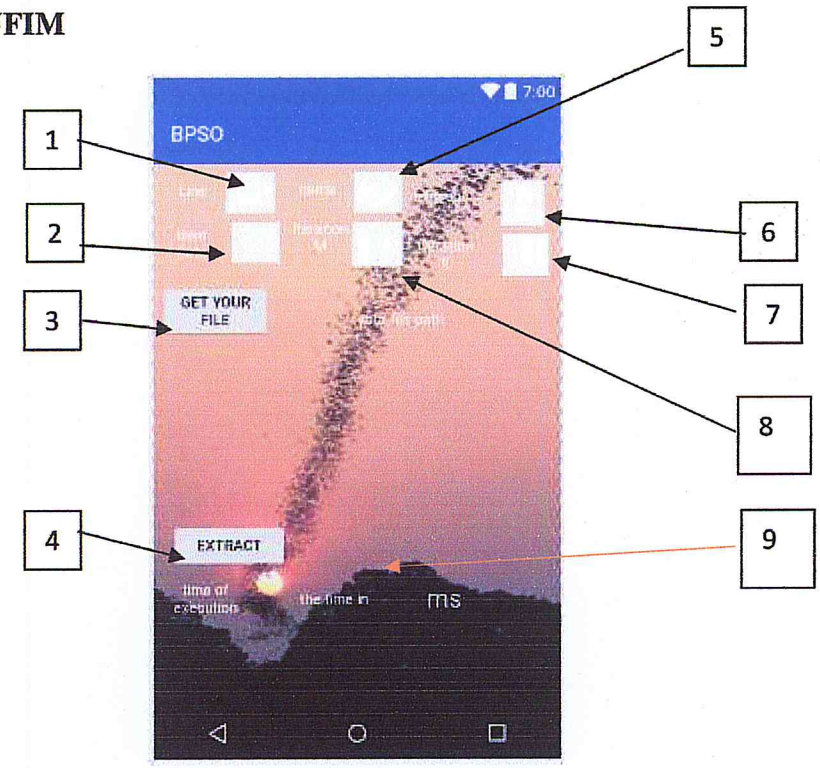


Figure 5.4: L'interface d'application BPSO

1 : nombre des transactions dans le dataset
2 : nombre de population
3 : bouton pour lire le fichier de datasets
4 : bouton pour extraire les items les plus fréquents
5 : nombre des items
6 : min sup (seuil)
7 : Max Itération interne
8 : Max itération externe
9 : le temps d'exécution

4.3.BSO-UFIM

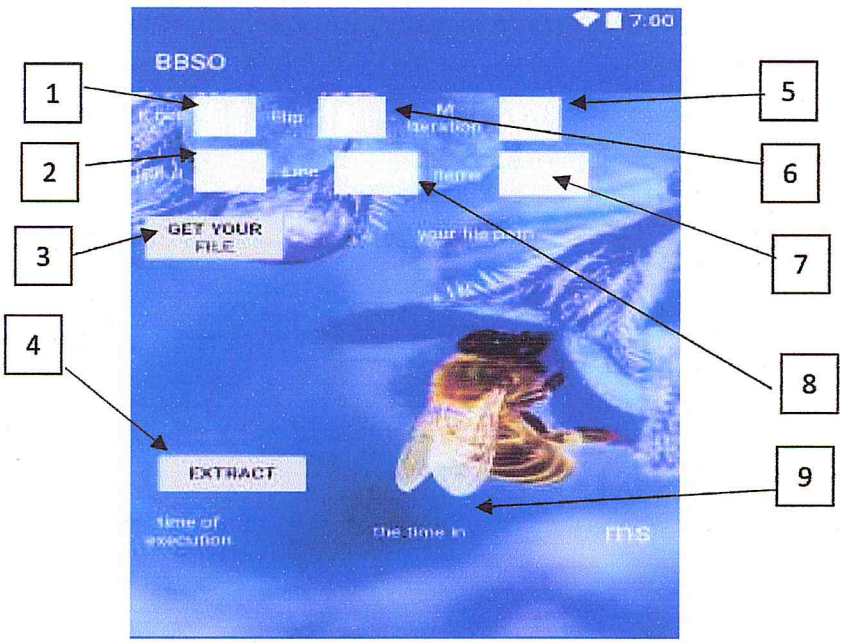


Figure 5.5 : l'interface d'application BSO-UFIM

1 : nombre d'abeilles
2 : minsup
3 : bouton pour lire le fichier de datasets
4 : bouton pour extraire les items les plus fréquents
5 : nombre d'itération
6 : flip
7 : nombre des items
8 : nombre des transactions
9 : le temps d'exécution

