MA-004-198-1

UNIVERSITE SAAD DAHLAB DE BLIDA
Faculté des Sciences
Département de Informatique

MEMOIRE DE MASTER

En Informatique

Spécialité : Génie des systèmes informatiques

Réseau Bayésien pour l'extraction de motifs fréquents a partir de données de masse

Par
KEBADJI BILLEL
SEGHIER CHOUAIB

Promotrice
Mlle F/Z Zahra

MA-004-198-1

Président Jury
Sidemoa Ridha
Sidoumoa Mohamed Rédha
Jury :
- Laaroussi ~~Mohamed~~.
- ARKAM ~~Asmaa~~

Aroussi Sanaa
ArKam Meriem

2012/2013

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 des bases de données de taille qui se mesure en téraoctets, pétaoctets, exaoctets et même, dans certains secteurs, en zettaoctets, les algorithmes classiques d'extraction de motifs sont devenus inadéquat dans l'ère de Big Data (données massives ou de masse).Ainsi, notre travail consiste à appliquer une méthode intelligente, distribuée et parallèle basée sur les réseaux Bayésiens dans le but d'extraire des motifs fréquents à partir de données de masse.

Mots clés : Extraction de motifs fréquents, Données de masse, Apprentissage de réseaux Bayésiens, Big Data, Mapreduce.

Abstract

Exractionfrequantitemset is a technique used in data mining, it relies on relatively simple principles. Its objective is to find patterns which frequently appear in a data set. Or, with databases size which is measured in terabytes, pétaoctets, exaoctets and even, in certain sectors, in zettaoctets, classical algorithms pattern extraction have become inadequate into the era of Big Data (massive data or mass). Thereby our job is to implement an intelligent method, parallel and distributed based on Bayesian networks in the purpose extract frequent patterns from mass data.

Keywords: Exractionfrequantitemset, mass data, Learning Bayesian networks, Big Data, Mapreduce.

Remerciements

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

Nos remerciements les plus sincères, accompagnés de toute notre gratitude vont tout d'abord à notre promotriceMlle. 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 nombres 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 ses encouragements et ses précieux conseils durant toute ma vie.

A ma mère pour son amour inconditionnel et sa présence à mes côtés dans les moments difficiles.

A ma sœur Madina, et mes frères Oussama, Mohamed Chawki pour leur encouragement.

A la petite princesse : « Roumaissa ».

Et spécialement a Asmaa Sâad qui m'a apporté son soutien durant cette année.

A toutes ma famille.

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

A mon binôme Chouaib et à toute sa famille.

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

BILLEL

Dédicaces

Je dédie ce travail ;

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

A mon père Allah Yerahmo mère pour son amour inconditionnel et sa présence à mes côtés dans les moments difficiles.

A mes sœur Amel et Fatima Zohra, et mes frères Tahar, Walid Sofiane pour leur encouragement.

A le petit prince : « Mohamed ».

A toutes ma famille.

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

A mon binôme Billel et à toute sa famille.

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

CHOUAIB

Table des matières

Liste des figures

Liste des tableaux

Liste des équations

INTRODUCTION GENERALE

Grâce aux techniques d'extraction des connaissances, les bases de données volumineuses 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. La recherche des règles d'associations commence par une étape algorithmiquement difficile qui produit des motifs potentiellement intéressants à partir des données.

1. Problématique

L'extraction de motifs fréquents est une technique utilisée en fouille de données, elles'appuie sur des principes relativement simples. Son objectif est de trouver les motifs qui apparaissent fréquemment dans un ensemble de données. Or, avec des bases de données de taille qui se mesure en téraoctets, pétaoctets, exaoctets et même, dans certains secteurs, en zettaoctets, les algorithmes classiques d'extraction de motifs sont devenus inadéquat dans l'ère de Big Data (données massives ou de masse). Le terme « données massives » désigne des ensembles de données tellement volumineux qu'il devient difficile de les gérer et traiter avec des outils classiques de gestion de base de données etde traitement d'ensemble de données.

Notre travail consiste à appliquer uneméthode intelligente dans le but d'extraire des motifs fréquents à partir de données. Tout le problème de la fouille de données réside dans le choix de la méthode adéquate à un problème donné. La méthode pour laquelle nous avons opté dans notre travail est l'apprentissage des réseaux Bayésiens (structure et paramètre).

Les réseaux Bayésiens sont la combinaison des approches probabilistes et de la théorie de graphes. Autrement dit, ce sont des modèles qui permettent de représenter des situations de raisonnement probabiliste à partir de connaissances incertaines. Ils ont une représentation efficace pour les calculs d'une distribution de probabilités.

La construction d'un modèle de réseau bayésien par apprentissage à partir des données peut être comparée à la discipline du Data Mining, par conséquent, l'apprentissage des réseaux Bayésiens à partir des données peut être considéré comme un outil de Data Mining.

2. Objectif du travail

L'objectif de notre travail est l'utilisation des réseaux Bayésiens pour la recherche et l'extraction des motifs pertinents à partir des données massives. Le réseau Bayésien obtenu à

partir d'un ensemble de données en utilisant des algorithmes d'apprentissage de structure et des paramètres de ce dernier révèle les relations qui existent entre les attributs de cet ensemble de données. Ces relations sont représentées par des arcs reliant les nœuds d'un réseau Bayésien, ces derniers représentent les attributs d'un ensemble de données. Autrement dit, on peut utiliser les méthodes d'apprentissage des réseaux Bayésien à partir de données pour extraire des règles d'association. Par conséquent, pour extraire des motifs.

Dans le cadre des données massives, l'application des algorithmes d'apprentissage des réseaux Bayésiens (Structure et paramètres) d'une manière classique est devenu inadéquate, vu la taille énorme des ensembles de données à traiter. Ainsi, nous avons utilisé une méthode qui peut être exécutée en parallèle pour l'apprentissage du réseau Bayésien. Cette méthode se concentre sur la mise en parallèle l'exécution de l'algorithme sur des clusters de machines.Elle est basée sur le modèle de programmation Mapreduce.

Il existe différents algorithmes d'apprentissage de structure des réseauxBayésien. L'algorithme K2 est le plus utilisé parce qu'il maximise la probabilité de la structure dans l'espace des DAGs (espace des graphes acycliques dirigés)à partir d' un ensemble de données en respectant la contrainte de l'ordre d'énumération (un ordre d'énumération des variables). Cette contrainte permet de restreindre l'espace de recherche et de limiter la recherche auxarcs intéressants seulement.Quant à L'apprentissage des paramètres,la méthode fréquentiste(maximumde vraisemblance) est la plus utilisée dans le des données complète.

3. Organisation du mémoire

Le mémoire se réparti en quatre grandes partie. Nous commence tout d'abord par un état de l'art sur domaine de la fouille de données (Data Mining) et l'extraction de motifs.Nousexpliciterons aussi les principaux algorithmes d'extraction de motifs avec une comparaison entres les algorithmes.

Dans le deuxièmes chapitre nous présenterons les réseaux Bayésiens, leur utilité et quelles sont les propriétés fondamentales qui en font une modélisation particulièrement avantageuse.

Dans le troisième chapitre nous allons étudier la méthode qu'on a utilisée pour l'extraction de motifs à partir de données massives en utilisant le modèle de programmation MapReduce.

Et dans le quatrième chapitre enfin nous validons la méthode adoptée, et on termine par une conclusion générale.

Chapitre I : La fouille de données et L'extraction de motifs

I.1 Introduction

La fouille de données (data-mining en anglais) est un domaine de recherche en plein essor visant à exploiter les grandes quantités de données collectées chaque jour dans divers domaines d'application de l'informatique. Ce domaine pluridisciplinaire se situe au confluent de l'intelligence artificielle (notamment de l'apprentissage automatique), des statistiques et des bases de données. On lui donne d'autres appellations, comme par exemple extraction de connaissances dans les données, traitement de motifs de données ou encore exploration de données. Les méthodes d'exploration des données fournissent à l'expert des solutions pour l'aide à la décision.

I.2 Techniques de la Fouille de données

Une grande partie des algorithmes utilisés en fouille de données proviennent de l'apprentissage automatique (Machine learning). L'apprentissage statistique **[01]** est un paradigme qui regroupe unensemble de méthodes et d'algorithmes permettant d'extraire l'informationpertinente à partir de données, ou d'apprendre des comportements à partir d'exemples.On distingue deux grandes problématiques en apprentissage statistique: l'apprentissage supervisé d'une part, et l'apprentissage non supervisé d'autrepart.

I.2.1 Apprentissage supervisées

Chaque objet étudié est étiqueté par une valeur de classe. Par exemple, s'il s'agit de données médicales concernant des patients, la classe définit le degré d'atteinte de la maladie. Pour des produits de fabrication industrielle, la classe est déterminée par la qualité de fabrication ; Avant l'apparition récente des techniques de fouille de données à base de motifs, les méthodes reconnues concernent l'utilisation de réseaux de neurones, de réseaux Bayésiens et les arbres de décision**[02].**

- **Réseaux de neurones :**

Les réseaux neurones (ou réseaux connexionnistes) utilisent l'analogie avec l'architecture physiologique du cerveau humain : les neurones sont des entités élémentaires qui reçoivent des signaux en entrée et transmettent à d'autres neurones des signaux de sortie qui résultent d'une combinaison des signaux d'entrée. Les premiers neurones d'entrée sont reliés aux valeurs des attributs d'un objet. Par exemple pour la reconnaissance d'images, ce sont les pixels allumés ou éteints. Les neurones de sortie indiquent la valeur finale de la décision, c'est-à-dire la classe de l'objet. Des neurones intermédiaires sont organisés en couches et l'ensemble constitue un réseau.

Pendant sa phase d'apprentissage, des objets sont présentés au réseau et lorsque la réponse diffère de la classe supervisée, un algorithme de rétro-propagation modifie les comportements des neurones intermédiaires. Techniquement, un tel réseau calcule des équations d'hyper-plan séparateur des classes, selon un algorithme de descente de gradient.

Les réseaux de neurones ont connu un rapide succès, particulièrement pour le traitement des images. Cependant, il est très difficile d'expliquer comment la décision est rendue par ces réseaux, du fait de la grande complexité de leur architecture.

- **Réseaux bayésiens :**

Fondés sur la notion de probabilité conditionnelle, les réseaux bayésiens permettent de calculer la distribution jointe sur un ensemble de données à l'aide de procédés stochastiques.

Leur architecture est obtenue par apprentissage à partir des données, mais cette étape reste la partie difficile de la mise en œuvre de cette technique. En revanche, les décisions rendues par ces réseaux sont plus compréhensibles que celles fournies par les neurones.

- **Arbres de décision**

Un arbre de décision permet de représenter les objets étudiés sous une forme arborescente, selon une hiérarchie des attributs déterminée par un calcul d'entropie. Ces méthodes sont populaires pour la présentation synthétique des données qu'elles fournissent, ainsi que pour la clarté des explications concernant la décision rendue.

I.2.2 Apprentissage non supervisées

Aucune classe n'est attribuée a priori.Les algorithmes classiques de classification non-supervisée sont les méthodes à base de k-moyenne ou de nuées dynamiques et permettent de segmenter l'ensemble des objets en un nombre défini de classes homogènes. À partir de centroïdes choisis aléatoirement, les classes sont déterminées par fusion ascendante, de manière à minimiser la distance interclasses et à maximiser la distance extra-classes. De façon duale, une classification peut être réalisée selon une technique hiérarchique descendante : l'ensemble des objets est découpé en classes de plus en plus fines par l'utilisation d'une distance**[02].**

La mise au point d'une distance adéquate reste un problème épineux pour ces méthodes. Elles souffrent également d'un manque de lisibilité des décisions rendues.

I.2.3 L'extraction des itemsets (Motifs)

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, 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 »**[03], [04], [05].**Le fichier comporte 10 observations (transactions) et 4 items (voir figure 1.1).

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

Figure 1.1 : Base de transactions**[03].**

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 itemsetest 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 $\subset$A et que SUP(B) = SUP(A). Autrement dit, l'itemset est générateur si tous ses sous itemsets ont un support strictement supérieur.

I.2.3.1Les algorithmes d'extractiondes itemsets (Motifs)

Dans cette section, nous allons présenter les lignes directrices des algorithmes les plus importants parus dans la littérature, en nous focalisant essentiellement sur l'étape de découverte des itemsets. Un premier survol de ces algorithmes permet de les classer selon la

technique adoptée pour l'exploration de l'espace de recherche, à savoir « Tester-et-générer » et « Diviser-pour-régner ».

- **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 (e.g. 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 proposéla technique « Diviser-pour-régner » dans cette technique on va éviter le problème de goulot d'étranglement et le perde de l'espace mémoire et le temps d'exécution moins long que la technique précédente. Bref, les algorithmes qui sont implémenté dans la technique Divisé-pour-ranger sont plus performant qui sont implémenté en Tester-et-générer.

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

Dans cette sous-section, nous allons passer en revue les algorithmes les plus connus dans la littérature opérant selon la technique « Tester-et-générer ». Avant d'aborder la description des algorithmes, nous allons présenter une structure générale ou générique des algorithmes entrants dans cette catégorie.

Notons que la structure générique, présentée par l'algorithme 1.1, indique globalement les différentes étapes que suit un algorithme pour l'extraction des itemsets**[03]**.

Algorithme 1.1 : algorithme générique.

Entrée: K : Contexte d'Extraction, minsup
Sortie: Ensemble des itemsets fréquents

1: Initialiser l'ensemble de candidats de taille 1
2: **tant que** ensemble de candidats non vide **faire**

3: **Étape d'élagage (ou de test)**
1) Calculer le support des candidats
2) Élaguer l'ensemble de candidats par rapport à minsup
3) (Éventuellement) calculer les fermetures des candidats retenus
4: **Étape de construction**
1) Construire l'ensemble de candidats à utiliser lors de l'itération suivante
2) Élaguer cet ensemble en utilisant les propriétés structurelles des itemsets fermés et/ou des générateurs minimaux.
5: **fin tant que**
6: **retourner** Ensemble des itemsets fréquents.

1. Apriori et OCD

L'algorithme Apriori de Agrawal et Srikant**[05, 06]** et l'algorithme OCD de *Mannila et al***[07]** ont été proposés indépendamment en 1994dans le domaine de l'apprentissage des règles d'association introduit par *Agrawal et al*dans « *Mining Association Rulesbetween Sets of Items in Large Databases*» **[08]**.Ces algorithmes, qui procèdent de la même manière, sont des algorithmes itératifs de recherche des itemsets fréquents par niveaux. Cela signifie que durant la $k^{ème}$ itération, un ensemble d'itemsets candidats de taille k est généré et un balayage du contexte est réalisé afin de supprimer les candidats infréquents. L'ensemble des k-itemsets fréquents ainsi générés est utilisé lors de l'itération $k + 1$ suivante pour générer les candidats de taille $k + 1$. Ces algorithmes réalisent donc μ itérations afin de déterminer tous les itemsets fréquents dans l'ordre croissant de leurs tailles, μ étant la taille des plus grands itemsets fréquents.

Le pseudo-code de l'algorithme est présenté dans l'algorithme 1.2. Les notations utilisées sont présentées dans la table 1.

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*.

Tableau 1.1- Notions utilisées dans l'algorithme Apriori.

Procédure Apriori-Gen(F_{k-1}) La procédure Apriori-Gen reçoit un ensemble F_{k-1} de (k-1)-itemsets fréquents comme paramètre. Elle retourne un ensemble C_k de k-itemsets candidats qui est un sur-ensemble de l'ensemble des k-itemsets fréquents.

Algorithme 1.2 : Extraction des itemsets fréquents avec Apriori.

Entrée : contexte $\mathcal{B}$; seuil minimal de support *minsupport* ;
Sortie : ensembles F_k des k-itemsets fréquents ;

1) $F_1 \leftarrow$ {1-itemsets fréquents} ;
2) **pour** ($k \leftarrow 2$; $F_{k-1} \neq \varnothing$; k++) **faire**
3) $\quad C_k \leftarrow$ Apriori-Gen(F_{k-1}) ;
4) $\quad$ **pour chaque** objet $o \in \mathcal{B}$ **faire**
5) $\quad\quad C_o \leftarrow$ Subset(C_k, o) ;
6) $\quad\quad$ **pour chaque** candidat $c \in C_o$ **faire** $c.support$++ ;
7) $\quad$ **fin pour**
8) $\quad F_k \leftarrow \{c \in C_k \mid c.support \geq minsupport\}$;
9) **fin pour**
10) **retourner** $\bigcup_k F_k$;

Algorithme DIC (*Dynamic Itemset Counting*):

L'algorithme DynamicItemsetCounting (DIC) a été proposé par Brin et al. **[09]** en 1997. Il est dans le domaine d'apprentissage des règles d'association. Il permet d'améliorer l'efficacité de la recherche des itemsets fréquents par niveaux en diminuant le nombre de balayages du contexte réalisés. Le principe de l'approche consiste à définir une fenêtre M correspondant à un nombre d'objets du contexte et effectuer les lectures du contexte par blocs de M objets. Lorsque la fin du contexte est atteinte, un balayage complet a été effectué et les lectures reprennent au début du contexte. Après la lecture de chaque bloc, les supports des itemsets candidats sont mis à jour, de nouveaux itemsets candidats sont créés et les itemsets pour lesquels tous les objets du contexte ont été parcourus sont soit insérés dans l'ensemble des itemsets fréquents, soit marqués comme itemsetsinfréquents selon leurs supports.Le pseudo-code de l'algorithme est présenté dans l'algorithme 1.3. Les notations utilisées sont présentées dans la table 1.2.

