REPUBLIQUE ALGEREINNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE SAAD DAHLEB

Faculté des Sciences

Département d'informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Master en informatique

Extraction des itemsets fréquents à partir des flux de données incertaines

Réaliser par :

Mr. Megdoud Aimene

Promotrice :

Mme. Zahra Fatma Zohra

2015/2016

Résume

L'extraction de motifs fréquents est une technique utilisée en fouille de données, elle s'appuie sur des principes relativement simples. Son objectif est de trouver les motifs qui apparaissent fréquemment dans un ensemble de données. Or, avec les flux de données, les données arrivent à n'importe quel moment d'une façon continue dans les tailles sont indéfinie. Ainsi notre travail consiste à proposer une méthode qui peut faire l'extraction des motifs fréquents incertains à partir des flux de données, en gagnant un gain temps et de consommation de mémoire.

Abstract

The extraction of frequent patterns is a technique used in data mining, it relies on relatively simple principles. his goal is to find the patterns that appear frequently in a data set. Now, with the data flow, data arrives at any time in the sizes are undefined and the data continues. So our job is to find a method that can extract the common reasons unclear from the data stream, earning a gain time and memory consumption.

Remerciements

Je remercie Dieu le tout puissant de m'avoir donné le courage et la volonté d'achever ce travail et sans Lequel il n'aurait jamais été accompli.

Mes 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 mon projet, et surtout pour la confiance qu'elle nous a accordé pour la réalisation de ce projet.

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

Je remercie les nombres de jury pour nous avoir fait l'honneur de juger notre travail.

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

Table des matières

Introduction général

Grâce aux techniques d'extraction des connaissances, les sources de données sont devenues des sources riches et fiables pour la génération et la validation de connaissances. Le Data Mining (fouille de données) est le noyau de processus d'extraction des connaissances, il couvre plusieurs domaines, l'analyse de données, les bases de données, l'apprentissage, les statistiques, les systèmes à base de règles. Il dispose d'outils performants afin de structurer et d'extraire des connaissances : la classification, la segmentation, la recherche de règle d'association, ...etc, dont l'extraction des itemsets fréquents à partir des différente sources de donnés fait partie.

1. Problématique

Le contexte dans lequel les travaux d'extractions des itemsets ont été définis ces dernières années considérait que les données, sur lesquelles la fouille était réalisée, étaient disponibles dans des bases de données statiques. Aujourd'hui, suite au développement de nouvelles applications, nous devons faire face à de nouveaux modèles dans lesquels les données sont disponibles sous la forme de flots. En outre, ces flots peuvent contenir des données incertaines.

Les divers méthodes et techniques d'extraction qui existent ont leurs avantages et leurs inconvenants, cependant l'extraction des motifs fréquents à partir des flux de données incertaines est une question difficile, vu que les données arrivent à n'importe quel moment et d'une taille illimité, les algorithmes classiques ne repend plus aux demandes. Pour cela des une catégorie d'algorithmes et des méthodes d'extraction de motifs fréquent à partir de flux de donnés est née.

2. Objectif du travail :

Notre travail consiste à développer une méthode qui permet d'extraire des itemsets fréquents à partir des flux de données incertaines. Notre méthode se base sur un algorithme inspiré d'une méta-heuristique connue appelé optimisation par colonies d'abeilles. Vu la nature des flots de données, cette méthode a été utilisée principalement dans le but de diminuer le temps nécessaire pour l'extraction des itemsets fréquents ainsi que l'espace mémoire.

3. Organisation de mémoire

Le mémoire se réparti en cinq grandes partie. Nous commençons tout d'abord par un état de l'art sur l'extraction de motifs fréquents. Nous expliciterons aussi les principaux de deux célèbres algorithmes d'extraction de motifs avec une comparaison entre ces algorithmes.

Dans le deuxième chapitre nous présenterons les flux de données incertains et leurs modèle et algorithmes d'extraction de motifs fréquents.

Dans le troisième chapitre nous allons parler des méta-heuristiques pour l'extraction des connaissances et leurs algorithmes conçu pour ça.

Dans le quatrième chapitre nous allons parler de notre solution et nous allons détailler nos idées et les méthodes utilisées.

Dans le cinqième chapitre enfin nous validons la méthode adoptée, et on termine par une conclusion générale.

CHAPITRE I

Extraction des motifs fréquents à partir des données

1) INTRODUCTION

L'exploration de données, connue aussi sous l'expression de fouille de données, forage de données, prospection de données, *data mining*, ou encore extraction de connaissances à partir de données, a pour objet l'extraction d'un savoir ou d'une connaissance à partir de grandes quantités de données, par des méthodes automatiques ou semi-automatiques [1].

Elle se propose d'utiliser un ensemble d'algorithmes issus de disciplines scientifiques diverses telles que les statistiques, l'intelligence artificielle ou l'informatique, pour construire des modèles à partir des données, c'est-à-dire trouver des structures intéressantes ou des motifs selon des critères fixés au préalable, et d'en extraire un maximum de connaissances.

2) L'EXTRACTION DES ITEMSETS

La recherche des régularités dans les bases de données est l'idée principale du data mining. Ces régularités s'expriment sous différentes formes. Dans l'analyse du panier d'achats de consommateurs par exemple, l'extraction des itemsets consiste à mettre en exergue 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 »[2] [3] [4].

Définissons les termes utilisés dans ce contexte :

- **Item** : Un item [03] correspond à un produit. Nous avons 4 items (S1, S2, S3 et S4) dans notre fichier.
- **Support** : Le support [03] d'un item est égal au nombre de transactions dans lesquelles il apparaît.
- **Itemset** : Un itemset [03] 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.
- **Itemset fréquent** : Un itemset est dit fréquent [03] si son support est supérieur à un seuil défini à l'avance, paramètre de l'algorithme de recherche.
- **Superset :** Un superset [03] est un itemset défini par rapport à un autre itemset.
- **Itemset fermé (closeditemset)** : Un itemset fréquent est dit fermé [03] si aucun de ses supersets n'a de support identique. Autrement dit, tous ses supersets ont un support strictement plus faible.
- **Itemset maximal (maximal itemset)**. Un itemset est dit maximal [03] si aucun de ses supersets n'est fréquent.
- **Itemset générateur (generatoritemset)** : Un itemset A est dit générateur [03] s'il

N'existe aucun itemset B tel que B ⊂A et que SUP(B) = SUP(A). Autrement dit, l'itemsetest générateur si tous ses sous itemsets ont un support strictement supérieur

3) LES ALGORITHMES D'EXTRACTION DES ITEMSETS A PARTIR DES DONNEES CERTAINES

Les algorithmes d'extraction de motifs fréquents peuvent être catégorisés suivant la manière avec laquelle l'algorithme explore l'espace de recherche des motifs. En effet, nous avons deux classes d'algorithmes : Algorithmes de type « générer et tester » et de type « diviser pour régner ». Dans cette section, on va introduire un l'algorithme représentatif de chaque classe.

a) LES ALGORITHMES DE TYPES « GENERER ET TESTER »

Générer et tester est une méthode de résolution de problème [5]. La méthode consiste à générer l'ensemble des solutions candidates et à vérifier *a posteriori* si chacune des solutions candidates générées est une solution valide du problème, et l'un des algorithmes connu dans cette catégorie est:

ALGORITHME APRIORI :

L'algorithme **A Priori** est un algorithme d'exploration de données conçu en 1994, par Rakesh Agrawal et Ramakrishnan Sikrant [6], dans le domaine de l'apprentissage des règles d'association. Il sert à reconnaitre des propriétés qui reviennent fréquemment dans un ensemble de données et d'en déduire une catégorisation.

L'algorithme Apriori s'exécute en deux étapes :Soient minsupp l'indice de support minimum donné, et minconf l'indice de confiance donné.

- Génération de tous les itemsets fréquents
- Génération de toutes les règles d'associations de confiance à partir des itemsets.

b) LES ALGORITHMES DE TYPE « DIVISER POUR REGNER » :

C'est une technique adoptée pour l'exploration de l'espace de recherche. 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. [7].

FP-TREE ALGORITHME :

L'algorithme **FP-tree** apporte ainsi une solution au problème de la fouille de motifs fréquents dans une grande base de données transactionnelle. En stockant l'ensemble des éléments fréquents de la base de transactions dans une structure compacte, on supprimant la nécessité de devoir scanner de façon répétée la base de transactions. De plus, en triant les éléments dans la structure compacte, on accélère la recherche des motifs.

Cette méthode s'appelle **FP-tree** (*FrequentPattern growth*). Elle consiste d'abord à compresser la base de données en une structure compacte appelée **FP-tree** (*Frequent Pattern tree*), puis à diviser la base de données ainsi compressée en sous projections de la base de données appelées bases conditionnelles [8].

c) **AVANTAGES ET INCONVENIENTS**

L'avantage majeur de l'algorithme est qu'il ne fait que deux balayages de la base des transactions. Le premier balayage étant pour trouver les k-itemset puis trier le résultat pour construire la liste des items fréquents dans T en ordre de fréquence. Le deuxième balayage ayant pour but la construction de la structure FP-tree. Aussi il peut être qualifié de :

- Complet :
    - puisqu'il contient toutes les informations sur les éléments fréquents
    - à chaque élément fréquent correspond un chemin dans l'arbre.
- Concis :
    - puisqu'il ne contient que les éléments fréquents. les items sont classés en ordre de fréquence décroissante.
    - Plus un item est fréquent, plus il sera utilisé par les itemsets fréquent. Par conséquent son nœud sera utilisé souvent.

Néanmoins, malgré sa structure compacte, cela ne garantit pas, dans le cas où la base de transactions est trop volumineuse, que toute la structure du FP-tree tiendra en mémoire centrale.

De plus la construction de la strcuture FP-tree peut s'avérer longue et pourrait consommer beaucoup de ressources du système [8].

4) EXTRACTION D'ITEMSETS FREQUENTS A PARTIR DE DONNEES INCERTAINES

En informatique, les données incertaines sont des données qui contiennent du bruit qui fait dévier des valeurs correctes, prévues ou originales. À l'ère des Big Data, l'incertitude ou la véracité des données est l'une des caractéristiques déterminantes de ces dernieres. Les données incertaines se trouvent en abondance aujourd'hui sur le web, les réseaux de capteurs, et au sein des entreprises, tant dans leurs sources structurées et non structurées. Afin de prendre des décisions fiables sur la base des données issues du monde réel, Les méthodes utilisées doivent nécessairement tenir compte des différents types d'incertitude présents dans ces masses de données. Les données incertaines peuvent provenir de multiples sources tels que:

- Erreurs de mesure
- Intégration de données de multiples sources
- Processus automatiques imprécis (extraction d'information,
- traitement du langage naturel. . .)
- Jugement humain imparfait

Les analyses basées sur des données incertaines auront un effet sur la qualité des décisions ultérieures, de sorte que le degré et le type d'inexactitudes dans ces données incertaines ne peuvent pas être ignorés. En effet, on trouve dans la littérature dédiée à l'extraction de motifs fréquents qui font l'objet de notre travail plusieurs algorithmes qui prennent en charge les données incertaines.

4.1) LES ALGORITHMES DE TYPE GENERER ET TESTER L'ALGORITHME U-APRIORI

L'U-Apriori est la version incertaine de l'algorithme Apriori. Créer par par R. Agrawal Pour faire face au problème d'extractions des items fréquents à partir des données incertaines. En utilisant le model probabiliste, l' *expected support* d'un itemset **X** dans un ensemble de données T de taille D, noté $expSup(\mathbf{X}, T)$ est calculé en utilisant la formule suivante:

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

Tels que $P(x, T_i)$ est la probabilité existentielle de l'item x (voir la **Figure 1**)

TID	items
1	{a:0.4, b:0.8, e:0.2}
2	{c:0.9, e:0.7, f:0.1, d:0.4}
3	{a:0.8, b:0.8}
4	{c:0.3, e:0.3, l:0.5, m:0.8}
5	{a:0.2, b:0.9, f:0.3, d:0.9}

Probabilistic dataset (Uncertain data)

Figure 1 : ensemble de données probabilistes (incertaines)

U-APrioriAlgorithm

Entrée: context β ; seuil minimal de support minsupport;
Sortie: ensembles F_k des k-itemsets frequents;

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 $c_k \times c_{k+1} \geq SM$
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_kF_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*.

- **Déroulement de L'Algorithme U-Apriori :**

Étape-1: Scannez la base de données de transaction pour obtenir le support S de chaque1-itemset, comparer avec minsup (MS), et d'obtenir un ensemble de 1- itemsets fréquentes.

Etape-2: Utiliser k-1. itemsets pour générer un ensemble de candidats k-itemset. Et utiliser la propriété Apriori pour filtrer les ensembles non fréquentés k-itemsets de cet ensemble.
Étape-3: Scannez la base de données de transaction pour obtenir le support S de chaque k-itemsets de l'ensemble candidat, comparer S avec MS pour un ensemble k-itemsets fréquents.

Étape-4: Pour chaque item fréquent de l'ensemble l, générer tous sous-ensembles non vides de l.

Étape-5: Générer les règles d'association satisfaisant un paramètre de confiance donnée.

