REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SIENTIFIQUE

UNIVERSITE SAAD DAHLEB DE BLIDA
FACULTE DES SCIENCES
DEPARTEMENT D'IFORMATIQUE

MEMOIRE DE FIN D'ETUDES

Pour l'obtention

D'un diplôme de master en informatique.

Option : Génie logiciel

THEME :

Extraction de motifs séquentiels fréquents à partir des données incertaines de masse

---

Réalisé par :
Hadj Ali Hiba
YKHLEF Rabia

Promotrice et encadreur :
Mme. Zahra Fatma Zohra

REMERCIEMENTS

Tout d'abord, je tiens à remercier tous les membres du jury d'avoir accepté de juger mon travail de thèse.

Avec l'aide de dieu est achevé le présent travail, cependant nous tenons à exprimer nos sincères remerciements à certaines personnes notamment Mr Hocine et Mr Rabia de IconSoftWare dont les conseils et encouragements ont été précieux durant la réalisation de ce travail.

Nous remercions très vivement M. Zahra pour nous avoir encadrés, ainsi pour sa disponibilité et ses remarques constructives qui nous ont été très utiles tout au long de notre projet.

Nous n'oublions pas non plus nos enseignants, tout au long du cycle d'études à l'université du Saad Dahleb De Blida dont les enseignements ont exercé sur nous une influence certaine.

Un grand merci à vous tous, nous tenons à remercier notre famille pour son soutien, son encouragement et sa patience pendant tous ces longs mois.

Un nouveau chapitre de notre vie commence et nous terminons en remerciant tous ceux qui, d'une façon ou d'une autre sont déjà là pour nous aider à l'écrire en nous accordant toute leur confiance, leur amitié et leur soutien.

Et Finalement, DIEU ... Merci

DÉDICACE

Toutes les lettres ne sauraient trouver les mots qu'il faut...

Tous les mots ne sauraient exprimer la gratitude, le respect, la reconnaissance...

Malgré les obstacles qui s'opposaient

En dépit des difficultés qui s'interposaient

À cœur vaillant rien d'impossible

À conscience tranquille tout est accessible

Nous prions dieu, souhaitant que le fruit de nos efforts fournis

Jour et nuit, nous mènera vers la réussite

Aussi, c'est tout simplement que je dédie ce mémoire :
À ma famille : mes chères parents, Fatiha et Ahmed
À mon mari Yazid et ma belle-mère Nfissa,
À mes frères et mes sœurs,
Je vous suis très reconnaissante, pour votre soutien, et aide précieuse.
À mes professeurs,
À ma binôme Rabia, À mes amis, Au IconSoftwar, en souvenir de notre Sincère et profonde amitié et des moments agréables que nous avons passés ensemble.

Hadj Ali Hiba.

DÉDICACE

Grace à dieu, nous avons pu achever ce modeste mémoire que nous tenons à dédier :

Nos très chers parents qui ont été d'un grand soutien moral durant toutes ces années de connaissance et de savoir, et nous ont donné la force et volonté de poursuivre les études.

A nos très chers frères et sœurs, à nos cousins et cousines.

A nos amis qui nous ont soutenus dans les moments les plus difficiles.

A tous j'espère qu'en consultant ce modeste mémoire vous trouver un sentiment de joie et que ceci puisse être un témoignage de reconnaissance.

A tous ma famille YKHLEF.

A ceux et celles qui nous somme plus aimes et plus chères.

RÉSUMÉ

Les bases de données de séquences incertaines sont largement utilisées pour modéliser des données horodatées inexactes ou imprécises dans de nombreuses applications réelles, où les algorithmes de l'extraction de motifs séquentiels traditionnels sont inapplicables en raison de l'incertitude des données et d'évolutivité.

Notre travail consiste à développer un algorithme d'extraction de motifs séquentiels fréquents à partir de données incertaines de masse. En effet, inspiré par le célèbre algorithme d'une méta-heuristique très connue, nous proposons un algorithme basé sur la méthode d'optimisation par essaims des abeilles (Bees Swarm Optimization **BSO**). En se basant aussi, sur le modèle de programmation MapReduce pour, notre algorithme procède d'une façon distribuée et parallèle pour qu'il soit adapté au contexte de Big Data.

Nous avons mené plusieurs expérimentations sur des ensembles de données incertaines synthétiques, et les résultats expérimentaux montrent que notre algorithme est efficace et évolutive.

Mot clé: motifs séquentiels, données incertaines, Big data, BSO, hadoop, MapReduce.

ABSTRACT

In this thesis, we propose an extraction algorithm of sequential patterns from uncertain data mass.

Uncertain sequence databases are widely used to model inaccurate stamped data or inaccurate in many real applications, where algorithms for extracting traditional sequential patterns are inapplicable because of data uncertainty and scalability.

We are interested in our work with big data techniques to treat the mass of data and improve the execution time, we used a well-known meta-heuristic algorithm based on the method Bees Swarm Optimization (BSO) with MapReduce programming model to perform the extraction algorithm of frequent sequences in parallel uncertain.

We conduct extensive experience in all of synthetic uncertain data, and experimental results show that our algorithm is efficient and scalable.

Keyword: sequential pattern, Data uncertain, big data, met heuristic, hadoop, MapReduce.

ملخص

في هذه الأطروحة، نقترح خوارزمية استخراج نماذج متسلسلة من كتلة البيانات غير مؤكدة

وتستخدم على نطاق واسع قواعد البيانات تسلسل غير مؤكدة إلى نموذج البيانات مختومة غير دقيقة أو غير دقيقة في العديد من التطبيقات الحقيقية، حيث خوارزميات لاستخراج نماذج متسلسلة التقليدية غير قابلة للتطبيق بسبب عدم اليقين البيانات وتطويره

ونحن مهتمون في عملنا مع تقنيات البيانات الكبيرة لعلاج كتلة من البيانات وتحسين وقت التنفيذ، استخدمنا خوارزمية ما مع مابريديوس نموذج (BSO) وراء الكشف عن مجريات الأمور المعروفة على أساس النحل طريقة سرب الأمثل البرمجة لأداء خوارزمية استخراج تسلسل متكررة في غير مؤكد متواز

نحن إجراء خبرة واسعة في كل من قدم بيانات غير مؤكد الاصطناعية، وتظهر النتائج التجريبية أن لدينا خوارزمية فعالة وقابلة للتطوير

TABLE DES MATIÈRES

TABLE DES MATIÈRES

TABLE DES MATIÈRES

TABLE DES MATIÈRES

Chapitre 3 : Méta-heuristique

Chapitre 4 : Solution proposé

Chapitre 5 : Implémentation et évaluation

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION GÉNÉRALE

INTRODUCTION GÉNÉRALE

INTRODUCTION GÉNÉRALE

1. Contexte générale

La fouille de donnée est un domaine de recherche en plein essor visant à exploiter les grandes quantités de donnée collectées chaque jour dans divers domaines d'application de l'informatique. Ce domaine pluridisciplinaire se situe au confluent de l'intelligence artificielle, des statistiques et des bases de données. On lui donne d'autres appellations, comme par exemple extraction des connaissances dans les données, traitement de motifs de données ou encore exploration de données.

L'idée sous-jacente de la fouille de données est donc d'extraire les connaissances cachées à partir d'un tas de données disponible. Diverses formes de connaissances peuvent êtres apprises à partir de données : elles peuvent être sous forme de règles, de modèles, de régularités, de concepts etc....

Pour faire face à l'explosion du volume des données, un nouveau domaine technologique a vu le jour : le Big Data. Inventées par les géants du web, ces solutions sont dessinées pour offrir un accès en temps réel à des bases de données géantes.

Les algorithmes classiques de fouille de données sont devenus indaptés en ère de Big Data à cause du volume très gégantesque de données ainsi que d'autres critères caractérisant les données de masse. Par conséquent, le dévéloppement de nouveaux algorithmes de Data Mining ou bien l'adaptation des enciens algorithimes est devenu obligatoire.

2. Problématique

Lorsque les données sont ordonnées selon une relation d'ordre, par exemple le temps, l'extraction de motifs séquentiels est bien adaptée. En effet, les motifs séquentiels permettent d'établir des corrélations entre les événements au cours du temps.

La plupart des données stockées par les entreprises, les administrations ou autres se prêtent à l'extraction de tels motifs. Cette technique permet, par exemple, d'extraire des motifs sur les habitudes de consommation des clients d'un magasin. En effet, un consommateur peut effectuer plusieurs achats à des dates différentes.

L'extraction de motifs séquentiels peut s'appliquer dans de nombreux contextes autres que l'étude du panier de la ménagère. Ils peuvent être ainsi utilisés pour fouiller le comportements des internautes et permettre ainsi d'améliorer la structure du site concerné. Ils

peuvent s'appliquer également en bioinformatique, en musique, sur du texte (l'orde sera par exemple celui des phrases dans le texte),etc.

Le problème de l'extraction de motifs séquentiels (ESF) classique ou de trouver des séquences fréquentes d'événements dans les données avec une composante temporelle, les données devant être minées et déterministes, mais il est reconnu que les données obtenues à partir d'un large éventail de sources de données sont essentiellement incertaines.

Ce mémoire traite de l'ESF dans le cadre des ensembles de données incertaines, plus précisément applés probabilistes. Plus particulièrement, nous étudions le problème de l'extraction des motifs séquentiels a partir de masse de données incertaines.

Le terme « données massives » ou « Big Data » désigne des ensembles de données tellement volumineux qu'il devient difficile de les gérer et traiter avec les outils classiques de gestion de base de données et de traitement de données.

L'extraction de motifs séquentiels a partir de masse de données incertaines pose plusieurs problèmes à savoir:

- La prise en charge de l'incertitude des données dans la procedure de l'extraction des motifs séquentiels.
- Parmi les caractéristiques de big data, il y a le volume, la varéité et la véracité. Le volume et l'espace de recherche des séquences qui est gigantesque implique un temps d'exécution très important.
- Le nombre importants des séquences. Le problème classique en data mining aussi touche le volume des solutions de sorties qui pourra être plus grands que les données d'entrées donc quelle sont les solutions les meilleures et significatives…

3. Objectif

L'objectif de l'extraction de motifs séquentiels fréquents est de trouver des séquences qui apparaissent fréquemment dans un ensemble de données. Or, les algorithmes classiques d'extraction de motifs séquentiels fréquents sont devenus inadéquat dans l'ère de Big Data.

Ainsi, ce travail consiste à proposer une méthode d'extraction des séquences fréquentes à partir des données incertaines, dans un contexte de Big Data.

4. Organisation du mémoire

Dans ce mémoire nous avons fait des études bibliographiques concernant l'extraction des motifs séquentiels fréquents a partir des données certaines et incertaines dans le contexte du big data, aussi sur les big data et les méta heuristique.Ce mémoire est structuré comme suit :

D'abord nous avons rédigé le chapitre 1 ***Big data***, ce chapitre est consacré aux diférents dimension, caractéristiques et techologies de big data, nous allons étudier le modèle MapReduce : un modèle de programmation mis au point par Google à travers l'étude de son principe de fonctionnement, ses caractéristiques et la concrétisation de son implémentation dans le framework Hadoop.

Après nous avons crée le chapitre 2 ***Etat de l'art*** qui contient une étude bibliographique, il s'agit de faire des études sur :

- Les différentes notions du processus de l'extraction des connaissances à partir des données, sur les types des motifs, les motifs fréquents ou non .
- Les algorithmes classiques d'extraction de séquences fréquentes à partir des données certaines.
- Les algorithmes classiques d'extraction de séquences fréquentes à partir des données incertaines.
- Les algorithmes d'extraction des séquences fréquentes à partir des données incertaines, continues dans le contexte de Big data.

Ensuite nous avons constituer le 3ième chapitre ***Méta-heuristique***, dans ce chapitre on va aborder l'ensemble des méthodes utilisées en recherche opérationnelle et en intelligence artificielle pour résoudre des problèmes d'optimisation réputés difficiles, sur l'algorithme génétique et BSO.

Ultérieurement nous avons concevoir le 4ième chapitre ***Solution proposé***, dans ce chapitre nous détaillerons notre conception d'un algorithme BSO-UFSPM pour l'extraction des motifs séquentiels à partir des données incertaines parallèle basée sur le paradigme MapReduce en utilisant le framework Hadoop.

Enfin, Le dernier chapitre, chapitre 5 ***test et validation*** sera réservé pour l'implémentation et évaluation de notre approche avec le Hadoop.

CHAPITRE 1 :

BIG DATA.

1. Introduction

Si le *Big Data* est devenu un terme à la mode dans les milieux décisionnels, il est parfois difficile de s'accorder sur une définition unique : on parle tantôt de technologies de traitement, tantôt de modèle économique.

Dans ce chapitre nous allons présenter les nouvelles technologies de Big Data, avec ses différentes dimension et mesure qu'on va citer ci-dessous.

2. Big Data

2.1. Définition de Big Data

Les Big data se caractérisent par les hauts volumes, hautes vitesses et hautes variétés des actifs d'information qui utilise des formes innovantes de traitement de l'information pour une meilleure compréhension et de décision. [w1]

Big Data est un volume massif de données à la fois structurées et non structurées qui est si grand qu'il est difficile de traiter en utilisant des techniques de base de données et de logiciel traditionnel. [1]

Les données qui dépassent la capacité de traitement des systèmes de base de données conventionnel sont trop gros, se déplacent trop rapidement ou ne correspondent pas aux structures des architectures de ses bases de données. Pour gagner de la valeur à partir de ces données, vous devez choisir une autre façon de le traiter. [2]

Il existe trois caractéristiques principales de Big Data [3] :

- ✓ les données elle-même,
- ✓ les analyses de données,
- ✓ la présentation des résultats des analyses.

2.2. L'origine des données du Big Data

Les données traitées par le Big Data proviennent notamment [4] :

Du Web : journaux d'accès, réseaux sociaux, e-commerce, indexation, stockage de documents, de photos, de vidéos, linked data, etc. (exemple : Google traitait 24 petaoctets de données par jour avec MapReduce en 2009).

Plus généralement, de l'internet et des objets communicants : réseaux de capteurs, journaux des appels en téléphonie.

Des sciences : génomique, astronomie, physique subatomique, climatologie (exemple : le centre de recherche allemand sur le climat gère une base de données de 60 petaoctets).

Données commerciales : (exemple : historique des transactions dans une chaîne d'hypermarchés).

Données personnelles :(exemple : dossiers médicaux).

Données publiques :(open data).

2.3. Les mesures de Big Data

Les mesures de données sont [5] :

- ✓ 1 Mégaoctet = 10^6 octets
- ✓ 1 Gigaoctet = 10^9 octets
- ✓ 1 Téraoctet = 10^{12} octets
- ✓ 1 Pétaoctet = 10^{15} octets
- ✓ 1 Exaoctet = 10^{18} octets
- ✓ 1 Zettaoctet =10^{21} octets

2.4. Les dimensions de Big Data

De nombreux responsables informatiques et autorités du secteur tendant à définir le Big Data selon quatre grandes caractéristiques.

- **Volume**

Le volume décrit la quantité de données générées par des entreprises ou des personnes. Le Big Data est généralement associé à cette caractéristique. Les entreprises, tous secteurs d'activité confondus, devront trouver des moyens pour gérer le volume de données en constante augmentation qui est créé quotidiennement. [w2]

- **Variété**

Variété fait référence aux nombreuses sources et types de données structurées et non structurées. Nous avons utilisé pour stocker des données provenant de sources telles que des tableaux et des bases de données. Maintenant, les données se présente sous la forme d'e-mails, photos, vidéos, suivi des dispositifs, des fichiers PDF, audio, etc. [w3]

- **Vélocité**

La dimension vélocité correspond à la vitesse à laquelle doivent être collectées, analysées et exploitées les données Big Data. De plus en plus souvent, ces données sont à traiter en temps réel. [w4]

- **Véracité**

Les données doivent être fiables et vérifiables pour pouvoir être prises en compte dans l'analyse. [w5]

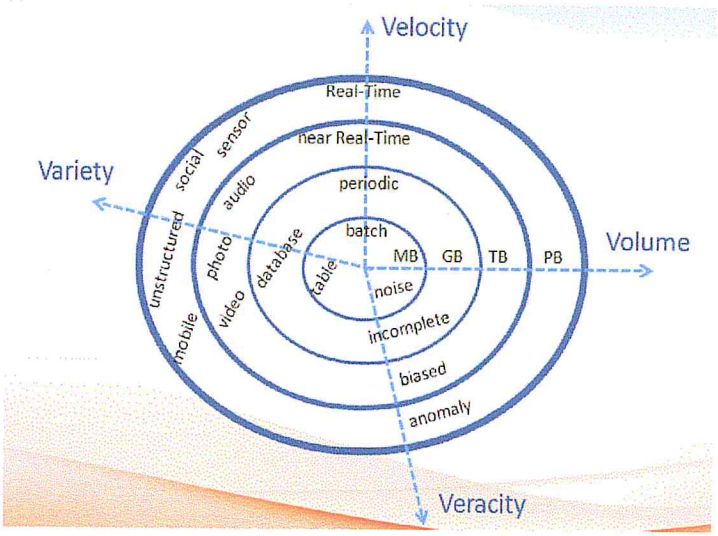


Figure 1.1 : Les dimensions de Big Data 4V. [w6]

3. MapReduce

Pour gérer le Big Data nous proposons l'utilisation de MapReduce qui est le patron d'architecture de développement informatique, inventé par Google, dans lequel sont effectués des calculs parallèles, et souvent distribués, de données potentiellement très volumineuses, typiquement supérieures en taille à 1 téraoctet.