M	Taille de la fenêtre : nombre d'objets lus avant de mettre à jour les supports des itemsets candidats.
C_k	Ensemble de *k*-itemsets candidats. Chaque élément de cet ensemble possède quatre champs : *itemset*, *support*, *nombrelusetsituation*.
C	Ensemble de tous les itemsets candidats des ensembles C_k quelle que soit leur taille.
P	Ensemble des itemsets candidats des ensembles C_k dont le *support* a été modifié après la lecture de *M* objets.
F_k	Ensemble de *k*-itemsets fréquents. Chaque élément de cet ensemble possède deux champs : *itemset* et *support*.

Tableau 1.2 – Notions utilisées dans l'algorithme DIC

L'algorithme utilise un compteur *nombrelu* un marqueur *situation* pour chaque itemset candidat. Le compteur *nombrelu*indique le nombre d'objets du contexte qui ont été considérés jusque-là pour déterminer le support de l'itemset. Lorsque *nombrelu* = $|\beta|$, tous les objets du contexte ont été considérés et le support de l'itemset est définitif. La valeur du marqueur *situation* dépend d'une part si le support actuel de l'itemset a été calculé sur l'ensemble du contexte ou sur une partie seulement et d'autre part si cet itemset est fréquent ou non selon cette valeur du support. Les valeurs du marqueur *situation* correspondants aux quatre états possibles de l'itemset sont :

FC (Fréquent Confirmé) pour un itemset dont le support calculé sur la totalité du contexte est supérieur à *minsupport* : l'itemset est un itemset fréquent.

IC (Infréquent Confirmé) pour un itemset dont le support calculé sur la totalité du contexte est inférieur à *minsupport* : l'itemset est un itemsetinfréquent.

FP (Fréquent Potentiel) pour un itemset dont le support n'est pas entièrement calculé, dans le cas ou sa valeur actuelle est supérieure à *minsupport*.

IP (Infréquent Potentiel) pour un itemset dont le support n'est pas entièrement calculé, dans le cas ou sa valeur actuelle est inférieure à *minsupport*.

L'algorithme Apriori correspond à l'application de cette méthode pour une fenêtre de la taille du contexte M = $|\beta|$: les 1-itemsets sont déterminés fréquents ou infréquents confirmés à la fin de la première lecture d'un bloc (contexte entier), les 2-itemsets à la fin de la deuxième lecture d'un bloc, etc. En conséquence, il n'est pas nécessaire pour Apriori d'associer un compteur *nombrelu* et un marqueur *situation* aux itemsets candidats.

Algorithme 1.3 : Extraction des itemsets fréquents avec DIC

Entrée : contexte $\mathcal{B}$; seuil minimal de support *minsupport* ; taille de la fenêtre de lecture M ;

Sortie : ensembles F_k des k-itemsets fréquents ;

1) $C_1 \leftarrow$ {1-itemsets avec *situation* = *IP*} ;
   // Lecture d'un bloc et mise à jour des supports
2) **lire** M objets dans $\mathcal{B}$;
3) **pour chaque** objet o lu **faire**
4) **pour chaque** candidat $c \subseteq o$ **tel que** $c.situation = IP$ ou FP **faire**
5) $c.support++$;
6) **insérer** c dans P ;
7) **fin pour**
8) **fin pour**
   // Création de nouveaux candidats
9) **pour chaque** candidat $c \in P$ **tel que** $c.situation = IP$ et $c.support \geq minsupport$ **faire**
10) $c.situation \leftarrow FP$;
11) **pour chaque** sur-ensemble x de c **tel que** $|x| = |c| + 1$ et $x \notin C_{|x|}$ **faire**
12) **si** ($\forall s$ sous-ensemble de x de taille $|x| - 1$ nous avons $s.situation =$
13) FC ou FP) **alors insérer** x dans $C_{|x|}$ avec $x.situation = IP$;
14) **fin pour**
15) **fin pour**
   // Mise à jour des compteurs d'objets testés
16) **pour chaque** candidat $c \in C$ **tel que** $c.situation = IP$ ou FP **faire**
17) $c.nombrelu \leftarrow c.nombrelu + M$;
18) **si** ($c.nombrelu = |\mathcal{B}|$) **alors faire**
19) **si** ($c.support \geq minsupport$) **alors** $c.situation \leftarrow FC$;
20) **sinon** $c.situation \leftarrow IC$;
21) **finsi**
22) **fin pour**
   // Test d'arrêt
23) **si** ($\exists c \in C \mid c.situation = FP$ ou IP) **alors aller** ligne 2 ;
24) $F_k \leftarrow \{c \in C_k \mid c.situation = FC\}$;
25) **retourner** $\bigcup_k F_k$;

Algorithme Max-Miner :

L'algorithme Max-Miner a été proposé par Bayardo**[10]** en 1998. C'est un algorithme d'extraction des itemsets fréquents maximaux dans le domaine d'apprentissage des règles d'association. Il effectue simultanément une recherche du bas vers le haut et une recherche du haut vers le bas dans le treillis des itemsets fréquents. La recherche du bas vers le haut est une

recherche par niveaux dont le résultat pour chaque itération k (liste des k-itemsets fréquents) est utilisé pour la recherche du haut vers le bas. La recherche du haut vers le bas est réalisée en associant à chaque itemset candidat la liste des items qui ajoutés au candidat peuvent donner un itemset fréquent. Étant donné un ordre défini sur les items, cette liste contient tous les items fréquents dans le contexte qui sont plus grands dans l'ordre que le plus grand item contenu dans le candidat. Cette liste est appelée *liste d'items extensions* des candidats. L'union de chaque itemset candidat avec sa liste d'item extensions constitue l'ensemble d'itemsets maximaux candidats.

Après chaque itération, les items de cette liste dont l'union avec l'itemset candidat donne un itemsetinfréquent sont supprimés de la liste.

Le pseudo-code de l'algorithme est présenté dans l'algorithme 1.4. Les notations utilisées sont présentées dans la tableau 1.3. Chaque élément de l'ensemble C est un groupe candidat contenant deux itemsets.

C	Ensemble de groupes candidats. Chaque élément c de cet ensemble possède : • un champ itemset candidat $h(c)$ et un champ support($h(c)$) ; • un champ liste d'items extensions $t(c)$; • un champ support ($h(c) \cup t(c)$) ; pour chaque item $i \in$ t(c) un champ support (h(c) ∪ {i}).
FM	Ensemble des itemsets fréquents maximaux parmi les itemsets fréquents découverts.

Tableau 1.3 - Notations utilisées dans l'algorithme Max-Miner.

Algorithme SPADE :

SPADE**[11]** utilise une représentation verticale de la base de données pour extraire des motifs séquentiels. Maintient le principe d'une recherche en largeur d'abord (breadth-first). Cette approche est une extension des travaux d'extraction d'itemsets fréquents CHARM **[12]** qu'on va la voir après. Dans les bases de données verticales, la base de données devient un ensemble de n-uplets de la forme<*itemset : (sequence_ID, event_ID)*>. L'ensemble des paires *ID* d'un itemset donné forme l'identifiant de la liste (ID_list) de l'itemset. Pour découvrir les k-séquences (séquences contenant k items), l'algorithme SPADE joint les *ID_lists* de deux éléments de l'ensemble des $k-1$-séquences fréquentes. La longueur de la liste résultante est égale au support de la k-séquence générée. La procédure s'arrête quand aucune séquence fréquente ne peut être générée ou qu'aucune séquence ne peut être jointe. L'utilisation de base de données verticale permet d'améliorer l'étape de vérification des séquences candidates.

Algorithme 1.4 : Extraction des itemsets fréquents maximaux avec Max-Miner.

Entrée : contexte $\mathcal{B}$; seuil minimal de support $minsupport$;
Sortie : ensemble FM des itemsets fréquents maximaux ;

1) $C \leftarrow \varnothing$;
2) $FM \leftarrow$ Gen-Initial-Groups$(\mathcal{B}, C, minsupport)$;
3) **tant que** $C \neq \varnothing$ **faire**
4) **lire** contexte $\mathcal{B}$;
5) Support-Count$(\mathcal{B}, C)$;
6) **pour chaque** candidat $c \in C$ **tel que** $h(c) \cup t(c)$ est fréquent **faire**
7) $FM \leftarrow FM \cup \{h(c) \cup t(c)\}$;
8) **fin pour**
9) $C_{new} \leftarrow \varnothing$;
10) **pour chaque** candidat $c \in C$ **tel que** $h(c) \cup t(c)$ est infréquent **faire**
11) $FM \leftarrow FM \cup$ Gen-Sub-Nodes$(c, C_{new}, minsupport)$;
12) **fin pour**
13) $C \leftarrow C_{new}$;
14) **supprimer** de FM les itemsets f **tel que** $\exists f' \in FM$ avec $f \subset f'$;
15) **supprimer** de C les groupes c **tel que** $\exists f' \in FM$ avec $h(c) \cup t(c) \subset f'$;
16) **fin tant que**
17) **retourner** FM ;

Algorithme GSP (Generalized Sequential Patterns):

GSP **[11]** a été l'une des premières propositions pour résoudre la problématique des motifs séquentiels introduite par **[12]**. S'appuie sur l'antimonotonie du support (le support d'une super-séquence est inférieur ou égal au support de toutes ses sous-séquences). Les auteurs, en définissant la problématique de l'extraction de motifs séquentiels, ont également proposé un algorithme reprenant les principes d'Apriori, conçu pour l'extraction de règles d'association. Cet algorithme permet l'extraction de motifs séquentiels en effectuant plusieurs passes sur les données avec une approche de génération et validation de candidats. Maintient les principes d'une recherche en largeur d'abord (breadth-first). GSP généralise également la définition **[12]** en incluant des contraintes de temps, des fenêtres glissantes, et des hiérarchies définies par l'utilisateur.

GSP est un algorithme multipasse. La première passe détermine le support (ie. le nombre de séquence contenant cet élément) de chaque élément et permet de ne travailler que sur les

éléments ayant le support minimum et générer les séquences candidates. Le support de ces candidats étant déterminé lors des passages suivants.

La génération des candidats se fait en 2 phases : la jointure et l'élagage. La jointure se fait en joignant l'ensemble des (k-1)-séquences fréquentes L_{k-1} avec lui-même, ce qui consiste en considérant les séquences s_1, $s_2 \in L_{k-1}$ à rajouter le dernier élément de s_2 à s_1. Si l'élément constituait un événement à lui tout seul dans s_2 alors il constituera un événement dans s1. De la même manière s'il faisait partie d'un événement dans s_2 alors il sera rajouté au dernier événement de s_1. La jointure est valable uniquement si en enlevant le premier élément de s1 on obtient la même sous-séquences qu'en retirant le dernier élément de s_2. Enfin, l'élagage consiste à retirer les candidats ayant des sous-séquences ou des sous-séquences contiguës dont le support serait inférieur au minsupp.

Une fois l'ensemble des k-séquences candidate généré, chaque candidat est dénombré dans la base de données. Afin d'optimiser le dénombrement, une table de hachage est utilisée ainsi que la mise en place d'une durée maximale entre éléments d'une séquence ce qui permet de limiter les recherches.

Algorithme PSP(Prefix-tree for Sequential Patterns)
PSP **[14]** s'appuie sur un arbre des préfixes afin d'améliorer la génération de candidats. C'est un algorithme pour résoudre la problématique des motifs séquentiels. Il est basé sur la méthode générer-élaguer. Le principal problème de PSP est le nombre de passes dans la base de données. Pour une séquence de longueur k, k passes sont effectuées. Ceci provoque une perte de temps.

Algorithme SPAM
SPAM**[15]** parcourt en profondeur l'espace de recherche et représente la base de données sous formes de vecteurs de bits, ce qui permet un calcul du support efficace. C'est un algorithme pour résoudre la problématique des motifs séquentiels.

Algorithme CLOSE

Pasquier et al. **[02]**ont proposé cet algorithme pour la découverte des règles sociatives. Il est basé sur l'élagage de l'espace de recherche des itemsets fermés. Ainsi étant donné un contexte d'extraction *K*, CLOSE génère toutes les règles associatives en trois étapes successives :

- découvrir des itemsets fermés fréquents ;
- dériver tous les itemsets fréquents à partir de ceux fermés obtenus durant lapremière étape ;
- pour chaque itemset fréquent i, générer toutes les règles dérivables et ayant uneconfiance au moins égale à *minconf*.

FFC*k*Ensemble des k-itemsets fréquents candidats.
FC*k*Ensemble des k-itemset fermés fréquents.
Chaque élément de ces ensembles possède trois champs :

i) **gen**: le générateur.
ii) **supp**: le support.
iii) **ferm**: la fermeture.

Tableau 1.4 - Notations utilisées dans l'algorithme CLOSE.

Algorithme 1.5 : L'algorithme CLOSE.

Entrée: $\mathcal{K}$ *:Contexte d'Extraction*, minsup
Sortie: $FC=\cup_k FC_k$ *: Ensemble des itemsets fermés fréquents*

{/* **Initialisation***/}
1: FFC_1={1-itemsets}
2: **pour** (k=1 ; FFC_k.gen $\neq \emptyset$;$k++$) **faire**
3: FFC_k.ferm=$\emptyset$
4: FFC_k.supp=$\emptyset$
5: FFC_k=GEN-FERMETURE(FFC_k)
{/* **Étape de construction***/}
6: **pour tout** c $\in$ FCC_k **faire**
7: **si** c.supp $\geq$ minsup **alors**
8: $FC_k = FC_k \cup$ c
9: **fin si**
10: FFC_{k+1}=GEN-GENERATEUR(FC_k)
11: **fin pour**
12: **fin pour**
13: **retourner** $FC=\cup_k FC_k$

Discussion :

Les deux facteurs principaux de l'efficacité, et donc des temps de réponses, des algorithmes d'extraction des itemsets fréquents sont le nombre de balayages du jeu de données réalisés et le nombre d'itemsets candidats considérés par l'algorithme. L'importance du nombre de balayages réalisés est liée au coût des opérations d'entrée/sortie. L'importance du nombre d'itemsets candidats vient du fait que les opérations portant sur ces derniers constituent la majeure partie du temps de calcul CPU de l'algorithme.

Les algorithmes présentés dans cette section ont été développés pour des applications concernant des bases de données grandeséparses et petites. Ces algorithmes leur temps d'exécution varie de quelques secondes à quelques minutes. Ces temps de réponse sont

relativement faibles car dans les données de ce type les plus long itemsets fréquents/fréquents maximum ne contiennent qu'un nombre limité d'items et le nombre total d'itemsets fréquents/fréquents maximum (dont le nombre d'itemsets candidats considérés dépend) est réduit. Dans le cas de contextes pour lesquels les plus longs itemsets fréquents/fréquents maximum sont grands, c'est à dire pour une valeur de μ élevée, les performances de ces algorithmes se dégradent considérablement :

Apriori et OCD réalisent μ itérations et donc μ balayages du contexte pour extraire les itemsets fréquents. Pour des valeurs élevées de μ, ceci entraîne des temps d'exécutions importants, les opérations d'entrée/sortie étant très coûteuses en temps. Ce problème est essentiel dans le cas de données denses pour lesquelles la taille des jeux de données est plus importante pour un nombre d'objets et un nombre d'items identiques.

Pour DIC, après la $\frac{|\beta|éme}{M}$itération, $\frac{|\beta|}{M}$ ensembles d'itemsets candidats (de tailles différentes) sont considérés simultanément lors de chaque itération. De plus, chacun de ces ensembles C_k de k-itemsets candidats est un sur-ensemble de l'ensemble $C_k$généré par Apriori et OCD. Se posent alors les problèmes de l'espace de stockage des ensembles d'itemsets candidats traités simultanément lors des lectures du jeu de données et du coût total du calcul des supports des candidats qui est plus important que pour Apriori ou ODC. En effet, DIC détermine les supports de certains itemsets candidats infréquents qui ne sont pas considérés par Apriori et OCD.