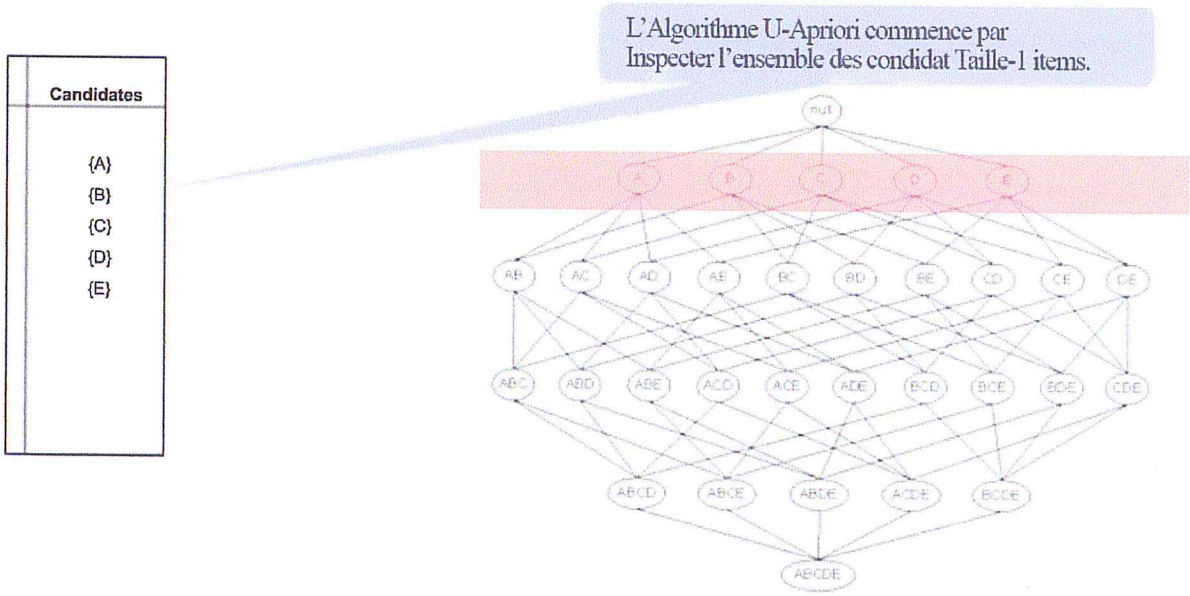

Figure 2 : Inspection des items candidats par L'Algorithme U-Apriori –

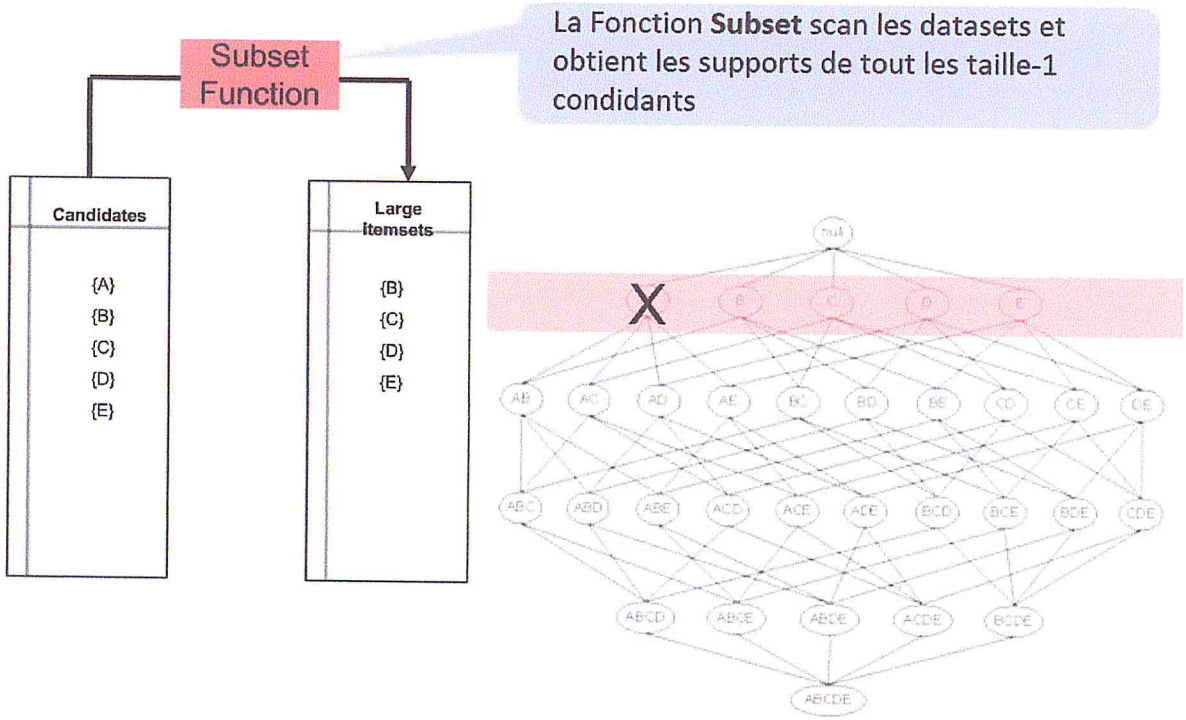

Figure 3 : les supports des itemsets de L'Algorithme U-Apriori

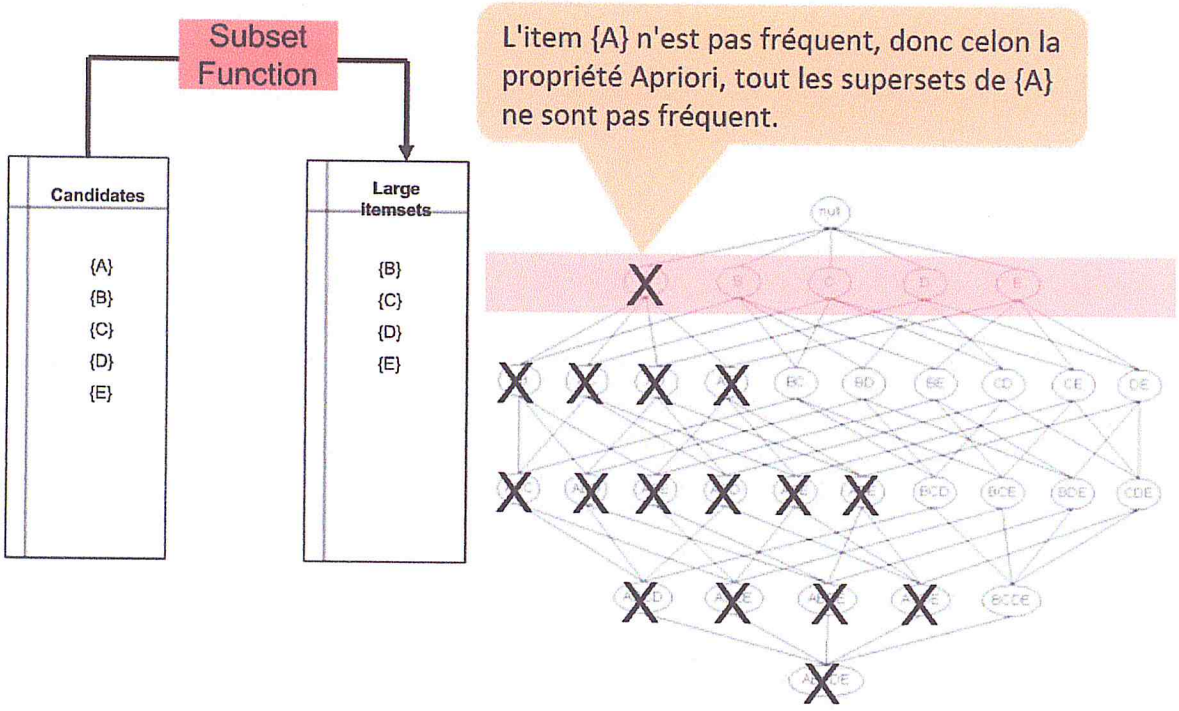

Figure 4 : Elimination des items et leurs super-sets qui ne sont pas fréquent

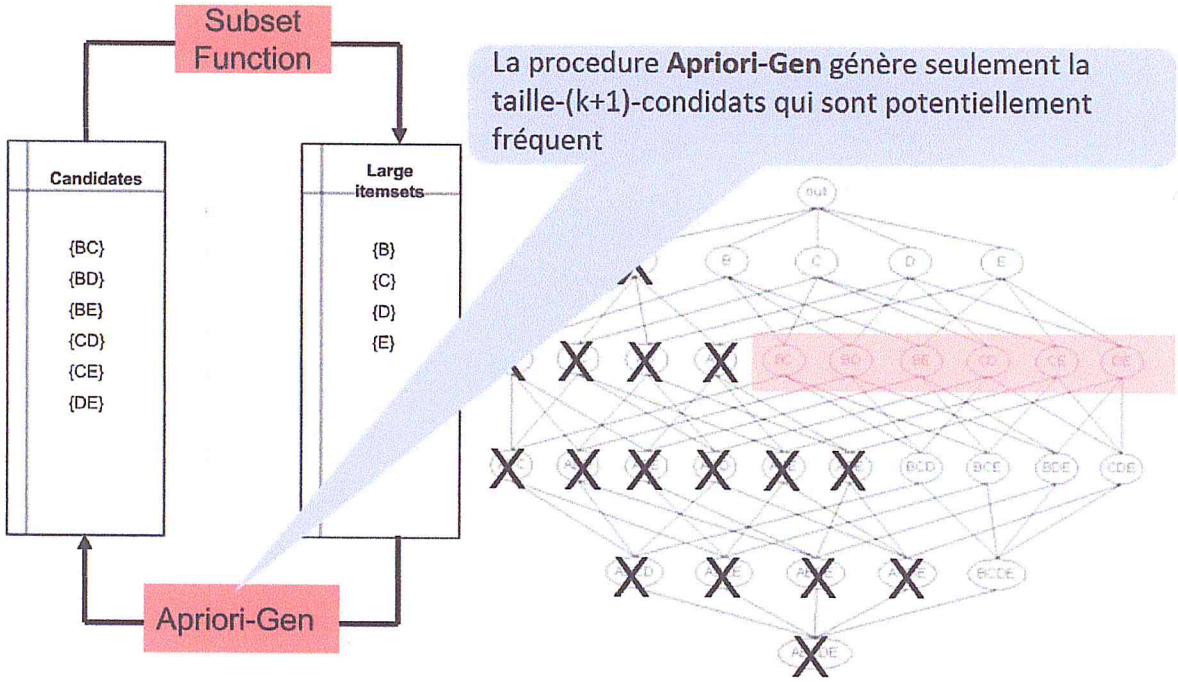

Figure 5 : Génération des candidats qui sont potentiellement fréquent -

1 (90%)	2 (80%)	4 (5%)	5 (60%)	8 (0.2 %)	991 (95%)

Le support estimé de {1, 2} contribué par la transaction 1 est 0.9 x 0.8 = 0.72

Les items-candidats	Support estimé
{1 ,2}	0.72
{1 ,5}	0.54
{1 ,8}	0.0018
{4 ,5}	0.03
{4 ,8}	0.0001

-Résultat de chaque item-candidat (Support estimé)-

On appelle cette modification l'algorithme Apriori, ceci sert comme une approche de brut-force dans le domaine du data-mining des données incertaines.

L'Algorithme U-Apriori a traversé l'arbre de hachage et à trouver une entité correspondante a {4, 8} Donc il incrémente le support.

4.1) Les Algorithmes de type Diviser-pour-régner

- **Algorithme UF-croissance (UF-growth algorithm)**

Pour représenter les données incertaines efficacement, 'UF-arbre, est une variante de FP-arbre. Chaque nœud dans l'UF-arbre stocke un item, son « expected support » 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 support minimal minsup

Sortie : UF-arbre, arbre des motifs fréquents

Etape 1 : scanner la base de données et accumuler les supports estimés de chaque item pour trouver les items fréquents où le support estimé ≥ 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 [5].

5) LE DEVELOPPEMENT DES SOURCES DE DONNEES ET LES OUTILS DU DATA-MINING

De nos jours, les techniques d'exploration de données peuvent être utilisées dans des domaines complètement différents avec des objectifs bien spécifiques.

La génération de modèles à partir des données continues n'est pas un phénomène récent. Pour qu'il y-ait une création de modèle il faut qu'il y-ait une collecte de données [9].

Cependant le développement de la technologie et des Algorithmes ne cesse d'augmenter jusqu'à ce que ou les données sont devenu relative à tout moment et les sources sont devenu continue et d'une taille illimité. A n'importe quel moment on peut recevoir des données. Ce phénomène connu sous le nom de « Flux de données » (Data-stream).

Un flux de donnée (Data-stream), est une séquence infinie d'éléments générés de façon continue à un rythme rapide. Son caractère de rapidité nécessite des algorithmes capable de traité rapidement ces données dans un temps pre-réel [10].

DOMAINE D'APPLICATION DES FLUX DE DONNEES :

- Logs de sites Internet
- Tickets de communications téléphoniques
- Indices Boursiers
- Données de capteurs
    - Consommation
    - Trafic routier
    - Géologiques

Les flux de données peuvent être incertains, et pour cela on va faire une étude plus détaillé dans le chapitre suivant.

CONCLUSION

Un aperçu de base sur les algorithmes d'extraction de motifs fréquents à partir de données certaine et incertaines a été fait dans ce chapitre. Plusieurs variantes de ces algorithmes existantent dans la littérature. Dans le chapitre suivant on s'intéresse à l'extraction des itemsets fréquents à partir des flux de données incertaines.

CHAPITRE II

Algorithmes d'extraction des itemsets fréquents à partir des flux de données incertains

1) INTRODUCTION

Les flux de données incertaines sont des données continues et infinies qui changent habituellement avec le temps, ils sont générés par plusieurs dispositifs tels que des capteurs sans fil dans de nombreuses applications réelles telles que le suivi des patients, la surveillance de l'environnement, etc. Ces données sont la plupart du temps incertaines. Par conséquent, l'extraction des motifs fréquents à partir des flux de données incertaines est une question difficile en raison de deux propriétés principales [11]. La première propriété c'est l'arrivée continue des données, les données sont également sans limite. Par conséquent, nous ne pouvons pas analyser toutes les données pour extraire les motifs fréquents incertaines, alors que tous les algorithmes d'extraction de motifs fréquents à partir des données statiques doivent balayer au moins deux fois l'ensemble de données. La seconde propriété est l'évolution des distributions de données avec le temps, un motif fréquent par rapport au temps t peut devenir fréquent à l'heure t+1. Par conséquent, pour l'extraction des motifs à partir des flux de données incertains sont approprié aux algorithmes en demande.

2) EXTRACTION DES ITEMSETS A PARTIR DES FLUX DE DONNES INCERTAINES

Depuis 2009, les chercheurs ont proposé une variété d'algorithmes pour l'extraction des motifs fréquents (par exemple, les itemsets, motifs séquentiels, et les modèles de sous-graphe) à partir de flux de données incertaines. Le tableau 3 résume les recherches les plus représentatifs et les algorithmes proposés dans ce domaine de recherche.

Les flux de données sont traités en utilisant des modèles différents. Les algorithmes d'exploration des motifs fréquents à partir des données séquentiellement incertaines sont utilisés dans trois modèles généraux à savoir: modèle des fenêtres coulissantes, modèle de temps à la lumière, et le modèle de point de repère.

Tableau 3 : les recherches les plus représentatifs et les algorithmes proposés dans ce domaine de recherche.