3.1. Définition

MapReduce est un modèle de programmation massivement parallèle adapté au traitement de très grandes quantités de données. MapReduce est un produit de Google Corp. Les programmes adoptant ce modèle sont automatiquement parallélisés et exécutés sur des clusters (grappes) d'ordinateurs. [1]

MapReduce est un framework pour l'analyse de big data, il traite les données non structurées, sans schéma et pour de très grands clusters. Il est fait pour le partitionnement et parallélisassions automatiques. Le modèle MapReduce est entrés dans de nombreuses variations comme Hadoop (Apache), Hadoop++, Amazon MapReduce, etc. [w7]

3.2. Les fonctions de MapReduce [w8]

- **Map (carte)** : la fonction Map serre a :
  - Lire les données d'entrée.
  - Calculer une liste de couples clef/valeur intermédiaires.
  - Fait son traitement et produit une sortie.
  - La phase de Map correspond à notre atelier de transformation.
- **Reduce** : la fonction Reduce permet d'assembler les résultats de la fonction Map, elle serre a :
  - Lire les sorties produites par la phase de map,
  - Fait son traitement et produit les données de sortie
  - Ces valeurs intermédiaires sont regroupées par étiquette et soumises à une fonction d'assemblage appliquée à chaque groupe.
  - Générées par les différentes instances de la fonction Map.
  - Effectue un traitement.
  - La phase reduce à notre atelier d'assemblage.

4. Les technologies de Big Data

Le Big Data comprend les données structurées, semi-structurées et non structurées, tel que :

- Les données structurées : sont les données mises en forme à utiliser dans un système de gestion de base de données [6].
- Les données semi-structurées : ont été traitées dans une certaine mesure [6]. XML ou texte HTML marqués sont des exemples de données semi-structurées [w9].
- Les données non structurées : sont des données dans le format dans lequel elles ont été collectées, pas de formatage est utilisé .Quelques exemples de données non structurées sont formatés pour permettre le stockage, l'utilisation et la production d'informations [6].

Pour analysées les données structurées ou semi-structurées en utilisant un logiciel comme hadoop et les données non structurées peuvent utiliser des logiciels tels que NoSQL, MongoDB et TerraStore.

4.1. Hadoop

4.1.1. Définition

Hadoop développe des logiciels open-source pour fiabilité de l'informatique distribuée. La bibliothèque de logiciels Hadoop est un cadre qui permet de distribuer le traitement des

grands ensembles de donnée à travers des grappes d'ordinateurs à l'aide de modèles de programmation simples. Il est conçu pour évoluer à partir de serveur simples à des milliers de machines, chacune offrant locale calcul et stockage. Plutôt que de compter sur du matériel à fournir la haute disponibilité, la bibliothèque elle-même est conçu pour détecter et gérer les échecs à la couche d'application, de sorte que la prestation d'un service hautement disponible sur un groupe d'ordinateurs, chacun d'eux pouvant être sujettes à échecs. [w10]

4.1.2. L'architecture de Hadoop

hadoop Contient deux serveurs [w11] :

a. Le JobTracker

Le démon JobTracker est votre liaison entre votre application et Hadoop lui-même. Il y a un JobTracker configuré par cluster Hadoop et lorsque vous soumettez votre code à exécuter sur le cluster Hadoop, il est de la responsabilité du JobTracker de construire un plan d'exécution. Ce plan d'exécution consiste à déterminer les nœuds qui contiennent des données pour fonctionner sur, en organisant des nœuds pour correspondre avec des données, le suivi des tâches en cours d'exécution, et la relance de tâches si elles échouent.

b. Le TaskTracker

Similaire à la façon dont le stockage de données suit l'architecture maître / esclave, l'exécution de code suit également l'architecture maître / esclave. Chaque nœud esclave aura un démon TaskTracker qui est responsable de l'exécution des tâches qui lui sont envoyées par le JobTracker et communiquer l'état de la tâche (et un battement de cœur) avec le JobTracker.

4.1.3. Les composants de Hadoop

Contient deux composants principaux qui sont :

- ✓ **Stockage :** le système Hadoop Distributed File (HDFS)
- ✓ **Traitement :** MapReduce

a. Le système Hadoop Distributed File (HDFS)

Définition

HDFS est un Système de fichiers distribué associé à Hadoop. C'est là qu'on stocke les données d'entrée et de sortie, etc. [w12]

Architecteur de HDFS

Contient deux serveurs standards [w11] :

✓ **Le NameNode**

Le NameNode est le maître de la HDFS qui contrôle daemons DataNode esclave ; il comprend où toutes vos données sont stockées, comment les données sont divisées en blocs, quels nœuds ces blocs sont déployés, et la santé globale du système de fichiers distribués. En bref, il est le nœud le plus important dans l'ensemble du cluster Hadoop.

✓ **Le DataNode**

Chaque nœud esclave dans votre cluster Hadoop sera l'hôte d'un DataNode. Le DataNode est responsable de l'exécution de gestion des données : Il lit ses blocs de données à partir du HDFS, gère les données sur chaque nœud physique, et fait rapport à la NameNode avec le statut de gestion des données.

b. MapReduce

Hadoop utilise le MapReduce comme modèle de programmation [w11] :

Définition

C'est un paradigme de programmation fonctionnelle pour analyser un seul enregistrement dans vos HDFS. Il assemble ensuite les résultats dans une solution consommable. Le mappeur est responsable de l'étape de traitement de données, tandis que le réducteur reçoit le signal de sortie des mappeurs et trie les données qu'appliquent à la même clé.

Analyse des données avec MapReduce

Map

Est une étape résout essentiellement un petit problème : le partitionner de Hadoop divise le problème en petits sous-ensembles réalisables et affecte ceux Map processus pour résoudre.

Reduce

Elle combine les résultats des processus de cartographie et constitue la sortie de l'opération de MapReduce.

La figure1.2 suivante présente un schéma général qui résume l'architecture de Hadoop.

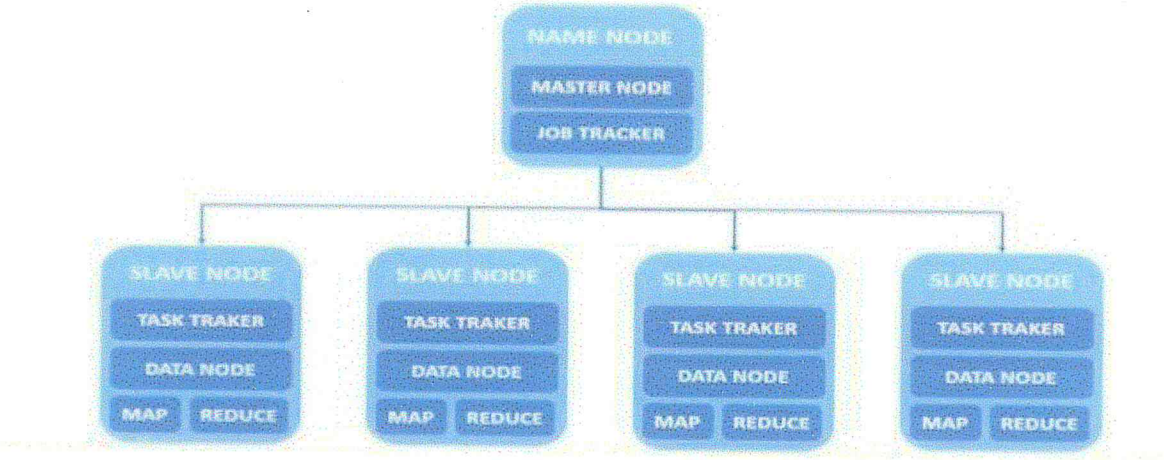


Figure1.2 : schéma général de l'architecture de Hadoop [w11]

4.2. No SQL

4.2.1. Définition

Not only SQL, il se réfère à un groupe éclectique et de plus en plus familière des systèmes de gestion de données non relationnelles ; où les bases de données ne sont pas construits principalement sur les tables, et généralement ne pas utiliser SQL pour la manipulation de données [w13].

4.2.2. Type de base de données NoSQL

Nous classons NoSQL bases de données dans quatre catégories de base, chacune adaptée à différents types de tâches [w14]

- **Clé-valeur**

Les données sont stockées en clé-valeur : une clé plus un BLOB (dans lequel on peut mettre : nombre, date, texte, XML, photo, vidéo, structure objet).

- **Orientées colonne ou colonnes de BigTable**

Ces bases de données se rapprochent des bases de données relationnelles, à ceci près qu'elles permettent de remplir un nombre de colonnes variable.

- **Orientées document**

Ces bases de données stockent des données semi-structurées : le contenu est formaté JSON ou XML, mais la structure n'est pas contrainte.

- **Graphe**

Ces bases de données, basées sur la théorie des graphes, sont gérées par nœuds, relations et propriétés. Elles gèrent des données spatiales, sociales ou financières (dépôts/retraits).

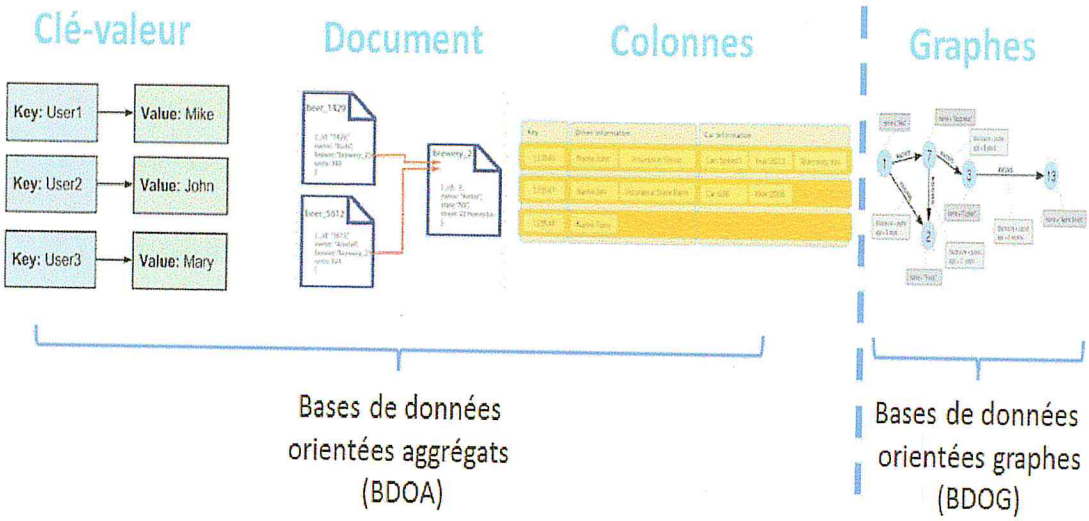


Figure 1.3 : architecture des bases des données NoSQL [w14].

4.3. Spark

4.3.1. Définition

Permettant d'écrire simplement des applications distribuées et proposant des bibliothèques de traitement classique. Entre-temps, avec une performance remarquable, il peut travailler sur des données sur disque ou des données chargées en RAM. Certes, il est plus jeune mais il dispose d'une communauté énorme. C'est aussi un des projets Apache ayant une vitesse de développement rapide. En somme, c'est une solution qui s'avère être le successeur de MapReduce, d'autant qu'il a l'avantage de fusionner une grande partie des outils nécessaires dans un cluster Hadoop. [w15]

4.3.2. Les fonctionnalités de Spark [w16]

✓ Spark supporte également les évaluations paresseuses des requêtes, ce qui aide à l'optimisation des étapes de traitement. Il propose une API de haut-niveau pour une meilleure productivité et un modèle d'architecture cohérent pour les solutions big data.

✓ Les opérateurs réalisent des opérations externes lorsque la donnée ne tient pas en mémoire, ce qui permet de traiter des jeux de données plus volumineux que la mémoire agrégée d'un cluster. Spark essaye de stocker le plus possible en mémoire avant de basculer sur disque. Il est capable de travailler avec une partie des données en mémoire, une autre sur disque.

✓ Il est nécessaire d'examiner ses données et ses cas d'utilisation pour évaluer ses besoins en mémoire car, en fonction du travail fait en mémoire, Spark peut présenter d'importants avantages de performance. Les autres fonctionnalités proposées par Spark comprennent :

CHAPITRE 2 :

ETAT DE L'ART

CHAPITRE 2. EXTRACTION DE MOTIFS SÉQUENTIELS À PARTIR DES DONNÉES INCERTAINES.

1. Introduction

L'extraction de motifs séquentiels est devenue un domaine très actif de la fouille de données [7].

Dans ce chapitre, nous citons d'abord les concepts préliminaires relatifs aux motifs séquentiels, les différentes notions du processus d'extractions des connaissances à partir des données, et les différentes approches pour l'extraction de motifs séquentiels en générale, ensuite j'expliciterai les travaux connexes qui traitent les données incertaines, plus précisément les bases de données probabilistes qui font l'objectif de mon travail.

2. Extraction de connaissance à partir des données

2.1. Le processus ECD

L'extraction de connaissances dans les bases de données (ECBD) est le processus non trivial d'extraction d'informations valides, implicites, potentiellement utiles et ultimement compréhensibles à partir de grandes bases de données [8].

L'Extraction de Connaissances à partir des Données (ECD) est un processus itératif et interactif d'analyse d'un grand ensemble de données brutes afin d'extraire des connaissances exploitables par un utilisateur analyste qui y joue un rôle central. [9]

Ce processus est divisé en trois étapes, la première étape est le prétraitement, cette étape prépare les données cibles. Elle contient plusieurs opérations, le nettoyage des données, l'intégration, la sélection et la transformation des données. L'étape principale du processus d'ECD est l'étape de fouille de données dont des différents algorithmes peuvent être appliqués afin de découvrir des connaissances. A la fin, l'étape de post-traitement évalue les résultats de l'extraction par rapport aux choix de l'utilisateur et aux connaissances du domaine. L'étape de data mining (l'étape principale) est nécessaire dans notre projet, elle est la phase la plus importante (fouille de données).

La Figure 2.1 récapitule ces différentes phases ainsi que les enchaînements possibles entre ces phases. Cette séparation est théorique car en pratique, ce n'est pas toujours le cas. En effet, dans de nombreux systèmes, certaines de ces étapes sont fusionnées [10].

CHAPITRE 2. EXTRACTION DE MOTIFS SÉQUENTIELS À PARTIR DES DONNÉES INCERTAINES.

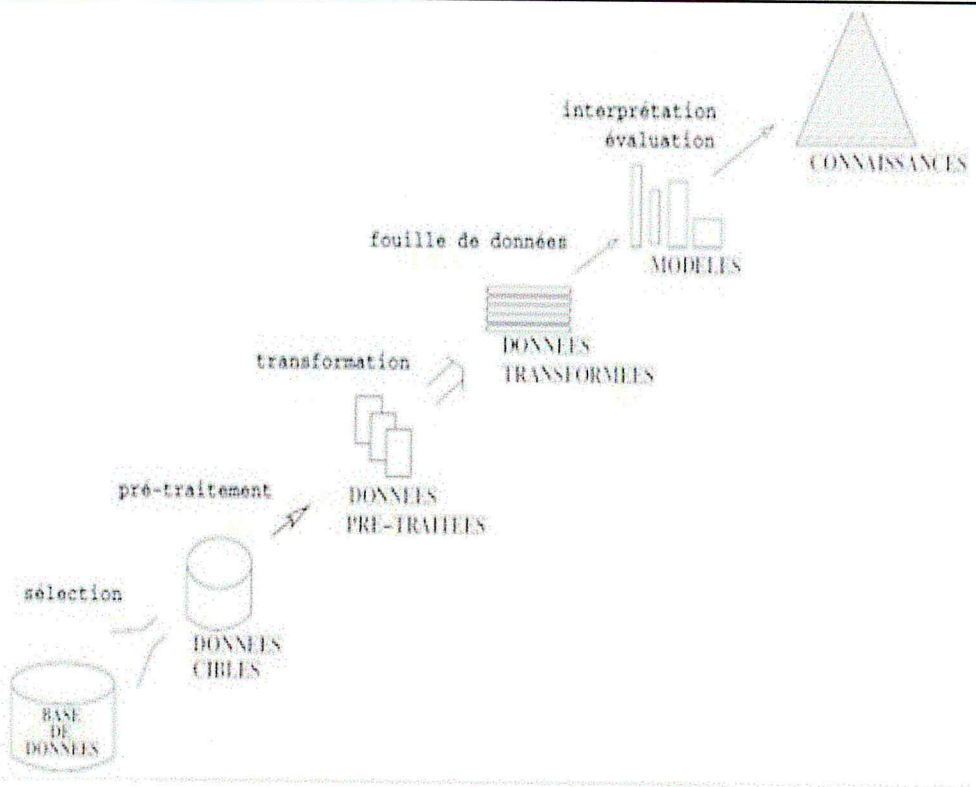


Figure 2.1 : Les différentes étapes de processus d'ECD [w17].

2.2 L'étape de fouille de données

Le datamining signifie littéralement « fouille de données » ou « forage de données». Ce procédé, basé sur une série d'algorithmes ou modèles de datamining permet d'extraire des informations à partir de données, informations qui, grâce à l'analyse, se convertissent en connaissances. [11]

Le datamining, ou fouille de données, est l'ensemble des méthodes et techniques destinées à l'exploration et l'analyse de bases de données informatiques (souvent grandes), de façon automatique ou semi-automatique, en vue de détecter dans ces données des règles, des associations, des tendances inconnues ou cachées, des structures particulières restituant l'essentiel de l'information utile tout en réduisant la quantité de données. [12]

Fouille de données est le processus non trivial d'extraction des connaissances implicites, précédemment inconnues et potentiellement utiles à partir de données. [13]

En bref, le datamining est l'art d'extraire des informations (ou même des connaissances) à partir des données. [14]

- **Méthode de fouille de données** [15]

Les méthodes d'exploration des données fournissent à l'expert des solutions pour l'aide à la décision.

On distingue généralement deux types de données : supervisées et non supervisées :

✓ **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. Donc l'objectif est d'extraire et exploiter l'information présente dans un jeu de donnée.

Dans ce cas l'expert est employé pour étiqueter correctement des exemples. L'apprenant doit alors trouver ou approximer la fonction qui permet d'affecter la bomme étiquette à ces exemples.

✓ **Apprentissage non supervisées**

Aucune classe n'est attribuée a priori. Suivant le type des données, les décisions concernent : la classification supervisée : elle consiste à proposer une valeur de classe pour un objet dont la classe est inconnue. Un médecin peut ainsi adapter le traitement d'un patient en fonction de ses attributs ; la classification non supervisée (ou clustering) : cette méthode permet de constituer des groupes homogènes d'objets, pour par exemple grouper des patients qui ont le même comportement ; le calcul d'associations : elles formulent les corrélations présentes dans les données et sont utilisées à des fins de classification ou de caractérisation de classe.