L'algorithme Max-Miner a été proposé concurremment au algorithme Close et A-Close et, ne disposant pas d'implémentation complète de cet algorithme, nous n'avons pas pu réaliser d'expérimentations permettant de comparer les performances de ces trois algorithmes. Les résultats des expérimentations de comparaison de l'algorithme Max-Miner avec les algorithmes d'extraction des itemsets fréquents montrent qu'il permet de réduire les temps d'extraction des itemsets fréquents dans le cas de valeurs de μ élevées, c'est à dire si les plus longs itemsets fréquents maximaux contiennent un nombre important d'items. Les expérimentations des algorithmes Close et A-Close montrent également que ces deux algorithmes réduisent les temps d'extraction des itemsets fréquents pour de tels jeux de données. Toutefois, l'algorithme Max-Miner nécessite une quantité importante de mémoire afin de stocker les informations concernant les groupes candidats de chaque itération. De plus, cet algorithme est soumis au problème de l'extraction de règles d'association à partir de données denses et fortement corrélées lié à la taille moyenne des itemsets fréquents qui est élevée pour les jeux de données de ce type. Les résultats expérimentaux démontrent que cette approche est particulièrement efficace pour l'extraction des motifs fréquents maximaux. Mais dans le cas de l'extraction de tous les motifs fréquents, les performances se dégradent considérablement du fait du coût du dernier balayage.

Les algorithmes du type GSP ou PSP seront très efficaces dans le cas de grandes bases de données avec des séquences moyennement longues. Par contre, les algorithmes comme SPADE, SPAM seront eux très efficaces dans le cas où un très grand nombre de candidats de

même taille sont générés. Il n'est pas possible de dire quel algorithme est meilleur que les autres, dans la mesure où leurs performances sont étroitement liées aux types de données manipulées. L'utilisation des algorithmes sur différents domaines d'application et le fait que les données sources peuvent varier rapidement ont donné lieu à de nombreux travaux de recherche autour de la fouille de données incrémentale. L'une des raisons essentielles du développement des approches autour des motifs séquentiels est leur capacité d'adaptation à de très nombreux problèmes.

La performance de l'algorithme CLOSE a été évaluée par rapport à l'algorithme APRIORI, en menant diverses expérimentations sur une machine biprocesseur IBM PowerPC 43P240. La plate-forme utilisée est *AIX* 4.1.5, avec une fréquence d'horloge de 166 MHz, 1 Giga octets de mémoire centrale et 9 Giga octets de mémoire secondaire. Deux types de contextes ont été utilisés durant ces expérimentations : bases synthétiques (typiquement des contextes épars) et la base *Census*(contexte dense). Les résultats de ces expérimentations ont montré que l'algorithme APRIORI donnait de meilleurs résultats sur les bases synthétiques. Tandis que sur le contexte dense *Census*, l'algorithme CLOSE est plus performant qu'APRIORI, surtout pour des valeurs de supports faibles. A noter que pour l'algorithme CLOSE, aucune technique de gestion du buffer n'a été proposée. En effet, les auteurs supposent que l'ensemble FC*k*desitemsets fermés fréquents et l'ensemble FC*k*+1 des itemsets candidats, utilisés dans la phase de construction d'un passage *k*, peuvent tenir en mémoire centrale. Cependant, on trouve qu'une telle quantité d'information est difficilement tenable en mémoire centrale, surtout pour des contextes denses et des valeurs de supports très faibles.Le tableau 1.5 montre une comparaison entre différents algorithmes d'extraction de motifs.

Tableau 1.5 : Comparaison entre algorithmes Tester-et-générer.

Algorithmes	**Complexité**	**Temps de réponse**	**Performances**		
			Base de données éparses	Petite base de données	Dense base de données
Apriori et OCD	Exponentielle	Très élevés	✓	✓	X
DIC	Exponentielle	Super élevés	✓	✓	✓
Max-Miner	Exponentielle	Elevés	✓	✓	✓
GSP	Exponentielle	Très élevés	✓	✓	✓
SPADE	Exponentielle	Elevés	✓	✓	✓
PSP	Exponentielle	Très élevés	✓	✓	✓
SPAM	Exponentielle	Elevés	✓	✓	✓
CLOSE	Exponentielle	Très élevés	✓	✓	✓

b. Les algorithmes de type « diviser-et-régner» :

Algorithme CLOSET :

L'algorithme CLOSET**[03]** proposait d'adopter une structure de données avancée, où la base de données peut être compressée pour arriver à achever le processus de fouille de données. L'idée motivant cette structure de donnée compacte, basée sur la notion de trie est que lorsque plusieurs transactions se partagent un item, alors elles peuvent être fusionnées en prenant soin d'enregistrer le nombre d'occurrences des items. L'algorithme CLOSET effectue le processus d'extraction des itemsets fermés en deux étapes successives :

Les items des transactions sont ordonnés par support décroissant. Ensuite, l'arbre FP-Tree est construit. Cette structure de donnée est construite comme suit. Premièrement, le nœud racine est créé et étiqueté par «*root*». Pour chaque transaction du contexte, les items sont traités et une branche est créée pour chaque transaction. Dans chaque nœud du FP-tree, il y a un compteur qui garde trace du nombre d'apparitions de l'item dans le contexte d'extraction. Spécifiquement dans le cas où une transaction présente un préfixe commun avec une branche du FP-tree, alors le compteur de chaque nœud appartenant à ce préfixe est incrémenté de 1 et une sous-branche va être créée contenant le reste des items de la transaction.
Au lieu d'une exploration en largeur des itemsets fermés candidats, il effectue une partition de l'espace de recherche pour effectuer ensuite une exploration en profondeur d'abord. Ainsi, il commence par considérer les 1-itemsets fréquents, et examine seulement leur sous-contexte conditionnel. Un sous-contexte conditionnel ne contient que les items qui co-occurrent avec le 1-itemset en question. Le FP-Tree conditionnel associé est construit et le processus se poursuit récursivement.Pseudo code de l'Algorithme CLOSET :

Algorithme 1.6 : L'algorithme CLOSET.

Entrée: $\mathcal{K}$ *:contexte d'extraction* ,minsup
Sortie: $FC=\cup_k FC_k$ *:Ensemble d'itemsets fermés*

```
1: FC=∅
2: {/*Trier le contexte d'extraction K avec une valeur du support décroissante */}
3: K̂= TRI(K)
4: Freq1= { 1-itemset frequents}
5: pour tout i ∈ Freq1 (dans un ordre croissant) faire
6:    FCi=MINE-FERME(i,K̂)
      {/*MINE-FERME construit le sous-contexte conditionnel de i et effectue un par-
      cours en profondeur d'abord par des appels récursifs pour l'extraction des item-
      sets fermés fréquents*/}
7:    FC=FC ∪FCi
8: fin pour
9: retourner FC
```

Algorithme FELINE:

A. Bouchahda et *al* **[03]**, Dans cette approche on applique deux algorithmes FELINE-F et FELINE-H.

- FELINE-F : une approche pour l'extraction des motifs fréquents fermés.
- FELINE-H : une approche pour l'extraction des motifs fermés fréquents et/ou motifs fréquent.

L'arbre CATS :

L'arbre CATS est un arbre préfixé contenant tous les composants d'un arbre FP-TREE. À chaque item correspond un nœud dans la table entête, auquel on associe la fréquence totale de cet item dans la base de transactions.

Algorithme FELINE-F :

Notre objectif consiste à extraire les itemsets fermés fréquents à partir d'un arbre CATS. À la différence de l'extraction des itemsets fréquents, durant le processus d'extraction des itemsets fermés, il peut y avoir des itemsets préfixes qui ne sont pas susceptibles de produire des itemsets fermés fréquents. Ainsi, ils ne peuvent pas être utilisés pour étendre des itemsets fermés et nous devons ainsi les détecter et les éliminer. Il prend en entrée un arbre CATS qui constitue l'espace de recherche de l'algorithme. Ensuite, l'espace de recherche est divisé en sous-espaces de recherche conformément à la liste des items fréquents. L'exploration de chaque sous-espace de recherche donnera lieu à un ensemble d'itemsets fermés candidats. Par la suite, nous devons vérifier l'existence de ces candidats dans l'arbre CFI afin de le mettre à jour**[03]**.

Algorithme 1.7 : L'algorithme FELINE-F.

Entrée: Un arbre CATS, minSup.
Sortie: complet des itemsets fermés fréquents
1: Trier les items de l'entête par ordre décroissant
2: **Pour tout** item fréquent I **faire**
3: **Si** (il n'existe pas un nœud dans CFI portant le label I avec un support supérieur ou égal à celui de l'item I) **alors**
4: Construire I-Tree {/* l'arbre CATS conditionnel condensé de l'item I */}
5: **Si** (I-Tree est un chemin unique) **alors**
6: Générer tous les itemsets fermés fréquents à partir de I-Tree ;
7: **Pour tout** itemset fermé fréquent X généré **faire**
8: **Si** (X n'existe pas dans CFI) **alors**
9: Mettre à jour l'arbre CFI
10: **sinon**
11: MINE-FERME (I-Tree, minSup)
12: **Fin Si**
13: **Fin Pour**
14: **Fin Si**

15: **Fin Si**
16: **Fin Pour**
17: **Retourner L'arbre CFI**

Algorithme FELINE-H :

FELINE-H**[03]** est similaire à l'algorithme FELINE-F, l'algorithme FELINE-H prend en entrée un arbre CATS qui constitue l'espace de recherche de l'algorithme. Ensuite, un arbre conditionnel condensé CATS est construit pour chaque item fréquent. Pour chacun de ces arbres, nous calculons la moyenne des supports des items fréquents relatifs à cet arbre.

Discussion :

D'après les expérimentions qui ont fait dans **[03]**, l'algorithme FELINE-F est appliqué lorsque la moyenne des supports des items fréquents relatifs à cet arbre CATS est supérieure à (*m* * *minSup*) sinon l'algorithme FELINE-H est appliqué.

Ce dernier s'exécuté sur des bases denses permet d'améliorer les temps de réponse comparativement à FELINE-F. Il est à remarquer que la différence au niveau temps d'exécution entre l'algorithme FELINE-F et l'algorithme FELINE-H décroît au fur et à mesure que la valeur du *minSup*croît. Ceci peut s'expliquer par le fait que, lorsque la valeur du *minSup*devient de plus en plus grande, le nombre d'items vérifiant le *minSup*devient de plus en plus petit. Par conséquent, l'arbre CATS associé va contenir de moins en moins de nœuds et sa taille devient de plus en plus réduite.

Les gains en termes de temps de réponse, suite à l'exécution de l'algorithme FELINE-H sur des bases éparses, sont insignifiants. En effet, les temps d'exécution des algorithmes FELINE-F et FELINE-H sont équivalents. Ceci se justifie par le fait que la plupart des arbres conditionnels condensés CATS sont constitués d'un chemin unique ce qui implique que, pour les bases éparses, appliquer l'algorithme FELINE-H revient à appliquer l'algorithme FELINE-F.Le tableau 1.7 montre une comparaison entre les algorithmes CLOSET, FELINE-F, FELINE-H.

Les performances de l'algorithme CLOSET **[03]**ont été évaluées comparativement à l'algorithme CHARM**[03]**. Les expérimentations ont été menées sur un Pentium PC, fonctionnant sur la plate-forme Windows NT, avec une fréquence d'horloge de 233 MHz, 128 Méga octets de mémoire centrale. Les contextes de test utilisés lors des expérimentations sont de deux types : contexte épars (les bases synthétiques) et contextes denses Les résultats de ces expérimentations ont montré que l'algorithme CLOSET est plus performant que l'algorithme CHARM sur les contextes épars et dense.

Tableau 1.6 : Comparaison entre algorithmes diviser-et-régner.

Algorithmes	Complexité	Temps de réponse	Performances	
			Base de données éparses	Dense base de données
CLOSET	Exponentielle	Haut Game	✓	✓
FELINE-F	Exponentielle	Haut Game	✓	X
FELINE-H	Exponentielle	Haut Game	✓	✓

I.3 Les Réseaux Bayésiens et l'extraction de motifs fréquents

Les travaux de **[16]**ont montré l'utilisation des connaissances de l'utilisateur par la dentition de règles. Cette approche a ensuite étéformalisée par un réseau de croyances **[17]**permettant d'extrairel'ensemble minimum des règles d'association en fonction des connaissances du domaine. Cetype d'approche présente cependant une limitation.

En effet, une règle est jugée intéressante sielle diffère des règles définies dans le réseau de croyance, et non pas par rapport à ce que l'onpourrait inférer de ces croyances. Cette notion d'inférence a été développée dans **[18]; [19]**. Ces auteurs décrivent l'utilisation d'un réseau bayésien pour calculer l'intérêt d'ensembles d'attributs extraits à l'aide d'un algorithmede type Apriori **[20].** La différence entre le support estimé sur les donnéeset le support inféré à partir du réseau bayésien est calculée pour chaque ensemble d'attributs.Les motifs les plus intéressants sont ceux pour lesquels la divergence entre les connaissancesde l'utilisateur et ce qui est observé dans les donnéesréelles est la plus forte.Ces ensembles d'attributs sont ensuite soumis à l'utilisateur pour une éventuelle mise à jour de la structure ou des paramètres du réseau bayésien.

On peut dire que les RBsont été utilisé dans le peu de travaux trouvé dans la littérature en tant que modèle de connaissances du domaine, les RBsont pour but de faciliter la découverte de motifs fréquents, dans les travaux présentés dans **[19]**et **[21]**,l'approche envisagée par les auteurs repose sur l'estimation de la fréquence des motifs à partir du RB et la comparaison de cette estimation avec la fréquence constatée sur le jeu de données.

I.4Conclusion

Dans ce chapitre nous avons explicité ce qui est la fouille de donnée et l'extraction des itemsets et les différentes approches d'extraction des motifs fréquents, avec une discussion de l'efficacité de ces algorithmes. Les critères de comparaison sont : le temps d'exécution et la qualité des résultats et aussi l'espace mémoire. Dans le prochain chapitre on va voir les réseauxBayésiens et comment les construire.

Chapitre II : Réseaux Bayésiens

II.1 Introduction

Un modèle graphique est une famille de distribution de probabilités définie en termes de graphe orienté ounon orienté. Il est constitué de nœuds qui représentent des variables aléatoires et des arcs représentant les relations de dépendances entre ces variables. On distingue deux formes de modèles graphiques probabilistes : les modèles orientés et les modèles non orientés, basés respectivement sur les graphes acyclique orientés connus sous le nom de réseaux Bayésiens (RB) et les graphes non orientés. Ces modèles graphiques probabilistes bénéficient non seulement des avantages mais aussi ils représentent les avantages liés à leur représentation graphique. En effet ils permettent de visualiser la structure et les propriétés de dépendance conditionnelles du modèle probabiliste correspondant. Ainsi les RB sont une union entre la théorie des probabilités et la théorie des graphes. Ils constituent une technique d'acquisition, de représentation et de manipulation de connaissance et on les utilise, surtout, pour leur capacité d'effectuer des inférences dans un contexte d'incertitude et aussi pour leurs algorithmes d'apprentissages. Ils sont pour prévoir, contrôler et simuler le comportement d'un système, à diagnostiquer les causes d'un phénomène observé, à analyser des données et à prendre des décisions **[22]**.

II.2 Réseau Bayésiens :

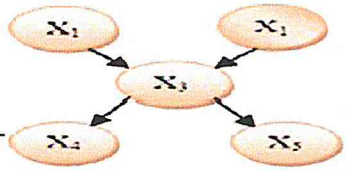
Définition:

Un réseau Bayésien (RB) **[2]**est un graphe orienté sans circuits (GOSC) (DAG pour Direct Acyclic Graph) qui définit une factorisation d'une distribution de probabilité jointe sur des variables qui sont représentées par les nœudsde ce graphe, cette factorisation est donnée par les liens orientés du DAG, plus précisément un graphe orienté sans circuits G, G=(V, E) tel que V dénote l'ensemble des nœuds(ou les sommets) et E dénote l'ensemble des liens orientés (ou les arcs) reliant les nœudsde ce graphe, la distribution de la probabilité jointe $P(X_v)$ sur l'ensemble des variables X_v indexé par V est

$$P(X_v) = \prod_{v\epsilon V} P(X_v|X_{pa(v)}) \qquad (2.1)$$

Où $X_{pa}(v)$ dénote l'ensemble de variables parents de la variable X_V pour chaque nœud.

Autrement dit, un réseau Bayésien se compose de deux parties. Une partie qualitative, qui est un graphe orienté sans circuits dont les nœuds représentent les variables aléatoires, et une partie quantitative définissant l'ensemble de fonctions de probabilité conditionnelles (FPCs). Un exemple d'un réseau Bayésien sur les variables $X = (X_1, \ldots, X_4)$ est montré sur la Figure suivant, les deux parties qualitative et quantitative sont représentées.