Algorithme (Acronyme) [11]	Modèle d'incertitude	measure de Support	Stratégie d'exploration de l'espace de recherche	Structure de données	Format De la source de données	Model de traitement du Data streams	Exactitude
SUF-growth [11]	Théorie de probabilité	Expected support	Diviser-pour-regner	Tree (SUF-tree)	Horizontal data streams	Sliding widow	Exacte
UF-streaming [11]	Théorie de probabilité	Expected support	Diviser-pour-regner	Tree (UF-Tree + FP-Stream)	Horizontal data streams	Sliding widow	Approximative
TUF-streaming [11]	Théorie de probabilité	Expected support	Diviser-pour-regner	Tree UF-Tree	Horizontal data streams	Time fading model	Approximative
LUF-streaming [11]	Théorie de probabilité	Expected support	Diviser-pour-regner	Tree LUF-Tree	Horizontal data streams	Landmark model	Approximative
UHS-Stream [11]	Théorie de probabilité	Expected support	Diviser-pour-regner	Hyper-linked array (UH-struct) + Tree (IS-tree)	Horizontal data streams	Sliding widow	Approximative
TFUHS-Stream [59]	Théorie de probabilité	Expected support	Diviser-pour-regner	Hyper-linked array (UH-struct) + Tree (IS-tree	Horizontal data streams	Time fading model	Approximative
Lixin et al [12]	Théorie de probabilité	Probabilistique support	Genérer-pour-tester	-	Horizontal data streams	Sliding widow	Approximative
FEMP [13]	Théorie de probabilité	Probabilistic support	Genérer-pour-tester	PF-Tree	Horizontal data streams	Sliding widow	Exacte

a) LES ALGORITHMES BASE SUR LE MODELE DES FENETRES COULISSANTE

Dans ce modèle, le calcul de la fonction d'intérêt se fait sur une fenêtre de taille fixe dans le flux. Les éléments de la fin de la fenêtre sont retirés de l'étude tandis que les nouveaux éléments

du flux prennent leur place. Leng et Hao [11] ont proposé deux algorithmes à base des fenêtres coulissantes. Un algorithme exact appelé SUF-Growth et un algorithme approximatif appelé UF-streaming pour L'extraction des motifs fréquents à partir des flux de données incertaines. UF-streaming est inspiré du FP-Growth algorithme [12] utilisé pour l'extraction des itemsets fréquents à partir de données précises et statiques. L'algorithme UF-Growth qui extrait les itemsets fréquents à partir de données incertaines, fonctionne de la manière suivante:

- Il applique du premier lot aux transactions entrant dans un flux de données incertaines, un preMinsup (un seuil qui est inférieur au minimum habituel) pour trouver des itemsets qui ne sont pas vraiment fréquents (minsup support $\geq>$ preMinsup). preMinsup est utilisé pour éviter une ignorance non voulu d'un itemset. Comme les flux de données qui ne sont pas nécessairement réparties uniformément, un itemset X ayant preMinsup $\leq$ expSup (X) $<$ minsup est actuellement rare, mais peut devenir fréquente plus tard; donc, X n'aurais pas une valeur pour l'instant.
- Les itemsets découverts "fréquents" sont stockés et conservés dans une autre structure d'arbre appelé UF-courant, dans lequel chaque chemin représente un itemset «fréquent» et chaque noeud contient (i) l'article et (ii) une table de fenêtre (contenant une liste de w valeurs attendues de soutien, un pour chaque lot de transactions).

Les étapes d'extraction ci-dessus sont répétées lorsque chacun des lots de transactions suivantes dans le flux de données incertaines arrivent.

L'algorithme exact SUF-Growth, retourne à l'utilisateur seulement les itemsets vraiment fréquents (à savoir, ceux avec le soutien attendu $\geq$ minsup). Il construit d'abord un SUF-Tree, puis des chemins pertinents en utilisant une approche diviser pour régner. Dans SUF-tree, les éléments sont disposés selon un certain ordre canonique (par exemple, l'ordre lexicographique). L'ordre des éléments n'est pas affecté par l'évolution continue.

Xia et al. [12] ont proposé des algorithmes basés sur deux tableaux hyper-liés aux flux d'exploitation des données incertaines, UHS-Stream et TFUHS-stream. L'algorithme UHS-Stream traite les données de flux en lots. Comme UF-streaming, UHS-Stream applique d'abord un algorithme de type incertain, qui est l'UH-mine statique qui trouve les itemsets potentiellement fréquentes dans le lot en cours. Ensuite, ces jeux d'éléments sont maintenus dans une structure arborescente globale appelée IS-Tree dans lequel les motifs fréquents et des

sous-ensembles d'éléments fréquents sont maintenues (par exemple, des itemsets sous-fréquent peut devenir fréquente plus tard). Chaque noeud dans la structure IS-Tree contient: (i) de l'article, (ii) E [SEST (X)] (le support estimé et attendue des formes de jeu d'éléments, en traversant de la racine à ce noeud), et (iii) Δ (l'erreur maximale possible de E [SEST (X)]). A chaque itération de l'algorithme, les itemsets stockées dans l'IS-tree sont mis à jour en fonction des informations extraites du lot en cours de traitement. Lorsque les requêtes des utilisateurs pour extraire les motifs fréquents sont exécuté, l'algorithme UHS-Stream traverse l'IS-arbre et fournit tous les motifs fréquents.

Les algorithmes susmentionnés se basent sur « l'expected support » pour calculer la fréquence d'un itemeset. Le support probabiliste est une autre mesure de fréquence proposée pour calculer la pertinence d'un itemset, a également été utilisé par certains algorithmes pour l'extraction des motifs fréquents de flux de données incertaines.

Reza et al. [13] ont proposé la FEMP (exploitation rapide et exacte des flux de données probabilistes) algorithme pour l'extraction exacte des itemsets fréquents dans les flux de données probabiliste avec des fenêtres coulissantes. Lorsqu'une transaction est ajoutée ou supprimé de la fenêtre, FEMP peut efficacement mettre à jour le support probabiliste de l'itemset en utilisant des opérations de mise à jour développées par les auteurs. Il représente le premier travail qui découvre exactement itemsets fréquents probabilistes à partir de flux de données incertaines, ce qui signifie que la fonction de distribution exacte pour l'extraction des items probabilistes est suivie et calculé, à tout moment avec un faible temps de réponse. Un autre algorithme d'extraction de motifs fréquents probabiliste est proposé par Lixin et al. [14]. Il améliore l'algorithme Apriori pour obtenir des itemsets fréquents probabilistes, et il fait des calculs arithmétiques en temps polynômial pour obtenir la probabilité des motifs fréquents, puis stocke les résultats dans un PF-Tree. Il mine la nouvelle fenêtre de base et met à jour le PF-Tree pour obtenir les motifs fréquents probabilistes de la fenêtre coulissante en cours d'exécution.

b) LES ALGORITHMES BASES SUR TIME FADING

Le modèle « Time fading » vise à prendre le flux de données pour tenir compte de l'extraction des motifs fréquents tout en donnant plus d'importance aux données récentes en pesant à l'aide d'un facteur $\lambda \in (0,1)$, aux données qui vient dans le temps avec k étapes précédentes est λk

- Module d'extraction des itemsets qui utilise « UF-Growth » avec un preMinsup aux itemsets "fréquentes" de chaque lot des flux de données incertaines.
- Le module d'extraction Stream qui maintient ces modèles sub-frequent extraits de chaque lot du flux dans une structure arborescente appelée LUF-flux de sorte que chaque nœud de l'arbre représente un itemset et contient des informations sur son soutien attendu dans le courant (à partir du point de repère à l'heure actuelle).
- Module d'élagage qui élimine ces itemsets sub-frequent et qui ne sont pas vraiment fréquents au moment où les utilisateurs demandent l'obtention des repenses.

3) DISCUTION

Généralement les algorithmes d'extraction des motifs fréquents à partir des flux de données incertaines étudiés ci-dessus, font partie des algorithmes qui utilisent la théorie des probabilités pour modéliser l'incertitude et donc, les motifs fréquents sont calculés en utilisant le support d'exception et le probabilistique support. En outre, trois modèles de traitement des flux de données sont principalement adoptées par ces algorithmes. Chaque modèle a ses avantages et ses inconvénients lorsqu'il est utilisé pour le traitement de l'extraction des motifs fréquents à partir des flux de données incertaines - Le modèle de fenêtre coulissante est le modèle le plus utilisé, il favoris de nouvelles données par rapport les anciennes. Nous avons une fenêtre de taille fixe les données les plus anciens sont supprimées lorsque les nouveaux sont arrivés. Cela représente un avantage pour la taille des motifs insignifiants et inconvénient parce que nous avons une perte d'informations rapide.

- Le modèle de temps-fading prend en compte toutes les données en favorisant le nouveau lot de données en introduisant un facteur de fading. L'avantage est que nous disposons de données plus anciennes pour calculer les motifs fréquents, mais le problème c'est la taille croissante des structures de données qui conservent les informations des modèles.
- Le modèle Landmark prend en compte toutes les données à partir d'un point de repère. Ainsi, d'une part, nous n'avons pas une perte d'information, mais en revanche nous n'avons pas le luxe de la taille des données insignifiantes et donc, une taille croissante des structures de données qui stockent des informations.
  Par conséquent, chaque algorithme propose des solutions pour faire face à ces questions.

Les Algorithmes approximatifs tels que UF-streaming, TUF-streaming, LUF-streaming, UHS-Stream, et TFUHS-Streamd qui utilisent preMinsup <minsup pour l'extraction des motifs qui peuvent être non fréquents afin d'éviter l'élagage possible des motifs vraiment fréquents souffre de certains problèmes potentiels:

- Trouver la valeur appropriée au preMinsup n'est pas facile car l'algorithme peut manquer quelques itemsets fréquents. Surtout, quand les preMinsup ont été trop.
- Pour trouver des itemsets vraiment fréquentes, une étape de post-traitement et une structure de données supplémentaire pour stocker les itemsets minées sont nécessaires.
- Les algorithmes utilisent un mode «immédiat» pour l'extraction des motifs fréquents.

En conséquence, beaucoup de calculs pourraient être gaspillés, en particulier lorsque de nombreux demandes d'extraction de motifs faites par les utilisateurs.

4) CONCLUSION

Chaque modèle étudier dans ce chapitre à ces avantages et ces inconvenant. Le modèle à base de Landmark ses données octroie car à chaque fois il stock les données comme historique, contrairement aux algorithmes « à base de temps-fading » et « Les fenêtres coulissante ». Par conséquence leurs inconvenant c'est que le premier supprime les anciens données de ça structure et le deuxième favorise les nouveaux données sur les anciennes.

Cependant les recherches ne sont pas arrêter ici, les chercheurs ont développé d'autres algorithmes et des méthodes dans le domaine de l'intelligence artificiel et des méta-heuristiques que nous allons découvrir dans le prochain chapitre.

CHAPITRE III

Meta-heuristiques pour l'extraction des connaissances

1) INTRODUCTION

La majorité des problèmes d'extraction de connaissances peuvent s'exprimer comme des problèmes d'optimisation combinatoire. Or, de nombreux problèmes d'optimisation combinatoire sont NP-difficiles et ne pourront donc pas être résolus de manière exacte dans un temps "raisonnable" puisque la capacité de calcul des machines évolue linéairement alors que le temps nécessaire à la résolution de ces problèmes évolue exponentiellement. Lorsqu'on s'attaque à des problèmes réels, il faut se résoudre à un compromis entre la qualité des solutions obtenues et le temps de calcul utilisé.

Au milieu des années 1970 sont apparues des méthodes qui supervisent l'évolution de solutions fournies par des heuristiques. Ces méthodes assurent un compromis entre diversification – quand il est possible de déterminer que la recherche se concentre sur de mauvaises zones de l'espace de recherche – et intensification – on recherche les meilleures solutions dans la région de l'espace de recherche en cours d'analyse. Ces algorithmes ont été appelés "méta-heuristiques" et ont pour objectif de trouver des solutions dont la qualité est au-delà de ce qu'il aurait été possible de réaliser avec une simple heuristique [18].

Dans ce chapitre, nous introduirons des différentes méthodes classiques d'optimisation (Méta-heuristiques) classées en deux groupes : les méthodes à solution unique et les méthodes à population de solutions. Nous verrons, pour chaque méthode, ses différentes applications dans l'extraction de connaissances en insistant sur :

- la fonction d'évaluation,
- la représentation des solutions,
- les opérateurs.

2) LES META-HEURISTIQUES A SOLUTION UNIQUE

Les méthodes itératives à solution unique sont toutes basées sur un algorithme de recherche de voisinage qui commence avec une solution initiale, puis l'améliore pas à pas en choisissant une nouvelle solution dans son voisinage.

Nous présenterons ici les méthodes les plus utilisées et leur utilisation en extraction de connaissances: les méthodes de descente, le recuit simulé et la recherche tabou.

a) LES METHODES DE DESCENTE (HILL CLIMBING)

Les méthodes de descente sont assez anciennes et doivent leur succès à leur rapidité et leur simplicité [19][20]. A chaque pas de la recherche, cette méthode progresse vers une solution voisine de meilleure qualité.

La descente s'arrête quand tous les voisins candidats sont moins bons que la solution courante : c'est-à-dire lorsqu'un optimum local est atteint.

On distingue différents types de descente en fonction de la stratégie de génération de la solution de départ et du parcours du voisinage : la descente déterministe, la descente stochastique et la descente vers le premier meilleur.

Algorithme 1 Méthode de descente générique

Procédure : φ fonction de coût
Variable locale : S solution courante
Choix d'une solution initiale S_0 :
Solution courante $S \leftarrow S_0$:
(a.) Génération des candidats par voisinage :
Choix du meilleur candidat C :
if $\varphi(C) < \varphi(S)$ **then**
  $S \leftarrow C$:
  Aller en (a.) :
end if
return S

Algorithme de la méthode de descente générique.

b) LE RECUIT SIMULE (SIMULATEDANNEALING)

Le recuit simulé est une technique d'optimisation de type Monte-Carlo généralisé à laquelle on introduit un paramètre de température qui sera ajusté pendant la recherche. [21] Elle s'inspirées méthodes de simulation de Métropolies (années 50) en mécanique statistique.

L'analogie historique s'inspire du recuit des métaux en métallurgie : un métal refroidi trop vite présente de nombreux défauts microscopiques, c'est l'équivalent d'un optimum local pour un problème d'optimisation combinatoire.