L'Objectif est d'extraire et exploiter l'information présente dans un jeu de donnée. Dans ce cas l'expert n'existe pas. L'algorithme doit découvrir par lui-même la structure des données.

- Clustering (K-means, CAH).
- Caractérisation, discrimination.
- Extraction de règles d'association.
- Analyse de cas extrêmes.
- Extraction de séquence.

3. Les règles d'association

- **Définition**

CHAPITRE 2. EXTRACTION DE MOTIFS SÉQUENTIELS À PARTIR DES DONNÉES INCERTAINES.

Les règles d'association sont des déclarations qui aident les relations entre les données et de découvrir des indépendants dans une base de données relationnelle ou autre information référentiel.

Une règle d'association comporte deux parties, un antécédent (si) et conséquents (alors).

Un antécédent est un élément trouvé dans les données. Une conséquence est un élément qui se trouve en combinaison avec l'antécédent.

- **Extraction de règles d'association** [16]

Introduit par Agrawal et al. L'extraction de règles d'association est l'un des principaux problèmes du KDD. Ce problème fut développé pour l'analyse de bases de données de transactions de ventes, chacune constituée d'une liste d'articles achetés, afin d'identifier les groupes d'articles achetés le plus fréquemment ensemble. Une règle d'association sera par exemple : « Les clients qui achètent de la bière ont tendance à acheter des cacahuètes ».

Une telle règle d'association permettra de définir une stratégie commerciale dans le but d'augmenter les ventes, en plaçant par exemple la bière et les cacahuètes dans le même rayonnage du magasin.

Afin de ne générer que les relations significatives entre les ensembles d'articles, des mesures d'utilité (le support) et de précision (la confiance), empruntées aux statistiques, sont associées à chaque règle d'association. Les règles générées sont celles dont le support et la confiance sont supérieurs ou égaux à des seuils minimaux définis par l'utilisateur en fonction de ses objectifs et du type de données traitées.

3.1 Extraction de motifs

3.1.1 Motif

- **Définition**

Un motif est une expression dans un langage qui décrit des relations dans un sous ensemble de données avec une certaine certitude. [17]

Un motif est un ensemble de propriétés ou attributs tandis qu'une association est de la forme A →B où A et B sont des motifs. [18]

Les motifs sont des items sets, des sous-séquences, des sous arbres ou des sous-graphes apparaissant dans un jeu de données et vérifiant un seuil de support minimum fixé par l'utilisateur.

CHAPITRE 2. EXTRACTION DE MOTIFS SÉQUENTIELS À PARTIR DES DONNÉES INCERTAINES.

- **Type de motif**

On a 4 types de motifs qui sont présenté ci-dessous :

✓ **Type 1 : Les items sets**

L'item set est un ensemble d'un ou plusieurs éléments. [19]

✓ **Type 2 : Les graphes**

Un motif de graphe décrit un ensemble de contraintes à satisfaire par le graphe d'interprétation pour qu'une dépendance soit produite. Formellement, un motif de graphe est constitué d'un ensemble de motifs de nœud et de relation entre ces motifs. [20]

Type 3 : Les arbres

Les arbres attribués sont des arbres dans lesquels les nœuds sont associés à des items sets. Les arbres attribués peuvent être utilisés dans de nombreuses applications de fouilles de données spatio-temporelles. Dans le cas d'études épidémiologiques, par exemple, l'espace géographique peut être découpé en zones qui sont représentées par les nœuds de l'arbre, les items sets décrivent les caractéristiques de ces zones à un temps donné et les arêtes symbolisent des relations de voisinage avec d'autres zones au temps suivant. [21]

✓ **Type 4 : Les séquences**

Les motifs séquentiels permettent d'extraire des motifs qui prennent en compte le temps. [22]

En effet, les motifs séquentiels permettent d'établir des corrélations entre les événements au cours du temps. [8]

L'extraction de motifs séquentiels peut s'appliquer dans de nombreux contextes. Ils peuvent être ainsi utilisés pour fouiller les comportements des internautes et permettre ainsi d'améliorer la structure du site concerné. Ils peuvent s'appliquer également en bio-informatique, en musique, sur du texte (l'ordre sera par exemple celui des phrases dans le texte, etc. [23]

- **Les motifs fréquents et non fréquents**

Les motifs peuvent êtres fréquents ou non fréquents :

✓ **Motif fréquents**

En règle générale, un motif fréquent est une conjonction de littéraux qui couvre au moins σ les instances de la base de données. La valeur σ est appelé minimal seuil de fréquence, elle est donnée par un utilisateur.

C'est un ensemble d'éléments présents dans un nombre « suffisamment grand » de lignes d'une base de données. Le cadre mathématique qui suit est classique, et formalise. [24]

✓ **Motif rare (non fréquents)**

Les motifs rares représentent les motifs qui apparaissent rarement dans un ensemble de données tel que les symptômes non usuels ou les effets indésirables exceptionnels qui peuvent se déclarer chez un patient pour une pathologie ou un traitement donné. Un motif est dit rare ou infréquenté si son support est inférieur ou égal à un support maximal (noté max_sup). [w18]

3.1.2Extraction de motif séquentielle traditionnelle

Le problème de l'extraction de motifs séquentiels a été bien étudié dans la littérature dans le contexte des données déterministes, et de nombreux algorithmes ont été proposés pour résoudre ce problème, y compris GSP, PreFixSpan, SPADE et FreeSpan [25].

- **Algorithme GSP (Generalized Sequential Pattern)**

GSP [26] fournit toutes les séquences fréquentes à partir d'une base de séquence en respectent un support minimal (min-sup). L'algorithme GSP effectue plusieurs passages dans la base de données :

- Le premier passage compte tous les éléments simples de 1-séquence.
- A partir des éléments fréquents sont formées un ensemble de candidat de 2-séquences.
- Un autre passage est fait pour recueillir leur soutien. Les fréquents 2-séquences sont utilisées pour générer des candidats de 3-séquences, et ce processus est répété jusqu'à ce qu'aucunes séquences plus fréquentes ne soient trouvées.

Pseudo code de l'algorithme GSP

F_1 = {Fréquent 1-séquences} ;

Pour (K=2 ; $F_{k-1} \neq \emptyset$; K=K+1) **Faire**

C_k= un ensemble de candidats K-séquences

Pour toutes les entrées-séquencesε dans la base de donnée **Faire**

Tout compte incrément$\alpha \epsilon C_k$contenu dansε

F_k= {$\alpha \epsilon C_k$| α.sup>= min-sup} ;

ensemble de toutes les séquences fréquente = $\cup_k F_k$;

- **Algorithme SPADE** (Sequential Pattern Discovery using Equivalent Classes)

L'algorithme SPADE [27] utilise une représentation verticale d'une base de données pour extraire des motifs séquentiels. Il travaille comme suit :

- Le premier passage : permet d'associer à chaque item une liste d'identifiant (id-séquence, id-transaction). Qui répertorie toutes ses apparition dans la base de données et de calculé son support.
- Les événements non fréquents ne sont associés à aucune liste.
- La génération et d'élagage de candidat : les candidats qui ne sont pas fréquents n'existent pas dans la base de séquence mais on utilise leur représentation verticale.

Pseudo code de l'algorithme SPADE

(**Entrer** : séquences de bases de données S, **Entrer** : Entier min-sup,
Entrée/Sortie : ensemble F)
Pour tout item A_i appartient S **Faire**
Ti ← {}
Pour tout item A_j appartient S, j>= i et toute combinaisonα of A_iA_j
Faire
L (α)= temporel TID liste de rejoindre L (A_i) avec L (A_j)
Si (sup (α)>= min-sup **alors**
Ti ← Ti U { α }
F=F U α
Fin si
Fin pour
SPADE (T, min-sup, F)
Fin pour

- **Algorithme PreFixSpan (Prefix-projected Sequential pattern)**

L'algorithme PreFixSpan [28] permet l'extraction de motif séquentiel à l'aide du paradigme motif growth. Cet algorithme utilise la méthode divisé pour régner. Il passe à plusieurs passages qui sont les suivants :

- Le premier passage concerne la base de données, permet d'extraire l'ensemble des 1-séquences fréquentes.
- Chaque motif séquentiel est considéré comme un préfixe. L'ensemble complet des motifs séquentiels est ainsi partitionné en différents sous-ensembles par rapport à différents préfixes.
- Extraire les sous-ensembles de motifs séquentiels, des bases de données,
- projetées sont construites et fouillées récursivement.

Pseudo code de l'algorithme PreFixSpan

```
(Entrer : Base de donnée Dα,
Entrer : Séquenceα,
Entrer : Entiermin_supp,
Entrée/Sortie : Ensemble F)
F1← {fréquent items dans Dα }
Pour tout item bi appartiens F1
    Faire
        β= (α1 → … → (αn U {bi}))
        γ= ( α1 → … → α → (bi))
        Si supp ( β, Dα)>= min_supp
            Alors
                F ← F U { β }
                D' ← (Dα) |   β
                 prefixspan (D',β, min_supp, F)
        Fin si
        Si supp ( γ , Dα)>= min_supp Alors
                F ← F U { γ }
                D' ← (Dα) | γ
                prefixspan (D',γ, min_supp, F)
        Fin si
Fin pour
```

4. Extraction de motifs séquentiels à partir des données probabilistes

Pour bien comprendre cette grandes partie, il est important d'abord de définir, comprendre et illustrer quelque notion de base par exemple : les bases de données probabilistes, qu'est-ce que les données incertaines, ainsi que les données certaines ?

4.1 Les données incertaines

- **Définition de donnée**

Une donnée est un élément (signe ou un symbole) brut, qui n'a pas encore été interprété, mis en contexte.

CHAPITRE 2. EXTRACTION DE MOTIFS SÉQUENTIELS À PARTIR DES DONNÉES INCERTAINES.

Une donnée est en générale mesurable, une mesure étant caractérisé par le type de données. Soit quantitatif comme poids, âge. En effet c'est des calculs numériques sur ces type ou qualitatif comme date et nom, il ne peut pas effectuer de calculs [w19].

- **Définition des données incertaines**

Une valeur sera déclarée incertaine si la validité de la donnée reste douteuse au stade de validation indiquée dans l'information.

Dans la mesure du possible, la qualification douteuse doit être une étape transitoire de la validation de la donnée et doit être réservé à des avancements intermédiaires de la validation [w20].

- **Exemple sur les données incertaines** [w20].

Dans cet ensemble de données incertaines des dossiers des patients, chaque transaction représente la visite d'un patient à un médecin de bureau. A noter que le patient peut souffrir de plusieurs maladies en même temps (C'est à dire, plusieurs éléments peuvent apparaître ensemble dans la même transaction).Chaque article (Représentant une maladie potentielle) dans la transaction est associé à une existentielle probabilité exprimant la probabilité d'un patient ayant la maladie chez cette visite.

Avec cette notion, chaque élément dans une transaction dans des bases de données traditionnelles. Des données précises peuvent être considérées comme un élément avec une probabilité de 100% d'être présent dans la transaction.

D'autres exemples de données incertaines incluent des ensembles de données d'images satellitaires, où chaque élément d'une opération exprime la probabilité de la présence d'un objet capturé dans une image.

Ce ne sont que quelques exemples des nombreuses situations de la vie réelle dans lequel les données sont incertaines. En d'autres termes, l'espace de recherche pour fréquent extraction de motifs séquentielle à partir des données incertaines peut être beaucoup plus grande que celle d'être des données certaines.

4.2 La base de données probabiliste

CHAPITRE 2. EXTRACTION DE MOTIFS SÉQUENTIELS À PARTIR DES DONNÉES INCERTAINES.

L'incertitude inhérente aux données présentes dans de nombreuses applications telles que les bases de données de capteurs, des annotations de texte, et la recherche d'information a motivé la nécessité de manipuler des données imprécises au niveau de la base de données.

Les exemples incluent les bases de données de capteurs (valeurs mesurées ont des erreurs), annotation texte (annotations sont rarement parfaites), la recherche d'information (la correspondance entre un document et une requête est souvent une question de degré ou de confiance), les données scientifiques (sorties de modèles, les estimations, expérimentale mesures et données hypothétiques), et les données de nettoyage (plusieurs alternatives pour une valeur incorrecte). Bien que les bases de données existantes offrent de grands avantages pour le traitement de ces données, ils ne fournissent pas de support direct à l'incertitude dans les données.

Par conséquent, ces applications sont soit contraintes de gérer l'incertitude à l'extérieur de la base de données, ou contraindre les données sous une forme qui peut être représenté dans le modèle de base de données.

En raison de l'importance de la nécessité de soutenir des données incertaines, plusieurs chercheurs ont abordé ce problème. Un vaste corpus de travail porte sur la modélisation des données incertaines. [w20]

Pour résumé on peut dire que les base des données probabiliste contient des données incertaines et calcule leur probabilité d'incertitude.

5. Le problème de l'extraction de motifs séquentiels fréquents à partir des données incertaines

Le problème de l'Extraction de motifs Séquentiels Fréquents (ESF), est de trouver des séquences fréquentes d'événements dans les données avec une composante temporelle.

Dans l'extraction de motifs séquentiels fréquents classique, la base de données d'événement se compose d'un tuple <eid, e, σ > où :

- ✓ e est un événement.
- ✓ σ est une source.
- ✓ eid est l'identifiant d'un événement qui comprend un horodatage.

CHAPITRE 2. EXTRACTION DE MOTIFS SÉQUENTIELS À PARTIR DES DONNÉES INCERTAINES.

Un tuple peut enregistrer une transaction de vente au détail (des événements) par un client (source), ou d'une observation d'un objet/personne (événement) par un capteur/appareil photo (source).

Depuis l'événement identifiants ont un horodatage, la base de données d'événement peut être considérée comme une collection de séquences de source, l'un par source, contenant une séquence d'événements (par ordre d'horodatage) associé à cette source, et le problème d'ESF classique est de trouver des modèles d'événements qui ont un ordre temporel et qui se produisent dans un nombre important de séquences sources [29].

I. Les niveaux d'incertitude dans ESF

L'incertitude dans ESF peut se produire dans trois niveaux différents : La source, l'événement et l'horodatage peuvent tous être incertaine.

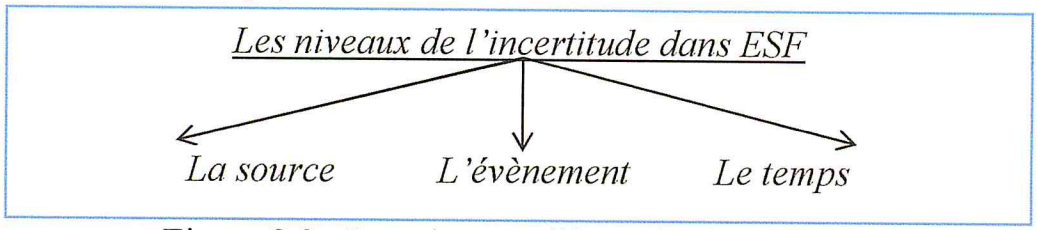


Figure 2.2 : Les niveaux d'incertitude dans ESF.

L'incertitude dans le temps ne semble pas bien adaptée à l'approche de base de données probabiliste, et nous nous concentrons sur l'incertitude (L'une des) source et l'événement.

a. La source

On note l'incertitude ou niveau de source par SLU (*Source-LevelUncertainty*). Dans SLU, la source attribut de chaque tuple est incertaine :

Chaque tuple contient une distribution de probabilité sur les sources possibles (niveau attribut incertitude) [30].

Dans les situations suivantes, un événement est enregistré de façon déterministe, mais la source n'est pas facilement identifiable :

i. Un client (source) achète certains articles (événement) à partir d'un magasin, et fournit des informations d'identité. Cependant, plusieurs correspondances peuvent émerger dans la base de données des clients :
   Les coordonnées du client peuvent être incomplètes ou incorrectes, ou la base de données du client lui-même peut être probabiliste à la suite de «déduplication» ou le nettoyage. [31]

ii. Un véhicule / personne est identifié par un capteur / caméra à l'aide des méthodes telles que la biométrie ou reconnaissance automatique des plaques en anglais Automatic Number Plate Recognition (ANPR), qui sont par nature bruyante.
Pour des motifs miniers tels que "10% des voitures passent par la caméra X, puis la caméra Y et plus tard caméra Z", nous considérons chaque voiture comme source et chaque observation comme un événement.

Dans de tels scénarios, il est certain que l'événement a eu lieu (par exemple un client a acheté certains éléments, un véhicule / personne sont entrés dans une zone) mais la source associée à cet événement est incertaines.

Alors que de grandes bases de données de type (i) sont facilement disponibles dans les organisations commerciales, grandes bases de données de type (ii) sont maintenant de plus en plus communes, un exemple notable étant UK de police la base de données ANPR [31].

b. L'évènement

La notation de l'incertitude au niveau d'évènement est ELU (*Event-LevelUncertainty*), dans ELU la source du tuple est certaine, mais les événements sont incertains.

De tels scénarios peuvent être modélisés comme une séquence de sortie des événements par une source, mais à chaque événement ayant seulement une certaine probabilité d'avoir vraiment eu lieu, comme ci-dessus, en utilisant l'incertitude au niveau de l'attribut.

Par exemple le système PEEX [32] qui agrège observations fiables d'employés utilisant la technologie RFID antennes à des emplacements fixes dans les événements de plus haut niveau d'incertitude tels que "au moment 103, Alice et Bob étaient réunis dans la chambre 435 avec une probabilité de 0,4".

Ici, la source (chambre 435) est fixe, mais l'événement ({Alice, Bob}) est incertain, et ne dispose que de 0,4 probabilité d'avoir eu lieu [29].