$P(X_1)$	$X_1=0$	$X_1=1$
	0,9	0,1

$P(X_2)$	$X_2=0$	$X_2=1$
	0,99	0,01

$P(X_4/X_3)$	$X_4=0$	$X_4=1$
$X_3=0$	0,98	0,02
$X_3=1$	0,77	0,33

$P(X_5/X_3)$	$X_5=0$	$X_5=1$
$X_3=0$	0,34	0,66
$X_3=1$	0,15	0,85

$P(X_3/X_1,X_2)$	$X_3=0$	$X_3=1$
$X_1=0,X_2=0$	0,95	0,05
$X_1=0,X_2=1$	0,65	0,35
$X_1=1,X_2=0$	0,81	0,19
$X_1=1,X_2=1$	0,30	0,70

Figure 2.1: Exemple d'un réseau Bayésien (partie qualitative et quantitative).

a. Une représentation graphique de la causalité

La représentation graphique la plus intuitive de l'influence d'un événement, d'un fait, ou d'une variable sur une autre, est probablement de relier la cause à l'effet par une flèche orientée.

Figure 2.2 : Représentation graphique de la causalité.

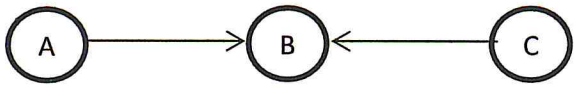
Supposons que A et B soient des événements, qui peuvent être observés ou non, vrais ou faux. Du point de vue du sens commun, le graphe ci-dessus peut se lire comme ceci. «La connaissance que j'ai de A détermine la connaissance que j'ai de B».

Cette détermination peut être stricte, c'est-à-dire que, sachant avec certitude que A est vrai, je peux en déduire B avec certitude. Il peut aussi s'agir d'une simple influence. Dans ce cas, cela signifie que, si je connais A avec certitude, mon opinion sur B est modifiée, sans que je puisse toutefois affirmer si B est vrai ou faux.Avant d'aller plus loin, il est important de comprendre que, bien que la flèche soit orientée de A vers B, elle peut cependant fonctionner dans les deux sens, et ce même si la relation causale est stricte.

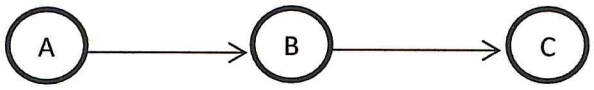
b. Les chemins de circulation d'information

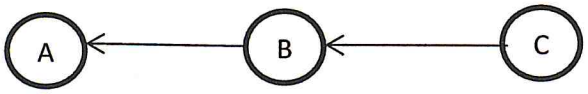
Dans l'exemple ci-dessus, nous avons vu qu'une information certaine se propage dans un graphe en modifiant les croyances que nous avons des autres faits. Nous allons étudier quels chemins cette information peut prendre à l'intérieur d'un graphe. Nous allons considérer les

trois cas suivants, qui décrivent l'ensemble des situations possibles faisant intervenir trois événements.



Connexion Convergente





Connexion en série

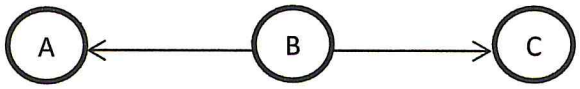


Figure 2.3- Circulation de l'information

c. D-séparation:

On dira que X et Y sont d-séparés par Z si pour tous les chemins entre X et Y, l'une au moins des deux conditions suivantes est vérifiée :

- Le chemin converge en un nœud W, tel que W ≠ Z, et W n'est pas une cause directe de Z.
- Le chemin passe par Z, et est soit divergent, soit en série au nœud Z.

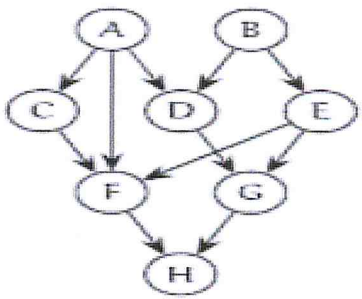


Figure 2.4 -Exemple sur la D-séparation.

- C et G sont d-connectés.
- C et E sont d-séparés.
- C et E sont d-connectés en donnant l'évidence sur G.
- A et G sont d-séparés en donnant l'évidence sur D et E
- A et G sont d-connectés en donnant l'évidence sur D.

d. La Couverture Markovienne :

Le Critère de Markov est un critère équivalent au critère de d-Séparation, mais dans certains cas, il est plus efficace parce qu'il demande moins des chemins possibles entre les sommets concernés.

Théorème de Markov :

La couverture de Markov d'une variable A contient :

- Ses parents
- Ses enfants
- Les parents de ses enfants

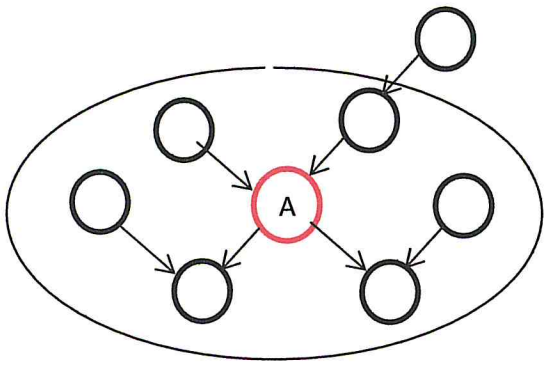


Figure 2.5 - La couverture de Markov.

Une variable A est indépendante des autres variables des réseaux si les nœuds de sa couverture de Markov sont connus.

II.3 Construction d'un RéseauxBayésiens à partir des données

La procédure de construction **[24]**d'un réseau Bayésien à partir des données est constituée de deux étapes. La première consiste à construire la structure du réseau qui est un graphe orienté sans circuits en utilisant des algorithmes d'apprentissage de structure. La deuxième étape consiste à estimer les paramètres du réseau qui sont définit généralement par les tables de probabilité conditionnelles.

II.3.1Apprentissage de la structure

Pour construire la structure d'un réseau Bayésien**[24]**, on a deux sources d'informations,des informations provenant des expertes du domaine, et des données

statistiques. Parconséquent on a deux méthodes de construction d'un modèle de réseau Bayésien, le choixde la méthode à utiliser est imposé par la source des informations disponibles.

- **La construction d'un réseau Bayésien en collaboration avec les experts du domaine**

La construction d'un réseau Bayésiens **[24]**en utilisant les connaissances des experts dudomaine est une tâche difficile et consommatrice du temps, elle est menée par un ensemblede spécialistes. Un expert en réseaux Bayésiens dirige le processus de construction, il posedes questions pertinentes, et explique les suppositions encodées dans le modèle au reste dugroupe des spécialistes.

- **La construction d'un réseau Bayésien à partir des données statistique (Modélisationbasé sur les données)**

La modélisation basée sur les données consiste à identifier la structure d'un réseauBayésien à partir d'une source de données**[24].** La base de données peut être complète, commeelle peut être incomplète par conséquent, les méthodes d'apprentissage de structure desréseaux Bayésiens se divise en deux classes :

La classe des algorithmes d'apprentissage de structure àpartir des données complètes : il existe différents méthodes d'apprentissages de structures à partir de données complètes parmi la méthode basées sur la recherche d'indépendances conditionnelles,son procédure est de faire des tests sur les indépendances conditionnelles permettant au final de retrouver la structure recherchée comme l'algorithme PC et IC **[25]**, algorithme BNPC**[25]**, L'algorithme QFCI**[26].** Et des méthodes de recherche employant un score comme l'algorithme K2 **[26].**

Et la classe des algorithmes d'apprentissage de structure à partir des donnéesincomplètes parmi ces algorithmes : L'algorithme EM et l'algorithme SEM**[27]**.

II. 3.2 Apprentissage de paramètres :

L'apprentissage des paramètres d'un réseau bayésien se fait à partir de données relatives au problème à modéliser, consiste à calculer des probabilités a posteriori (après observation d'événements), dans ce cas on distinct deux type de donnée (données complètes et données incomplètes).

a. A partir de données complètes :

Nous cherchons ici à estimer les distributions de probabilités (ou les paramètres des lois correspondantes) à partir de données disponibles. L'estimation de distributions de probabilités, paramétriques ou non, est un sujet très vaste et complexe. Nous décrirons ici les

méthodes les plus utilisées dans le cadre des réseaux bayésiens, selon que les données à notre disposition sont complètes ou non.

- **Apprentissage statistique**

  Dans le cas où toutes les variables sont observées, la méthode la plus simple et la plus utilisée est l'estimation statistique qui consiste à estimer la probabilité d'un événement par la fréquence d'apparition de l'événement dans la base de données. Cette approche, appelée maximum de vraisemblance (MV).

- **Apprentissage bayésien :**

  L'estimation bayésienne suit un principe quelque peu différent. Il consiste à trouver les paramètres les plus probables sachant que les données ont été observées, en utilisant des a priori sur les paramètres.

b. A partir de données incomplètes :

Les réseaux bayésiens (BNs) sont des structures fondamentales pour l'inférence probabiliste et modélisation. Paramètres du réseau BN, leurs tables de probabilités conditionnelles, peuvent avoir besoin d'être appris. Lorsque la structure de BN est connu, mais les données sont incomplètes, plusieurs algorithmes ont été utilisés : Espérance Maximisation (EM), par chaînes de Markov, méthodes de Monte Carlo **[28]**, comme l'échantillonnage de Gibbs, et les méthodes de descente de gradient. La méthode la plus couramment utilisée est fondée sur l'algorithme itératif EM (Expectation Maximisation).

Et d'autres algorithmes comme MMHC **[29]** et MDL **[30]**. Ces méthodes choisir la structure de BN de candidat basé sur la fonction de score quantitative.

II.4 Conclusion :

Les réseaux Bayésiens demeurent un outil puissant dans la modélisation de problèmes complexes et le raisonnement à partir de l'incertain. Nous avons introduit dans ce chapitre la notion de ces modèles graphiques probabilistes. Dans le prochain chapitre on va présenter comment utiliser les réseaux Bayésiens pour extraire les motifs fréquents.

Chapitre III :

Extraction de motifs fréquents à partir des données massives en utilisant les réseaux Bayésiens

III.1Introduction

Dans ce chapitre, on va expliquer comment utiliser les Réseau Bayésiens pour l'extraction de motifs pertinents à partir des données de masse. Pour ce fairenous présentons à travers ce chapitre en premier lieu, comment extraire les motifs fréquents à partir d'une base de données.En deuxième lieu, nous étudions la façon d'extraire les motifs fréquents à partir des données massive en utilisant les réseaux Bayésiens.

III. 2 Réseau bayésien pour l'extraction de motifs fréquents

La représentation des connaissances et le raisonnement à partir de ces représentationsa donné naissance à de nombreux modèles. Les modèles graphiques probabilistes, et plusprécisément les réseaux bayésiens, se sont révélés des outils très pratiques pour lareprésentation de connaissances incertaines et le raisonnement à partir d'informationsincomplètes, dans de nombreux domaines comme la bio-informatique, la gestion du risque,le marketing, la sécurité informatique, le transport, etc. La partie graphique des RBs offreun outil intuitif inégalable et attractif dans de nombreuses applications où les utilisateursont besoin de "comprendre" ce que raconte le modèle qu'ils utilisent. La construction deces modèles à partir de données permet aussi de découvrir des connaissances utiles auxexperts, en allant – sous certaines réserves - jusqu'à la découverte de relations causales.

Grâce aux capacités de modélisation offertes par les réseaux Bayésiens, ces derniersont été utilisés entant qu'un modèle de représentation et d'interprétation de connaissances.En outre, les réseaux Bayésiens sont équipés d'un mécanisme d'inférence, ce qu'il les apermis d'être utiles pour la validation des connaissances extraites sous forme des règles d'association dans le cadre de la fouille de données. Cependant, Les méthodes proposées dans le cadre de l'apprentissage de structure ont transformé les réseaux Bayésiens non seulement en outil de représentation des connaissances, mais aussi en outil de découverte de celles-ci.

La construction d'unmodèle (Structure et paramètres) de RB par apprentissage à partir d'un ensemble de données permet de représenter d'une façon compacte et claire les relations de dépendance et d'indépendance qui existent entre les attributs de cet ensemble sous formes des règles représentées par des arcs reliant les nœuds (attributs) d'un RB. Par conséquent

l'apprentissage des RBs peut être une méthode d'extraction des règles d'associations, ce qui implique qu'ils peuvent êtres aussi utilisés pour extraire des motifs fréquents.

Dans un RB,une règle d'association comme par exemple : Fumeur →(Cancer du poumon, Bronchite) est représenté par deux arcs, un reliant le nœud Fumeur au nœud Cancer du poumon et un autre reliant le nœud Fumeur au nœud Bronchite. {Fumeur} est un motif et {Cancer du poumon, Bronchite} un motif et cette règle on peut la choisir comme motifs fréquents.

Nous avons choisis les RBs pour extraire les motifs fréquents pour les raisons suivantes :

- Ils offrent une représentation compacte et intuitive des relations "cause-effet", en général, des relationsde dépendance et d'indépendance conditionnelle.
- Ils représentent un modèle cohérent et fondé mathématiquement qui intègre l'incertitude.
- La possibilité de construire automatiquementun modèle de RB à partir des données complètes et aussi à partir des données incomplète (ce qui 'est très intéressent dans le cas des bases de données avec des valeurs des champs manquantes).
- La possibilité de construire automatiquementun modèle de RB à partir des données avec des variables cachés.

La procédure de construction d'un RB à partir des données commence premièrement par la phase d'apprentissage de structure qui a pour but de trouver le meilleur graphe représentant la tâche à résoudre.Nous avons choisis l'algorithme K2qui est un algorithme efficace et rapide en comparaison avec le reste des algorithmes d'apprentissage de structure des RBs.

III. 2.1 La version classique de l'Algorithme classique K2

Cooper &Hersovits**[26]**ont proposé une méthode largement utilisée dans l'apprentissage de structure des RB. Cette méthode est appelée K2. L'algorithme K2 présente un inconvénient de demander un ordre d'énumération des variables en paramètre d'entrée. Le but de cette méthode est donc de maximiser la probabilité de la structure sachant les données dans l'espace des DAG (espace des graphes acycliques dirigés) en respectant la contrainte de l'ordre d'énumération. Cette contrainte permet de restreindre l'espace de recherche et de le limiter par la recherche seulement des arcs intéressants. En effet, l'algorithme K2 teste l'ajout de parents

en respectant cet ordre de la manière suivante : si X_i est le premier sur la liste, alors il ne pourra pas avoir de parents. Si non, si X_i est cita avant X_j alors il ne pourra y avoir d'arc de X_j vers X_i. Autrement dit, l'ensemble des parents choisis pour un nœud est le sous-ensemble de nœuds qui augmente le plus le score parmi l'ensemble des nœuds le précédent dans l'ordre d'énumération.Ainsi, l'espace de recherche est réduit à l'ensemble des DAG respectant cet ordre.

Comme on vient de dire l'algorithme classique K2 nécessite un ordre total a priori sur les nœuds afin de s'assurer que le graphe résultant soit sans cycle. En partant de qu'il augmente le plus la probabilité de la structure résultant, s'exprimée par la fonction *g*. K2 s'arrêt lorsque l'ajout de parent ne permet plus d'améliorer la probabilité.

La fonction de score **[31]** dans K2 est la suivante:

$$g(i, \pi_i) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i-1)!}{(N_{ij}+r_i-1)!} \prod_{k=1}^{r_i} N_{ijk}! \qquad (3.1)$$

Où :

- n le nombre de variables ;
- $\emptyset_i$: liste de toutes les instanciations possibles de x_i dans D ;
- $q_i = |\emptyset_i|$;
- r_i le nombre de modalités de la variable X_i ;
- N_{ijk} le nombre d'occurrence ou le Xi=k et les parent(Xi)=j ;
- $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$ (3.2)

Un réseau bayésien est un DAG G= (V, E) dans lequel ce qui suit est titulaire **[32]** :

(1) V est l'ensemble des variables aléatoires et qui fait les nœuds du réseau.

(2) E est l'ensemble des arcs orientés reliant des paires de nœuds. Une flèche de nœud X en nœud Y signifie que X a une influence directe sur Y (X, Y ∈ V et X ≠ Y).

(3) Chaque nœud possède un CPT (une table de probabilité conditionnelle) qui quantifie les effets que les parents ont sur le nœud. Les parents de nœud X sont tous ceux qui ont des flèches pointant vers elle.