5. Résultats et discutions

Remarque : le min représente la valeur que l'utilisateur saisi dans l'application mais le vrai minsup il se calcule comme suivant

minsup= min*nombre des transactions

- **AG-UFIM**

Tableau 5.1 : résultats d'AG-UFIM

Transactior	items	min	minsup	itemsets	M	K	time in s	Ram mg
1000	20	0.00001	0.01	100	10	2	1.3	1
1000	30	0.002	2	150	20	2	3151.4	11
1000	40	0.0045	4.5	200	40	3	5164.9	1
1000	50	0.0009	0.9	300	50	3	9354.9	3
2000	20	0.0003	0.6	400	100	3	12403.8	1
2000	30	0.0002	0.4	600	200	3	28766.8	1
2000	40	0.0005	1	1000	400	4	78039.3	2
2000	50	0.0015	3	1000	500	5	106.0	0.5
5000	20	0.00001	0.05	1000	600	5	125.3	2
5000	30	0.0002	1	2000	500	3	1532.0	10
5000	40	0.0045	22.5	3000	500	3	552.2	4
10000	40	0.0005	5	5000	3000	3	812.2	25
10000	50	0.00015	1.5	5000	3000	5	700.0	18

On remarque que le temps d'exécution augmente lorsque les paramètres suivants augmentent nombre des transactions(line) , items, population, itération et taux de mutation.

Et la qualité de population augmente lorsque le taux de mutation et taux de croisement augmentent.

Notre test terminé dans 10000 transactions et 70 items paracerque il prend plus que 90 min alors on l'ignore parque ce n'est pas pratique pour nous.

- **BPSO-UFIM**

Dans le PSO-UFIM le temps d'exécution augmente aussi par rapport l'augmentation des paramètres suivantes : nombre des transactions(line), items, itemsets, M l'itération de l'extérieur, et K l'itération de l'intérieur.

Et à propos de qualité de M population augmente a chaque fois on augmente nombre de population.

Tableau 5.2 : résultats de BPSO-UFIM

Transactior	items	min	minsup	itemsets	M	K	time in s	Ram mg
1000	20	0.00001	0.01	100	10	2	1.3	1
1000	30	0.002	2	150	20	2	3.2	11
1000	40	0.0045	4.5	200	40	3	5.2	1
1000	50	0.0009	0.9	300	50	3	9.4	3
2000	20	0.0003	0.6	400	100	3	12.4	1
2000	30	0.0002	0.4	600	200	3	28.8	1
2000	40	0.0005	1	1000	400	4	78.0	2
2000	50	0.0015	3	1000	500	5	106.0	0.5
5000	20	0.00001	0.05	1000	600	5	125.3	2
5000	30	0.0002	1	2000	500	3	140.5	10
5000	40	0.0045	22.5	3000	500	3	552.2	4
10000	40	0.0005	5	5000	3000	3	812.2	25
10000	50	0.00015	1.5	5000	3000	5	700.0	18

- **BSO-UFIM**

Tableau 5.3 : résultats de BSO UFIM

k Bee	flip	M Ieteractior	seuil	minsup	Transactior items	time in s	cest	ramir Mg
5	2	3	0.0000001	0.0005	500	20	9 / 1.0	16	2
5	3	3	0.0000075	0.00375	500	30	1270.7	16	3
5	4	5	0.00001	0.005	600	45	5502.8	26	8.5
6	3	4	0.0000035	0.01875	750	34	8.6	25	4
5	5	5	0.0000003	0.00225	750	60	10.9	46	1.5
8	3	5	0.0000005	0.0044	880	28	3.8	41	5
6	5	4	0.0000004	0.0044	1100	28	4.0	25	5.5
6	4	6	0.0000005	0.011	2200	29	11.6	37	8
7	3	2	0.0000003	0.0075	2500	44	8.0	15	9
6	3	2	0.0000008	0.028	3500	25	3.8	13	14.5
7	3	2	0.0000001	0.0044	4400	54	12.1	15	15
6	5	6	0.0000006	0.03	5000	24	38.7	37	16
7	5	3	0.0000003	0.0165	5500	36	20.7	22	22
8	4	6	0.0000005	0.0045	3900	26	63.3	49	23
7	5	6	0.0000002	0.0014	7000	37	58.4	43	19.5
6	4	5	0.0000004	0.028	7000	50	38.5	31	21
8	2	5	0.0000008	0.06	7500	75	#VALEUR!		
6	3	2	0.0000001	0.0078	7800	56	27.6	19	27.5
5	5	5	0.0000003	0.0267	8900	34	35.0	26	28
7	4	3	0.0000006	0.054	9000	50	33.6	22	30