II. Calcul des séquences fréquentes

De la sémantique des mondes possibles, nous obtenons les définitions des séquences fréquentes sous deux mesures :

- ✓ Soutien attendu (Expected support)
- ✓ frequentness probabiliste (probabilistic frequentness) [33]

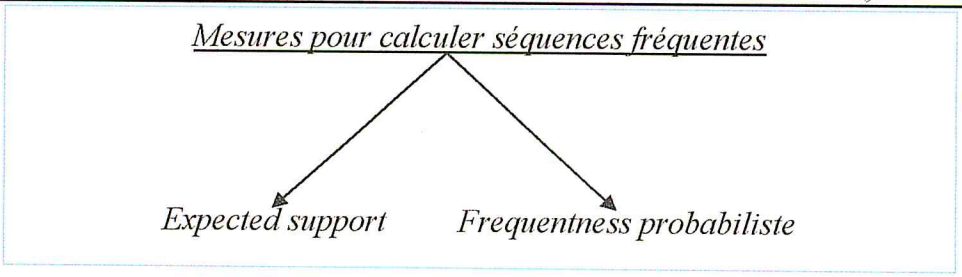


Figure 2.3 : Les mesures de calcul des séquences fréquentes.

III. Notion de base pour calculer Expected support

Laisser I= $\{i_1,i_2, ..., i_q\}$ un ensemble d'éléments (Items) et S=$\{\sigma_1,..., \sigma_m\}$ un ensemble de sources. Un évènement e$\subseteq I$ est une collection d'éléments(Items).

La base de donnée D=<$r_1,r_2,...,r_n$> est une liste ordonnée d'enregistrement tels que chaque $r_i \in$D est de la forme (eid,e_i, σ_i),où eid est un identifiant d'un événement-id unique, y compris un horodatage (les événements sont triés par cette horodatage), e_i est un événement et σ_i est une source.

Une séquence s = <$s_1,s_2, ...,s_a$> est une liste ordonnée des événements. La longueur d'une séquence s est le nombre total d'éléments.

Pour tout entier k, le k-séquence est une séquence de longueur k. Laisser S=<$s_1,s_2, . . .,s_q$> et T=<$t_1,t_2, . . ., t_r$> sont deux séquences.

On dit que S est une sous-séquence de T, noté S <=T, s'il existe des entiers 1<=i_1<i_2<.... <i_q<=rtelle que$s_k \subseteq ti_j$, pour k=1,....q.

Pour une séquences et la source i, laissez X_i (s, D) est une variable d'indicateur, dont la valeur est 1 si s est une sous-séquence de la séquence source pour la source i, et 0 sinon.

Pour toute séquence s, définir son support à D, notée Sup(s, D) = $\sum_{i=1}^{m} X_i$(s, D)tel que i ayant de 1 jusqu'à m.

L'objectif est de trouver de telle sorte que Sup (s, D) $\geq \theta_m$ de toutes les séquences pendant un certain seuil défini par l'utilisateur 0≤θ≤1 [33].

ELU [33] **:**

CHAPITRE 2. EXTRACTION DE MOTIFS SÉQUENTIELS À PARTIR DES DONNÉES INCERTAINES.

Une base de données probabiliste Dp est une collection de p-séquences $Dp_1,\ldots, Dp_m$ où Dp_i est le p-séquence de la source i dans S.

p-séquence est l'analogue naturel de la séquence source dans le SPM classique, et chaque p-séquence est de la forme < $(e_1,c_1)\ldots.$ $(e_k; c_k)$>, Oùe_j est l'événement (par ordree_{id}) et c_j est la confiance qui est réellement survenu.

Dans les exemples, nous écrivons un p-séquence < ({a, d}, 0.4); ({a, b}, 0.2)>comme (a, d: 0.4) (a, b: 0.2).

Un exemple d'une base de données aléatoire au niveau de l'événement se trouve dans le tableau 1 (L).

La sémantique des mondes possibles de Dp_i est comme suit. Pour chaque e_jd'événement dans un p-séquence Dp_i, il y a deux sortes de mondes ; celui dans lequel e_j a lieu et l'autre où il ne fait pas.

Laisser se produit = $\{x_1,\ldots,x_l\}$, où 1 <= x_1<.....<x_l<= K, être les indices d'événements qui se produisent dans Di*.

Alors Di*=<$e_{x1},\ldots, e_{xl}$>, et Pr (Di*)= $\prod_{j \in \text{occur}}$ Cj*$\prod_{j \notin \text{occur}}$ 1 − Cj.

Un ensemble de mondes possibles de Dpi, notée par PW (Dp_i), tel que :

$$\text{PW}(Dp_i) = \text{PW}(Dp_1) * \ldots * \text{PW}(Dp_m).$$

Pour tout D*∈PW(Dp) tel que D*= (D1*,…, Dm*) la probabilité de D* est donnée par

$$\Pr[D^*] = \prod_{i=1}^{m} \Pr(Di*).$$

	P-séquence
Dx_p	(a, d : 0.6) (a, b : 0.3) (b, c : 0.3)
Dy_p	(a, d : 0.4) (a, b : 0.2)
Dz_p	(a : 1.0) (a, b : 0.5) (b, c : 0.3)

(L)

()	(1-0.4) * (1-0.2) = 0.48
{(a, d)}	(0.4) * (1-0.2) = 0.32
{(a, b)}	(1-0.4) * (0.2) = 0.12
{(a, d) (a, b)}	(0.4) * (0.2) = 0.08

(R)

Tableau 2.1 : Une base de données au niveau de l'événement incertain (L), et tous les mondes possibles de Dp_y ainsi que leurs probabilités (R). [33]

PW (Dx_p)	{() = 0.084}{(a, b) = 0.126}{(a, b) = 0.036} {(b, c) = 0.196} {(a, d) (a, b) = 0.054} {(a, d) (b, c) = 0.294} {(a, b) (b, c) = 0.084} {(a, d) (a, b) (b, c) = 0.126}
PW (Dy_p)	{() = 0.48}{(a, d)= 0.32}{(a, b) =0.12} {(a, d) (a, b)= 0.08}
PW (Dz_p)	{(a)= 0.35}{(a) (a, b) = 0.35} {(a) (b, c) = 0.15} {(a) (a, b) (b, c)=0.15}

Dx*	(a, d) (b, c)	0.294
Dy*	(a, d)	0.32
Dz*	(a) (a, b)	0.35

Tableau 2.2 :L'ensemble des mondes possibles pour chaque p-séquence dans (L). [33]

Dans le tableau 2.1 (L). En tout, il y a 8 * 4 * 4 = 128 mondes possibles de Dp, un tel monde possible est représenté (R).

Remarque : dans cette exemple Pr [D*] = 0.29 * 0.32 * 0.35 = 0.03.

SLU :

Une base de données probabiliste Dp est une liste ordonnée d'enregistrement <r1,…, rn> tel que chaque ri est de la forme (e_{id}, e, w), ou eid est l'identifiant de l'évènement, e est l'évènement et w est la probabilité de distribution de S.

La distribution w contient une paire de la forme (σ, c), où $\sigma \in S$ et $0 < c \leq 1$ est la confiance que l'événement e est associée à σ source.

Les mondes possibles sémantique de Dp est la suivante,Un monde possible D* de Dp est généré en prenant chaque e_i de l'événement, à son tour, et en l'attribuant à l'une des sources possibles σi∈Wi.

Ainsi, chaque enregistrement ri = (e_{id},e_i, Wi) ∈Dp prend la forme ri' = (e_{id}, e_i, σi), pour certains σi∈ S dans D*.

En énumérant toutes les combinaisons possibles, nous obtenons l'ensemble des mondes possibles.

Nous supposons que les distributions associées à chaque ri record en Dp sont stochastique indépendante ; la probabilité d'un éventuel monde D * est donc :

$$\Pr[D^*] = \prod_{i=1}^{m} \Pr wi[\sigma i].$$

Un exemple peut être trouvé dans le tableau 2.3.

Eid	évènement	W
e_1	(a, d)	(X : 0.6)(Y : 0.4)
e_2	(a)	(Z : 1.0)
e_3	(a, b)	(X : 0.3)(Y : 0.2)(Z : 0.5)
e_4	(b, c)	(X : 0.7)(Z : 0.3)

(L)

	P-séquence
Dxp	(a, d : 0.6) (a, b : 0.3) (b, c : 0.7)
Dyp	(a, d : 0.4)(a, b : 0.2)
Dzp	(a : 1.0)(b, c : 0.3)

(R)

Tableau 2.3 : la base de données au niveau de la source incertaine (L) transformé en p-séquences (R).

A noter que la représentation de p-séquence est la même que la base de données aléatoire au niveau de l'événement du tableau 1.

Par exemple, un possible monde D* de la base de données du tableau 3 peuvent être générés par attribuer des événementse_1,e_3 et e_4 de X avec des probabilités 0.6, 0.3 et 0.7 respectivement, et e_2 à Z avec une probabilité de 1,0, et Pr [D *] = 0,6 × 1,0 × 0,3 × 0,7 = 0,126.

6. Travaux connexes

Un grand nombre de techniques de base de données et de data mining traditionnels ont été étendues à être appliquée sur des données incertaines.

6.1 Extraction de motifs séquentiels à partir de données incertaine

Muzammal et Raman [34], proposent l'algorithme de SPM dans la base de données probabiliste en utilisant expected support pour mesurer motif frequentness, qui a la faiblesse dans le secteur minier motifs séquentiels de haute qualité.

L'algorithme SPM dans la base de données probabiliste examine le problème de trouver toutes les séquences fréquentes dans les bases de données probabilistes SLU.

L'algorithme SPM utilise la mesure expacted support, qui a l'avantage qu'il peut être calculé de manière efficace pour des bases de données SLU, Probabilistic frequentness est prouvable intraitable pour les bases de données SLU.

L'algorithme SPM passe par 4 étapes principales qui sont :

- ***Etape 1 :*** le calcul de la mesure expected support.
- ***Etape 2 :*** L'optimisation : il existe 3 sous-programmes optimisés
  - **1)** pour le calcul de tous les fréquentes 1-séquences
  - **2)** pour le calcul de soutien supplémentaire
  - **3)** pour l'élagage probabiliste.
- ***Etape 3 :*** Génération du candidat : cette étape décrit deux méthodes de génération de candidat pour énumérer tous les fréquentes séquences.
- ***Etape 4 :*** Evaluation.

6.2 L'algorithme d'extraction de motifs séquentiels incertains (USPM) [35]

Cet algorithme est basé sur le principe de l'algorithme PreFixSpan classique. Il a fait deux modification principale pour l'incertitude, ceci apparait dans le pseudo code suivant :

Pseudo code de l'algorithme USPM :

Algorithme : USPM$(s, D|_s)$

Entrer: motif séquentielle$s$, base de donnéesprojetéeincertaine$D|_s$,

$minsup=t_s$, $minprob= t_p$

Sortie: L: un ensemble de motifs séquentielle frequents

Trouvertous les items fréquents$I = \{i_1, i_2, \dots, i_n\}$Dans $D|_s$

si$D|_s = \varphi$ *or* $I = \varphi$ **alors**

return L

fin

Pour chaque*item* $i \in I$ **faire**

$s' = s + \{i\}$

Pour$d_i|_{o_1,\dots,o_n} \in D|_s$***Faire***

Pour $d_i|_{o_j} \in d_i|_{o_1,\dots,o_n}$**Faire**

construireuneséquenceprojetée $d_i|_{o_s}$de $d_i|_{o_j}$

calculer$T(o_{s'})$de$T(o_{sj})$ dans$d_i$par:

$$P(< T_1, \dots, T_n >) = \sum_{i=1}^{p} P\,(< T_1, \dots, T_{ni} > \cap\, T_{ni})$$

fin

$$P(< T_1, \dots, T_{n_i} >) = \sum_{j=1}^{q} P\left(T_{(n-1)_j} T_{n_i}\right) P\left(< T_1, \dots, T_{(n-1)_j} > \cap\ T_{(n-1)_j}\right)$$

fin

calculer la probabilité de support $P(s' \leq d_i)$ by

$$P(s \leq d) = \sum_{i=1}^{N} P(s \leq o_{s_i})$$

fin

utilisé FFT pour calculer la distribution de Poisson binomiale de $sup(s')$

si$P(sup(s') \geq t_{s)} \geq t_p$**alors**

$$L = L \cup \{s'\};$$

$USPM(D|_{s'}, s');$

Fin **fin**

CHAPITRE 2. EXTRACTION DE MOTIFS SÉQUENTIELS À PARTIR DES DONNÉES INCERTAINES.

Après l'étude des travaux pour l'extraction des motifs séquentiels à partir des données incertaines dans le contexte des big data, nous avons trouvé un seul travail qui existe celui de Jiaqi Ge et al et qu'on présente par la suite [36].

6.3 Extraction des motifs séquentiels fréquents à partir des données incertaines dans le contexte de big data

La fonction MapReduce a pour objectifs de chercher et extraire des motifs séquentiels fréquents en temps réel (en parallèle). Cette recherche passe par plusieurs itérations. Dans chaque itération, les MapReduce utilisent une méthode pour rechercher k-longueur de motifs fréquents dans une grande masse de données. Les algorithmes suivants présentent les étapes de chaque fonction de MapReduce :

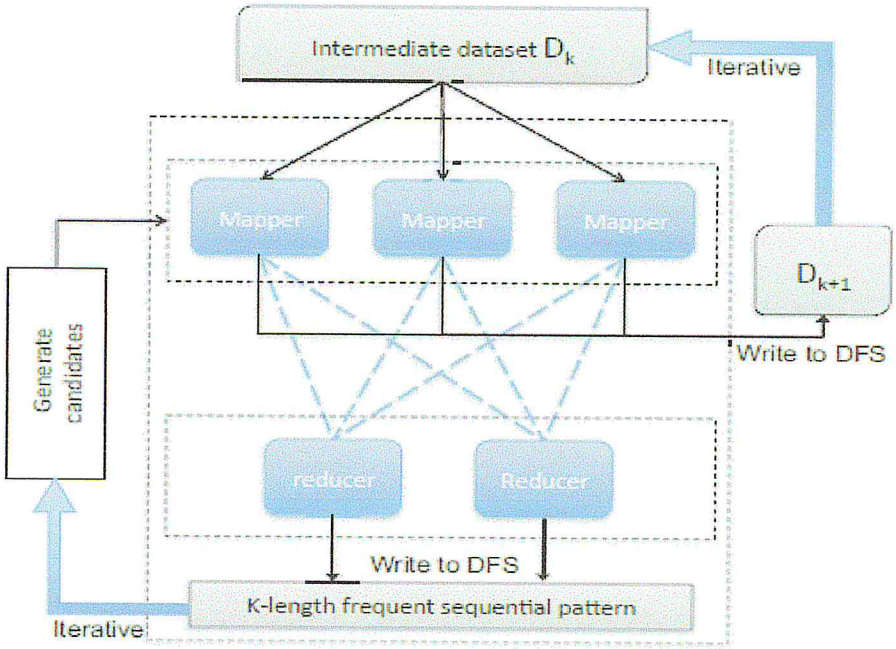


Figure 2.4 : Iterative MapReduce framework for uncertain sequential pattern mining. [36]

Algorithme Map

Map (key key, valeur valeur, contexte contexte)

d_{k-1}← pase (valeur) / * $d_{k-1} \in D_{k-1}$ analysé de la valeur * /

C_k ← DistributedCache.file

d_k ← construire à partir de C_k et d_{k-1}

pour chaque c ∈ Ck **faire**

p ← P ($C \subseteq d_k$) / * calculé en additionnant P_c(c) dans d_k * /

key← c;

valeur ← ‹p, p * (1 - p)› / * Valeur composited * /

context.collect (clé, valeur)

fin

DFS .file f = new DFSFile ("D_k");

f.append (d);

Algorithme combinateur

Combiner (key key, itératifs valeurs, contexte contexte)

$\mu \leftarrow 0, \sigma^2 \leftarrow 0$

pour chaque valeur ∈ valeurs **faire**

$\mu = \mu$ + value.μ

$\sigma^2 = \sigma^2$ + value. σ^2

fin

context.collect (key, ‹ μ, σ^2›)

Algorithme Reduce

Reduce (key key, itératifs valeurs, contexte contexte)

c ← key

μ ← 0, σ^2 ← 0

pour chaque valeur ∈ valeurs **faire**

μ = μ + value.μ

$\sigma^2 = \sigma^2$ + value. σ^2

fin pour

sup (c) ~ N (μ,σ^2)

t_s ← context.minsup, t_p← context.minprob

Si P (sup$(c) \geq t_s \geq t_p$) Alors

DFS.file f = new DFSFile («motif fréquent») ;

f.append (c) ;

fin

6.3.1Le déroulements des algorithmes précédents

Dans la première itération, la base de données d'origine est divisée en plusieurs ensembles de données, chaque ensemble représente le paramètre d'entrée dans la fonction map.

Dans le kème (k> 1) itération, les données d'entrée d'un mappeur est un morceau de D_{k-1}. Nous modifions la fonction split de données dans MapReduce pour vous assurer que toutes les occurrences dans une séquence sont entrées dans la même fonction de Map. Un ensemble de motifs candidats k-langueur, sont distribués aux mapper, qui est noté par C_k.

6.3.2 Fonction Map

La fonction map est représentée dans l'algorithme 1. Il construit d'abord d_k à partir de d_{k-1} etC_k, où $d_{k-1} \in D_{k-1}$ contient les occurrences de (k - 1)-langueur motifs fréquents dans une séquence incertaine. Étant donné un candidat modèle c, le mappeur calcule la probabilité de soutien $p = P(c \subseteq D_k)$ en utilisant la structure de données récemment mis à jour et délivre un couple clé-valeur ‹C, ‹μ, σ^2›› si$p = P(c \subseteq d_k)$.

Ici, $\mu = p$ et $\sigma^2 = p * (1 - p)$ sont la moyenne et la variance de la variable aléatoire de Bernoulli sup$(c|d_k)$. Par la suite, d_k est écrit au système de fichiers distribué (DFS) à utiliser dans la prochaine itération.