Soit Z un ensemble de n paramètres discrets, où une variable x_i dans Z a r_i affectations de valeurs possibles: $(v_{i1}, \ldots, v_{ij})$, Soit D une base de données avec m enregistrements (des

lignes), dont chacun contient une affectation de valeur pour chaque variable à Z. soit $B_s$désigner une structure de croyance du réseau contenant seulement les paramètres de Z. chaque variable $x_i$dans $B_s$dispose d'un ensemble de parents, que nous nous la représentons par une liste de $r_i$paramètres. Soit w_{ij} désigner le $j$ème enregistrement unique instancielle de L_i par rapport à D. Supposons qu'il y ait q_i enregistrements uniques instancielle de L_i. Définir N_{ijk} comme le nombre de cas dans D dans lequel la variable x_ia la valeur v_{ik} et L_i est instancié comme w_{ij}. Le pseudo-code suivant exprime l'algorithme de recherche heuristique, appelée K2 **[31].**

Algorithme 3.1. K2 Classique

Entrer *:*

1) *Un ensemble de n noeuds, une séquence de nœuds*
2) *Une limite supérieure u le nombre de noeuds parents*
3) *Une base de données D contenant m enregistrements*

Sortie *: des parents pour chaque nœud*

Etapes :

pour *i :=1 to n* ***faire***

$\pi_i := \emptyset$

$P_{old} := g(i, \pi_i)$ *// depuisl'équation (3.1)*

OKTo Proceed:=true

whileOKToProceed and $|\pi_i| < u$ *Do*

Soit z un noeud dans $Pred(x_i) - \pi_i$ *maximisant* $g(i, \pi_i \cup \{z_i\})$

$P_{new} := g(i, \pi_i \cup \{z_i\})$

Si $P_{new} > P_{old}$ *Then*

$P_{old} := P_{new}$

$\pi_i := \pi_i \cup \{z\}$

sinon*OKToProceed:=false*

Fin while

Output ('Noeud:', x_i*, 'parents de ce noeud:',* π_i*)*

Fin Pour

Pour chaque nœud i et ses parents π_i on utilise l'équation (3.1) pour calculer le score des structures candidates de chaque nœud, et utiliser *Pred* (x_i) pour désigner l'ensemble des nœuds qui précèdent x_t. Nous nous considérons que le parent précédent de chaque nœud en fonction de la séquence des nœuds. A la fin on aura tous les structures candidates de chaque nœud avec un score maximum et à la fin ont découvert les motifs fréquents.

Apres avoir un réseau bayésien défini par un graphe et la distribution de probabilité associée, il faudra estimer les probabilités conditionnelles de chaque nœud du réseau.Les tables de probabilités sont définies par des statistiques relatives au problème à résoudre (peuvent aussi être déterminées par des experts). Chacune des variables disposes d'une table de probabilités conditionnelles relatives aux variables causales dont elle dépend.

Dans notre cas nous avons estimé les probabilités conditionnelles de chaque nœud du réseau par apprentissage des paramètres (*critère : Apprentissage statique*). Et tous les variables sont observées (critère : Apprentissage de paramètres données complètes) et ont utilisé la méthode la plus simple et la plus utilisé est l'estimation statistique de probabilité d'un évènement par la fréquence d'apparition de l'évènement dans la base de données, cette méthode est appelée « maximum de vraisemblance ».

$$\hat{P}\left(X_i = x_k \middle| Pa(X_i = x_j)\right) = \hat{\theta}_{i,j,k}^{MV} = \frac{N_{ijk}}{\sum_k N_{ijk}}. \qquad (3.3)$$

Où N_{ijk} est le nombre d'événements dans la base de données pour lesquels la variable X_i est dans l'état x_k et ses parents sont dans la configuration x_j.

III.3Les Réseaux bayésiens pour l'extraction de motifs fréquents à partir de données massives:

Avec le développement rapide d'applications (e-santé, e-commerce,..), de plus en plus de données sont générées et collectés à des expériences scientifiques. Centré sur la gestion et l'analyse de données massives, les données du calcul intensif peuvent générer de nombreuses requêtes, chacune impliquant l'accès à des calculs supercalculateur de classe sur giga-octets ou téraoctets ou pétaoctets de données **[33]**. Ainsi, la découverte de connaissances à partir de données massives est naturellement indispensable pour les applications intensives de paradigmes données, telles que le Web 2.0 et le business intelligence à grande échelle.

III.3.1 Le modèle de programmation Mapreduce et la plate-forme Hadoop :

MapReduce est un cadre de programmation pour l'informatique distribuée sur des ensembles de données volumineux qui a été introduit par Google en 2004. C'est une abstraction qui permet aux utilisateurs de créer facilement des applications parallèles, tout en cachant les détails de la distribution des données, l'équilibrage de charge et la tolérance aux pannes **[34].**

MapReduce nécessite la décomposition d'un algorithme dans la Map et de réduire les étapes. Dans la phase de la Map, les données d'entrée sont divisés en blocs sont traitées comme un ensemble de paires d'entrée clé-valeur et en parallèle par de multiples mappeurs. Chaque mappeur s'applique à chaque donnée et l'attribuer une fonction de Map spécifié par l'utilisateur et produit en sortie un ensemble de paires de valeurs clés intermédiaires. Ensuite, les valeurs ayant la même clé sont regroupées (la phase de tri et de lecture aléatoire) et transmises à reduce, qui fusionne les valeurs appartenant à la même clé selon une fonction reduce définie par l'utilisateur.

Hadoop, une implémentation de MapReduce, fournit un cadre pour la distribution des données et des emplois, MapReduce spécifié par l'utilisateur à travers un grand nombre de nœuds de cluster ou de machines. Il est basé sur l'architecture maître / esclave. Le seul serveur maitre, appelé JobTracker, reçoit une affectation des tâches de l'utilisateur, distribue la Map et de réduire les tâches de nœuds esclaves (tasktrackers) et surveille leur progression. Stockage et la distribution des données aux nœuds esclaves sont traitées par le système de fichiers distribués Hadoop (HDFS). Dans le texte suivant, un nœud Hadoop peut désigner un TaskTracker ou machine JobTracker. La tache Map décrit le travail exécuté par un mappeur sur une partition d'entrée. La tache reduce traité les dossiers avec la même clé intermédiaire. Un mapper / réducteur peut avoir multiple tache Map / Reduce.

III.3.2 K2 basé sur Mapreduce pour l'apprentissage de structuredes RBs à partir des données de masse

L'architecture de l'algorithme K2basé sur MapReduce **[35]** est comme suit :

- Etape1 :Génération de structures candidates.
- Etape 2 : Les paramètres nécessaires dans l'équation (3.1) sont obtenus par le comptage statistique à partir de l'ensemble de données.
- Etape 3 : Décompose sur trois traitements :

  Etape 1 : Calcule du score de chaque structure candidate.

  Etape 2 : Calcule du score maximum de chaque nœud père.

  Etape 3 : A partir du fichier output du traitement des nœuds, chaque nœud père et son score consiste a trouvé dans le fichier output du traitement 1 la structure optimal local, après fusionner tous les structure optimal local pour trouver la structure optimal global.

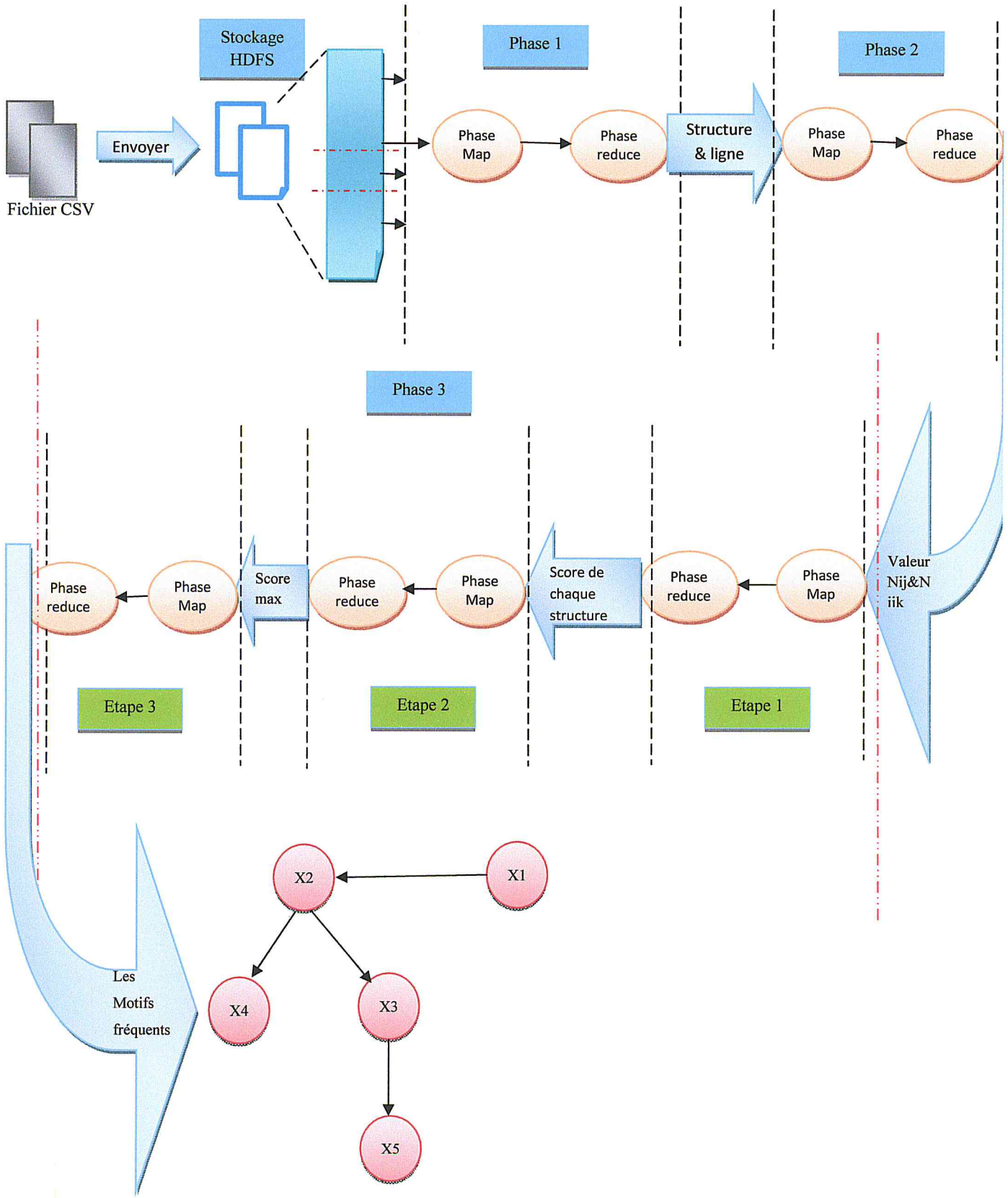


Figure 3-1L'architecture de l'algorithme K2 basé sur MapReduce

Dans cette architecture, les données massives seront divisées en *m*map traité de manière dynamique, dont les résultats seront collectées pour le résultat final par le processus reduce qui généralement unique.

III.3.2.1 Génération des structures candidates basé sur Mapreduce

Selon une séquence de nœuds seq et le Sucs (x_i), qui représente l'ensemble des nœuds qui suivant x_i dans la séquence Seq nœud prédéfini, les structures candidates du nœud x_i peuvent être obtenues. Par exemple, nous supposons Sucs (x_1) = {x_2, x_3}, Sucs (x_2) = {x_3}, Sucs (x_3) = {} et Sucs = {x_1, x_2, x_3}. Pour le nœud x_1, il y a 3 structures candidates ($x_1 \rightarrow x_2$), ($x_1 \rightarrow x_3$) et ($x_1 \rightarrow x_2$, $x_1 \rightarrow x_3$). Pour le nœud x_2, la structure de candidat ($x_2 \rightarrow x_3$).

Pour obtenir les successeurs nous développons l'algorithme 3.2, y compris le processus Map et le processus Reduce respectivement, pour chaque Map nécessite un nœud père et une séquence de nœuds et le nombre maximum de fils, et pourreducenous donne tous les structures candidates.

Algorithme 3.2 :
Entrer :

1. *Nœud père xi.*
2. *Séquence de nœud {x0,…,xn}.*
3. *Nombre maximum de fils.*

Sortie :*pour un nœud père tous les structures candidates d'un nœud père.*

Algorithme 3.2.1, *Map (String clef, String valeur)*
// Clef: clef d'enregistrement de données, Valeur : Contenu du dossier de données

```
String noeud_pere;
String seq_noeud[];
int taille_seq_noeud=n;
int pos_pere=m;
pour (i=1, n) {
        clef = noeud_pere&&sec_noeud[i];
        valeur=1;
        String srtct=clef;
        Insert.fichier_noeud_per(srtct);
        Générer <Clef, valeur>
        }
lire.fichier_noeud_pe(clef);
tantque(clef !=null){
```

```
        if(clef (dernier_position)==seq_neoud(dernier)){
        //passe à la structure suivant
        }
        Else {
                int k=postion_dernier_noeud_cle;
                pour (i=k , n){

                        clef=clef+seq_neoud[i];
                        valeur=1
                        if (clef.lentgh<=max){
                        Stringstrct=cle;
                        Insert.fichier_noeud_per(strct);
                        générer<Clef, valeur>
                        }
                        Else{ // passe à la structure suivant
                        }
                }
                }
}
Fin
```

Les résultats intermédiaires générés par les processus Map seront utilisé dans le processus reduce.

Algorithme 3.2.2,*Reduce(String clef, String valeur)*

// Clef: Structure candidate pour un noeud père.

clef =strcture_candidate;

valeur=L;

Générer <Clef, valeur>

III.3.2.2 Calcul nombre d'occurrence N_{ij} et N_{ijk} basé sur Mapreduce:

En général, r_i dans l'équation (3.1) est une constante, que nous ne calculons pas en parallèle. Supposons qu'il existe un ensemble de données d'échantillons fixés comme le Tableau 3.1, dans laquelle chaque colonne représente une variable unique et chaque enregistrement représente un cas sur les variables x_1, x_2 et x_3 qui dénotent le tabagisme, le cancer et la tuberculose, respectivement. Pour plus de simplicité, 0 et 1 sont utilisés pour désigner absent et présent respectivement. Dans cet exemple, r_i est 2.

Tableau 3.1 : Données simples**[35].**

Cas	Les valeurs des variables pour chaque cas		
	X_1	X_2	
1	Présent	Absent	Absent
2	Présent	Présent	Présent
3	Absent	Absent	Présent
4	Présent	Présent	Présent
5	Absent	Absent	Absent
6	Absent	Présent	Présent
7	Présent	Présent	Présent
8	Absent	Absent	Absent
9	Présent	Présent	Présent
10	Absent	Absent	Absent

Pour obtenir les paramètres de l'équation (3.1), nous développons l'algorithme 3.3, y compris le processus Map et le processus Reduce respectivement pour accéder aux données de l'échantillon et en additionnant les résultats intermédiaires. Dans chaque processus Map les paramètres N_{ijk} et N_{ij} seront obtenus comme résultats intermédiaires

Algorithme 3.3

Entrer :

1) *m enregistrements sur n variables*
2) *Enregistrer L_x ($1 \leq x \leq m$) y compris les valeurs de $\{s_1, s_2, ..., s_n\}$, ou Si est un 1 ou 0, désigne le i ème variable est présent ou absent respectivement.*
3) *la structure candidate (structure $(x_i;.....x_n)$ avec ($1 \leq i \leq n$)).*

Sortie : *Paire <clef/Valeur> de "N_{ij}" (ou "N_{ijk}") et la valeur de "N_{ij}" (ou "N_{ijk}").*

Algorithme 3.3.1*, Map (String clé, String valeur)*
// Clef: clef d'enregistrement de données, Valeur : Contenu du dossier de données
Calculer N_{ij}et N_{ijk} de chaque structure à partir de L_x ($1 \leq i \leq n$; $1 \leq j \leq q_i$; $1 \leq k \leq r_i$)
// Par équation (3.2)

Générer (Structure($x_0,...,x_n$) && Lx ($1 \leq i \leq n$; $1 \leq j \leq q_i$; $1 \leq k \leq r_i$) &&N_{ijk}, 1) ;
Générer (Structure($x_0,...,x_n$) &&Lx ($1 \leq j \leq q_i$) &&N_{ij}, 1) ;

Fin

Pour le premier enregistrement dans le tableau 3.1, nous pouvons obtenir la paire <clef/valeur>de la <structure(x_1=1, x_2 =0)&&L_1(x_1=1,x_2=0)&&N_{ijk}, 1> et <structure(x_1=1, x_2 =0)&&L_1(x_2=0)&&N_{ij}, 1> par l'algorithme 3.3.1, qui est le nombre d'enregistrements satisfaisant $x_1 = 0$, $x_2 = 0$ est égal à 0 et la valeur de N_{ijk} et N_{ij} est égal à 0. De même, nous pouvons obtenir pour les paires pour les autres lignes L_x.

Le résultat intermédiaires qui est obtenu par le processus map,< Structure(x_0,...,x_n) && L_x ($1\leq i\leq n$; $1\leq j\leq q_i$; $1\leq k\leq r_i$) &&N_{ijk} , 1) > et < Structure(x_0,...,x_n) && Lx ($1\leq j\leq q_i$) &&N_{ij}, 1> le processus reduce additionner les valeurs qui ont la même clef pour obtenir à la fin N_{ij}et N_{ijk} total pour chaque structure. Algorithme (3.3.2).