Si on le refroidit lentement, les atomes se réarrangent, les défauts disparaissent, et le métal a alors une structure très ordonnée, équivalente à un optimum global.
La méthode du recuit simulé, appliquée aux problèmes d'optimisation, considère une solution initiale est rechercher dans son voisinage avec une autre solution de façon aléatoire.
L'originalité de cette méthode est qu'il est possible de se diriger vers une solution voisine de moins bonne qualité avec une probabilité non nulle.

Ceci permet d'échapper aux optima locaux. Au début de l'algorithme, un paramètre T, apparenté à la température, est déterminé et décroît tout au long de l'algorithme pour tendre vers 0.
De la valeur de ce paramètre va dépendre la probabilité P d'acceptation de solutions détériorâtes (plus la température T est élevée, plus cette probabilité sera forte).

c) LA RECHERCHE TABOU (TABUSEARCH)

La recherche Tabou a été introduite par F. Glover [23] et a montré sa performance sur de nombreux problèmes d'optimisation.
Les idées de bases de la recherche Tabou se retrouvent également dans le travail de P. Hansen [23]. Elle n'a aucun caractère stochastique et utilise-la notion de mémoire pour éviter de tomber dans un optimum local.

Le principe de l'algorithme est le suivant : à chaque itération, le voisinage (complet ou sous-ensemble de voisinage) de la solution courante est examiné et la meilleure solution est sélectionnée.
En appliquant ce principe, la méthode autorise de remonter vers des solutions qui semblent moins intéressantes mais qui ont peut-être un meilleur voisinage.

Le risque est de cycler entre deux solutions. Pour éviter les phénomènes de cyclane, la méthode à l'interdiction de visiter une solution récemment visitée. Pour cela, une liste taboue contenantes attributs des dernières solutions visitées est tenue à jour. Chaque nouvelle solution considérée enlève de cette liste la solution la plus anciennement visitée.
Ainsi, la recherche de la solution courante suivante se fait dans le voisinage de la solution courante actuelle sans considérer les solutions appartenant à la liste taboue.

Cette méthode ne s'arrête pas d'elle-même et il faut déterminer un critère d'arrêt en fonction du temps de recherche que l'on s'octroie.

Ce critère peut être, par exemple, l'exécution d'un certain nombre d'itérations ou la non-amélioration de la meilleure solution pendant un certain nombre d'itérations. Ainsi, tout au long de l'algorithme, la meilleure solution doit être conservée car l'arrête fait rarement sur la meilleure solution.

Il existe de nombreuses variantes de recherches Tabou impliquant des techniques plus ou moins [24].

Algorithme 3 Méthode générique de Tabou

Procédure : φ fonction de coût
Variables locales : S solution courante, liste tabou L, Meilleure solution M, itération courante K, nombre d'itération N
Paramétrages : Taille de la liste Tabou, Critère d'aspiration
Choix d'une solution initiale S_0 :
Solution courante $S \leftarrow S_0$:
Meilleure solution $M \leftarrow S$:
$K \leftarrow 0$:
while $K < N$ **do**
  $K \leftarrow K + 1$
  Mise à jour de L
  Génération des candidats E par opération de voisinage
  $C \leftarrow best(E)$
  if $(\varphi(S) < \varphi(M))$ OU C n'est pas tabou OU C vérifie l'aspiration **then**
    $S \leftarrow C$
  else
    $E \leftarrow E \setminus C$
  end if
end while
return S

Algorithme de la méthode générique de Tabou.

3) LES META-HEURISTIQUES A POPULATION DE SOLUTIONS

Les méthodes d'optimisation à population de solutions améliorent, au fur et à mesure des itérations, une population de solutions. L'intérêt de ces méthodes est d'utiliser la population comme facteur de diversité. [18]

a) LES ALGORITHMES GENETIQUES (GENETIC ALGORITHMS)

Les algorithmes génétiques ont été largement utilisés dans le cadre de l'extraction de connaissances.
On les retrouve dans toutes les tâches principales. Leur application en sélection d'attributs surtout été réalisée en optimisation mono-objectif à l'aide de méthodes enveloppantes notamment avec le classifier K-Nearest-Neighbor, [22] [41] [42] avec un réseau de neurones [24] ou avec des tables de décisions euclidiennes [25][43]. En optimisation multi-objectif, les travaux sont plus rares. Leur application à la sélection d'attributs pour le clustering est relativement peu fréquente.

Les algorithmes génétiques sont régulièrement utilisés pour réaliser une tâche de clustering. Les différents algorithmes proposés diffèrent aussi bien par l'approche de clustering utilisée que par leur application à des données réelles. Ainsi il existe des travaux ayant comme approche la méthode des médïoides, la méthode des Kmeans, ou une approche hiérarchique
[26]. Les applications sont diversifiées : données spatiales, données d'expression génique et de données de bio-puces.

b) LES COLONIES DE FOURMIS (ANTS SYSTEM) :

L'histoire de l'intelligence en essaim remonte à l'étude du comportement de fourmis à la recherchée nourriture au départ de leur nid, par Goss, Deneubourg et leur équipe [31][32].
En se déplaçant du nid à la source de nourriture et vice-versa (ce qui, dans un premier temps, se fait Essentiellement d'une façon aléatoire), les fourmis déposent au passage sur le sol une substance Odorante appelée phéromone, ce qui a pour effet de créer une piste chimique.

Les fourmis peuvent sentir ces phéromones qui ont un rôle de marqueur de chemin : quand les fourmis choisissent leur chemin, elles ont tendance à choisir la piste qui porte la plus forte concentration de phéromones (Voir figure 1.2).

Cela leur permet de retrouver le chemin vers leur nid lors du retour. D'autre part, les odeurs peuvent être utilisées par d'autres fourmis pour retrouver les sources de nourriture détectées par leurs consœurs. Il a été démontré expérimentalement que ce comportement permet l'émergence

des chemins les plus courts entre le nid et la nourriture, à condition que les pistes de phéromones soient utilisées par une colonie entière de fourmis.

Le système de fourmis (Ants System - AS) est une méthode d'optimisation basée sur ces observations proposées par Dorigo. [27][28][29], Le système de fourmis a été employé avec succès sur des nombreux problèmes (voyageur de commerce, affectation quadratique, ...) mais les auteurs ont remarqué que l'AS n'a pas un comportement très exploratoire ce qui a conduit les auteurs à utiliser des hybridations du système de fourmis avec des recherches locales.
Les colonies de fourmis ont été utilisées en extraction de connaissances. On retrouve notamment leur utilisation pour effectuer des tâches de clustering. Ainsi, dans [30] Handl et Meyer proposent d'utiliser des colonies de fourmis pour regrouper des textes issus de moteur de recherche.

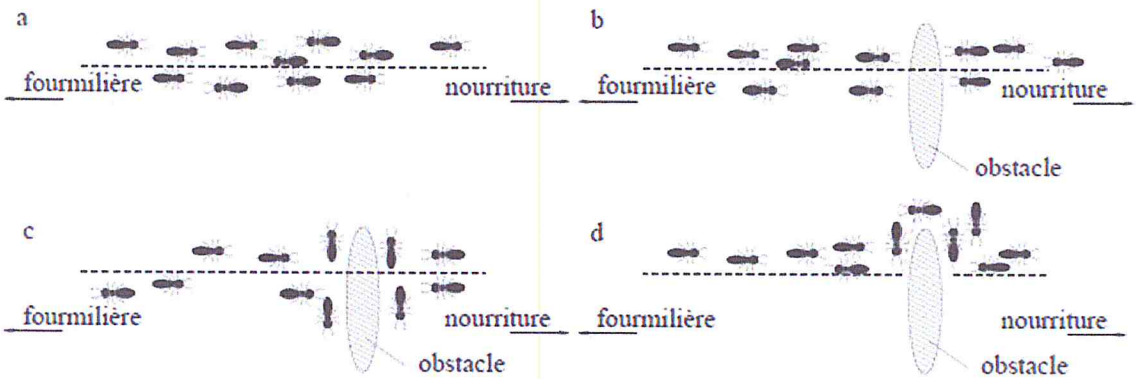

FIG. 1 – (a) Les fourmis suivent un chemin entre la fourmilière et la nourriture. (b) Un obstacle apparaît sur le chemin ; les fourmis choisissent entre prendre à droite et à gauche avec équiprobabilité.
(c) La phéromone s'évapore sur le chemin le plus long. (d) Toutes les fourmis choisissent le chemin le plus court [18].

Les fourmis utilisent les pistes phéromones pour marquer leur trajet par exemple entre le nid et une source de nourriture [18]. Une colonie est ainsi capable de choisir (sous certaines conditions) le plus court chemin vers une source à exploiter, sans que les individus aient une vision globale du trajet.

FIG. 2– Expérience de sélection des branches les plus courtes par une colonie de fourmis : « A » au début de l'expérience « B » à la fin de l'expérience.

ALGORITHME DE COLONIES DE FOURMIS DE BASE (ANT SYSTEM) :

Algorithme 4.1 Algorithme de colonies de fourmis de base : le "Ant System".

Pour $t = 1, \ldots, t_{max}$

  Pour chaque fourmi $k = 1, \ldots, m$

    Choisir une ville au hasard

    Pour chaque ville non visitée i

      Choisir une ville j, dans la liste J_i^k des villes restantes, selon la formule 4.1

    Fin Pour

    Déposer une piste $\Delta\tau_{ij}^k(t)$ sur le trajet $T^k(t)$ conformément à l'équation 4.2

  Fin Pour

  Évaporer les pistes selon la formule 4.3

Fin Pour

c) LES COLONIES D'ABEILLES :

Cette section décrit la biologie des abeilles, en se concentrant sur les aspects pertinente à des algorithmes d'optimisation.

- **ACCOUPLEMENT :**

Colonies d'abeilles domestiques contiennent une seule reine accouplée à un grand nombre de mâles et de des milliers de travailleurs. La reine est normalement le seul individu qui se reproduit dans la colonie, tandis que les travailleurs nettoyer le nid, occupe du fourrage pour l'alimentation et nourrir les couvains.

En raison de l'accouplement multiple par la reine, des ouvrières dans une colonie diffèrent génétiquement. Lorsque les travailleurs qui ne partagent pas le même père diffèrent dans leur seuil de la tâche, des résultats de spécialisation des tâches basées sur la génétique.

Les travailleurs issus du père A seront les premiers à démarrer la recherche de nourriture pour le pollen quand les réserves de pollen sont faibles. Parce qu'ils vont récolter le pollen avant la recherche du pollen lorsque seuil de travailleurs d'autres est atteint (par exemple, ceux qui n'ont pas désiré par le père de A), la majorité des butineuses de pollen seront des travailleurs qui partagent le même père. Cette diversité génétique est pensée pour permettre une colonie de répondre élastiquement à des changements dans l'environnement [18].
Les travaux empiriques ont montré que les colonies d'abeilles comprenant une main-d'œuvre diversifiée génétiquement à des résultats meilleurs.

- **COMPORTEMENT DE BUTINAGE :**

Les travailleurs non uniquement fourragent pour le pollen, ils ont aussi recueillir le nectar qui est stocké dans la colonie et devient miel. Alors que le pollen est utilisé assez rapidement car il est alimenté à le couvain en développement, [33] le nectar est stocké afin de permettre à la colonie de survivre à des périodes lorsque le fourrage n'est pas disponible. Les abeilles ont développé un mécanisme unique qui leur permet de recruter des membres de la colonie aux sources de nourriture trouvés et ça grâce à : le langage de la danse. L'utilisation du langage dedans permet une colonie d'exploiter rapidement et de monopoliser les sources de nourriture

rentables, tout en ignorant presque ceux qui sont de qualité médiocre. La danse des abeilles encode l'information sur la direction et la distance de la source de nourriture trouvée, les recrues potentielles, sont capables d'extraire cette information sur laquelle ils quittent la colonie et tenter délocaliser la source nourriture annoncé. Au cours d'une danse typique des progrès danseur avant de secouer vigoureusement son corps de droite à gauche [34].

Ceci est connu comme la «phase frétillante » de la danse. Après la phase de frétillement de l'abeille fait un virage brusque vers la gauche ou la droite, encerclant retour au début de la phase oscillante à nouveau. Ceci est connu comme la «phase retour danse frétillante'». À la fin de la seconde frétillante, l'abeille tourne danse sens opposé, de sorte qu'à chaque deuxième circuit de la danse, elle aura tracée fameux chiffre de motif huit de la danse frétillante voilà figure 3 qui montrent cette danse :

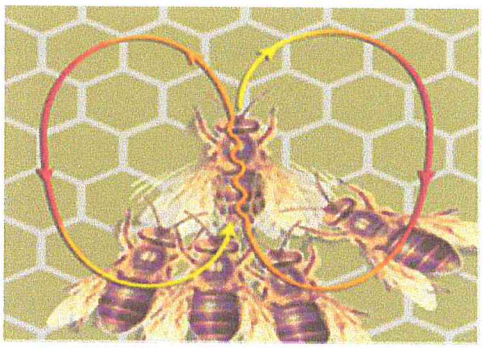

FIGURE 3 –Danse frétillante des abeilles

La phase la plus riche en information de la danse est la phase oscillante [33]. Pendant la phase oscillante l'abeille aligne son corps de telle sorte que l'angle de déviation de la verticale est similaire à l'angle de l'objectif de l'azimut actuel du soleil. La figure 5 montre cette notion sur direction :

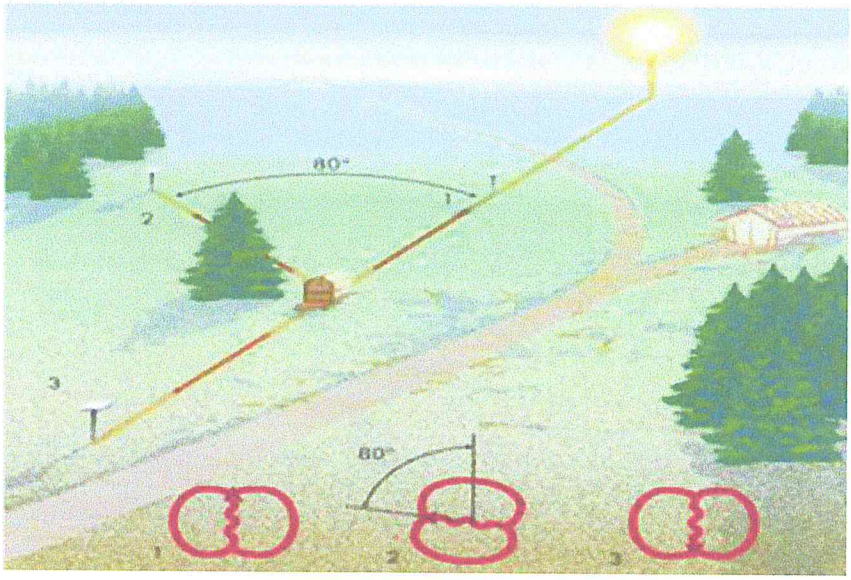

FIGURE 4: Direction du fourrage chez les abeilles

L'information de distance est codée dans la durée de la phase oscillante [34]. Danses pour les cibles proches ont des phases courtes frétillante tandis que des danses pour les cibles éloignées ont des phases frétillante prolongées.la figure 5 montre la relation entre nombre du tour et distance.