6.3.3Fonction combinateur

Nous concevons une fonction combinateur dans l'algorithme 2 pour aider à améliorer la performance du notre algorithme. Supposons une fonction de mapper émet n paires de (clé-valeur) ‹c,‹ μ_i, $\sigma^2{}_i$ › (i = 1,..., n) qui sont associés au motif identique c. Comme la valeur déposée de la sortie du mappeur est associatif et commutatif, ils peuvent être condensés à une seule paire ‹c, ‹$\sum_1^n \mu_i$, $\sum_1^n \sigma^2{}_i$ ››. Ensuite, chaque mappeur envoie une seule paire de (clé-valeur) au réducteur pour chaque motif candidat, ce qui réduit considérablement le coût total de la bande passante de brassage de données.

6.3.4Fonction Reduce

Dans l'entrée de cette fonction le couple de pair ‹Clé, Valeur› sous la forme de ‹c, ‹μ_i, $\sigma^2{}_i$››, où $\mu_i = \sum$ p et $\sigma^2{}_i = \sum$ p * (1 - p) Sont la moyenne et la variance partiellement agrégée du support probabiliste candidat c.

La fonction de reduce accumule la moyenne globale et de la variance de c dans l'ensemble de la base aléatoire et utilise la distribution gaussienne pour rapprocher la distribution de sup global d'appui (c). Compte tenu de minsup = t_s et minprob = t_p, si $P(\sup(c) \geq t_s) \geq t_p)$ le reduce émet les motifs séquentiels fréquents probabilistes dans le fichier, sinon, c n'est pas probabilistiquement fréquente et est rejeté par le réducteur.

Une itération de MapReduce est terminée après tout k-longueurs motifs séquentiels fréquents probabilistes sont découverts et écrites dans les fichiers DFS. Après cela, nous joignons k-longueur motifs fréquents pour générer (k + 1) motifs candidats-Longueur pour la prochaine itération. Ce processus se poursuit jusqu'à ce que tous les motifs fréquents soient découverts.

7. Conclusion

Dans ce chapitre nous avons vu les différents travaux connexes concernant l'extraction de motifs séquentiels à partir des données incertaines dans deux cas différents, séquentielle et parallèle

Notre but est de donner une meilleure vue du travail qui sera présenté dans les chapitres suivants.

CHAPITRE 3:

META-HEURISTIQUE

1. Introduction

Le principe de méta heuristique est inclut dans les méthodes de résolution de problème d'optimisation. Dans ce chapitre, nous citrons les différentes méthodes de résolution de problème d'optimisation et nous nous baserons sur les méthodes de méta-heuristique.

2. Les méthodes de résolution de problème d'optimisation

Ces méthodes sont regroupées en trois classes principales : la classe de la méthode exacte, la classe de la méthode approchée et des hybrides. La Figure.3.1 démontre le contenu de chaque classe.

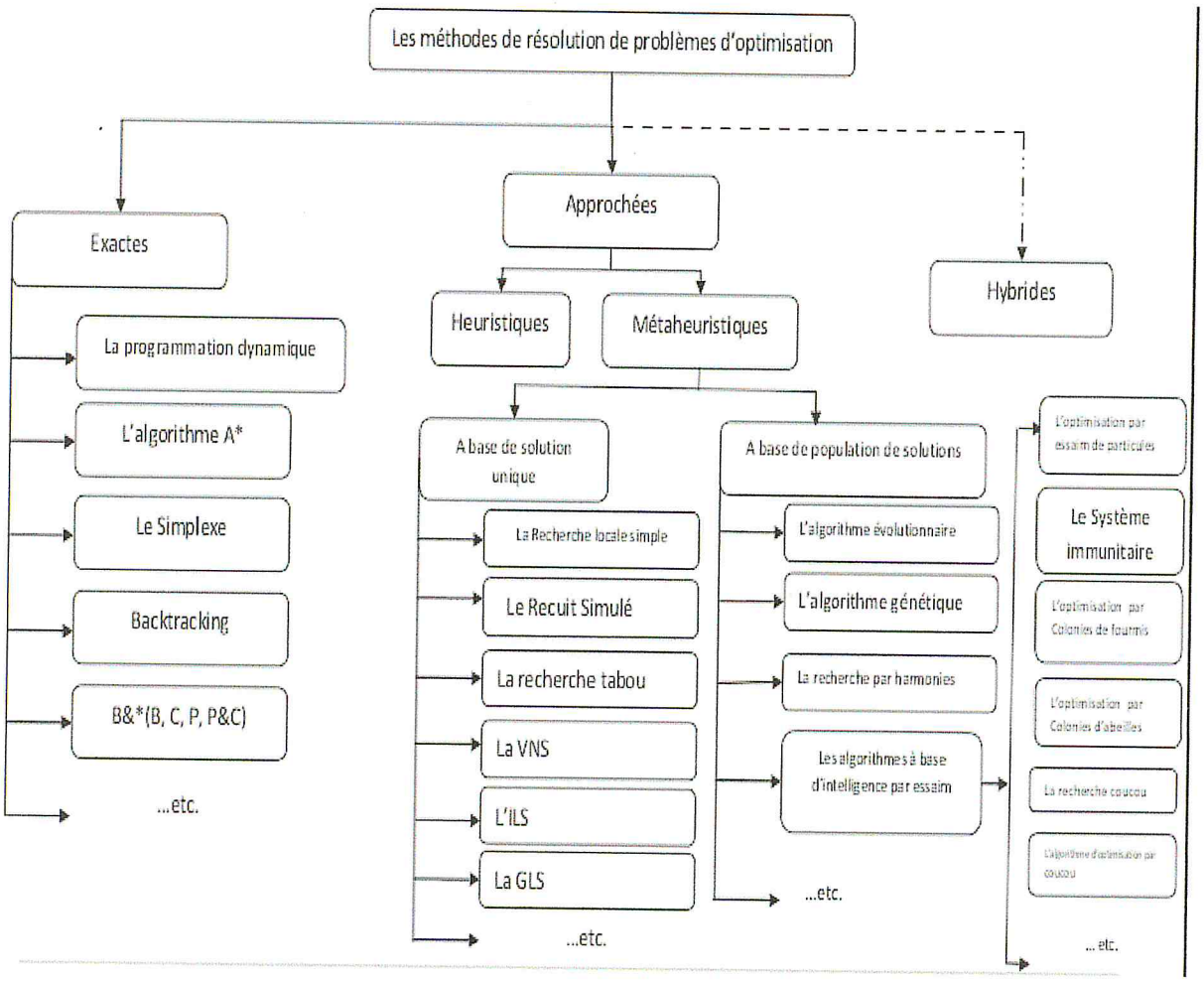


Figure.3.1 : Classification de méthode de résolution de problème. [37]

3. La classe des méthodes exactes [37]

Le temps et l'espace mémoire est nécessaire pour obtenir une solution optimale par une méthode exacte.

Il existe plusieurs algorithme dans cette classe : par exemple l'algorithme dynamique, l'algorithme A*, les algorithmes des séparations et évaluations, les algorithmes de retour arrière (Backtracking). Nous citons quelque unes dans ce qui suit :

3.1. L'algorithme de retour arrière (Backtracking)

Le principe de cet algorithme est de parcourir les valeurs des variables par instanciation. Afin d'obtenir une configuration consistante, le domaine de définition du variable x_i affecte une nouvelle valeur. Le processus est répété à chaque itération jusqu'à l'obtention d'une solution complète.

Si toutes les possibilités ont été essayées sans donner une bonne solution alors le problème est irréalisable

✓ **Les avantages de cet algorithme**

- Permet une exploitation complète de l'espace de recherche.
- Garantit l'aboutissement à la solution optimale.

✓ **Les inconvénients**

Il est très lent.

3.2. La méthode Branch and Bround (B&B)

C'est la méthode de séparation et d'évaluation en anglais Branch and Bround, ces méthodes résolu les problèmes combinatoires.

Cette méthode est basée sur la technique divisée pour régner Consiste à diviser le problème en sous problème représenté sous forme d'arborescence, tel que chaque nœud correspond à une solution partielle.

Les solutions partielles se forment de manière incrémentale dans l'arbre. Chaque solution partielle possède une borne supérieure et une borne inférieure. Ces deux bornes sont utilisées pour couper quelques branches de l'arbre et ainsi éviter d'explorer tout l'arbre.

Alors, si l'évaluation partielle d'un nœud x_i a montre que sa qualité est supérieure à la borne supérieure, le sous arbre sera élagué. Sinon, le nœud sera divisé en sous nœud. Ce processus se répété tant que reste des branche non parcourue. La recherche continue jusqu'à trouver une solution optimale, si elle existe.

✓ **L'avantage de cette méthode**

Permet de trouver de bonne solution en un temps de recherche raisonnable.

4. La classe des méthodes approchées

Le but d'utilisation des méthodes approchées est de fournir de très bonne qualité de solution en un temps de calcule raisonnable.

Il existe plusieurs méthodes approchées, ces méthodes sont utilisées pour la résolution de problèmes difficiles ou de problèmes dont on cherche des solutions en un bref délai.

Elles sont souvent classées en deux catégories : des méthodes heuristiques et des méthodes méta-heuristiques.

D'abord, nous citons quelques définitions des méthodes heuristiques puis des méthodes méta-heuristiques.

4.1. Les méthodes heuristiques

Une méthode heuristique (ou simplement une heuristique) est une méthode qui aide à découvrir la solution d'un problème en faisant des conjectures plausibles mais faillible de ce qui est la meilleure chose à faire. [38]

Les heuristiques sont des critères, des méthodes ou des principes pour décider qui, parmi plusieurs d'autres plans d'action permet d'être le plus efficace pour atteindre un certain but. [39]

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

4.2. Les méthodes de méta-heuristiques

Une méta-heuristique est un processus itératif qui subordonne et qui d'une heuristique en combinant intelligemment plusieurs concepts pour explorer et exploiter tout l'espace de recherche. Des stratégies d'apprentissage sont utilisées pour structurer l'information afin de trouver efficacement des solutions optimales ou presque optimales. [40]

Les techniques de méta heuristique sont classées en deux groupes : les méthodes à une solution unique et les méthodes à population de solution connues.

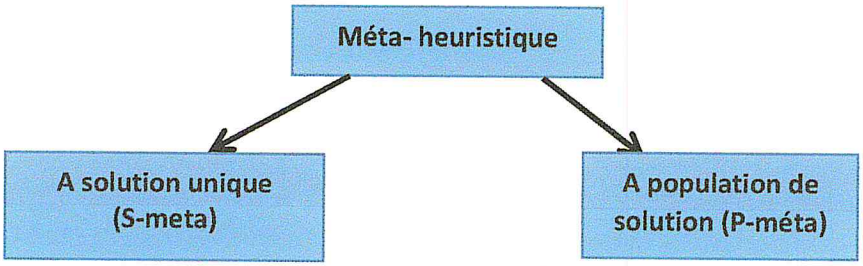


Figure 3.2 : classification des méthodes de méta-heuristique [40]

La plus part des méthodes de méta-heuristique utilisent le voisinage. Une méthode voisinage débute avec une configuration initiale à laquelle un processus itératif est appliqué. Il cherche à améliorer la configuration courante en le remplaçant par une de ses voisines en tenant compte de la fonction objectif. Ce processus est arrêté et donne la meilleure solution trouvé si le critère d'arrêt est atteint. Cette condition d'arrêt concerne une limite sur le nombre d'itération ou sur l'objectif à réaliser. [41]

Dans les figures suivantes (Figure.3.3, Figure.3.4, Figure.3.5) nous présentons les contenus de chaque méthode de méta-heuristique. Puis nous entrons dans les détails de chaque méthode.

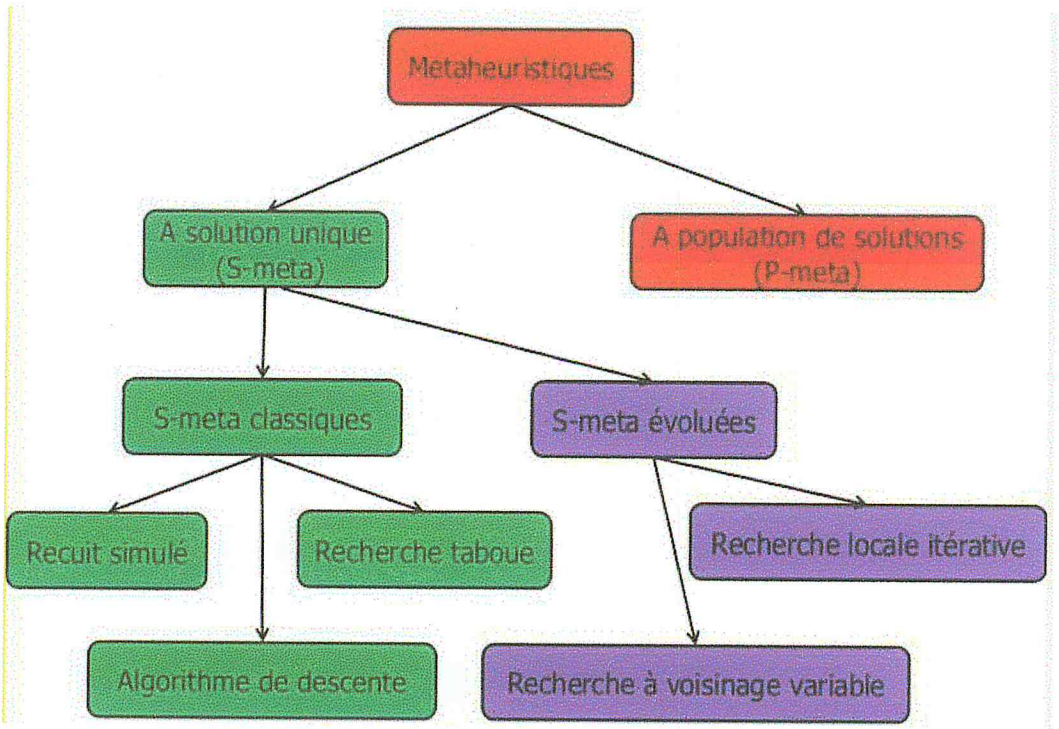


Figure.3.3 : Classification à solution unique (s-méta) [41]

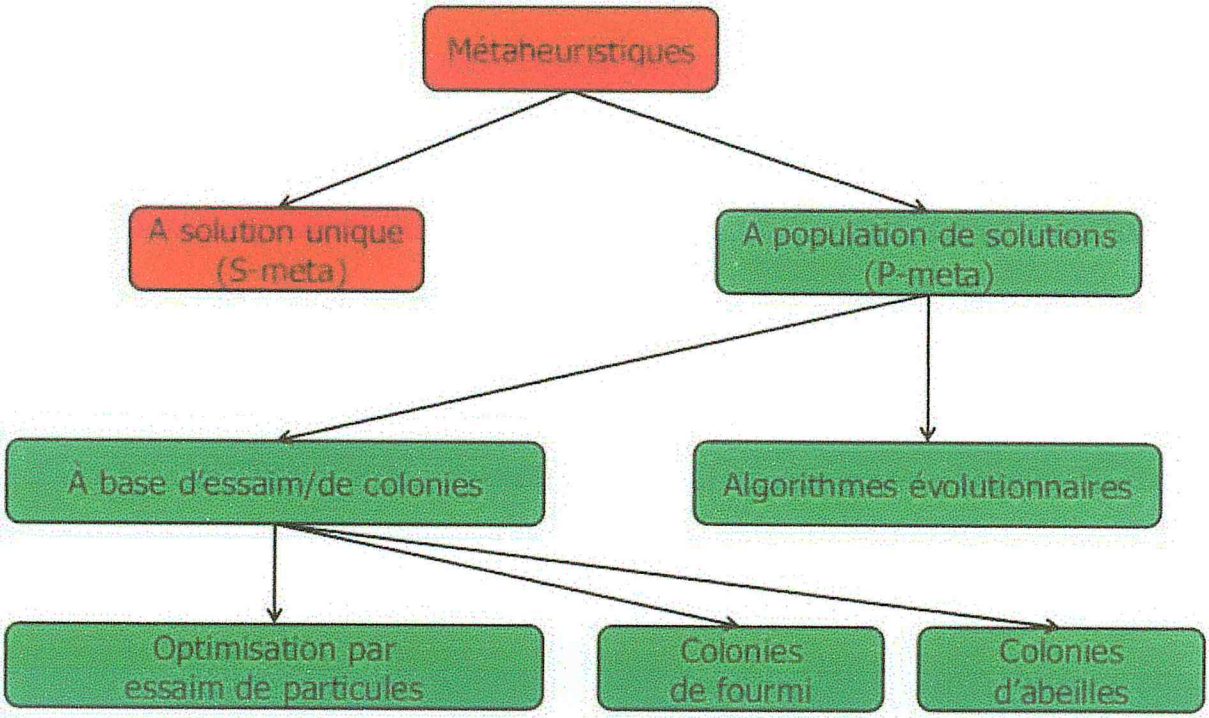


Figure 3.4 : Classification à population de solution (p-méta) à base d'essaim / de colonie. [41]