Dans ce test on a le truc tel que l'algorithme il doit vérifier le Sref initiale avant que les abeilles commencent leurs travails sur les sous Sref qui ont généré à la base de Sref initiale alors le temps d'exécution augmente depuis le début. Après à chaque fois on augmente chaque paramètre parmi ces paramètres nombre des transactions(line), items itération et sans oublie le minsup. Calcul non-stop à cause de la solution est inférieure à minsup alors il généré jusqu'à trouver la solution préfère (>= minsup).

6. La comparaison entre les trois méthodes

Parmi les trois méthodes la meilleure qualité trouvé est dans la troisième méthode BSO-UFIM mais elle a un inconvénient c'est que le temps dépasse le 30 min quand les transactions et les items dépasse 7500 transactions et 75 items.

Par contre dans la première méthode AG-UFIM elle a une qualité bien par rapport le PSO-UFIM et moins par rapport le BSO-UFIM, avec un temps d'exécution réduit par rapport le BSO-UFIM

La Deuxième méthode qui est le PSO-UFIM elle a des solutions de qualité moins par rapport les deux autres méthodes mais elle ne prend pas beaucoup du temps et elle peut faire mieux que les autre BSO-UFIM et AG-UFIM malgré les transactions, items et nombre de population qui représente nombre d'itération