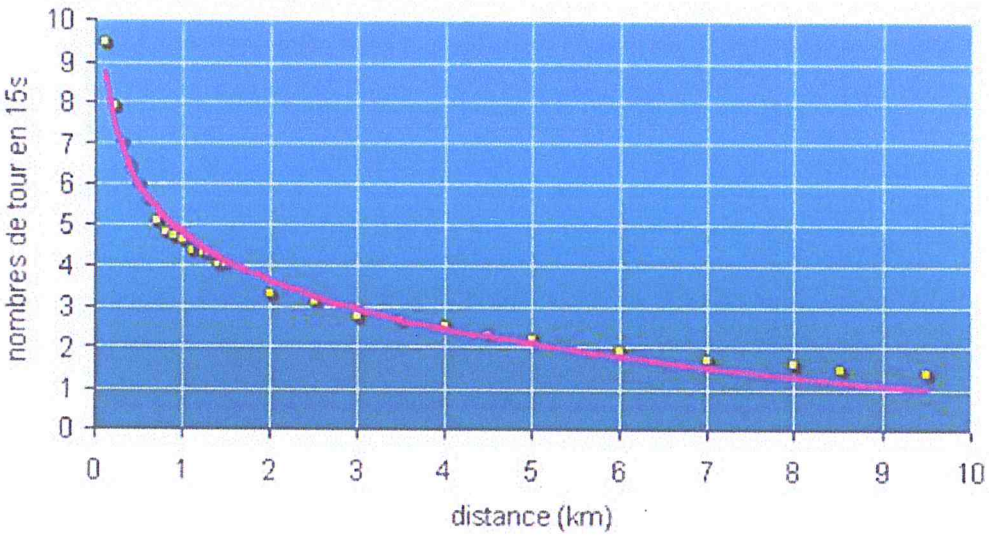

FIGURE 5–: Distance de la nourriture par rapport au nombre de tour de danse

- **MODELE DE COMPORTEMENT DE BUTINAGE :**

Tereshko développe un modèle de comportement de butinage d'une colonie d'abeilles sur la base des équations de réaction-diffusion Ce modèle minimal de la sélection du fourrage qui conduit à l'émergence de l'intelligence collective des essaims d'abeilles se compose de trois éléments essentiels : les sources de nourriture, employés butineurs et les non employés butineurs et le modèle définit deux modes principaux du comportement : le recrutement d'une source de nectar et de l'abandon d'une source [35].

- **SOURCES ALIMENTAIRES :**

La valeur d'une source de nourriture dépend de nombreux facteurs tels que sa proximité avec le nid, sa richesse, le goût de son nectar ou de la concentration de son énergie et la facilité d'extraction de cette énergie. Par souci de simplicité la «rentabilité » d'une source de nourriture peut être représentée par une seule quantité [36].

- **EMPLOYES BUTINEURS :**

Elles sont associées à une source de nourriture particulière dont ils sont actuellement exploitant ou qui sont «employés» là dans. Ils transportent avec eux des informations à propos de cette source particulière, sa distance et la direction du nid, la rentabilité de la source et de partager cette information avec une certaine probabilité [36].

- **NON EMPLOYES BUTINEURS :**

Ils sont sans cesse à regarder pour une source de nourriture à exploiter. Il existe deux types de non employé butineur : les scouts qui recherchent dans environnement du milieu de nid pour nouvelles sources de nourriture et des spectateurs qui attendent dans le nid d'établir une source de nourriture par le biais Del 'information partagée par les employés butineurs.
L'échange d'informations entre les abeilles est l'événement le plus important dans la formation de la connaissance collective. La partie la plus importante delà ruche à l'égard de l'échange

d'informations est la piste de danse. La communication entre les abeilles liées à la qualité des sources de nourriture a lieu dans la région de la danse. Cette danse s'appelle un la danse frétille. Depuis des informations sur toutes les sources de nourriture autour du riche est disponible pour un spectateur sur la piste de danse, probablement il peut assister à des danses de nombreuses employeurs et décide de se recruter à la source la plus rentable [37]. Il Ya une plus grande probabilité de assistant qui choisissent des sources les plus rentables depuis plus d'information est distribué sur les sources les plus rentables. Employés Butineurs partagent leurs informations avec une probabilité proportionnelle à la rentabilité de la source de nourriture, et le partage de cette information à travers la danse frétillante est plus longue dans la durée. Par conséquent, le recrutement est proportionnel à la rentabilité de la source des aliments. Afin de comprendre les caractéristiques de comportement de base de butineurs, nous allons examiner la figure 1. Supposons qu'il existe deux sources de nourriture découverts : A et B. Au tout début, un butineur potentielle commencera comme butineur chômeurs. Cette abeille n'aucune connaissance sur les sources de nourriture autour du nid. Il y-a deux options possibles pour une telle abeille :

- **(a)** Il peut être un éclaireur et commence à chercher autour du nid spontanément pour un aliment en raison de certaines démotivation interne ou indice externe possible(S sur la figure 1.7).

- **(b)** Il peut être une recrue après avoir regardé les danses frétillante et commence chercher une source d'aliment. (R sur la figure 1.7.Après avoir localisé la source de nourriture, l'abeille utilise sa propre capacité à mémoriser l'emplacement et commence alors

Immédiatement à l'exploiter. Ainsi, l'abeille va devenir une « butineuse travailleuse » [38]. La butineuse prend une chargée de nectar de la source et retourne à la ruche et décharge dans magasin d'alimentation. Après le déchargement de la nourriture, l'abeille a les trois options suivantes :

(a) Il devient un suiveur engagé après l'abandon de la source de nourriture (UF).
(b) Il danse, puis recrute camarades du nid avant de revenir à la même source alimentaire (EF1).
(c) Il continue de fourrage à la source de nourriture sans faire appel à d'autres abeilles (EF2).

Il est important de noter que pas tous les abeilles commencent butinage simultanément. Les expériences confirment que les nouvelles abeilles butinent commencent un taux proportionnel à la différence entre le nombre total d'abeilles et le nombre de nourriture présente.

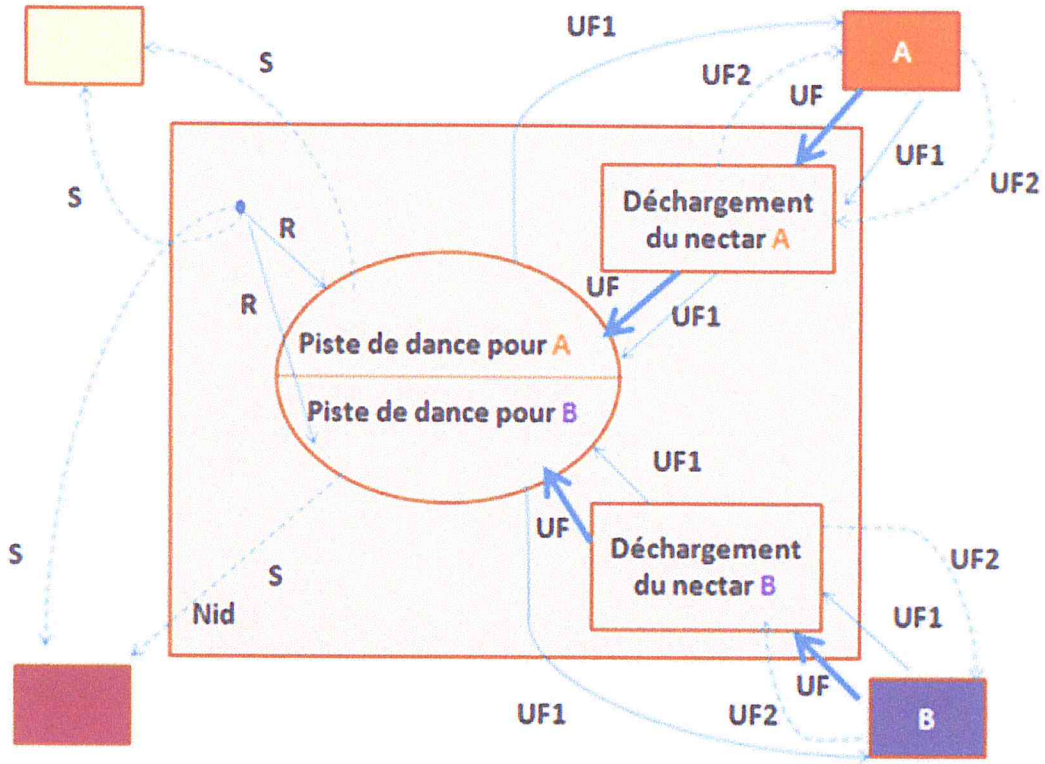

FIGURE1.7–: Comportement des abeilles mellifères butinent le nectar

- **ALGORITHMES D'INSPIRATION D'ABEILLES :**

Comme il est indiqué, les approches actuelles d'optimisation d'inspiration des abeilles sont basées sur l'un des deux comportements trouvés chez les abeilles : butinages ou de l'accouplement. Cette section décrit certains des algorithmes existants basés sur ces comportements.

- **ALGORITHMES D'OPTIMISATION A BASE D'ACCOUPLEMENT :**

Algorithmes d'optimisation basés sur accouplement puisent leur inspiration sur l'accouplement du comportement des reines des abeilles, la paternité mixte a été montré qu'elle améliorer performances d'une colonie, les chercheurs ont été inspirés pour développer algorithmes d'optimisation qui exploitent les principes qui sous-tendent le comportement d'accouplement de la reine [38]. Ces algorithmes sont également liés à la région de l'informatique évolutive car ils s'appuyer sur les principes de sélection, de croisement et la mutation.

- **ALGORITHMES A BASE COMPORTEMENT DE BUTINAGE :**

Des approches basées sur le Comportement de butinage s'inspirer des mécanismes sous-jacents au processus de butinage chez les abeilles. Outre les études expérimentales montre plusieurs support de modèles théoriques et de l'efficacité contour le processus décisionnel décentralisée de l'abeille lors la recherche de nourriture [39] Sherman et Visscher [40] ont étudié quand la danse- frétillante de recrutement est bénéfique : Les résultats suggèrent que ce recrutement augmente la quantité de nourriture d'une colonie lorsque les ressources sont rares. Beek-man et Lew [39] ont trouvé que le recrutement est le plus bénéfique si les succès moyen de localiser les correctifs nouveaux aliments tombe en dessous delà réussite moyen des recrues. En outre, ils ont montré que la communication

Facilite l'exploitation rapide des sources de nourriture très rentables lorsque les sources alimentaires de plusieurs qualités différentes sont présentes. Ainsi, la danse de communication des abeilles réglemente le compromis entre l'exploitation et l'exploration. Ces études soulignent l'utilité de comportement de butinage des abeilles en termes d'optimisation dans un environnement dynamique dans lequel

Les ressources sont rares et se distinguent de la qualité, comme c'est le cas dans de nombreux domaines problématiques d'optimisation. Inspiré par ces résultats des études du comportement du transfert d'information directe joue un rôle important dans des algorithmes basés sur la recherche de nourriture des abeilles qui sont exposées ci-dessous. Ceci est en contraste à des algorithmes d'optimisation de colonies de fourmis qui reposent sur communication indirecte par l'intermédiaire des phéromones artificielles [39].

- **ALGORITHME DE COLONIE D'ABEILLES ARTIFICIELLES (ABC) :**

L'algorithme de colonie d'abeille artificielle (ABC) a été introduit par Kara-boga [40] pour l'optimisation de fonction. Chaque solution représente Une position de nourriture potentielle dans l'espace de recherche et la qualité de la solution correspond à la qualité de la position alimentaire. Agents (abeilles artificielles) recherche d'exploiter les sources de nourriture dans l'espace de recherche.

L'ABC utilise trois types d'agents : les abeilles employés, les abeilles spectateur, et les scouts. Abeilles employé (EB) sont associés à des solutions actuelles de l'algorithme. À chaque étape de l'algorithme un EB tente d'améliorer la solution, il représente en utilisant une étape de recherche locale, après il va essayer de recruter des abeilles spectateur (OBs) pour sa position actuelle. OBs choisissent parmi les postes promus en fonction de leur qualité, ce qui signifie que de meilleures solutions attireront plus OBs. Une fois un OB a choisi un EB et donc une solution, il cherche à optimiser la position de l'EB par une étape de recherche locale. EB mise à jour sa position si un OB recruté était en mesure de repérer une meilleure position, sinon il reste sur sa position actuelle. En outre, un EB va abandonner sa position si elle n'était pas en mesure d'améliorer sa position pour certain nombre d'étapes. Quand un EB abandonne sa position, il devient un éclaireur, ce qui signifie qu'il sélectionne une position aléatoire dans l'espace de recherchée devient l'employé à cette position. L'algorithme peut être décrit plus en détail comme suit :

Algorithm 4 Algorithme de colonie d'abeille artificielle (ABC).