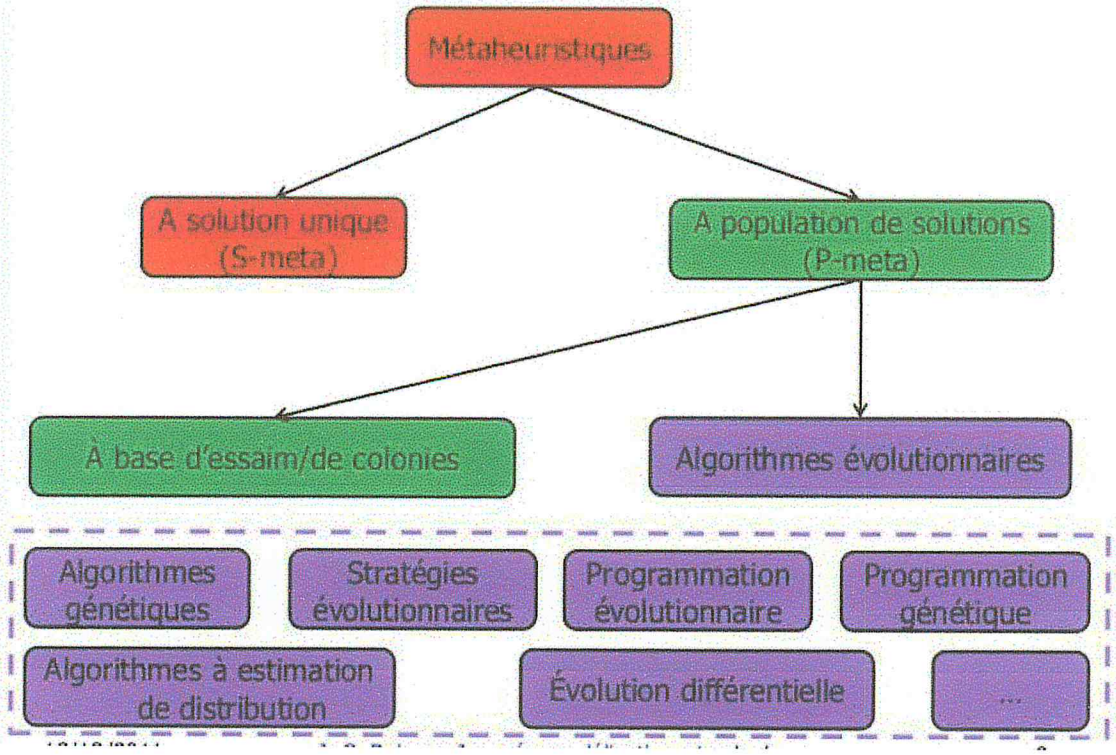


Figure.3.5 : Classification à population de solution (p-méta) à niveau d'algorithme évolutionnaire. [41]

4.2.1. Les méthodes à base de solution unique

a. Recuit simulie

Est une méthode basée sur la recherche locale dans laquelle chaque mouvement est accepté s'il améliore la fonction objective. Cette méthode est inspirée du processus de recuit utilisé en métallurgie pour améliorer la qualité d'un solde en cherchant un état d'énergie minimum. Cette méthode considère une solution initiale et cherche son voisinage d'une autre façon aléatoire. [43]

Algorithme Le recuit simulé

Début

Construire une solution initiale s ;

Calculer la fitness f(s) de s ;

Initialiser une valeur de la température T ;

$s_{best} = s$;

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

Générer une solution s' voisine de s;

Calculer f(s') ;

Calculer $\Delta(f)=f(s') - f(s)$;

Si $\Delta(f) \geq 0$ **alors** // cas de maximisation

$s_{best} = s'$

$S=s'$

Sinon Si $r < exp\frac{\Delta(f)}{T}$ **alors**

$s=s'$

Fin Si

Décroitre la température T ;

Fin Tant que

Retourner s_{best} ;

Fin

b. La descente

La descente est un algorithme d'amélioration. Le principe de cet algorithme est d'exploré le voisinage de la solution courante afin d'améliorer sa qualité. A chaque itération l'algorithme modifie un ensemble de composante de solution courante pour permettre le déplacement vers

une solution voisine de meilleure qualité. Le processus est répété jusqu'à la satisfaction du critère d'arrêt. Il existe trois types de descente : déterministe, stochastique et descente vers le premier meilleur voisin. L'algorithme suivant résume les étapes générales de la descente.

```
Algorithme La recherche locale simple (la descente)
Début
   Construire une solution initiale s ;
   Calculer la fitness f (s) de s ;
   Tant que la condition d'arret n'est pas verifies faire
         Modifier s pour obtenir une nouvelle solution voisine s';
         Calculer f ( s') ;
         Si f ( s') est meilleure que f (s) alors
               Remplacer s par s'
         Fin Si
   Fin Tant que
   Retourner s ;
Fin
```

c. Recherche taboue

Le principe de cet algorithme est de sélectionné le voisinage de solution à chaque itération. La méthode autorise de remonter vers des solutions qui semblent moins intéressent mais qui ont peut-être un meilleur voisinage. Des fois ce principe engendre des phénomènes des cyclages entre deux solutions, tandis que la méthode taboue à l'interdiction de visiter une solution récemment visité. Pour cela une liste taboue contenant les attributs des dernières solutions considérée et tenue à jour. Chaque nouvelle solution considérée enlève de cette liste la solution la plus anciennement visitée. Ainsi, la recherche de solution suivante se fait dans le voisinage de la solution actuelle sans considérer appartenant à la liste taboue. [44]

Algorithme La recherche tabou

Début

Construire une solution initiale s ;

Calculer la fitness $f(s)$ de s ;

Initialiser une liste tabou vide ;

s_{best}=s ;

Tant que le critère d'arrét n'est pas verifier **faire**

Trouver la meilleure solution s' dans le voisinage de s qui ne soit pas tabou ou qui verifier le critère d'aspiration;

Calculer $f(s')$;

Si fitness de (s') est meilleure que fitness de (s_{best}) **alors**

$s_{best} = s'$

Fin Si

Mettre à jour la liste tabou ;

$s = s'$

Fin Tant que

Retourner s_{best} ;

Fin

4.2.2. Les méthodes à base de solution population

Ces méthodes contiennent deux sous bases qui sont des essaims de colonie et algorithmes évolutionnaire.

4.2.2.1. Les essaims de colonie

a. Les colonies de fourmis (Ant Colony Optimization : ACO)

ACO [45] a été inspirée des études du comportement de fourmis réelles pour résoudre naturellement des problèmes relativement complexes. Le premier algorithme de ce méta heuristique a été appliqué pour résoudre le problème du voyageur de commerce, le principe de cet algorithme est simple. Lorsqu' une fourmi k se déplace de la ville i à la ville j, elle laisse une trace sur le chemin. De plus, elle choisit la prochaine ville à visiter à l'aide d'une probabilité basée sur un compromis entre l'intensité de la trace et la visibilité qui représente l'inverse de la distance entre i et j. (d_{ij})

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

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

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

Pseudo code de l'algorithme

S'il y'a une place libre dans la station de chargement **alors**
  Pour $t = 1$ à t_{max}
    Pour chaque fourmi $k = 1$ à m **Choisir**
      pour la première pièce de la file infini un routage au hasard suivant son type.
        Pour chaque pièce *i* contenu dans la deuxième place jusqu' au la n^{eme} place de la file infini. **Choisir**
          un routage j, parmi les routages possibles selon une probabilité dépendant de l'intensité de la trace et du nombre de pièces dans la file d'entrée de la première machine de ce routage et sa charge.
        Fin Pour
      Evaluation de la fonction objective. (Produit des charges de routages)
      Déposer une $\Delta\pi_4^1(t)$ piste sur le trajet T^K(t) (pour chaque routage j choisi pour la pièce i par la fourmi k).
      Fin Pour.
  Évaporer les pistes et modifier les intensités.
  Fin Pour.
Finsi.

b. Les essaims particulaires (Particle Swarm Optimization : PSO)

PSO [68] est basée sur la métaphore des interactions et communications sociales. Au départ de l'algorithme, un essaim est réparti au hasard dans l'espace de recherche, chaque particule ayant également un.

- ✓ chaque particule est capable d'évaluer la qualité de sa position et de garder en mémoire sa meilleure performance, c'est-à-dire la meilleure position qu'elle a atteinte jusqu'ici (qui peut en fait être parfois la position courante) et sa qualité (la valeur en cette position de la fonction à optimiser).
- ✓ chaque particule est capable d'interroger un certain nombre de ses voisins et d'obtenir de chacune d'entre elles sa propre meilleure performance (et la qualité afférente).
- ✓ A chaque pas de temps, chaque particule choisit la meilleure des meilleures performances dont elle à connaissance, modifie sa vitesse en fonction de cette information et de ses propres données et se déplace en conséquence.

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

Pseudo code

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

c. Colonie d'abeille

Le PSO étant initialement prévu pour l'optimisation continue, une autre famille d'algorithme utilise la notion de vol dans l'espace de solutions, pour les problèmes combinatoires.

L'algorithme à colonie d'abeilles, en anglais Artificial Bees Colony (ABC), [46] et [47] s'inspire du vol des abeilles à la recherche de fleurs pour butiner. L'ABC explore le voisinage des sources de nourriture pour converger, et, pour diversifier, des solutions sont créées aléatoirement. L'algorithme commence par la création aléatoire des sources de nourriture et leur évaluation.

Dans une première phase d'amélioration, une recherche locale est appliquée sur des sources de nourriture sélectionnées aléatoirement. En second, une recherche locale est faite sur des sources de nourriture sélectionnées au hasard, mais les meilleures évaluations ont plus de chance d'être choisies. Après, le nombre de sources de nourriture est réduit en ne gardant que les meilleures solutions. La troisième étape diversifie l'exploration en créant des solutions aléatoirement, jusqu'à atteindre la taille de la population initiale. Après, si le critère d'arrêt n'est pas atteint, l'algorithme revient à la première phase. Sinon, la meilleure solution est retournée par l'algorithme.

Pseudo code de cet algorithme

```
Paramètre m : Nombre de sources de nourriture sélectionnées pour la recherche de voisinage
Paramètre e : Nombre d'abeilles pour visiter les meilleures sources de nourriture
Paramètre n : Nombre d'abeilles pour trouver de nouvelles sources de nourriture
Paramètre T : Un critère d'arrêt
    Initialisation de la population avec des sources de nourriture aléatoires
    évaluation de la population
    Tant que T n'est pas atteint Faire
        Sélectionner m sources de nourriture pour faire des recherches locales
        Visiter le voisinage de la e meilleure source de nourriture
        Sélectionner les meilleurs voisinages visités par les abeilles
        Créer n sources de nourriture aléatoirement et les évaluer pour compléter la nouvelle
        population
    Fin Tant que
Retourne Meilleure source de nourriture
```

4.2.2.2. Les algorithmes évolutionnaires

Ces méthodes sont basées sur le principe de la sélection naturelle. Le principe de cette dernière stipule que les individus bien adaptés à l'environnement ont plus de chances de se reproduire et de survivre que les autres individus. En fait, la combinaison des caractéristiques

des individus peut former au fil des générations de nouveaux individus de plus en plus mieux adaptés à leur environnement et qui peuvent avoir plus de chances de survivre que leurs parents.

Les algorithmes évolutionnaires s'inspire de l'évolution naturelle des êtres vivants. Ils adoptent une sorte d'évolution artificielle pour améliorer la qualité des individus de la population.

Le processus d'évolution d'un algorithme évolutionnaire est basé sur trois opérations principales :

- ✓ **La sélection**

La sélection est appliquée pour choisir les meilleurs individus parents qui vont se reproduire pour construire de nouveaux individus enfants et appliqué aussi a la fin de chaque itération pour opter pour les individus qui vont construire une nouvelle population.

- ✓ **La reproduction**

Contient deux opérations, le croisement et la mutation. Elle utilise le croisement pour combiner les caractéristiques de nouveaux individus puis utilisent la mutation pour faire des modifications de certains individus pour améliorer leurs qualités.

- ✓ **L'évaluation**

L'évaluation consiste à mesurer la qualité de chaque individu. Donc calculer la fitness des individus.

La classe d'algorithme d'évolutionnaire principale contient plusieurs sous classe. Dont l'algorithme génétique.

L'algorithme génétique

Algorithme génétique [48] s'inspire des mécanismes biologiques tels que les lois de Mendel et la théorie de l'évolution. Son processus de recherche de solutions à un problème donné imite celui des êtres vivants dans leur évolution. Il utilise le même vocabulaire que celui de la biologie et la génétique classique, on parle donc de : gène, chromosome, individu, population et génération.

- ✓ **Un gène :** est un ensemble de symboles représentant la valeur d'une variable. Dans la plupart des cas, un gène est représenté par un seul symbole (un bit, un entier, un réel ou un caractère).
- ✓ **Un chromosome** : est un ensemble de gènes, présentés dans un ordre donné de manière qui prend en considération les contraintes du problème à traiter. Par exemple, dans le

CHAPITRE 4 :

SOLUTION PROPOSE

SOLUTION PROPOSÉ

1. Introduction

Parmi les méthodes approchées, on trouve les méta-heuristiques est un processus itératif qui subordonne et qui d'une heuristique en combinant intelligemment plusieurs concepts pour explorer et exploiter tout l'espace de recherche.

Les techniques de méta heuristique sont classées en deux groupes : les méthodes à une solution unique et les méthodes à population de solution connues, on trouve dans ses méthodes à population de solution, l'algorithme BCO étant initialement prévu pour l'optimisation continue.

Dans ce chapitre, tous d'abord nous allons présenter les étapes principales de la méthode d'optimisation par colonie d'abeilles (BCO) (Bees Swarm Optimization BSO) pour gérer et traiter notre propre méthode. Ensuite nous allons décrire la méthode d'extraction des motifs séquentiels à partir des données incertaines dans le contexte de big data avec son principe de fonctionnement, puis nous étudieront les différentes étapes de parcours et illustration avec des exemples d'utilisation de cette méthode, Enfin, nous montrerons dans un schéma la conception de notre solution.

2. Pseudo code de l'algorithme général d'optimisation par colonie d'abeilles (BCO) [49]

```
Début
    Laisser Sref soit la solution trouvée par Abeille Initiale ;
    Tanque
            la condition d'arrêt n'est pas satisfaite          Faire
Début
    Insérer Sref dans la liste tabou ;
    Déterminer l'espace de recherche de Sref ;
    Affecter une solution de l'espace de recherche à chaque abeille ;
    Pour
            chaque Abeille K                                   Faire
Début
    Rechercher en commençant par la solution affectée à elle ;
    Stocker le résultat dans le tableau Danse ;
Fin ;
    Choisissez la nouvelle solution de référence Sref ;
Fin ;
Fin ;
```

3. Solution proposé

Notre propre solution proposée serre à une extraction de motif séquentiel fréquent à partir des données incertaines dans le contexte des big data en utilisant l'algorithme des abeilles.

Dans notre cas on travaille avec des bases de données séquentielles probabilistes, d'après le deuxième chapitre on a vu qu'il existe 3 niveaux d'incertitudes, niveau des sources, niveau de temps et niveau d'évènement.

Dans notre solution l'incertitude sera au niveau d'évènement, les motifs séquentiels établissent des corrélations entre les événements au cours de temps, on prend un exemple d'un client qui fait plusieurs achats dans un magasin dans différent temps.

Exemple

ID Client	Temps de transaction	Item set
1	Janvier 1995	ab:0.5
1	Mai 2001	c:0.2
1	Juin 2002	abc:0.7
2	Avril 2014	d : 0.4
2	Septembre 2015	abcd:0.5
3	Novembre 2012	d:0.4
3	Février 2016	abcd:0.5
3	Mars 2016	db:0.7

Tableau 4.1 : Liste d'achat des clients.

D'après le tableau précédent on remarque que le premier client fait 3 achats dans un temps différent, le deuxième client fait deux achats dans différent temps et la même chose pour le 3ième client.

Donc la base de données séquentielle d'après l'exemple de tableau sera comme suit :

ID client	Motif séquentiel
1	(ab:0.5)(c:0.2)(abc:0.7)
2	(d:0.4)(abcd:0.5)
3	(d:0.2)(abcd:0.8)(bd:0.1)

Tableau 4.2 : Les motifs séquentiels.

Ce tableau présente des séquences identifier par le numéro du client, ou chaque séquence contient un ensemble d'achats ou bien ensemble d'item set.

4. L'algorithme BSO-UFSPM pour l'extraction des motifs séquentiels à partir des données incertaines

Notre solution propose 3 étapes nécessaires a passé pour cet extraction, au début on a une base de données séquentielle probabiliste qui contient des items set incertaines.

1ière étape

Le début de notre algorithme proposé contient une solution Sref donnée d'une façon aléatoire de type séquence et deux valeurs nécessaires pour débuter notre programme, une valeur MinSup et l'autre présente le nombre d'itération MaxIter qui sont donnée par l'utilisateur.

Tout d'abord en calcule la solution Sref qui est choisi par l'utilisateur si elle est fréquente ou pas.

Si Sref est fréquente on passe à la deuxième étape sinon on va lancer la fonction rundom qui donne une autre Sref aléatoire jusqu'on obtienne une solution fréquente.

Pour calculer la fréquentness de Sref on doit utiliser la fonction Expected Support.

a. La fonction Expected Support

Pour calculer la fréquentness de Sref on a besoin des paramètres d'entrés qui sont les suivants :

La solution Sref, notre base de données séquentielle, et enfin la valeur de MinSup.

Ensuite en compte combien de fois Sref existe dans notre base de données. A chaque fois qu'on trouve Sref est égale à une séquence dans notre base de donnée, on calcule le produit des probabilités des item set (des évènements) de cette séquence et on la sauvgarde.

Après on fait la somme des produits et sa sera la valeur de fréquentness de Sref. Donc l'expression de la fonction Expected Support est présentée comme suit :

$$\text{ExpSup} = \sum \prod_{n=1} S_n$$

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

2ième étape

Après avoir une solution Sref fréquente, dans la deuxième étape on utilise une méthode sur la solution Sref qui s'appelle stratégie de l'inverse, elle serre a trouvé un espace ou bien un ensemble de solutions dans notre base de données qui sont plus fréquentes que Sref.

1 | 1 | 1 | 0 | 0 | 0 | 0 | **1** | 1 | 0 | 0 | 0 | 0 | 0 | 1 |

1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | **0** | 0 | 0 | 0 | 0 | 0 | 1 |

1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | **1** | 0 | 0 | 0 | 0 | 1 |

1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | **1** | 0 | 0 | 0 | 1 |

1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | **1** | 0 | 0 | 1 |

1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | **1** | 0 | 1 |

1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | **1** | 1 |

1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | **0** |

Donc l'espace de recherche ou bien **l'ensemble des solutions** sont :

bc | d| e

ac| d| e

ab| d |e

abcd |d |e

abce |d |e

abc| ad |e

abc| bd |e

abc| cd |e

abc |e

abc| d| ae

abc| d| be

abc| d| ce

abc| d| de

abc| d

Après l'extraction de l'espace de solution, on donne à chaque k abeille une solution de l'ensemble des solutions, donc la valeur k sera le nombre des solutions, dans l'exemple précédent k=15, et on passe à la 3ième étape.