Algorithme 3.3.2, *Reduce(String clef, Iterator valeur)*

*// **Clef**: < Structure(x_0,...,x_n) && L_x ($1\leq i\leq n$; $1\leq j\leq q_i$; $1\leq k\leq r_i$) &&N_{ijk} , 1) > ou et < Structure(x_0,...,x_n) && L_x ($1\leq j\leq q_i$) &&N_{ij}, 1>, valeur: dans l'itérateur qui ont la même clef de la paire <clef/ Valeur>*
Result:=0
***Pour** chaque paire clef/valeur Faire*
valeur:=Interator.Next()
resultat:=resultat+valeur
***Fin** Pour*
Sortie <clef, résultat>

III.3.2.3 Calcul du score basé sur MapReduce

Pour tous les Sucs (x_1) = {x_2, x_3}, Sucs (x_2) = {x_3}, Sucs (x_3) = {} et Sucs = {x_1, x_2, x_3}. Le nœud x_1, il a 3 structures candidates ($x_1 \rightarrow x_2$), ($x_1 \rightarrow x_3$) et ($x_1 \rightarrow x_2$, $x_1 \rightarrow x_3$), Pour calculer le score de chaque structure candidate nous développons l'algorithme 3.4.

Algorithme 3.4. *Fonction de score basée sur MapReduce*
Entrer : structure candidate locales de chaque nœud père avec N_{ij} ou N_{ijk}.
Sortie : Score du structure candidate locales.

Algorithme 3.4.1. *Map (String clef, String valeur)*
// Clef: Structure(x_0,....x_n), Valeur: N_{ij}ou N_{ijk}
***If**(valeur=N_{ijk}) factorielle= Nijk!*

***Else** factorielle=1/(N_{ij}+1)!*

Générer <clef/valeur>paire <Structure(x_0,....,X_n), factorielle>
//Clef se compose du Structure(x_0,...,X_n), et valeur est factorielle de n_{ij}ou n_{ijk};

Les résultats intermédiaires générés par les processus Map sont formées en tant que <Clef/Valeur> paire <Structure(x_0,....x_n), factorielle>. Dans chaque processus reduce (Algorithme 3.4.2) produit tous les valeurs qui ont les mêmes clefs. Et on obtient à la fin du processus le score de chaque structure local.

Algorithme 3.4.2*Reduce(String clef, String valeur)*
// Clé: structure(x_0,....x_n), valeur: factorielle
Score:=1;
***Pour** chaque structure(x_0,....x_n)*
*Score =Score*factorielle*
FinPour
Généré le pair clé <structure(x_0,....x_n), Score>
Fin

Et les résultats générer par l'algorithme 3.4 de la formées <*structure(x_0,....x_n), Score*>sont utiliser dans l'algorithme 3.5 pour la détermination du score le plus élevé pour chaque nœuds pères.

Algorithme 3.5. *Maximum score pour les nœuds père basé sur MapReduce*
Entrer : structure(x_0,....x_n)et son Score
Sortie : le score maximum de chaque nœud père

Algorithme 3.5.1. *Map (String clef, String valeur)*
// Clef: structure(x_0,....x_n); Valeur: score
Nœud_père =L'extraction de nœuds père dans la structure locale
Générer clef/valeur paire < Nœud_père, score>
//Clef se compose du Nœud_père et valeur est le score.

Les résultats intermédiaires générés par les processus Map sont formées en tant que <Clef/Valeur> paire <Structure(x_0,....x_n), score>. Et dans chaque processus Reduce (Algorithme 3.5.2) calculle maximum du score pour le même clé (Nœud_père).

Algorithme 3.5.2 *Reduce (String clef, String valeur)*
// Clé: Nœud_pere , valeur: score
Maximum=valeur
itirateur:valeur
minimum=valeur ;
if*(minimum < = Maximum) maximum= Maximum;*
Else *maximum=minimum ;*
Endif
Généré le pair clé <Nœud_pere ; maximum>
Fin

Maintenant on prend en charge les résultat de l'algorithme (3.4) et (3.5), et pour chaque Nœud_père(x_i) et son score, on le compare en parallèleavec les résultats de l'algorithme (3.4) la structure(x_0,....x_n) et son score qui ont le même Nœud_père (x_i) et le même score, on

prendre en compte cette structure(x_0,....x_n) locale comme une structure optimale locale pour le Nœud_père (x_i).
Toutes les structures optimales locales disponibles reste que le fusionnement des structures optimales locales pour obtenir la structure optimale globale.

Exemple :

$x_1 \rightarrow x_2$ c'est la structure optimale locale pour le nœud x_1.

$x_2 \rightarrow x_3$ c'est la structure optimale locale pour le nœud x_2.

Par fusionnement on obtient $x_1 \rightarrow x_2 \rightarrow x_3$ comme une structure optimale globale.

III.3.3 Apprentissage de paramètre

Une fois la structure est créée (apprentissage de structure), il reste à remplir la table de probabilités conditionnelles de chaque nœud (apprentissage des paramètres).

Les tables de probabilités sont définies par des statistiques relatives au problème à résoudre (peuvent aussi être déterminées par des experts). Chacune des variables disposes d'une table de probabilités conditionnelles relatives aux variables causales dont elle dépend.

On va utiliser l'estimation statistique de probabilité d'un évènement par la fréquence d'apparition de l'évènement dans la base de données, cette méthode est appelée « maximum de vraisemblance ».

L'architecture de la méthode maximum de vraisemblance basée sur MapReduce est comme suite :Extraire les N_{ijk} et les N_{ij} pour chaque motifs ;calcul les paramètres pour chaque motifs.

III.3.3.1 Extraire les N_{ijk} et N_{ij} pour chaque motif

L'algorithme 3.6 calcul les paramètres N_{ijk} et N_{ij}qui seront obtenus comme résultats intermédiaires.

Algorithme 3.6:
***Entrer** : les motifs, le fichier Input de l'algorithme 3.4*
***Sortie** : N_{ijk} pour chaque motif, fichier_nij pour chaque motif*

***Algorithme 3.6.1.** Map (String clef, String valeur)*
Clef: clef d'enregistrement de données, Valeur : Contenu du dossier de données

```
if(motif ==structure){
        if(structure.contient(nijk)){
        clef=structure;
        valeur =la_valeur_nijk;// pourcette structure
        générer<Clef, valeur>;
        }
else{
        insert.FichierNij(structure && valeur(nij));
        }
```

}

Fin

Algorithme 3.6.2.*reduce (String Clef , intvaleur)*

Entre Clef : structure
Valeur : la valeur de Nijk pour structure
Sortie : toutes les structures avec valeur de nijkcorrespondanteà la structure
Générer<Clef, valeur>
Fin

III.3.3.2 Calcul les paramètres pour chaque motifs

On prend en charge les résultats de l'algorithme 3.6 pour calculer table de probabilité conditionnel de chaque motif(Algorithme 3.7).

Algorithme 3.7 :
***Entre** : fichier de N_{ij}, le fichier de sortie de l'algorithme (3.6)*
***Sortie** : table de probabilité de chaque motif*

Algorithme 3.7.1 *Map(String Clef, Double valeur)*

Entrer :
1. *Clef: clef d'enregistrement de données,*
2. *Valeur : Contenu du dossier de données*

Sortie : *probabilité de chaque structure*
structure =valeur
r = Lire.fichier_Nij
while*(r!=null){*
structure_nij=r
if*(strcture = structure_nij){*
int*nijk=valeur_nijk_structure*
int*nij=valeur_nij_structure_nij;*
double *d=nijk/nij;*
clef=structure;
valeur =d
Génère <Clef, valeur>
}
}

Algorithme 3.7.2 *reduce(string clef, double valeur)*

Entrer : clef, la structure
valeur : probabilité
sortie : la table de probabilité pour chaque motif
clef= structure

valeur =probabilité
Génère<Clef, valeur>

III.4Conclusion

Dans ce chapitre, nous avons explicité l'approche proposée pour l'extraction de motifs fréquents à partir de données massives basée sur l'apprentissage du réseau Bayésien.Nous avons présenté une méthode basée sur MapReduce pour l'apprentissage du réseau Bayésien à partir de données massives tout en se concentrant sur l'apprentissage de la structure DAG du réseau bayésien puis l'apprentissage de paramètres pour calculer de la table des probabilités conditionnelles. Dans le prochain chapitre, nous allons valider les résultats obtenus.

Chapitre IV : Tests et résultats

IV.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.

IV.2. Environnement de travail

Dans cette section nous présenterons les différents outils utilisés pour réaliser notre application.

Java (Version 3.3.2 Europa) Eclipse est un projet**[W01]**, décliné et organisé en un ensemble de sous-projets de développements logiciels, de la Fondation Eclipse visant à développer un environnement de production de logiciels libres qui soit extensible, universel et polyvalent, en s'appuyant principalement sur Java. Figurant parmi les grandes réussites de l'OpenSource, Eclipse est devenu un standard du marché des logiciels de développement, intégré par de grands éditeurs logiciels et sociétés de services. Les logiciels commerciaux Lotus Notes 8, IBM Lotus Symphony ou WebSphere Studio Application Developer sont notamment basés sur Eclipse. Son objectif est de produire et fournir des outils pour la réalisation de logiciels, englobant les activités de programmation (notamment environnement de développement intégré et frameworks) mais aussi d'ATL recouvrant modélisation, conception, testing, gestion de configuration, reporting... Son EDI, partie intégrante du projet, vise notamment à supporter tout langage de programmation à l'instar de Microsoft Visual Studio.

Cette dernière version est disponible depuis fin juin 2007**[W01]**, 310 développeurs répartis dans 19 pays ont écrit les 17 millions de lignes de codes qui la composent. Cette version porte officiellement 21 projets. Est compatible avec hadoop-0.19.1. On peut y écrire, compiler et exécuter les programmes dans le but est de faire un parallélisme dans l'exécution.Cette version peut être téléchargée gratuitement à partir du **[W02].**

Hadoop (version 0.19.1) Hadoop **[W03]**est un framework Java libre destiné à faciliter la création d'applications distribuées et échelonnables (scalables). Il permet aux applications de travailler avec des milliers de nœuds et des pétaoctets de données. Hadoop a été inspiré par les publications MapReduce, GoogleFS et BigTable de Google.

Hadoop a été créé par Doug Cutting et fait, en 2009, partie des projets de la fondation logicielle Apache. Il contient le HDFS est un système de fichiers distribué, extensible et portable développé par Hadoop à partir du GoogleFS. Écrit en Java, il a été conçu pour stocker de très gros volumes de données sur un grand nombre de machines équipées de disques durs banalisés. Il permet l'abstraction de l'architecture physique de stockage, afin de manipuler un système de fichiers distribué comme s'il s'agissait d'un disque dur unique.

Une architecture de machines HDFS (aussi appelée cluster HDFS) repose sur deux types de composants majeurs :

NameNode (nom de nœud) : ce composant **[W03]** gère l'espace de noms, l'arborescence du système de fichiers et les métadonnées des fichiers et des répertoires. Il centralise la localisation des blocs de données répartis dans le cluster. Il est unique mais dispose d'une instance secondaire qui gère l'historique des modifications dans le système de fichiers (rôle de backup). Ce NameNode secondaire permet la continuité du fonctionnement du cluster Hadoop en cas de panne du NameNode d'origine.

DataNode (nœud de données) : ce composant **[W03]** stocke et restitue les blocs de données. Lors du processus de lecture d'un fichier, le NameNode est interrogé pour localiser l'ensemble des blocs de données. Pour chacun d'entre-eux, le NameNode renvoie l'adresse du DataNode le plus accessible, c'est-à-dire le DataNode qui dispose de la plus grande bande passante. Les DataNodes communiquent de manière périodique au NameNode la liste des blocs de données qu'ils hébergent. Si certains de ces blocs ne sont pas assez répliqués dans le cluster, l'écriture de ces blocs s'effectue en cascade par copie sur d'autres DataNodes.

Chaque DataNode sert de bloc de données sur le réseau en utilisant un protocole spécifique au HDFS. Le système de fichiers utilise la couche TCP/IP pour la communication. Les clients utilisent le RemoteProcedure Call pour communiquer entre eux. Le HDFS stocke les fichiers de grande taille sur plusieurs machines. Il réalise la fiabilité en répliquant les données sur plusieurs hôtes et par conséquent ne nécessite pas de stockage RAID sur les hôtes. Avec la valeur par défaut de réplication, les données sont stockées sur trois nœuds : deux sur le même support et l'autre sur un support différent. Les DataNodes peuvent communiquer entre-eux afin de rééquilibrer les données et de garder un niveau de réplication des données élevé.

Hadoop dispose d'une implémentation complète de l'algorithme de MapReduce**[W03]**.

Cygwin

Cygwin **[W04]**est une collection de logiciels libres à l'origine développés par Cygnus Solutions permettant à différentes versions de Windows de Microsoft d'émuler un système Unix. Il vise principalement l'adaptation à Windows de logiciels qui fonctionnent sur des systèmes POSIX (tels que les systèmes GNU/Linux, BSD, et Unix). Cygwin simule un environnement Unix sous Windows, rendant possible l'exécution de ces logiciels après une simple compilation. Les programmes ainsi portés sur Cygwin, fonctionnent mieux sur Windows NT, Windows 2000 et Windows XP que sur les versions antérieures de Windows,

mais certains peuvent s'exécuter de façon tout à fait acceptable sur Windows 95 et Windows 98.La bibliothèque Cygwin est une DLL nommée cygwin1.dll.

Cygwin est :

- Une collection d'outils qui fournissent un regard Linux a un environnement Windows.
- DLL (cygwin1.dll) qui agit comme une couche API Linux offrant des fonctionnalités importantes Linux API.

La DLL Cygwin fonctionne actuellement avec tous récent, sorti commercialement versions 64 bits de Windows 32 bits x86 et, à partir de Windows XP SP3.

The most recent version of the Cygwin DLL is 1.7.25. Install it by running setup-x86.exe (32-bit installation) or setup-x86_64.exe (64-bit installation). **[W05]**

IV.3 Jeu d'essai

Les données à analyser par les méthodes de data mining sont parfois incomplètes, inconsistantes, erronées, incompatibles entre elles, inadaptées ou encombrantes. Ces types de données sont courants et se retrouvent régulièrement dans les bases de données et d'entrepôts de données.

On a utilisé la base de données **[W06]**ASIA de 8 attributs et de 500 enregistrementset la base de données ALARM **[W06]**de 37 variables et de 500enregistrements pour la démonstration et la validation de notre travail.

IV.4 Présentation de l'interface utilisateur

Dans qui suit nous allons présenter les deux interfaces réalisées qui permettent à l'utilisateur de spécifier la configuration qui définit la structure du réseau Bayésien souhaité et d'avoir les résultats souhaités.

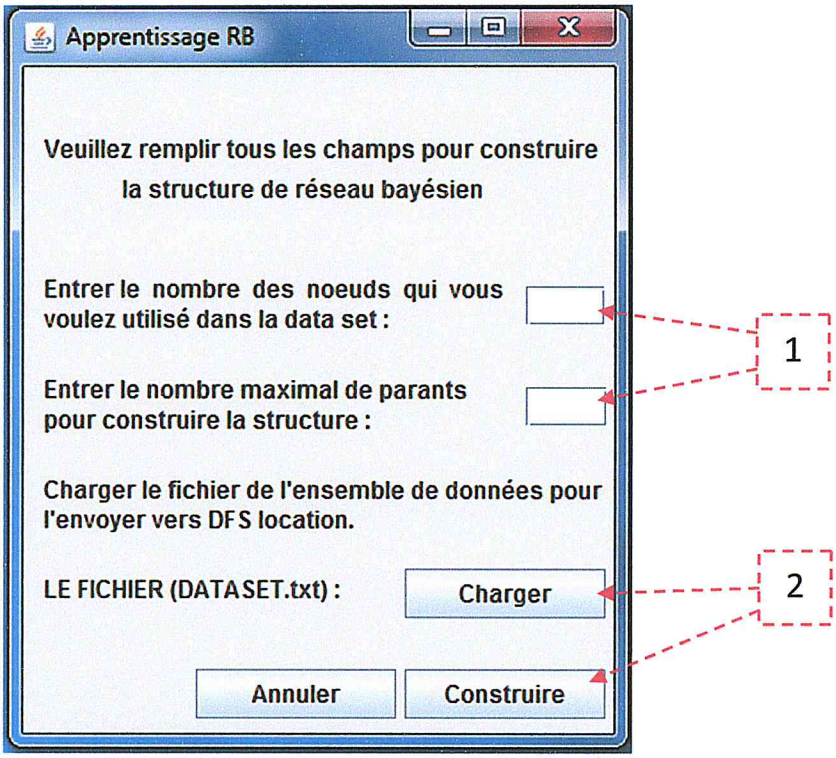


Figure 4.1- Interface pour la configuration du Réseau Bayésien

Dans cette interface nous avons 2 parties principales :

Partie 1 : c'est la zone du paramétrage de notre application qui permettra de spécifier une configuration pour les phases Map et Reduce pour obtenir une bonne performance du processus d'extraction de motifs fréquents. Le premier champ permet de spécifier le nombre de nœuds qu'ont utilisent, et les deuxièmes champs permet de spécifier le nombre maximal des parents pour la construction du réseau Bayésien.