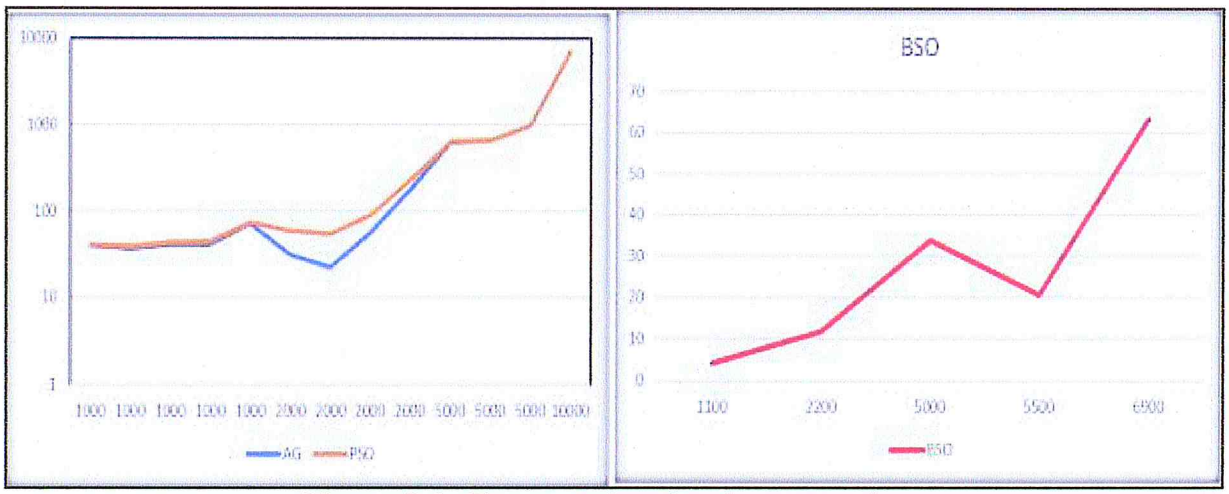


Figure 5.6 : Temps en fonction des transactions

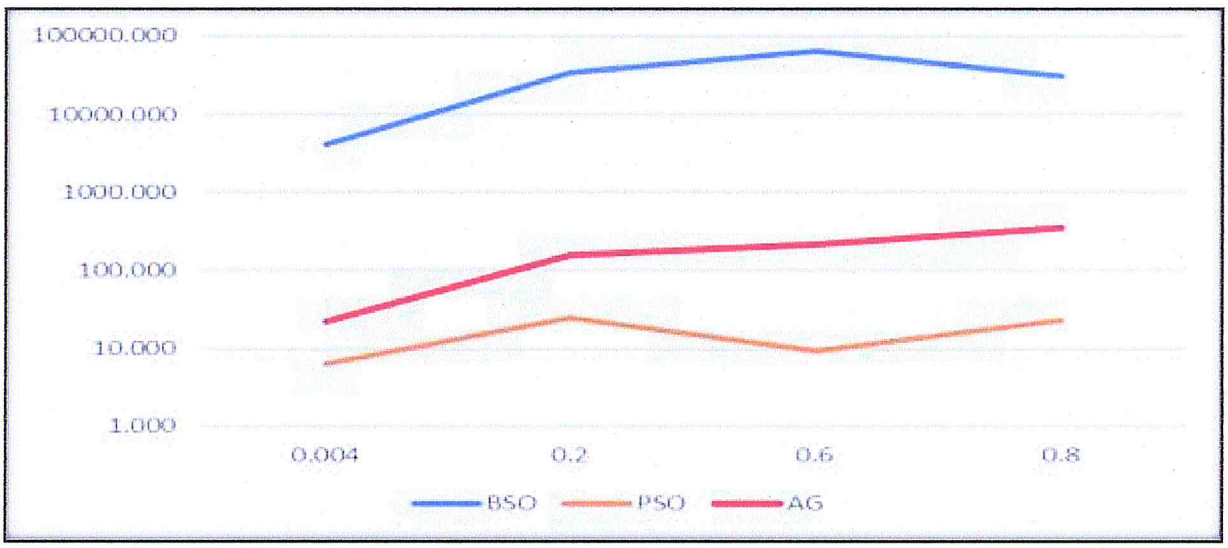


Figure 5.7 : Temps en fonction de minsup

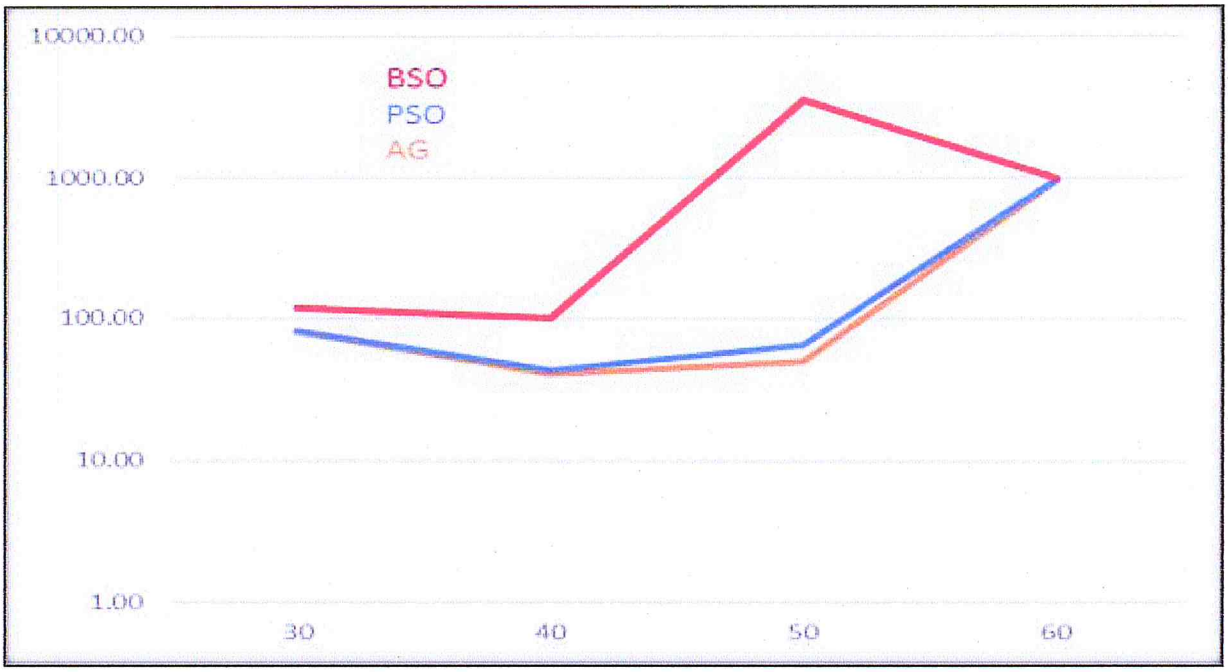


Figure 5.8 : Temps en fonction des items

7. Conclusion

Dans ce chapitre nous avons présenté notre travail qui consiste en l'implémentation de nos trois algorithmes ou nous avons testé leurs temps d'exécution par rapport a six paramètres d'entrer bien précis (nombre de transactions, nombre des items, nombres de population, seuil et taux de mutation et croisement) ou nous avons remarqué que le temps d'exécution augmente quand le nombre paramètres augmentes.