Paramètres : n, limite
la fonction objectif f (x), x = [x1, x2, ..., xd] T
Initialiser les positions alimentaires aléatoirement x_i i = 1, 2, ..., n
Initialiser les positions alimentaires aléatoirement x_i i = 1, 2, ..., n
évaluer fitness f (x_i) des individus
tant que condition d'arrêt n'est pas satisfait faire
phase employé :
Produire de nouvelles solutions avec k, j et ϕ aléatoirement

$$v_i^j = x_i^j + \phi i_i^j * (x_i^j - x_k^j) k \in 1, 2, ..., n, j \in 1, 2, ..., d, k \neq j, \phi \in [0, 1]$$

Évaluer les solutions
Appliquer le processus de sélection gourmande pour les abeilles employeuses.
phase spectateur :
Calcul des valeurs de probabilité pour des xi solutions

$$P_i = \frac{f_i}{\sum_{i=1}^{n} f_j} 2.8$$

Produire de nouvelles solutions de $x_i$sélectionné à l'aide P_i
Évaluer les solutions
Appliquer la sélection gourmande pour les spectateurs
phase de scout :
Trouver une solution abandonnée : Si la limite est dépassée, le remplacer par une nouvelle solution aléatoire.
Mémorisez la meilleure solution achevé
fin tant que
post-traitement des résultats et de visualisation

Algorithme de la méthode de colonie d'abeille [A4]

4) CONCLUSION

L'étude faite sur les Algorithmes précédente a montré que certains d'eux inspirer par la nature comme celui des abeilles ou des fourmis offre des méthodes et des mécanismes qui ont amélioré le domaine de l'extraction des connaissances, du côté de consommation de mémoire, de fluidité, et un gain temps gagné. Cependant ceci ne vas pas résoudre tous les problèmes du domaine, car chacun d'eux a des avantages et en même temps ses inconvenant.

CHAPITRE IV

Solution proposée

1) INTRODUCTION :

Les études faites dans les chapitres précédents ont montré des avantages et des inconvénients des algorithmes d'extraction de motifs incertains. Les chercheurs n'ont jamais cessés de trouver une solution qui est meilleure des précédentes et qui donne un coup d'avance au domaine de la fouille de données, avec une faveur de consommation bas de mémoire et une rapidité.

Dans ce chapitre on va essayer d'expliquer notre propre solution, qui est inspiré de deux méthodes étudié précédemment, qui sont, les flux de données et L'Algorithme de la colonie des abeilles.

Notre inspiration de l'algorithme de la colonie d'abeilles nous a laissé à prendre deux éléments importante de ces derniers : « L'exploitation des sources via des chemins manipulés » et « la communication rapide des informations » et si nous pouvons nommer notre petite idées ou notre système nommer « Les nano-abeilles ».

Ce système sera construit avec l'architecture «Multi-process » que nous parlerons dans la suite du chapitre.

2) LES NANO-ABEILLES :

Notre système consiste à construire des nano-abeilles c'est-à-dire des abeilles minuscule qui ne savent qu'extraire les motifs fréquents via des zones limités pour chacune d'elles. Ces zones sont calculées par le système général qui est responsable aussi de l'accueil de tous les résultats obtenu par nos nano-abeilles, l'extraction des motifs fréquents se fait via notre algorithme des nano-abeilles qui contient trois étapes que nous résumons dans le tableau 1.

Tableau 1 : Détails sur le fonctionnement de l'algorithme des nano-abeilles

Fonction	Détails
Etape 1	permet de récupérer la zone ou son nano-abeille commence à faire l'extraction, cependant elle retourne deux variables qui sont départ et arriver qui vont être utilisé par la fonction qui viendra ensuite, elle récupère aussi l'ensemble de solutions possibles, et le mini-support suggéré par l'utilisateur. Elle envoie un signal dès qu'elle finit de récupérer les paramètres pour permettre ainsi à L'étape suivante de commencé son travail.
Etape 2	permet d'extraire les motifs fréquents incertains à partir des flux de données, grâce à ses deux paramètres quelle prend « départ et arriver » elle permet de faire la fouille de données dans la zone spécifié en respectant le mini-support suggéré et le nombre de solutions possibles. Pour chaque solution possible un traitement de données et fait. Elle retourne le résultat obtenu et envoie un signal à l'étape qui s'occupe d'envoyé les résultats pour le système général qui s'occupe lui-même de réunir tous ces dernier.
Etape 3	Dès qu'elle reçoit un signal de la part de l'étape précédente, elle récupère les résultats dans ses paramètres et envoie ce résultat au system général cité précédemment. Elle envoie ainsi un signal après l'envoie des données à l'étape suivante.
Etape 4	dès qu'elle reçoit un signal de la part de la fonction précédente la « nano-abeille » termine sa tâche.

3) LES NANO-ABEILLES ET LES FLUX DE DONNES

Les données sont en état de croissance, ce qui signifie qu'elles ne sont pas statiques mais plutôt dynamiques, à chaque fois les données arrivent, ce qui signifie une difficulté d'extraction des motifs fréquents si à chaque fois on revient à calculer depuis le début cela deviendra quasiment impossible.

Les fenêtres coulissantes étaient toujours une bonne solution pour traiter les données arrivées simultanément. Mais les fenêtres coulissantes ont toujours une taille qui est fixe et qui représente le nombre de données à traité dans cette dernière. Cependant le traitement sera fait quand l'utilisateur voudrait le faire « Manuellement ».

De plus les données arrivées sont à chaque fois stocké l'un devant l'autre ce qui forme une longue queue de données.

Mais si nous essayons de mettre en place un mécanisme dans notre système général, qui permet à chaque fois que les données sont reçut dans flux de données, un générateurs de nano-abeilles sera exécuté, un nombre de ces derniers sera générais et seront suggéré à la fouille de ces données.

Notre système est composé de plusieurs sous-système et éléments qui permettent le fonctionnement correcte et rapide du système, et qui sont indispensable l'un a l'autre. La figure-2, explique le fonctionnement global de notre système proposé.

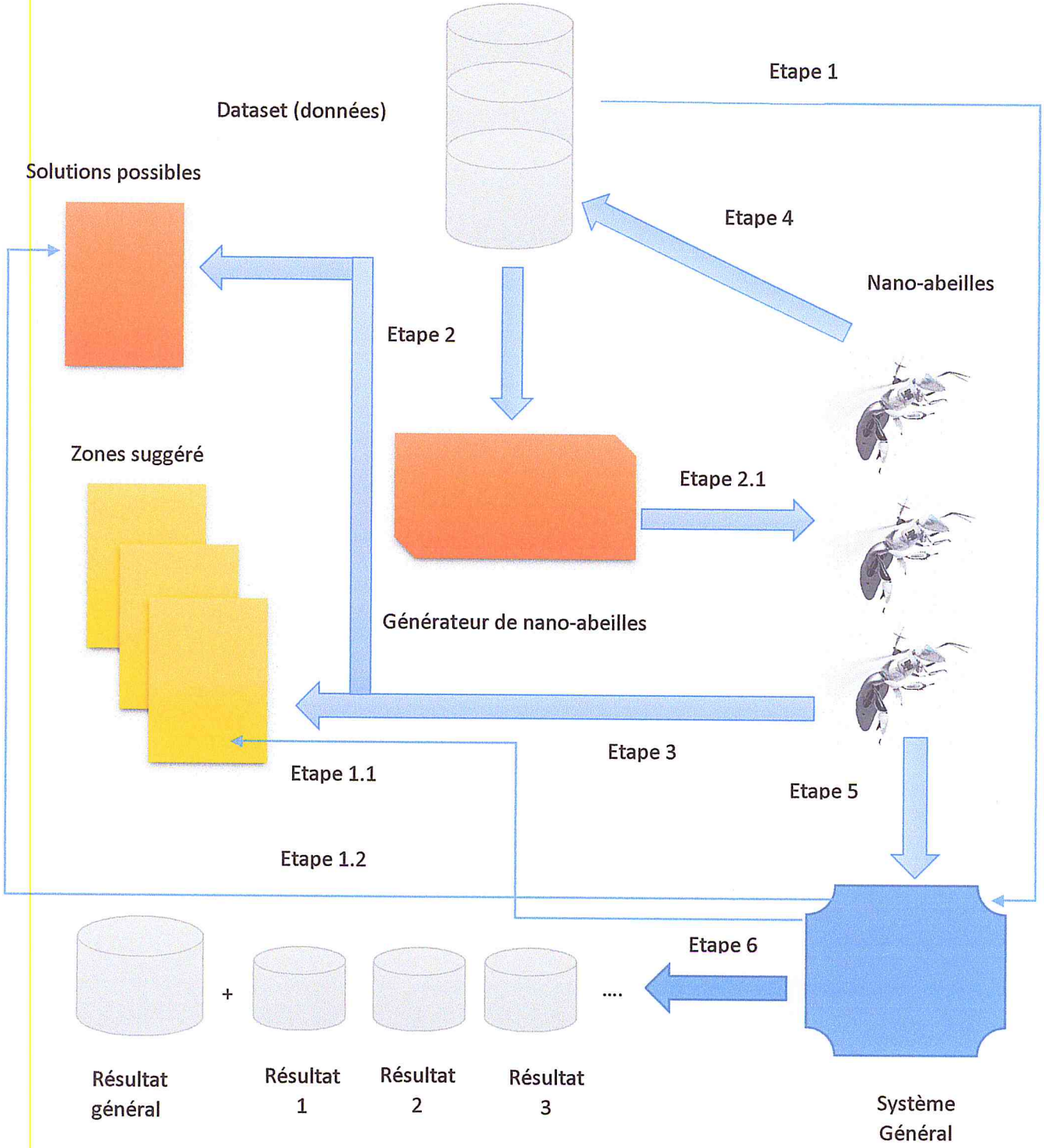

Figure 2 : Détail sur le fonctionnement du système proposé

Etape 1 : Dès que des données sont arrivées dans un temps t, un signal sera envoyé au système général.

Etape 1.1 : Le système général découpe les données reçu en plusieurs parties selon le nombre de nano-abeilles défini par l'utilisateur afin de facilité l'exploitation. Le tableau E1 explique cette étape.

Tableau E1 : Exemple de découpage d'un dataset de 60 000 transactions sur 6 nano-abeilles.

Numéro de Nano-abeille	Départ d'extraction	Fin d'extraction
1	0	10 000
2	10 000	20 000
3	20 000	30 000
4	30 000	40 000
5	40 000	50 000
6	50 000	60 000

Etape 1.2 : Le système général détecte les itemsets disponibles et génère une liste de solutions possibles grâce à un algorithme implémenté par nous-même, le tableau E2 explique cette étape.

Tableau E2 : Exemple de générations des solutions possibles dans un cas de 4 items, dans l'absence d'un item est égale à 0 et la présence vaut 1.

Nombre incrémentale	Conversion binaire	Egalité avec la solution finale.	Solution Final	Solution Final
1	0001	Pas d'égalité	1111	Incrémentation
2	0010	Pas d'égalité	1111	Incrémentation
3	0011	Pas d'égalité	1111	Incrémentation
15	1111	égalité	1111	Arrêt de l'incrémentation et génération des solutions

Etape 2 : Dès que des données sont arrivées, un signal sera envoyé au générateur des nano-abeilles.

Etape 2.1 : Une génération d'un nombre de nano-abeilles sera faite. Ce nombre est suggéré par l'utilisateur. La figure 3 nous explique cette étape.

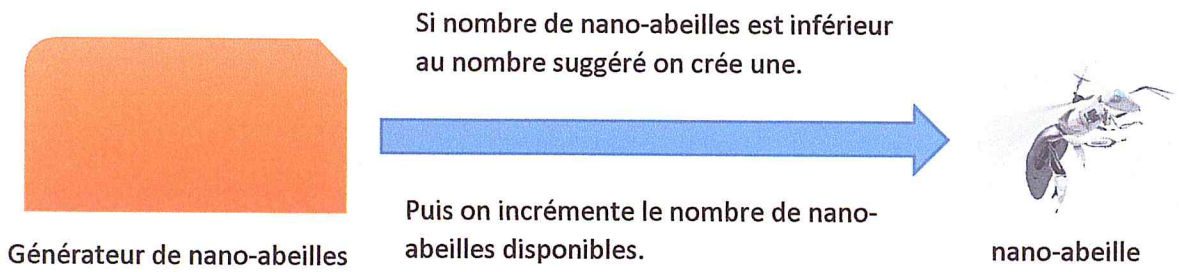

Figure 3 : Génération d'un nombre suggéré de nano-abeilles.

Etape 3 : Dès la création d'une nano-abeille, elle récupère la zone où elle doit faire l'exploitation des données, cette zone est suggérée par le système général dans l'étape du découpage des données.

Etape 4 : Après la récupération de la zone suggérée, l'extraction des motifs fréquents à partir de ces flux de données incertaines commence via des solutions suggérées par le système général dans « l'étape 1.2 ».

Etape 5 : Dès la fin de l'extraction, le résultat obtenu sera envoyé au système qui va aboutir à « l'étape 6 »

Etape 6 : Le système général fait réunir tous les résultats partiaux qui ont le même itemset pour obtenir un résultat général.

4) LES NANO-ABEILLES ET LE MULTI-PROCESS

Le Multi-process en informatique est une méthode ou les taches seront exécutées par des différents processus enfants, et les résultats seront communiqué au processus parent qui s'occupe lui-même d'organisé ces dernier afin d'avoir le résultat final.

Les avantages de cette méthode est son temps de rendu très rapides qui permet une accélération a la fouille de données, et sa consommation de mémoire réduite. La figure 3 résume le fonctionnement de la méthode Multi-process avec notre système.

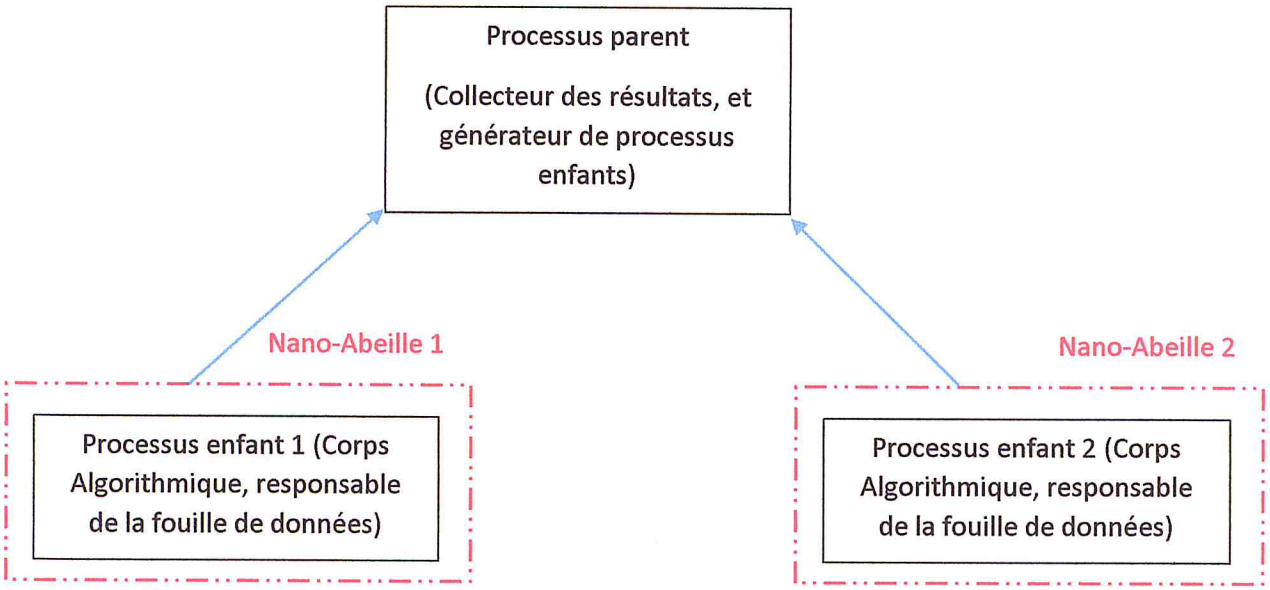

Figure 4 : Architecture Multi-process.