Partie 2 : Dans cette partie l'utilisateur charge la base de données de type *.txt*ou.*csv* pour l'extraction de motifs en construisant le réseau Bayésien.

Et d'où l'utilisateur lance le processus d'extraction de motifs (Le bouton Construire).

Le forma de la base de données étant TXT ou CSV. Nous avons choisi TXT ou CSV pour les avantages :

- permettent de stocker des données tabulaires dans un fichier texte.
- Chaque ligne du fichier correspond à une ligne du tableau. Les valeurs de chaque colonne du tableau sont séparées par un caractère de séparation, en général une virgule ou un point-virgule. Chaque ligne est terminée par un caractère de fin de ligne (line break).
- Toutes les lignes contiennent obligatoirement le même nombre de valeurs (donc le même nombre de caractères de séparation).
- il est possible de travailler avec des fichiers de plus de 65536 lignes.

- les fichiers permettent d'échanger facilement des données avec d'autres programmes car les formatsTXT et CSV sont universel.

Une foi le fichier chargé, est envoyée dans le fichier Input dans le répertoire HDFS pour le commencement du partage des lignes de la base de données sur des mapper et des reducer. Il commence à calculer toutes les structures candidates possible pour chaque nœud père, puis les valeurs de Nij et de Nijk et à la fin, il calcule le score de chaque structure optimale local.

La fenêtre suivante qui s'affiche après ces traitements

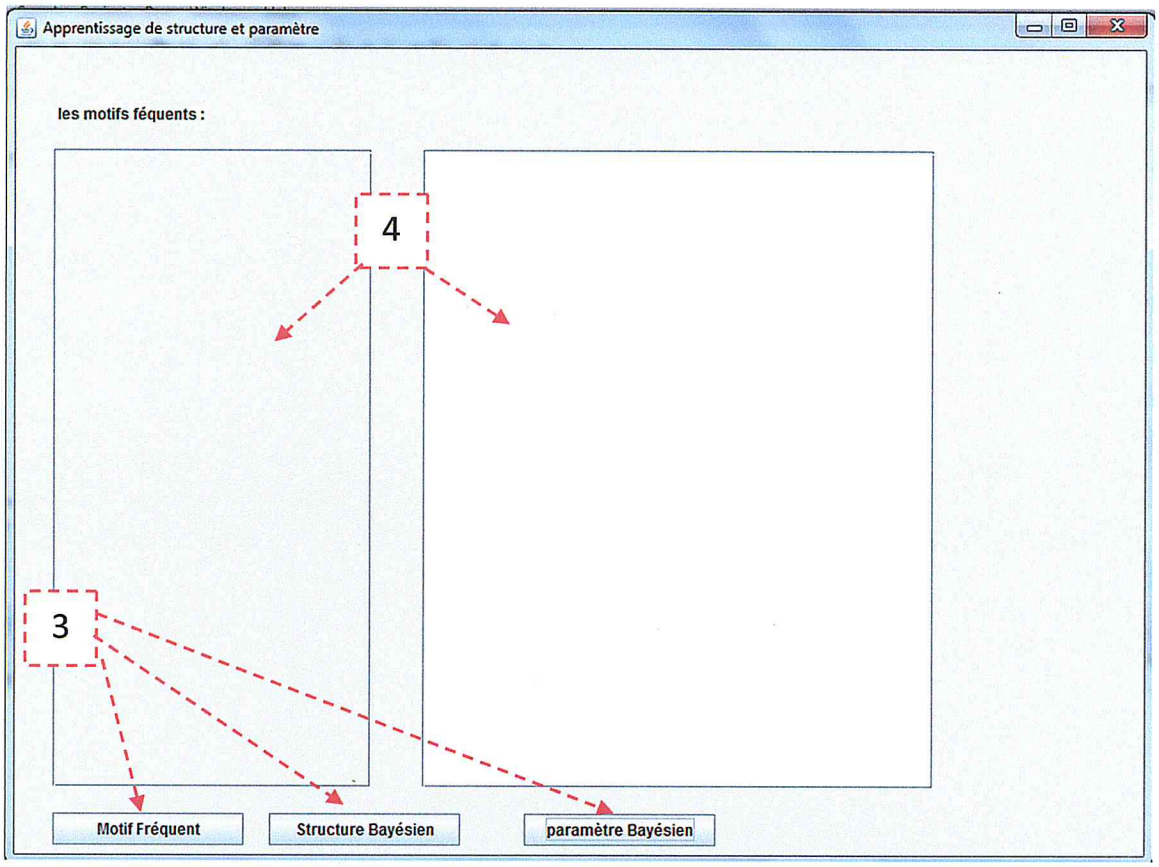


Figure 4.2- Interface de construction du Réseau Bayésien.

Phase 3 : Danscettephase l'utilisateur peut lancer le processus d'extraction de motifs.

- Le bouton *Motif fréquent* affiche tous les motifs fréquents qui construisent le réseau Bayésien.
- Le bouton *Structure Bayésien* affiche le réseau Bayésien à partir de motifs fréquents qu'on a calculé. Le réseau Bayésien est affiché dans une autre fenêtre.

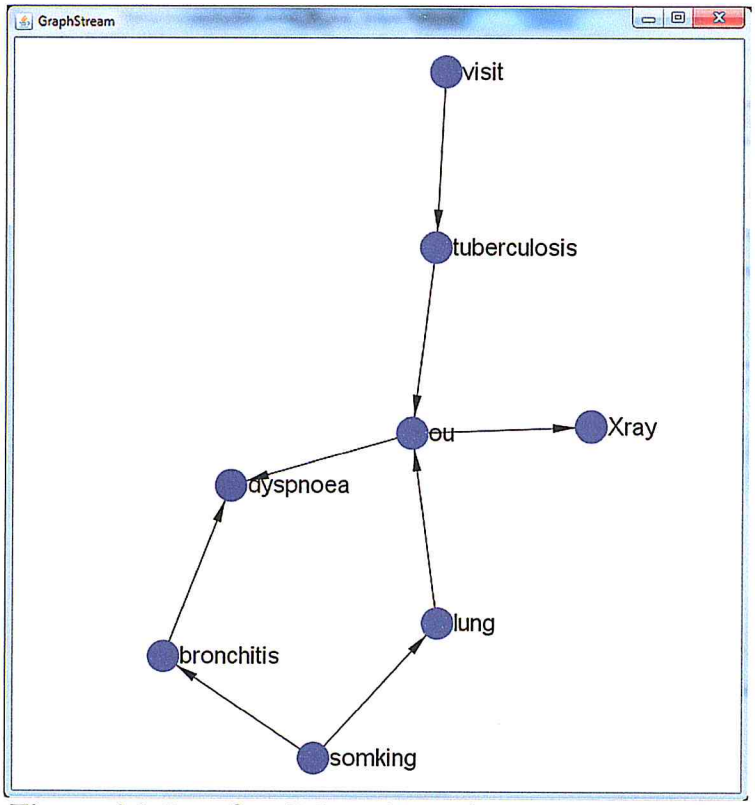


Figure 4.3- Interface Réseau Bayésien Structure -ASIA-

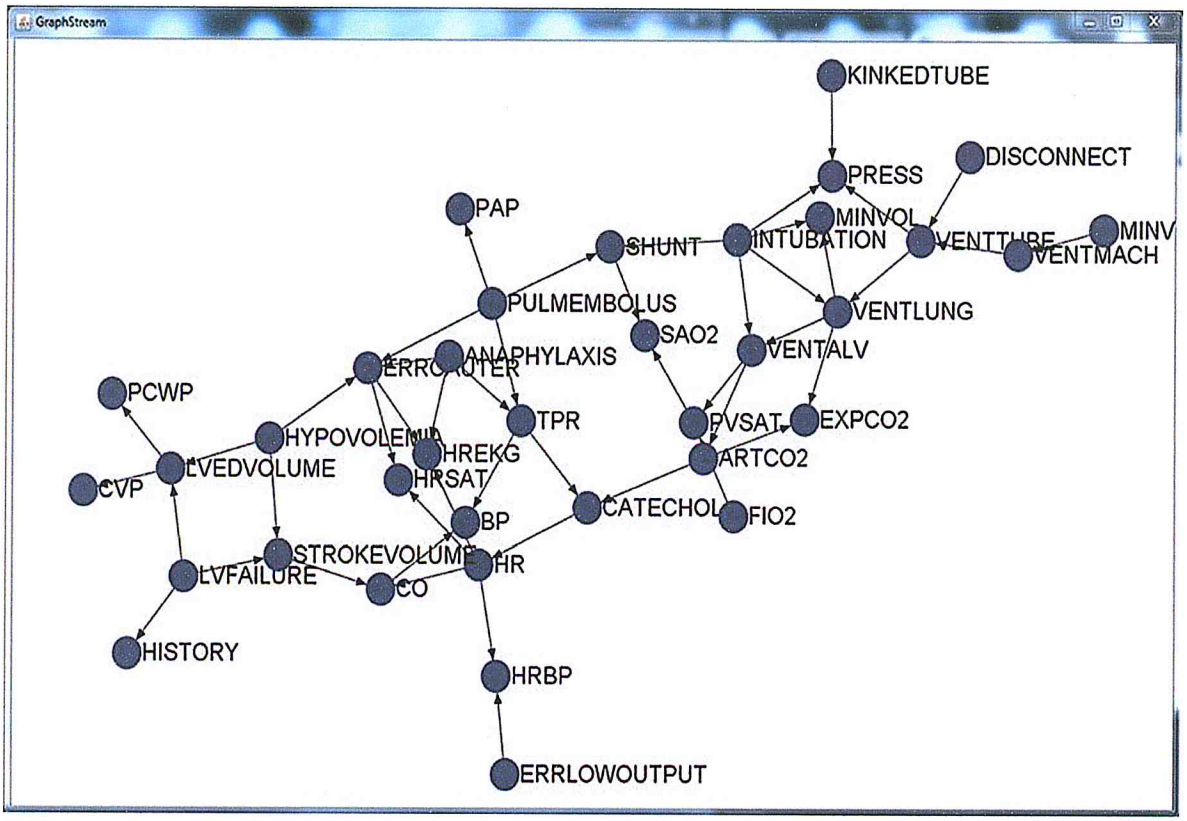


Figure 4.4- Interface Réseau Bayésien Structure -ALARM-

- Le bouton *Paramètre bayésien* permit de trouver les valeursde paramètre d'un motif demandé par l'utilisateur.

Phase 4 : Dans cette partie tous les motifs fréquents du réseau bayésien et les paramètres demandé par l'utilisateur sont affichés sous la forme de règles d'association.

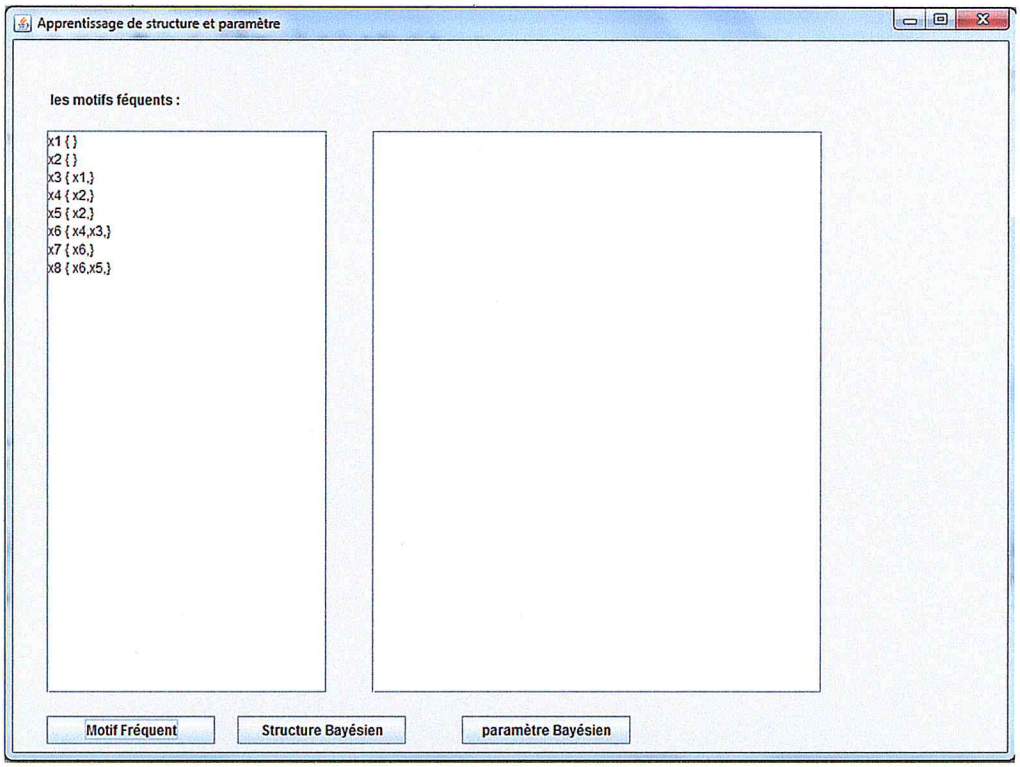


Figure 4.5- Les motifs fréquents base de données ASIA.