3ième étape

C'est l'étape la plus longue, elle consiste a donnée à chaque k ou bien à chaque abeille une solution des solutions de la 2ième étape.

Après, chaque abeille utilise la méthode de la stratégie de l'inverse sur sa propre solution et elles obtiennent un nouvel ensemble de solution (searchArea), comme par exemple :

On prend la première solution (bc | d| e) du résultat de l'étape 2 on obtient :

searchArea de (bc | d| e) est :

abc/d/e

c/d/e

b/d/e

bcd/d/e

bce/d/e

bc/ad/e

bc/db/e

bc/cd/e

bc/e

bc/de/e

bc/d/ae

bc/d/be

bc/d/ce

bc/d/de

bc/d

Après on va calculée la fréquentness de chaque solution obtenu, et on prenant les solutions fréquentes et on ignore les solutions non fréquentes, la meilleure solution on la sauvegarde avec sa valeur de fréquentness dans deux tableaux : Danse de taille K et le deuxième tableau résultat finale, c'est le tableau d'affichage, il contient tous les meilleures solutions obtenu de Sref, de la taille K*MaxIter. Tout ce travail sera dans la 1ière itération.

Dans les autres itérations on prend la meilleure solution de tableau Danse de l'itération précédente et on la considère comme une nouvelle solution Sref.

Les abeilles dans ce cas travaillent d'une façon séquentielle (continue).

Notre but est de faire en sorte que les abeilles travaillent en parallèles donc c'est pour cette raison qu'on utilise la technologie hadoop, hadoop travaille le parallélisme comme montrer dans le chapitre passé technologie de BigData.

Adaptation de l'algorithme au contexte du big data

C'est le même principe comme montrée dans la partie séquentielle. Dans ce cas on utilise deux fonction principale dans hadoop, fonction Map et Reduce.

La fonction Map permet de partager le travail des abeilles en même temps et La fonction Reduce rassemble tous les résultats des Map dans un seul résultat final.

1ière étape

Fonction Map 1

Map 1 (key key, valeur valeur, contexte contexte)

Entrée : Base de données ;

Début

Laisser Sref soit la solution trouvée par Abeille Initiale aléatoirement ;

Calculer le produit des probabilités de Sref

Envoyé le produit des probabilités de Sref à la fonction Reduce

Fin ;

Fonction Reduce 1

Reduce 1 (key key, itératifs valeurs, contexte contexte)

Entrée : MinSup ;

Début

Calculer Expected Support de Sref ;

Comparer Expected Support de Sref avec le MinSup ;

Si Sref >MinSup **alors** Sref est fréquente ;

Sinon Sref n'est pas fréquente ;

Fin ;

2ième étape

Fonction Map 2

Map 2 (key key, valeur valeur, contexte contexte)

Entrée : les solutions inversées ;

Début

Appliqué la stratégie de l'inverse sur toute les solutions inversées ;

Envoyé l'espace des solutions à la fonction Reduce ;

Fin ;

Fonction Reduce 2

Reduce 2 (key key, itératifs valeurs, contexte contexte)

Début

Réception de l'espace des solutions ;

Fin ;

3ième étape

Fonction Map 3

Map 3 (key key, valeur valeur, contexte contexte)

Entrée : Base de données ;

Début

Calcule la fonction Expected Support de chaque solution dans chaque ligne ;

Comparer entre eux ;

Envoie La grande valeur de chaque ligne à la class Reduce. ;

Fin ;

Fonction Reduce 3

Reduce 3 (key key, itératifs valeurs, contexte contexte)

Entrée : MaxIter ;

Début

Tanque int i<MaxIter **Faire**

Compare entre les valeurs qui sont envoyée par les Map ;

Sauvgarder la meilleure solution dans le fichier Taboulist et sa sera l'entrés de la prochaine itération. ; i++ ;

Les étapes de l'algorithme proposé sont présentées dans le schéma suivant :

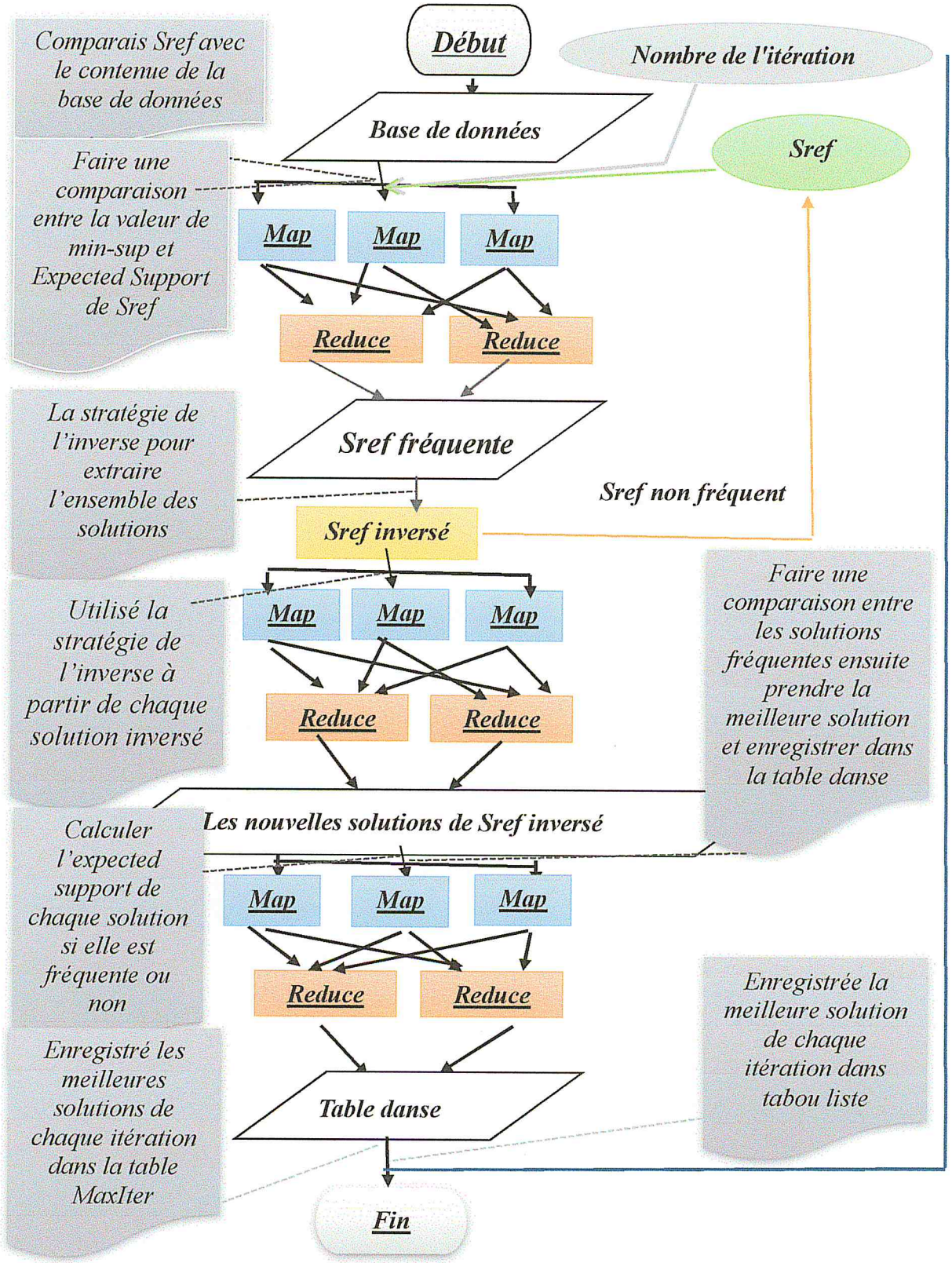


Figure 4.1 Conception de L'algorithme BSO-UFSPM.

5. Conclusion

Ce chapitre présente notre solution proposée. Elle intègre 3 étapes principales pour extraire les meilleures solutions fréquentes de notre base de données séquentielle incertaines.

Notre méthode proposé utilise le concept de l'algorithme BCO en utilisant les techniques de big data.

Dans le chapitre suivant on définit les outils et les logicielle de programmation pour implémenter et tester notre algorithme.

CHAPITRE 5 :

TEST ET VALIDATION .

1. Introduction

Après avoir abordé notre conception d'un algorithme parallèle du problème d'optimisation minimale pour extraire des séquences fréquentes à partir des données incertaines dans le contexte de big data avec le modèle de programmation MapReduce, en utilisant le principe de l'algorithme BCO, dans ce présent chapitre nous allons décrire notre implémentation avec Hadoop et mener une série de tests "Benchmarks" pour évaluer l'approche exposée dans le chapitre précédent.

Nous commencerons par présenter l'environnement de travail et des bases de données réel sur lesquelles les tests se feront. Après nous discuterons les résultats obtenus à l'issu des différents tests d'évaluation.

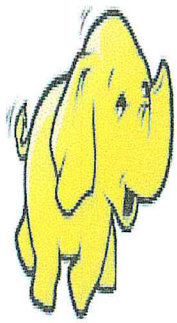
2. Présentation de l'environnement de travail

2.1 Systèmes d'exploitation

Nous utilisons le système Windows 7 son type est. 64-bit, sa RAM et de 4GO, son Disque dure et de 500 GO

Le processeur est Intel(R) Core(TM) i3 CPU M 330 @ 2.13GHz 2.13 GHz

2.2 Le Framework Hadoop



Hadoop est un framework libre et open source écrit dans le langage Java destiné à faciliter la création d'applications distribuées (au niveau du stockage des données et de leurs traitements) ainsi que le passage à l'échelle. Tous les modules de Hadoop sont conçus pour prendre en considération les pannes matérielles automatiquement par le framework.

Au niveau du cluster : le nœud ibnbadis15 est le maître et les nœuds 16-19 sont les esclaves.

2.3 Eclipse, Sun java 8

Eclipse : est un IDE, Integrated Development Environment (EDI environnement de développement intégré), un logiciel qui simplifie la programmation sous le langage JAVA en proposant un certain nombre de raccourcis. Il est gratuit et disponible pour la plupart des systèmes d'exploitation.

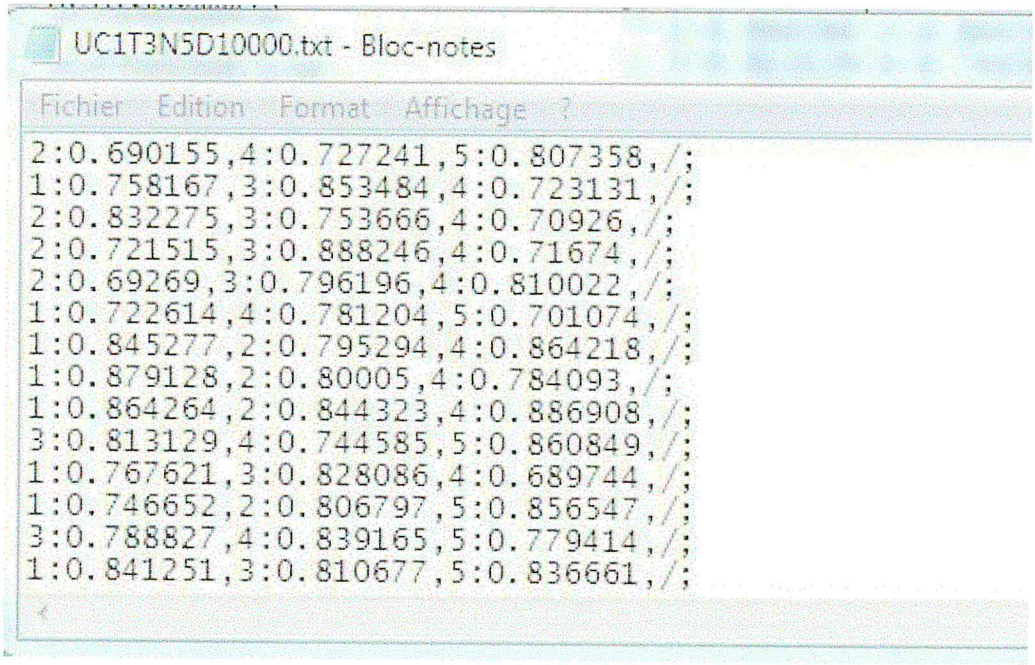
Nous avons utilisé Eclipse pour la programmation et l'implémentation des algorithmes ainsi que l'exportation des projets sous forme de .jar.

Sun java 8 : Le Java Development Kit (JDK) désigne un ensemble de bibliothèques logicielles de base du langage de programmation Java, ainsi que les outils avec lesquels le code Java peut être compilé. Pour l'exécution de Hadoop l'environnement java et prérequis, nécessitant installation d'une version 5 ou plus, nous avons choisi d'installer la version (Java7).

3. Présentation des données utilisées pour les tests "Benchmarks"

Les benchmarks que nous avons utilisé sont des fichiers texte qui contiennent le nombre des séquences dans la base de données (D), le nombre moyen d'item sets par séquence (c) , le nombre moyen d'items par item set (T), et enfin le nombre de différent items qui existe dans la base de données (N) .

Nous avons utilisé 3 benchmarks, afin de tester notre approche. Les figures suivantes représentent les benchmarks utilisés.

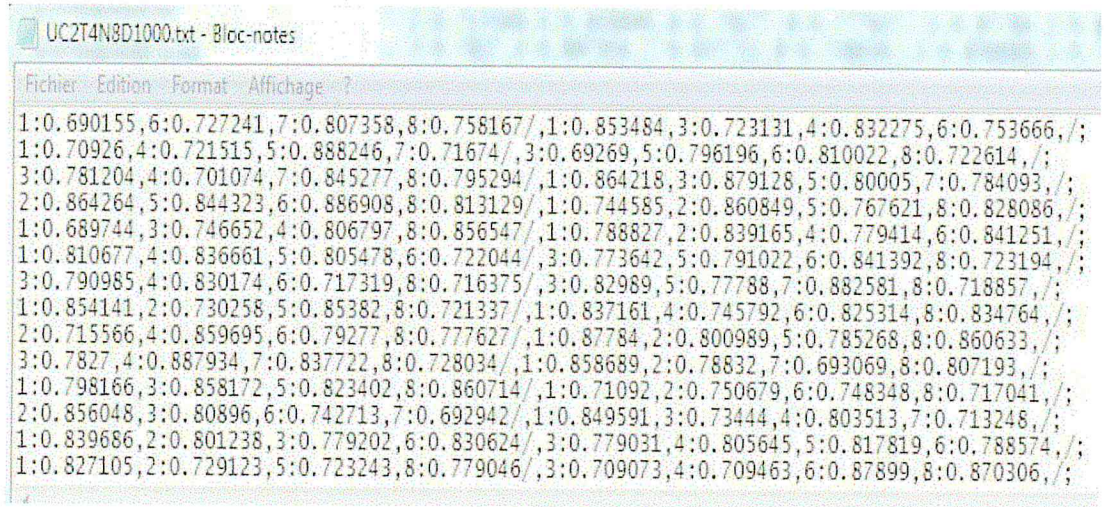

UC1T3N5D10000.txt - Bloc-notes

Fichier Edition Format Affichage ?

```
2:0.690155,4:0.727241,5:0.807358,/;
1:0.758167,3:0.853484,4:0.723131,/;
2:0.832275,3:0.753666,4:0.70926,/;
2:0.721515,3:0.888246,4:0.71674,/;
2:0.69269,3:0.796196,4:0.810022,/;
1:0.722614,4:0.781204,5:0.701074,/;
1:0.845277,2:0.795294,4:0.864218,/;
1:0.879128,2:0.80005,4:0.784093,/;
1:0.864264,2:0.844323,4:0.886908,/;
3:0.813129,4:0.744585,5:0.860849,/;
1:0.767621,3:0.828086,4:0.689744,/;
1:0.746652,2:0.806797,5:0.856547,/;
3:0.788827,4:0.839165,5:0.779414,/;
1:0.841251,3:0.810677,5:0.836661,/;
```

Figure 5.1 Benchmark 1 UC1T3N5D10000.

Dans ce Benchmark on trouve qu'il existe 10000 séquences, chaque séquence contient un seul item set, dans ce dernier il existe 3 items, les différents items qui existe dans ce Benchmark sont de 1 à 5.


UC2T4N8D1000.txt - Bloc-notes

Fichier Edition Format Affichage ?

```
1:0.690155,6:0.727241,7:0.807358,8:0.758167/,1:0.853484,3:0.723131,4:0.832275,6:0.753666,/;
1:0.70926,4:0.721515,5:0.888246,7:0.71674/,3:0.69269,5:0.796196,6:0.810022,8:0.722614,/;
3:0.781204,4:0.701074,7:0.845277,8:0.795294/,1:0.864218,3:0.879128,5:0.80005,7:0.784093,/;
2:0.864264,5:0.844323,6:0.886908,8:0.813129/,1:0.744585,2:0.860849,5:0.767621,8:0.828086,/;
1:0.689744,3:0.746652,4:0.806797,8:0.856547/,1:0.788827,2:0.839165,4:0.779414,6:0.841251,/;
1:0.810677,4:0.836661,5:0.805478,6:0.722044/,3:0.773642,5:0.791022,6:0.841392,8:0.723194,/;
3:0.790985,4:0.830174,6:0.717319,8:0.716375/,3:0.82989,5:0.77788,7:0.882581,8:0.718857,/;
1:0.854141,2:0.730258,5:0.85382,8:0.721337/,1:0.837161,4:0.745792,6:0.825314,8:0.834764,/;
2:0.715566,4:0.859695,6:0.79277,8:0.777627/,1:0.87784,2:0.800989,5:0.785268,8:0.860633,/;
3:0.7827,4:0.887934,7:0.837722,8:0.728034/,1:0.858689,2:0.78832,7:0.693069,8:0.807193,/;
1:0.798166,3:0.858172,5:0.823402,8:0.860714/,1:0.71092,2:0.750679,6:0.748348,8:0.717041,/;
2:0.856048,3:0.80896,6:0.742713,7:0.692942/,1:0.849591,3:0.73444,4:0.803513,7:0.713248,/;
1:0.839686,2:0.801238,3:0.779202,6:0.830624/,3:0.779031,4:0.805645,5:0.817819,6:0.788574,/;
1:0.827105,2:0.729123,5:0.723243,8:0.779046/,3:0.709073,4:0.709463,6:0.87899,8:0.870306,/;
```

Figure 5.2 Benchmark 2 UC2T4N8D10000.