5) CONCLUSION :

Dans ce chapitre nous avons expliqué et détailler nos idée, la création d'un système unique qui se base sur l'extraction des motifs fréquents à partir des flux de données incertains et la communication des résultats avait besoin d'un coup de pousse de l'architecture Multi-process afin d'accélérer notre système des nano-abeilles et d'avoir un rendu estimable dans un temps rapide.

Dans le chapitre suivant nous allons tester notre solution et analysé les résultats, et ainsi de voir les avantages et les inconvenant de notre travail.

CHAPITRE V

Tests & Résultats

1) INTRODUCTION :

Après avoir décrit notre solution, nous aborderons dans ce chapitre la partie implémentation 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) ENVIRONNEMENT DE TRAVAIL :

C++ (Qt 5.5) : Qt (prononcé officiellement en anglais *cute* mais couramment prononcé *Q.T.* (Kiu ti):

- une API orientée objet et développée en C++ par Qt-Development-Frameworks, filiale de Digia. Qt offre des composants d'interface graphique (widgets), d'accès aux données, de connexions réseaux, de gestion des fils d'exécution, d'analyse XML, etc. ;
- par certains aspects, elle ressemble à un Framework lorsqu'on l'utilise pour concevoir des interfaces graphiques ou que l'on conçoit l'architecture de son application en utilisant les mécanismes des signaux et slots par exemple [45].

Qt permet la portabilité des applications qui n'utilisent que ses composants par simple recompilation du code source. Les environnements supportés sont les Unix (dont GNU/Linux) qui utilisent le système graphique X Window System ou Wayland, Windows, Mac OS X et également Tizen. Le fait d'être une bibliothèque logicielle multiplateforme attire un grand nombre de personnes qui ont donc l'occasion de diffuser leurs programmes sur les principaux OS existants.

Qt supporte des bindings avec plus d'une dizaine de langages autres que le C++, comme Java, Python, Ruby, Ada, C#, Pascal, Perl, Common Lisp, etc.

Qt est notamment connu pour être le Framework sur lequel repose l'environnement graphique KDE, l'un des environnements de bureau par défaut de plusieurs distributions GNU/Linux.

Qui utilise Qt ?

Pour témoigner de son sérieux, une bibliothèque comme Qt a besoin de références, c'est-à-dire d'entreprises célèbres qui l'utilisent. De ce point de vue, pas de problème : Qt est utilisée par de nombreuses entreprises que vous connaissez sûrement : Adobe, Archos, Boeing, Google, Skype, la NASA, FireFox…

Qt est utilisée pour réaliser de nombreuses GUI, comme celle d'Adobe Photoshop Elements, de Google Earth ou encore de Skype !

3) INTERFACES ET EXPLICATIONS :

a) INTERFACE GENERAL:

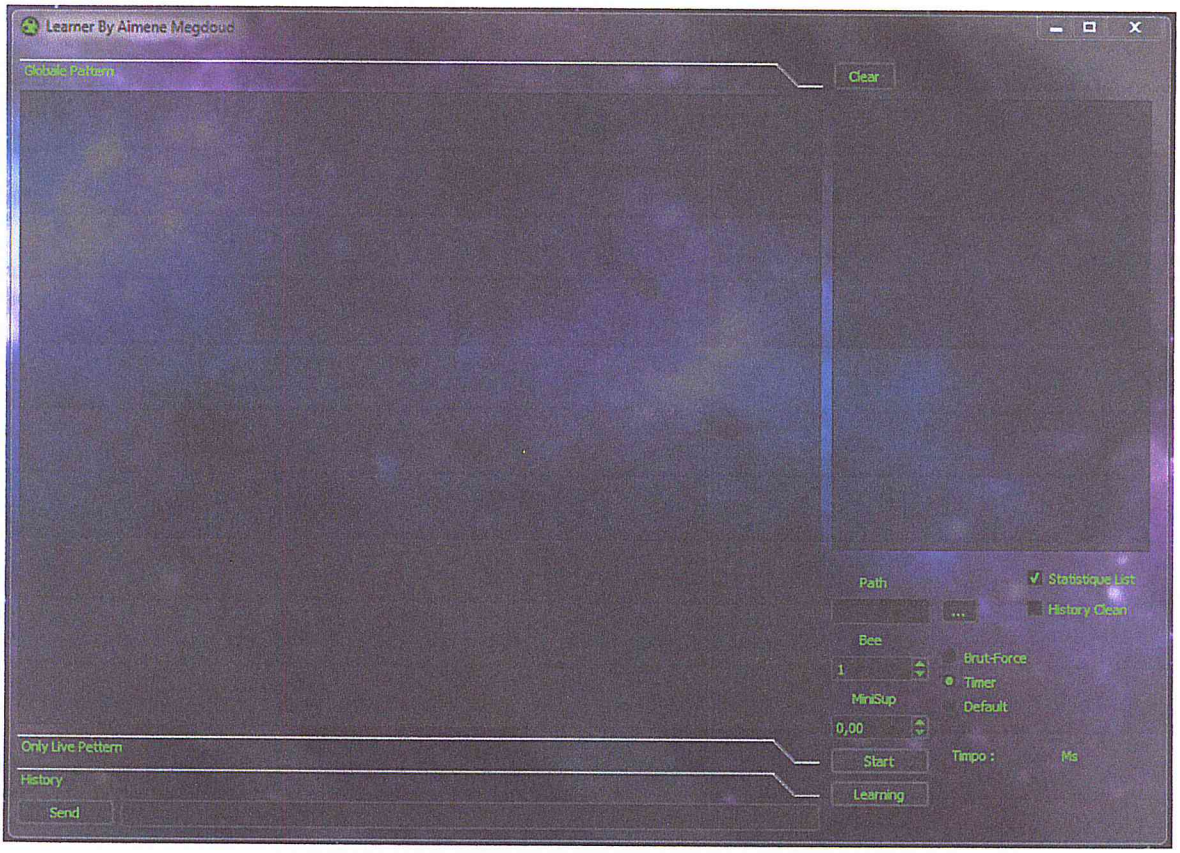

Figure 1 : Interface Général

Type	Nom	Détail
Afficheurs	Globale Pattern	Permet l'affichage des résultats obtenu des données existante déjà, et qui ne sont pas d'une source de flux de données.
	Only Live Pattern	Permet l'affichage des résultats du traitement obtenu des flux de données.
	History	Affiche les anciens résultats pour permettre à l'utilisateur l'étude facile et la recherche des anciens résultats.
	SW-Nano-Bee	Affiche les résultats généraux additionnés aux résultats statiques avec les résultats obtenus de la part des Nano-abeilles dans les flux de données.
Buttons	Clear	Permet d'effacer tous les résultats des fenêtres.
	Path	Ouvre une petite fenêtre qui demande le chemin du fichier des Data-sets.
	Start	Enregistre les data-sets dans une Mmap pour accélérer le traitement, puis divise les données a des zones selon le nombre de nano-abeille. Elle

		stock les zones dans un fichier ainsi les couples de solutions possibles dans un autre.
	Learning	Permet le lancement du corps générateur, c'est lui le responsable de la génération des nano-abeilles.
Spin-Box	Mini-Sup	Permet à l'utilisateur d'entrer la valeur du mini-support.
	Pre-MinSup	Permet à l'utilisateur d'entrer la valeur du proche du mini-support.

b) INTERFACE FLUX DE DONNEES :

C'est une interface conçue spécialement pour jouer le rôle des flux donnés. Elle permet d'envoyé des données d'une taille et d'ans temps définie par l'utilisateur.

Elle permet aussi de générer des données Aléatoirement dans des fichiers, pour faire des tests hors les flux de données.

Figure 2 : Interface du Flux de données

Type	Nom	Détail
Buttons	Add	Permet l'ajout des items a envoyé.
	Start	Permet le commencement dès l'envoie des données.
	Stop	Arrête l'envoie des données.
	Auto-built	Génère des données aléatoires dans un fichier.

c) INTERFACE DES NANO-ABEILLES :

Contrairement aux deux programmes précédents que chacun d'entre eux a une interface, les nano-abeilles leurs interface est invisible pour des raisons :

- Les Nano-abeilles peuvent être nombreux
- Réduction de taux de mémoire utilisé par le système
- Evitez le ralentissement causé par la consommation de mémoire graphique (GPU), la consommation d'un gros taux de pourcentage du processeur.
- Utilisation de la technologie Multi-process qui permet l'échange rapide entre un processus enfant et son parent.

Les Nano-abeilles sont visible dans le gestionnaire des taches sous formes de processus (figure 3)

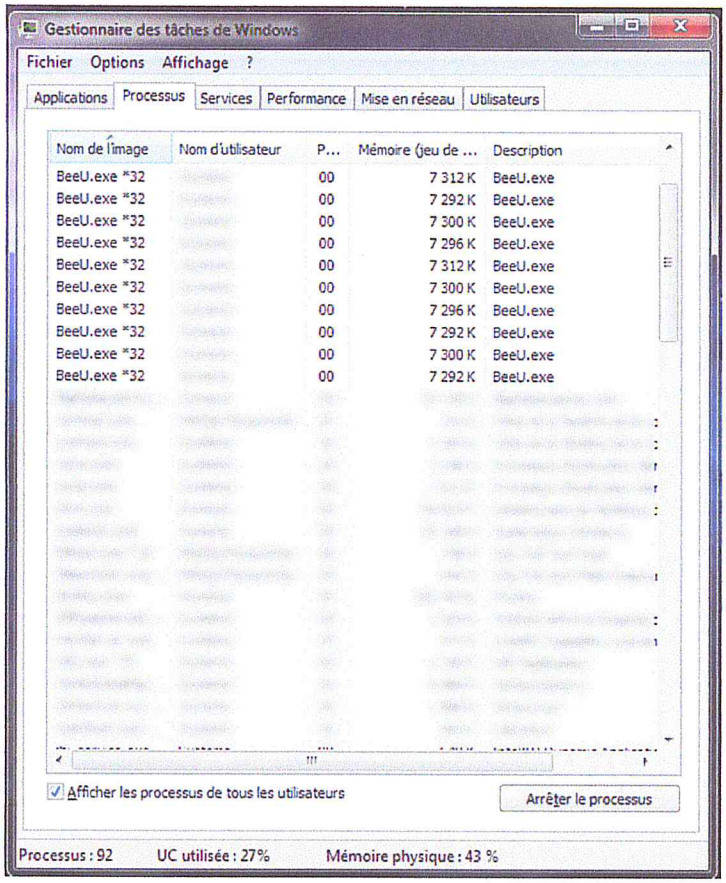

Figure 3 : Des processus enfants qui représentent le nombre des nano-abeilles entrain de traiter les données.

Remarque : Ces processus seront fermés après la fin de leurs travaillent pour libérer la mémoire utilisé.

4) TEST ET RESULTATS :

Avant de voir les résultats obtenu, nous allons monter comment fonctionne notre programme. Selon des étapes a respecté pour assurer le fonctionnement correct. La figure 4 montre cela.

Figure 4 : étapes à suivre pour le fonctionnement correct du système.

Etape 1 : Ajouter le lien ou se situe le fichier des données (Data-set)

Etape 2 : Définir le nombre de nano-abeilles qui peuvent faire le traitement d'un lot de données, cependant ce nombre peut être dupliqué à chaque fois que les données sont arrivées dans un flux de données, contrairement aux données statiques.

Etape 3 : Définir le Mini-support.

Etape 4 : Lancer les Mmap [44] pour accélérer le traitement des données, et fractionnez les données à des zones exploitable par deux variables (début, fin)

Etape 5 : Lancer le générateur des nano-abeilles dans des données statiques, par contre dans un flux de données il sera lancé automatiquement seul dès l'arrivée des données.