Conclusion Générale

1. Conclusions

L'extraction des motifs fréquents à partir des données incertaines dans l'environnement mobile devient de plus en plus importante. Aussi, la communauté scientifique cherche développe de nouveaux outils et méthode de traitement. Au cours de notre travail nous avons vu les différentes méthodes d'extraction d'itemsets fréquents. Et afin de comprendre et cerner la problématique de l'extraction d'itemsets fréquents à partir de données incertaines dans l'environnement mobile, nous avons étudié les différents algorithmes proposés dans la littérature dans ce contexte.

Les méthodes d'optimisation BSO, BPSO, et AG sont des métaheuristiques d'optimisation qui sont inspirées de la nature, ils sont représentés par des algorithmes qui peuvent être appliqués à de nombreux problèmes d'optimisation dans le management, l'ingénierie, et le contrôle.

Il y a des choses communes entre les AG, BPSO, et BSO. Néanmoins, chaque une a son principe d'exploration de l'espace de recherche et le mécanisme d'optimisation pour trouver la solution optimale.

Des résultats préliminaires ont montré que le développement de nouveaux modèles basés sur les principes des algorithmes d'extraction des motifs fréquents doit certainement contribuer à la résolution des problèmes assez complexes appliqués dans environnement mobiles.

Ce travail nous a permis d'avoir des nouvelles connaissances sur les nouvelles technologies d'extraction d'itemsets fréquents dans le mobile, des notions sur les problèmes par rapport la version desktop.

L'étude effectuée représente une ébauche à des solutions qui peuvent être efficaces dans le contexte de l'extraction des motifs fréquents dans des environnements mobiles. Ce domaine qui est encore dans ses débuts.

2. Perspectives

Les perspectives de ce travail préliminaire sont nombreuses comme :

Adapter une nouvelle structure plus optimale par rapport à notre méthode utilisée dans nos trois algorithmes utilisés :

- AG-UFIM :

✓ Elimination des itemsets qui ne sont pas valide.

✓ Reproduction des items valides.

- BSO-UFIM :

✓ Développer une meilleure méthode dans la recherche locale de l'abeille dans sa zone de recherche.

En ce qui concerne la mesure de calcule utilisé qui est l'Expected support ou on pourra développer une nouvelle mesure pour optimiser dans le balayage de la base de données.

Bibliographie

Bibliographie

[01] T. Green and V. Tannen, "Models for Incomplete and Probabilistic Information," Data Eng. Bull., vol. 29, no. 1, 2006.

[02] Tanagra , FREQUENT ITEMSETS . Octobre 2011.

[03] R. Lovin, "Mining Frequent Patterns", Avril 2012.

[04] R. Agrawal and R. Srikant. Fast algorithms for mining association rulesin large databases. In Proceedings of the 20th international conferenceon Very Large Data Bases (VLDB'94), pages 478-499. Morgan Kaufmann, September 1994.

[05] Leung, C.K.-S., Hao, B.: Mining of frequent itemsets from streams of uncertain data. In: IEEE ICDE, pp. 1663–1670 (2009).

[06] Tan, P.-N., Chawla, S., Ho, C.K., Bailey, J. (eds.) PAKDD 2012, Part II. LNCS (LNAI), vol. 7302, pp. 322–334. Springer, Heidelberg (2012).

[07] C. C. Aggarwal, Y. Li, J. Wang, and J. Wang. Frequent pattern mining with uncertain data. In KDD, pages 29–38, 2009.

[08] Leung, C.K.-S.and S.K.Tanbeer. "PUF-Tree: A Compact tree structure for frequent pattern mining of uncertain data." In: J. Pei, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD LNCS (LNAI), vol. 7818, pp. 13–25. Springer, Heidelberg 2013.

[09] Tan, P.-N., Chawla, S., Ho, C.K., Bailey, J. (eds.) PAKDD 2012, Part II. LNCS (LNAI), vol. 7302, pp. 322–334. Springer, Heidelberg (2012).