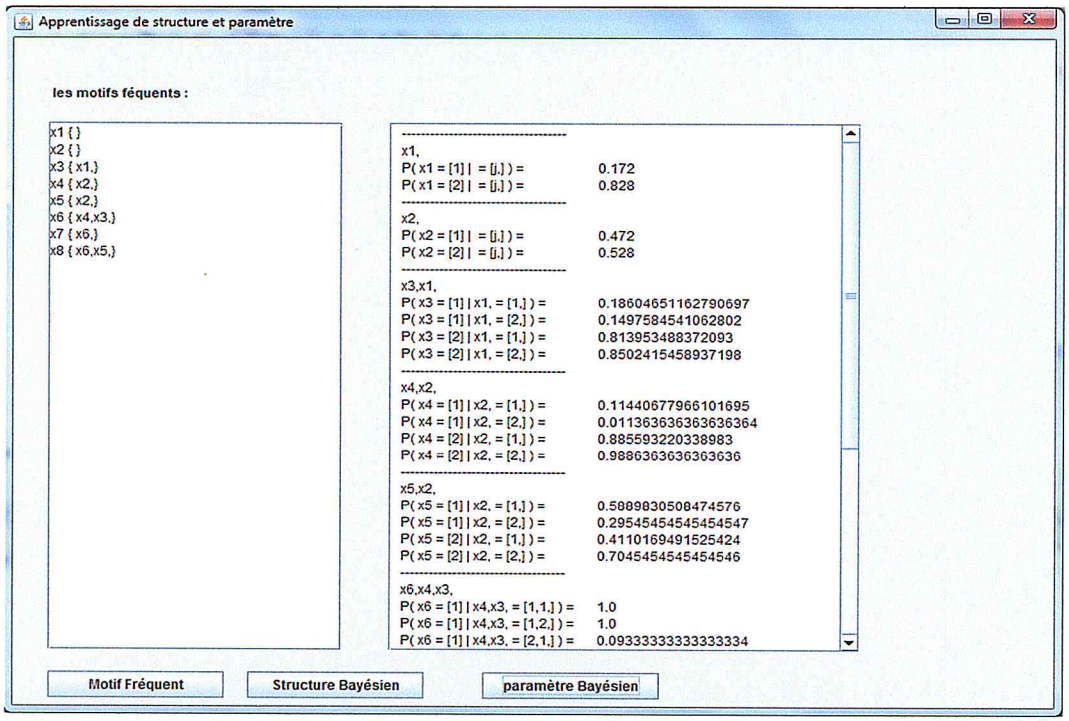


Figure 4.6- Interface Réseau Bayésien paramètre –ASIA-

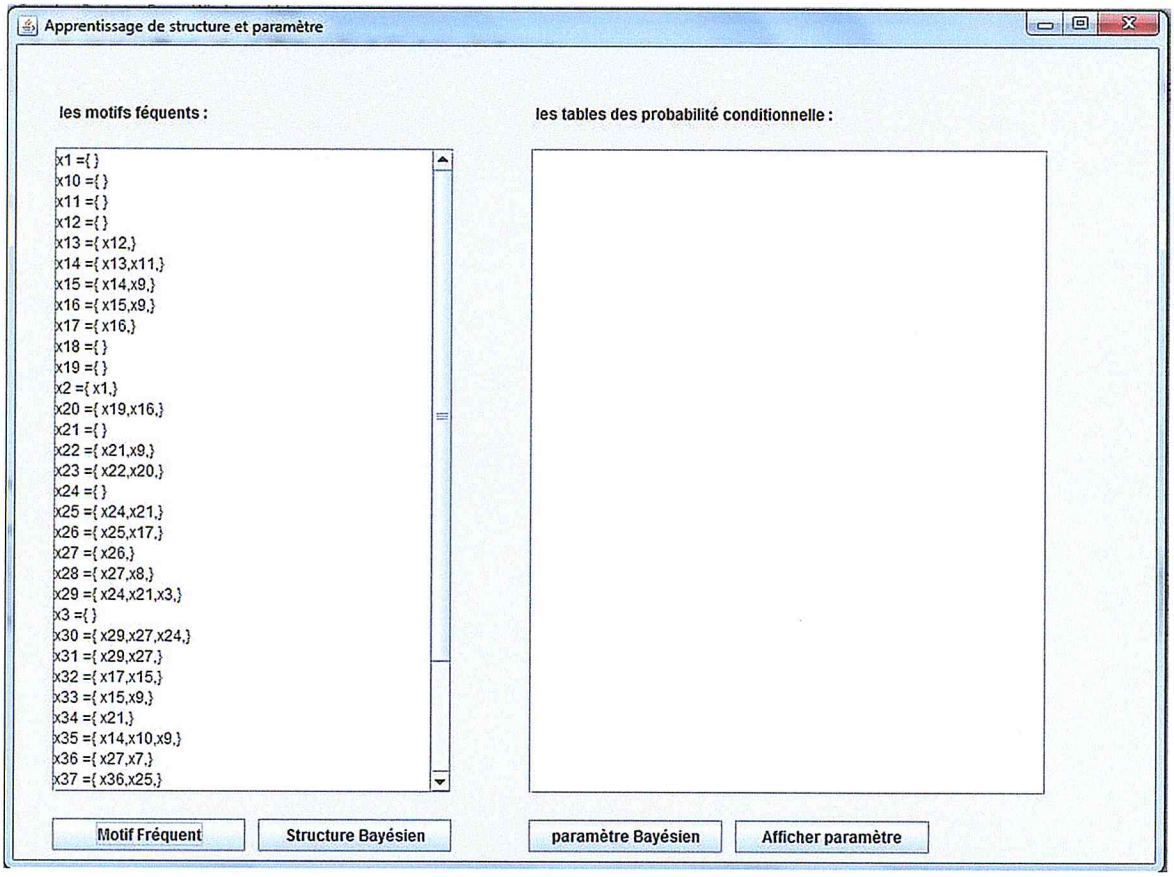


Figure 4.7- Les motifs fréquents base de données ALARM.

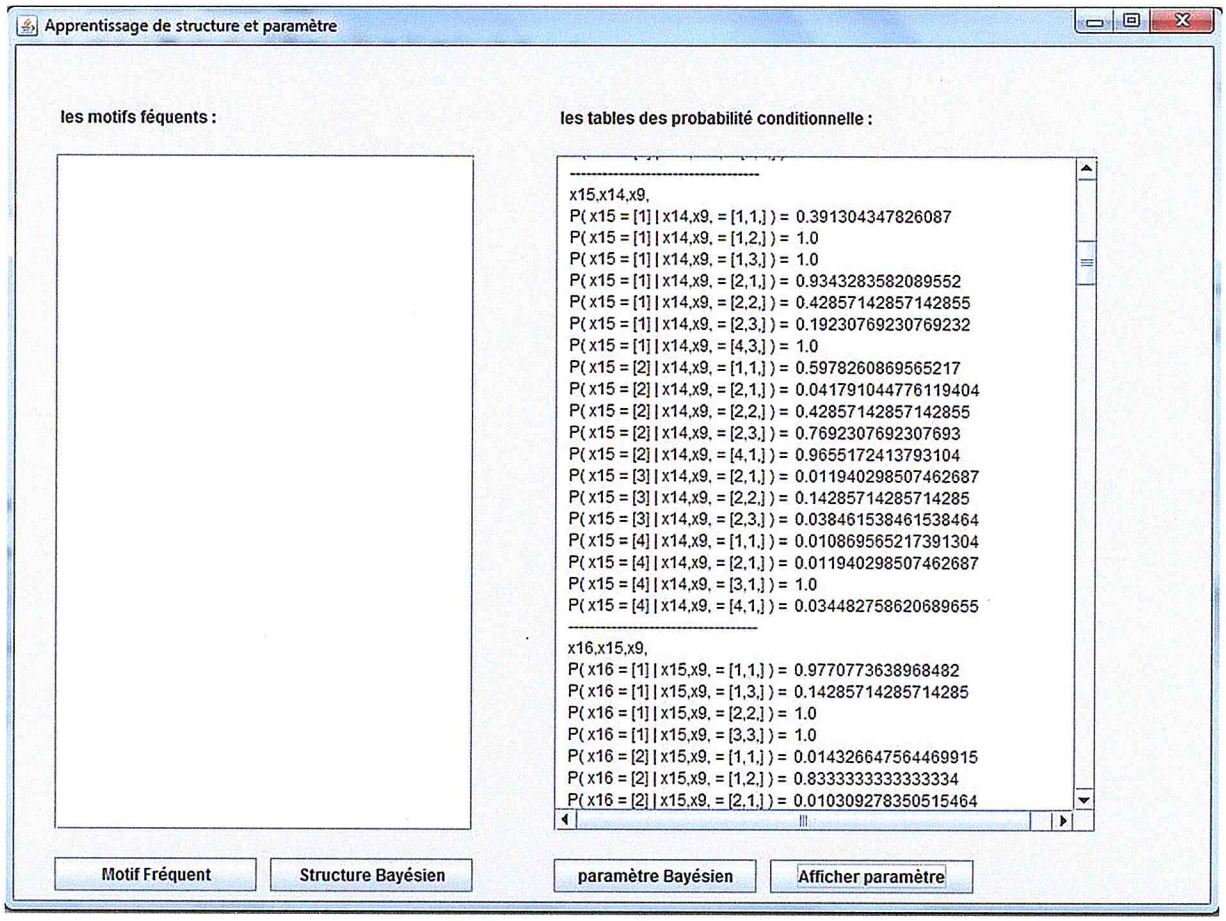


Figure 4.8- Interface Réseau Bayésien paramètre –ALARM-

IV.5 Evaluation des résultats

Cette phase est constituée de l'évaluation, qui mesure l'intérêt des motifs extraits. Nous avons également utilisé les mêmes données **[W06]** pour construire le réseau Bayésien par notre méthode, et nous avons obtenu la même structure que celle obtenue par des experts qui sont utilisé ladataset ASIA et ALARM. Ainsi, nous pouvons confirmer que les structures et les paramètres du réseau Bayésien et les motifs fréquents appris par nos algorithmes sont corrects et juste.

IV.6 Conclusion :

Nous avons présenté dans ce chapitre l'environnement de notre travail et les outils de développement utilisés et la base de données qu'on a utilisé pour valider notre travail,nous avons aussi présenté les différentes interfaces de notre application, puis nous avons présenté notre résultat obtenu.

Conclusion générale

1. Contributions

Le traitement de données de masse est un domaine de recherche en plein essor, 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 de masse, nous avons consacré le chapitre 1 à une étude comparative des versions classiques des algorithmes d'extraction de motifs fréquents. Comme conclusion de cette étude est que ces algorithme sont inadéquats dans le cadre de données de masse et doivent être adaptés.

Les réseaux bayésiens sont devenus très populaires à causes de leurs capacités de modélisation. Ils sont aussi utilisés pour l'extraction de connaissances grâce aux algorithmes d'apprentissage de structure. C'est ce qui nous a encouragés à les utiliser pour l'extraction de motifs fréquents.

L'apprentissage de réseaux Bayésiens à partir de Big data était le cœur de notre travail. Pour cela nous avons adapté un algorithme classique d'apprentissage de RB en utilisant le modèle de programmation Mapreduce tout en s'inspirant d'autres travaux qui existe dans la littérature. Nous avons proposé aussi une adaptation de la méthode classique d'apprentissage de paramètres basée sur Mapreduce.

Le facteur le plus important dans le cadre de Big data est le temps d'exécution. Or nous avons focalisé dans notre validation du travail sur la qualité des résultats obtenus, parce que nous n'avons pas une plateforme matérielle (ensemble de cluster de machines) sur laquelle, on peut faire des tests de performance.

2. Perspectives

Le travail mené dans ce mémoire a permis de pouvoir confronter les caractéristiquesd'une problématique d'intérêt – l'apprentissage des réseaux Bayésiens pour l'extraction de motifs – à celle d'adaptation des méthodes d'extraction de motifs à l'ère de Big Data. Il offre plusieurs perspectives de recherche:

- Faire des tests de performance sur une architecture matérielle réelle (Cloud) contenant des clusters de plusieurs machines.
- L'adaptation des algorithmes d'apprentissage des réseaux bayésiens à partir des données incomplètes avec des variables latentes à l'utilisation dans le cadre de données de masse incomplète, ce qui est très intéressant pour l'extraction de motifs fréquents.
- Pour la plateforme Hadoop, le développement de mécanisme d'enchainement des Maps et des Reduces sera très intéressent.

Referencesbibliographiques

[01] A new learning paradigm: Learning using privileged information Vladimir Vapnik ,AkshayVashistVolume 22 Issue 5-6, July, 2009.

[02] C. Borgelt, "A priori - Association Rule Induction / Frequent Item Set Mining"Intelligent Data Analysis and Graphical Models Research UnitEuropean Centre for Soft Computing c/ Gonzalo Gutiérrez Quiros s/n, 33600 Mieres, Spain

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

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

[05] 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.

[06] R. Srikant. Fast algorithms for mining association rules and sequentialpatterns. PhD thesis, University of Wisconsin, 1996.

[07] H. Mannila, H.Toivonen, and A. I. Verkamo. Efficient algorithms fordiscovering association rules. In AAAI'94 Workshop on Know ledge Discovery in Databases, pages 181-192. AAAI Press, July 1994.

[08] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In Proceedings of the 1993 ACM SIGMOD international conference on Management of Data (SIG-MOD'93), pages 207-216. ACM Press, May 1993.

[09] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In Proceedings of the 1997 ACM SIGMODinternational conference on Management of Data (SIGMOD'97), pages 255-264. ACM Press, May 1997.

[10] R. J. Bayardo. Efficiently mining long patterns from databases. In Proceedings of the 1998 ACM SIGMOD international conference on Management of Data (SIGMOD'98), pages 85-93. ACM Press, June 1998.

[11] R. Srikant et R. Agrawal : Mining sequential patterns : Generalizations and performance improvements. In Peter Apers, MokraneBouzeghoub et Georges Gardarin, éditeurs : Advances in Database Technology - EDBT'96, 5th International

Conference on Extending Database Technology, Avignon, France, March 25-29, 1996, Proceedings, volume 1057 de Lecture Notes in Computer Science, pages 3–17. Springer, 1996.

[12] Mohammed JaveedZaki et Ching-JiuHsiao : Charm : An efficient algorithm for closed itemset mining. In Robert L. Grossman, Jiawei Han, Vipin Kumar, HeikkiMannilaet Rajeev Motwani, éditeurs : SDM. SIAM, 2002.

[14] FlorentMasseglia, FabienneCathala et Pascal Poncelet : The psp approach for mining sequential patterns. In Jan M. Zytkow et Mohamed Quafafou, éditeurs : Principles of Data Mining and Knowledge Discovery, Second European Symposium, PKDD '98, Nantes, France, September 23-26, 1998, Proceedings, volume 1510 de Lecture Notes in Computer Science, pages 176–184. Springer, 1998.

[15] Jay Ayres, Jason Flannick, Johannes Gehrke et TomiYiu :Sequential pattern mining using a bitmap representation. In KDD, pages 429–435. 2002.

[16] Padmanabhan, B. et A. Tuzhilin (1998). A belief-driven method for discovering unexpected patterns. In Proceedings of the 1998 KDD International Conference on Knowledge Discovery and Data Mining, pp. 94–100.

[17] Padmanabhan, B. et A. Tuzhilin (2000). Small is beautiful : discovering the minimal set of unexpected patterns. In Proceedings of the 2000 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, pp. 54–63. ACM Press.

[18] Jaroszewicz, S. et D. A. Simovici (2004). Interestingness of frequent itemsets using Bayesian networks as background knowledge. In Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, pp. 178–186. ACM Press.

[19] Jaroszewicz, S. et T. Scheffer (2005). Fast discovery of unexpected patterns in data, relative to a bayesian network. In Proceedings of the 2005 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA. ACM Press.

[20] Agrawal, R., H. Mannila, R. Srikant, H. Toivonen, et A. I. Verkamo (1996). Fast discovery of association rules, Chapter 12, pp. 307–328. Menlo Park, CA, USA : American Association for Artificial Intelligence.

[21] SzymonJaroszewicz and Dan A.Simovici. Interestingness of frequent itemsets using Bayesian networks as background knowledge. In Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 178-186, New York, NY, USA, 2004. ACM Press.

[22] ZAABOT Zohra, les réseaux bayésiens. Application en reconnaissance de formes à partir d'informations complètes ou incomplètes. Mémoire MAGISTER en automatique. 2012Université MOULOUD MAMMERI, TIZI-OUZOU.

[23] Uffe, B., Kjarulff, A., Madsen, L., "Bayesian Networks and Influence Diagrams",Springer, Science+Business Media, New York (USA),(2008).

[24] ZAHRA FATMA ZOHRA -UTILISATION DES RESEAUX BAYESIENS POUR CALCULER LA FIABILITE DES SYSTEMES- Mémoire de magister Université saaddahleb de blida 2012

[25] Delaplace, A., "Approche évolutionnaire de l'apprentissage de structure pour les réseaux Bayésiens", Thèse de doctorat de l'université FRANÇOIS RABELAIS TOURS, (décembre 2007).

[26] François O., "De l'identification de structure de réseaux Bayésiens à la reconnaissance de formes à partir d'informations complètes ou incomplètes", Thèse de doctorat de l'Institut National des Sciences Appliquées de Rouen, (novembre 2006).

[27] Margaritis, D., "Distribution-free learning of bayesian network structure in continuous domains", In AAAI, (2005), 825–830.

[28] Christian P. Robert, George Casella, Méthodes de Monte-Carlo avec R Collection Pratique R 2011, pp 99-139

[29] IoannisTsamardinos, Laura E. Brown, and Constantin F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. Machine Learning, 65(1) :31–78, 2006.

[30] Wai Lam and Fahiem Bacchus. Learning bayesian belief networks : An approach based on the MDL principle. Computational Intelligence, 10(4) :269–293, 1994.

[31] Cooper, G., Herskovits, E.: A Bayesian method for the induction of probabilistic networks from data. Machine Learning. Machine Learning 9(4), 309–347 (1992)

[32] Pearl, J.: Probabilistic reasoning in intelligent systems: network of plausible inference. Morgan Kaufmann, San Mates (1988)

[33] Deshpande, A., Sarawagi, S.: Probabilistic graphical models and their role in database. In: Koch, C., Gehrke, J., Garofalakis, M.N., et al. (eds.) VLDB 2007, pp. 1435–1436. ACM (2007)

[34] J. Dean and S. Ghemawat. Mapreduce: Simplied data processing on large clusters. Communications of the ACM, 51(1):107-113, 2008.

[35] A Mapreduce-based method for learning Bayesien network from Massive Data. Qiyu Fang, Kun Yue, Xiaodong Fu3, Hong Wu, and Weiyi Liu, Y. Ishikawa et al. (Eds.): APWeb 2013, LNCS 7808, pp. 697–708, 2013.

[36] HAMIDAT FELLA et BOUZOUIDJAYASMINE -réaliser un système pour l'orientation automatique des patients en utilisant les réseaux bayésiens pour l'extraction des connaissances médicales à partir d'une grande masse de données – pour l'obtention dudiplôme d'ingéniorat en informatique de l'université SAAD DAHLEB de Blida 2011

Références webographiques

[W01] http://fr.wikipedia.org/wiki/Eclipse_(projet)Consulter le juillet 2013.

[W02] http://www.eclipse.org/downloads/packages/release/europa/winter.Consulter le juillet 2013.

[W03] http://fr.wikipedia.org/wiki/HadoopConsulter le juillet 2013.

[W04] http://fr.wikipedia.org/wiki/CygwinConsulter le juillet 2013.

[W05] http://www.cygwin.com/Consulter le 26 juillet 2013.

[W06] http://archive.ics.uci.edu/ml/datasets/FlagsMachine Learning Repository Septembre 2013