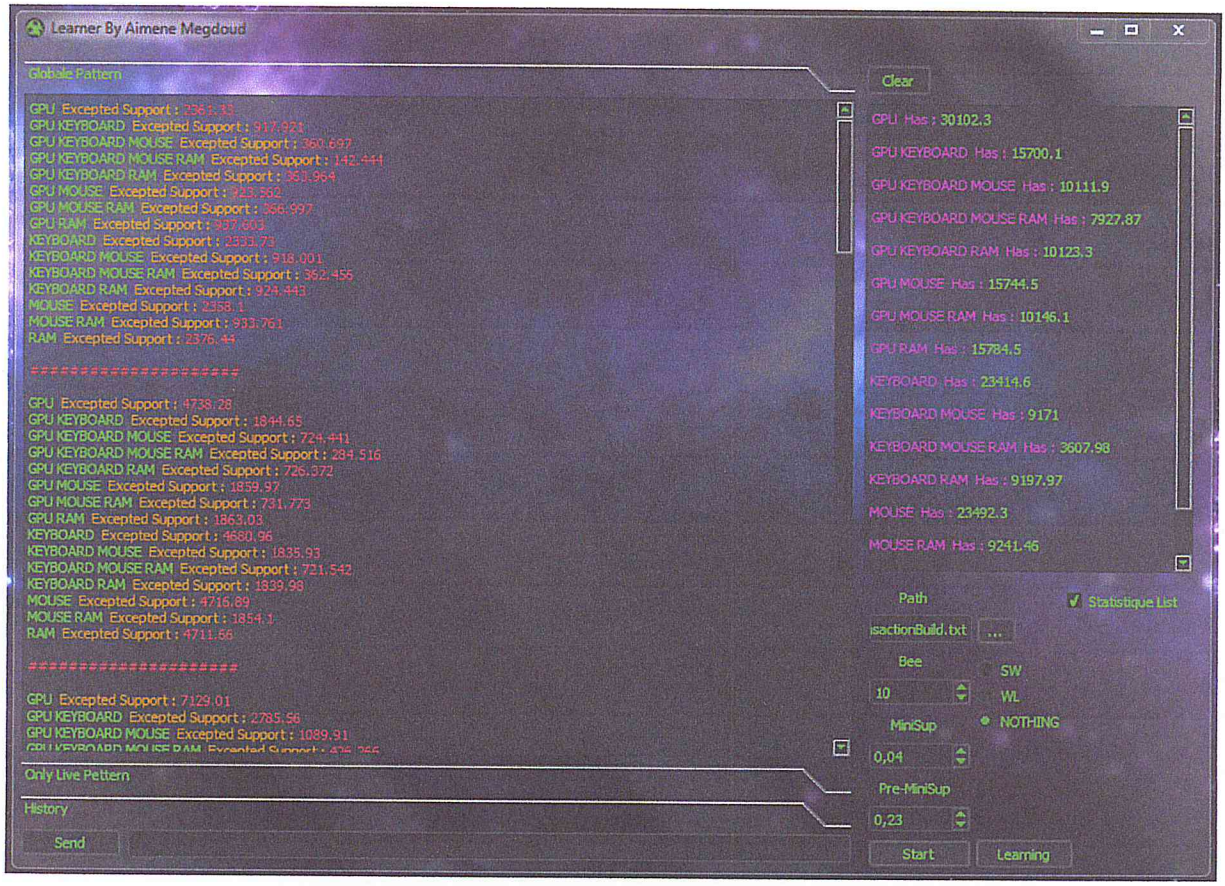

Figure 5 : Affichage des itemsets fréquents.

Remarque : Dans le cas où les données sont arrivées via les flux de données incertaines, toutes ces manipulations appliquées manuellement avant seront exécuté automatiquement par le programme. La figure 6 montre des données envoyées par notre flux de données programmer et le lancement traitement automatique du programme.

Figure 6 : Traitement des flux de données automatiquement.

AVANTAGES ET INCONVENANT

Les résultats obtenus des différents tests, ont montré un changement de temps de traitement avec le nombre de nano-abeilles utilisées. Nous avons testé l'extraction des motifs fréquents incertains sur des données d'une taille de 60 000 transactions en trois étapes différentes :

Etape 1 : Avec une seule Nano-abeille.

Etape 2 : Avec 5 Nano-abeilles.

Etape 3 : Avec 10 Nano-abeilles.

La Figure 6 montre le temps consommé en millisecondes dans les différentes étapes cité avant. Ainsi la figure 7 montre le taux de consommation de mémoire selon le nombre de nano-abeilles actuel. Notamment chacune de ces derniers se ferme à la fin de l'extraction, donc le nombre est de différentes tailles au cours du temps. Dans un cas ou y-a pas de données à traité le nombre sera 0 donc pas de consommation inutile.

A chaque fois que le nombre des nano-abeilles augmente, on aura un temps de repense rapide mais aussi une consommation de mémoire élevée.

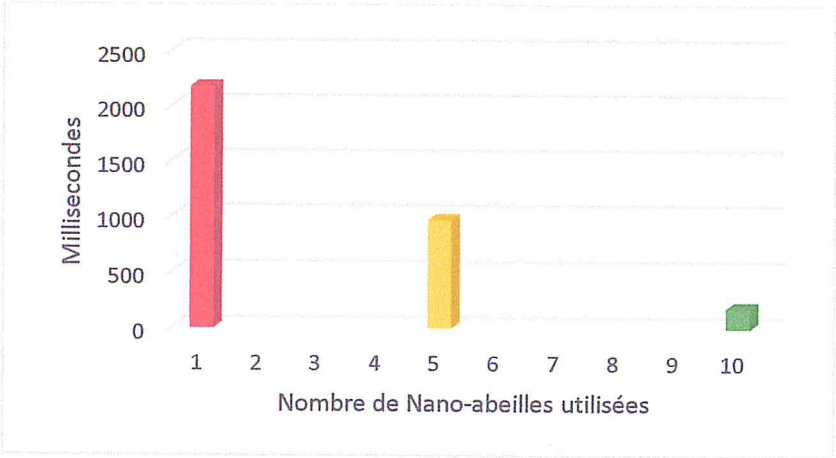

Figure 7 : Histogramme de la variation du temps selon le nombre de nano-abeilles utilisées.

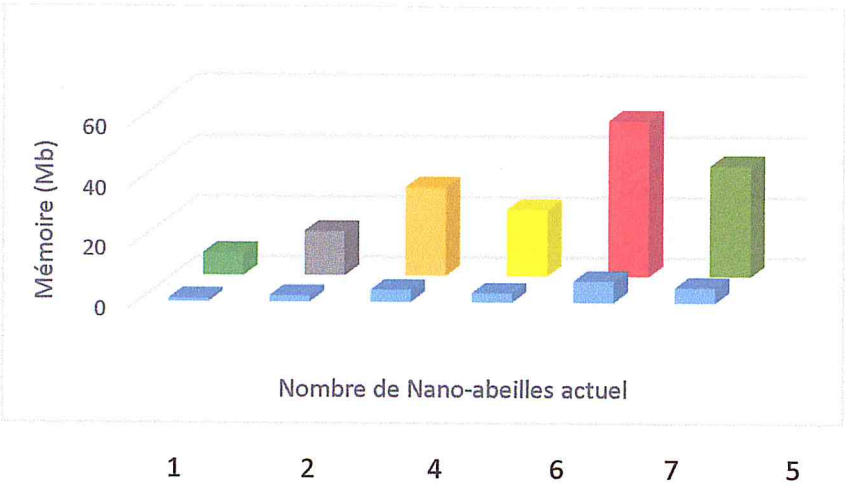

Figure 8 : Histogramme de la variation de consommation de mémoire selon le nombre de nano-abeilles actuel

5) CONCLUSION

Nos test ont montré une rapidité gagné avec l'architecture multi-process et notre système nano-abeille, chaque fois que le nombre de nano-abeilles utilisées est élevé on gagne un temps de traitement, par conséquence la consommation de mémoire augmente selon le nombre de

nano-abeilles utilisées. Une nano-abeille n'existe plus dès la fin de son travail, mais le taux de consommation de mémoire reste en état pas stable, cependant nous essayons d'améliorer notre système au fur et à mésure.

Conclusion general

L'extraction de motifs fréquents est une méthode traitement de données. Pour comprendre et cerner la problématique de l'extraction de motifs fréquents à partie de données, nous avons consacré les chapitres 1 et 3 à une étude comparative des versions des algorithmes d'extraction de motifs fréquents incertains. Comme conclusion de cette étude est que chaque algorithme à ces propres avantages et inconvenant.

Cependant l'étude faite sur les flux de données dans le chapitre 2 nous a données une autre vision sur l'extraction des motifs fréquents incertains. Nous avons découvert des algorithmes qui ont vu le jour afin d'aboutir à des meilleurs résultats dans ce domaine.

L'Algorithme de la colonie d'abeille était un des algorithmes qui nous ont marqué dans notre étude. Nous nous sommes inspirés ainsi de ça et nous avons fait notre propose système d'extractions de motifs fréquents à partir des flux de données incertaines.

Notre système nous l'avons appelée les nano-abeilles à montrer sa simplicité et sa fluidité de travail. Ce qui nous à laisser faire tel travail, c'est l'envie de faire de nouvelles essayes car si on veut avancer en avant et améliorer nos résultats pour cela il ne faut pas se basé que sur le savoir. Car le savoir à ces limites et l'imagination entoure l'univers.

Références

[2] FREQUENT ITEMSETS par Tanagra le 3.10.11, Octobre 2011.

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

[4] 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.

[10] Baptiste Csernel, Thésard, Fabrice Clérot, Superviseur FT R&D Georges Hébrail, Superviseur ENST Classification de Flux de Données par Échantillonnages sur Fenêtres Inclinées

[11] Carson Kai-Sang Leung, Boyu Hao, Mining of Frequent Itemsets from Streams of Uncertain Data, IEEE International Conference on Data Engineering, 2009.

[12] Nadungodage, C.H., Xia, Y., Lee, J.J., & Tu, Y. 2013. Hyper-structure mining of frequent patterns in uncertain data streams. In Knowledge and Information Systems (KAIS), 37(1), pp. 219–244.

[13] Reza Akbarinia, Florent Masseglia, Fast and Exact Mining of Probabilistic Data Streams, European Conference, ECML PKDD, 2013.

[14] LIU Lixin, ZHANG Xiaolin, ZHANG Huanxiang, Mining of Probabilistic Frequent Itemsets over Uncertain Data Streams, 11th Web Information System and Application Conference, 2014.

[15] Nadungodage, C.H., Xia, Y., Lee, J.J., & Tu, Y. 2013. Hyper-structure mining of frequent patterns in uncertain data streams. In Knowledge and Information Systems (KAIS), 37(1), pp. 219–244.

[16] Carson Kai-Sang Leung, Fan Jiang, Frequent Pattern Mining from Time-Fading Streams of Uncertain Data, DaWaK 2011, LNCS 6862, pp. 252–264, 2011.

[17] Carson Kai-Sang Leung, Fan Jiang, Yaroslav Hayduk, A Landmark-Model Based System for Mining Frequent Patterns from Uncertain Data Streams, Proceedings of IDEAS'11; 2011. pp. 249–250.

[18] MÉTAHEURISTIQUES POUR L'EXTRACTION DE CONNAISSANCES : APPLICATION À LA GÉNOMIQUE, LAETITIA JOURDAN, 26 Novembre 2003. – pp 17-216

[19] C. H. Papadimitriou. *The complexity of combinatorial optimization problems*. PhD thesis, Princeton, 1976

[20] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization : Algorithms and Complexity*. Prentice-Hall, 1982.
[KGV38] S. Kirkpatrick, D.C. Gelatt, and M.P.Vechhi. Optimization by simulated annealing. *Science*, 220 :671–680, May 1983.

[21] P. Hansen. The steepest ascent mildest descent heuristic for combinatorial programming. In *Congress on Numerical Methods in Combinatorial Optimization*, 1986. Capri, Italy.

[22] M. Pei, E.D. Goodman, W.F. Punch, and Y. Ding. Genetic algorithms for classification and feature extraction. In *Annual Meeting : Classification Society of North America*, June 1995.

[23] J D. Kelly and L. Davis. A hybrid genetic algorithm for classification. In *Proceedings of the 12th IJCAI*, pages 645–650. Morgan Kaufmann, 1991.

[24] D.W. Opitz and J.W. Shavlik. Using genetic search to refine knowledge-based neural networks. In Morgan Kaufman, editor, *Machine learning : Proceedings of the Eleventh International Conference*, pages 208–216, 1994.

[25] C. Guerra-Salcedo and D. Whitley. Genetic approach to feature selection for ensemble creation. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 236–243, Orlando, Florida, USA, 1999. Morgan Kaufmann.

[26] J. A. Lozano and P. Larra. Applying genetic algorithms to search for the best hierarchical clustering of a dataset. *Pattern Recognition Letters*, 20(9) :911–918, 1999.

[27] M. Dorigo, V. Maniezzo, and A. Colorni. Positive feedback as a search strategy. Technical report, Technical Report 91016, Dipartimento di Elettronica e Informatica, Politecnico di Milano, italie, 1991.

[28] M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System : Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B : Cybernetics*, 26(1) :29–41, 1996.

[29] M. Dorigo. *Learning and Natural Algorithms (in Italian)*. PhD thesis, DEI, Politecnico di Milano,Italy, 1992.

[30] J. Handl and B. Meyer. Improved ant-based clustering and sorting in a document retrieval interface. In J.J.M. Guervós, P. Adamidis, H-G. Beyer, J-L. Fernández-Villacañas, and H-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VII*, pages 913–923, Berlin, 2002. Springer.

[31] J.L. Deneubourg, J.M. Pasteels, and J.C. Verhaeghe. Probabilistic behaviour in ants : a strategy of errors ? *Journal of Theoretical Biology*, 105 :259–271, 1983.

[32] J.L Deneubourg and S. Goss. Collective patterns and decision-making. *Ethology and Evolution*, pages 295–311, 1989.

[33] k.von Frisch. The dance language and orientation of bees. Harvard University Press, 1967.

[34] K.]Polat, S Gunes, and A. Arslan. cascade learning system for classification of diabetes disease : Generalized discriminate analysis and least square support vector machine. Expert Systems with Applications, 34, 482–487., 2007.

[35] M Beekman and F.L.W. Ratnieks. Long range foraging by the honeybee apis mellifera. Functional Ecology, page 490–496, 2000.

[36] T. D. Seeley. The wisdom of the hive. Harvard University Press, Cambridge, MA, 1995.

[37] V Tereshko and A Loengarov. Collective decision-making in honey bee foraging dynamics. Computing and Information Systems Journal, pages 1352–9404.

[38] Conception d'un Classifieur Foul Utilisant Colonie D'abeille Pour Diagnostic Médical, Melle Nedjar Hadjira Hanane. Juillet 2012.

[39] Lew J.B. Beekman, M. Foraging in honeybeeswhen does it pay to dance. Behavioral Ecology 19, page 255–262, 2008.

[40] Takagi T. and Sugeno M. Fuzzy identification of systems and its applications to modelling and control, ieee trans. syst., man, cybern.

[41] M.L. Raymer, W.F. Punch, E.D. Goodman, P.C. Sanschagrin, and L.A. Kuhn. Simultaneous feature extraction and selection using a masking genetic algorithm. Technical report, Michigan State University, USA, 1997.

[42] H. Vafaie and K. de Jong. Robust feature selection algorithms. In IEEE Computer Society Press, editor, *In Proceedings of the Fifth Conference on Tools for Artificial Intelligence*, pages 356–363, Boston, 1993.

[43] J. Yang and V. Honoavar. *Feature Extraction, Construction and Selection : A data Mining Perspective*, chapter 1 : Feature Subset Selection Using a Genetic Algorithm, pages 117–136. H. Liu and H. Motoda Eds, massachussetts : kluwer academic publishers edition, 1998.