[10] Sanjay Madria and Takahiro HaraBig .Data analytics and knowledge discovery : 17th International Conference, DaWaK 2015, Valencia, Spain, September 1-4, 2015.

[11] Thomas Bernecker, Hans-Peter Kriegel, Matthias Renz, Florian Verhein, Andreas Zuefle. 15th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining (KDD'09), pp. 896-905, Paris, France, 2009.

[12] Lu, E.-C., Lee, W.-C., and Tseng, V.S.: 'A framework forpersonal mobile commerce pattern mining and prediction',Knowledge and Data Engineering,., IEEE Transactions on, 2012, 24,(5).

[13] Wang, L., Gu, T., Tao, X., and Lu, J.: 'A hierarchical approach to real-time activity recognition in body sensor networks', Pervasive and Mobile Computing, pp. 115-130., 2012, 8, (1).

[14] Muhammad Habib ur Rehman, Chee Sun Liew, Teh Ying Wah, Frequent pattern mining in mobile devices: A feasibility study, 2014 International Conference on Information Technology and Multimedia (ICIMU), pp. 351-356, Putrajaya, Malaysia, November 18 – 20, 2014.

[15] Agrawal, R., and Srikant, R.: 'Fast algorithms for mining association rules', in Proceedings of 20th International Conference on Very Large Data Bases, VLDB (Vol. 1215, pp. 487-499.

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

[17] Borgelt, C.: 'Keeping things simple: Finding frequent item sets by recursive elimination', In Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations, pp. 66-70.

[18] Zaki, M.J.: 'Scalable algorithms for association mining', Knowledge and Data Engineering, IEEE Transactions on, 2000, 12, (3), pp. 372-390.

[19] Zaki, M.J., and Gouda, K.: 'Fast vertical mining using diffsets', In Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 326-335.

[20] Lin, Yi, et al. "Making In-Memory Frequent Pattern Mining Durable and Energy Efficient." Parallel Processing (ICPP), 2016 45th International Conference on. IEEE, 2016.

[21] Srinivasan, Vijay, et al. "Mobileminer: Mining your frequent patterns on your phone." ,Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing. ACM, 2014.

[22] Evangeline, S. Jacinth, K. M. Subramanianl, and K. Venkatachalam. "Efficiently Mining the Frequent Patterns in Mobile Commerce Environment International Journal of Innovation and Scientific Research ISSN 2351-8014 Vol. 5 No. 1 Jul. 2014, pp. 30-39.

[23] E. A. Feigenbaum and J. Feldman. (Edirors). Computers and thought. McGraw-Hill Inc. pp.192. New York, 1963.

[24] J. Pearl. Heuristics: intelligent search strategies for computer problem solving. pp. 3. Addison-Wesley Publ. Co, London, 1984.

[25] I.H. Osman, G. Laporte. Metaheuristics: A bibliography. Ann. Oper. Res. Vol. 63, N° 5, pp. 513-623, 1996.

[26] R. Benabid , "Optimisation Multi-objectif de la Synthèse des FACTS par les Particules en Essaim pour le Contrôle de la Stabilité de Tension des Réseaux Electriques," Université de Laghouat mémoire de Magister, 2007.

[27] Dorigo. M., (1992). *Optimization, learning and natural algorithms*. Italy, PhD thesis, DEI, Politecnico di Milano.

[28] Eberhart R. C. and J. Kennedy, 1995. New optimizerusing particle swarm theory. Proceedings of the 6thInternational Symposium on Micro Machine and Human Science, pp. 39-43, Nagoya, Japan.

[29] D.T. Pham and A. Ghanbarzadeh. Multi-objective optimisation using the bees algorithm. IPROMS2007 - 2007 Innovative Production machines and systems, 2007.

[30] J.H. Holland. Genetic Algorithms and the optimal allocation of trials, SIAM Journal of

Computing. Vol. 2, N° 2, pp. 88-105, 1973.

[31] Goldberg, D.E., Genetic Algorithms in Search, Optimization and Machine Learning .Addison-Wesley : Reading, MA, 1989.

[32] Y.Djenouri ,H.Drias ,Bees swarm optimization using multiple strategies for association rule mining,USTHB,Vol6,Num4,2014.

جامعة البليدة
مكتبة
الكلية
كلية العلوم