Dans ce Benchmark il existe 10000 séquences, chaque séquence contient deux item sets, dans chaqu'un de ces item sets il existe 4 items, les différents items qui existe dans ce Benchmark sont de 1 à 8.

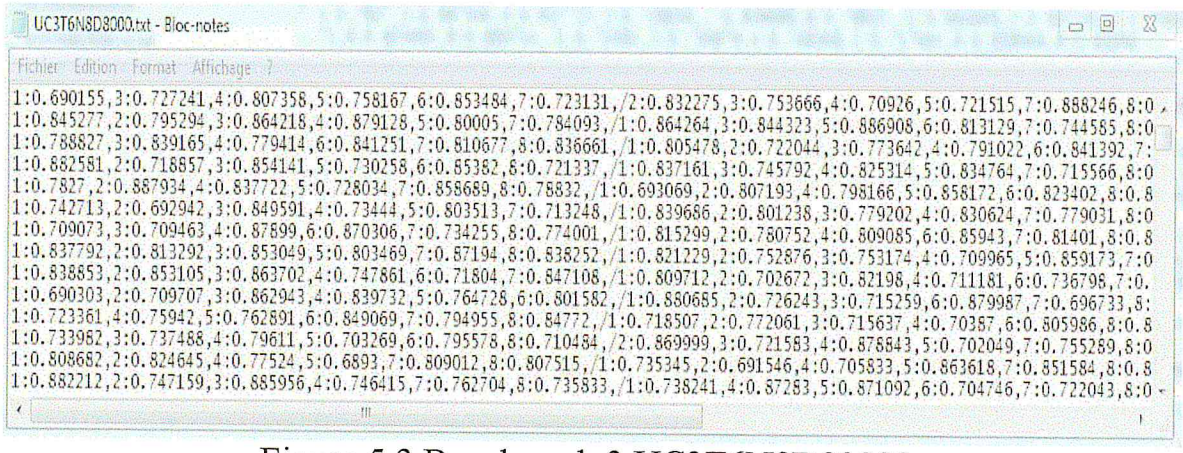


Figure 5.3 Benchmark 3 UC3T6N8D80000.

Dans ce Benchmark on trouve 80000 séquences, chaque séquence contient 3 item sets, dans chaqu'un d'eux il existe 6 items, les différents items qui existe dans ce Benchmark sont de 1 à 8.

4. Tests de validation

Nous avons validé notre algorithme avec des tests sur des benchmarks exposé dans la section précédente. Dans les tests suivants on va jouer sur les deux valeurs de MinSup et MaxIteration, et on compare leurs résultats avec le temps d'exécution qui est présenté par (ms) et par minute.

Première expérimentation

Au début, dans le premier test, nous avons fixé la valeur de MinSup qui est égale à 0 et nous avons joué sur la valeur de nombre des itérations les résultats sont montrer dans le tableau 5.1 le suivant :

MaxIteration	Temps d'exécution
1	13492
3	24766
5	39111
7	55540
9	51624
11	60377
13	64451

Tableau 5.1 : Résultat de test 1 avec le Benchmark 1 UC1T3N5D10000.

MinSup	Temps d'exécution
1	10381
5	10985
10	10854
15	9190
20	11355
25	13350
30	31174

Tableau 5.2 : Résultat de test 2 avec le Benchmark 1 UC1T3N5D10000.

Dans le tableau 5.2 nous avons fixé la valeur de Max Itération qui est égale à 1, et a chaque fois on a changé la valeur de Min Sup.

On va présenter les résultats de test 1 avec le Benchmark 1 UC1T3N5D10000 dans le graphe suivant.

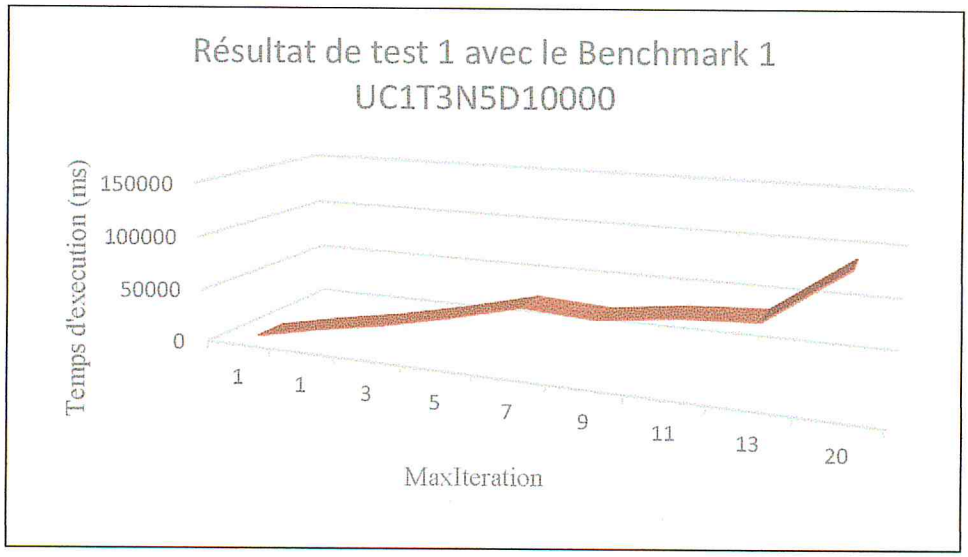


Figure 5.4 : Résultat de test 1 avec le Benchmark 1 UC1T3N5D10000.

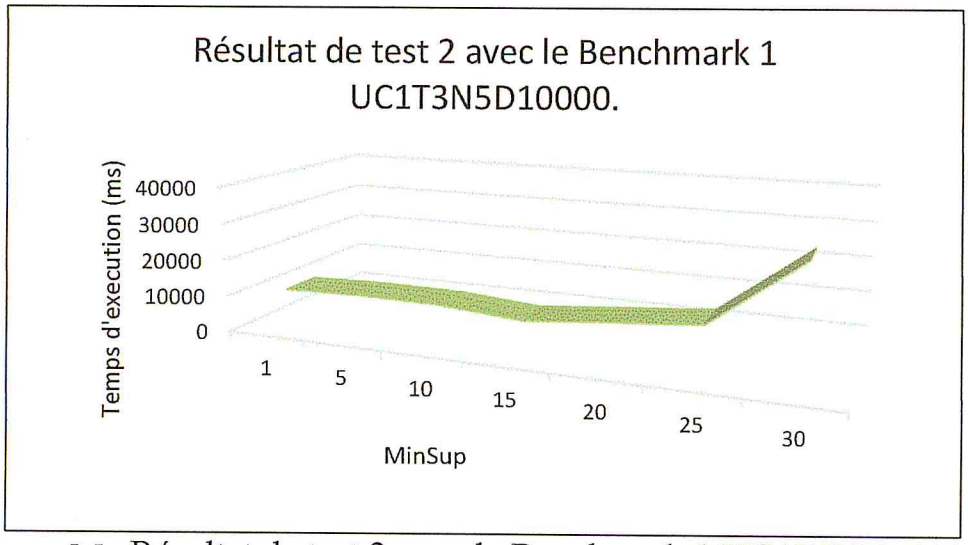


Figure 5.5 : Résultat de test 2 avec le Benchmark 1 UC1T3N5D10000.

Cette figure présente les résultats de test 2 qui sont présenté dans le tableau 5.2.

Dans le graphe de la figure 5.4 (les résultats de test 1 sur le Benchmark 1 UC1T3N5D10000), nous remarquant que, à chaque fois que nous augmentons la valeur de Max Itération le temps d'exécution augmente aussi.

Pour le graphe de la figure 5.5 (Résultat de test 2 sur le Benchmark 1 UC1T3N5D10000), nous remarquons, au début dans l'intervalle [1-15] le temps d'exécution reste stable, après dans l'intervalle [15-25] augmente progressivement, ensuite il augmente rapidement.

Deuxième expérimentation

Dans cette expérimentation nous allons agir sur le Benchmark 2 UC2T4N8D10000, c'est le même principe qu'on a fait dans la première expérimentation.

Le prochain tableau présente les résultats du test 1 :

Max Itération	Temps d'exécution (min)
1	102
3	153
5	196
7	205
9	271

Tableau 5.3 : Résultat de test 1 sur le Benchmark 2 UC2T4N8D10000.

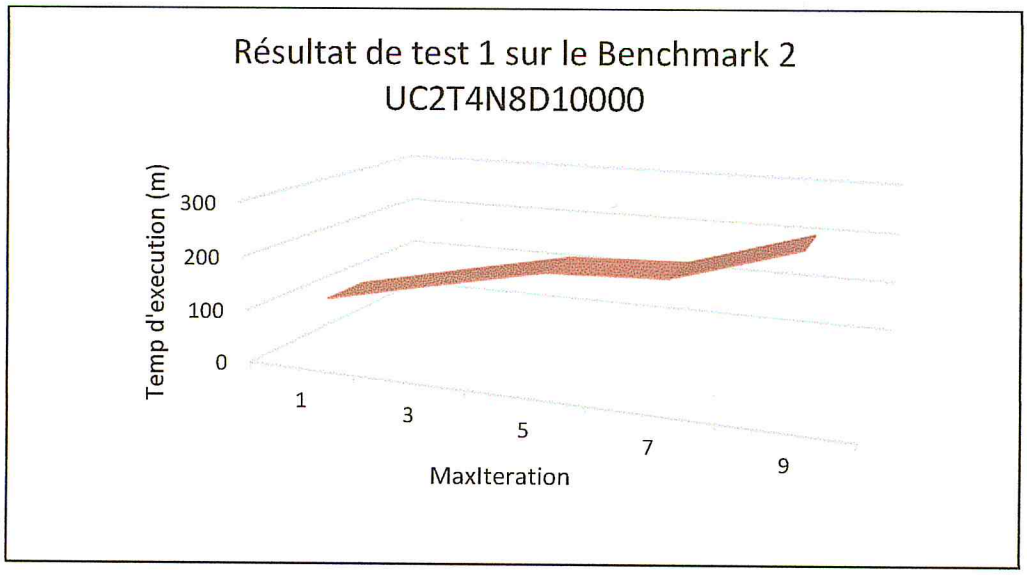


Figure 5.6 : Résultat de test 1 sur le Benchmark 2 UC2T4N8D10000.

MinSup	Temps d'exécution (min)
1	151
5	196
10	233
15	375
20	422

Tableau 5.4 : Résultat de test 1 sur le Benchmark 2 UC2T4N8D10000.

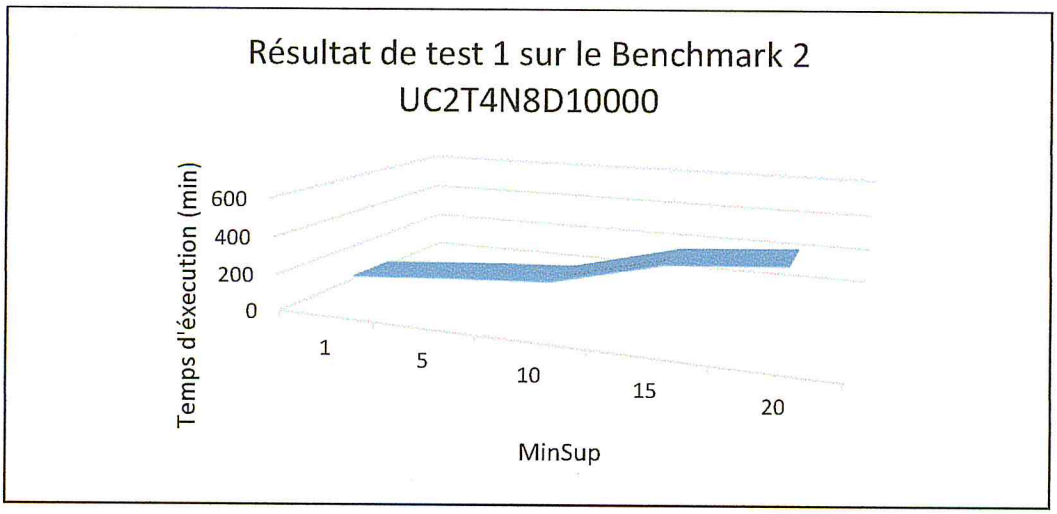


Figure 5.7 : Résultat de test 2 sur le Benchmark 2 UC2T4N8D10000

Dans la deuxième expérimentation menée, dans la Figure 5.6 : Résultat de test 1 sur le Benchmark 2 UC2T4N8D1000.0, on remarque que le temps d'exécution dans l'intervalle

[1-5] augmente progressivement ensuite il se stabilise dans l'intervalle [5-7] et enfin il augmente entre l'intervalle [7-9]

Dans la Figure 5.7 : Résultat de test 2 sur le Benchmark 2 UC2T4N8D10000, on remarque que le temps d'exécution dans l'intervalle [1-10] reste stable après dans l'intervalle [10-15] augmente et enfin il rentabilise

Troisième expérimentation

Max Itération	Temps d'exécution (min)
1	194
3	240
5	371
7	401

Tableau 5.5 : Résultat de test 1 sur le Benchmark 3 UC3T6N8D80000.

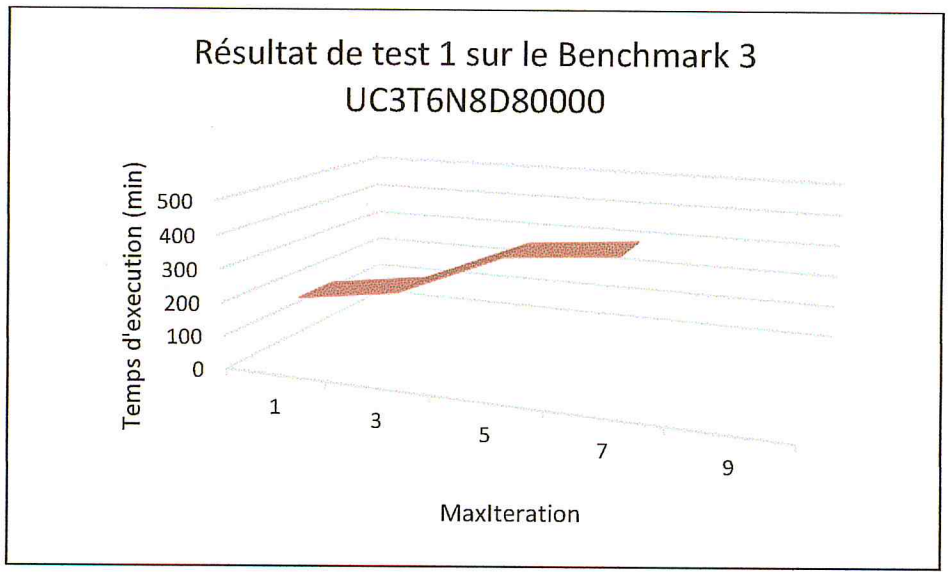


Figure 5.8 : Résultat de test 1 sur le Benchmark 3 UC3T6N8D80000

MinSup	Temps d'exécution (min)
1	200
5	241
10	293
15	377
20	489

Tableau 5.6 : Résultat de test 2 sur le Benchmark 3 UC3T6N8D80000.

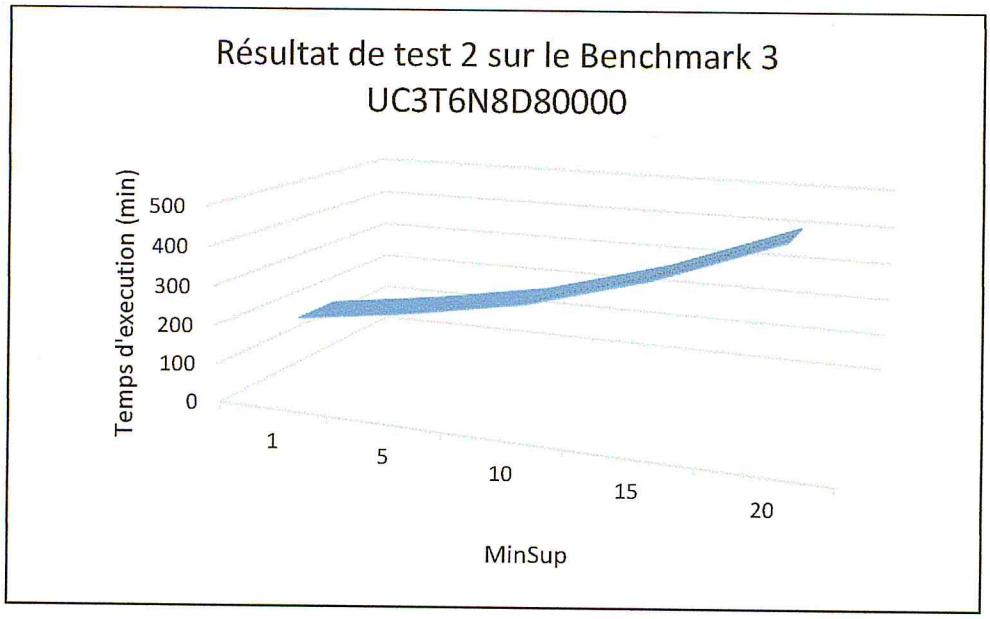


Figure 5.9 : Résultat de test 2 sur le Benchmark 3 UC3T6N8D80000

Dans la troisième expérimentation menée, dans la Figure 5.8 : Résultat de test 1 sur le Benchmark 3 UC3T6N8D80000, on remarque que les résultats de ce graphe correspondent aux résultat du premier graphe de la deuxième expérimentation.

Dans la Figure 5.9 : Résultat de test 2 sur le Benchmark 2 UC2T4N8D10000, on remarque que le temps d'exécution dans l'intervalle [1-10] reste stable après dans l'intervalle [10-20] augmente.

5. Conclusion

Dans ce dernier chapitre nous avons présenté l'implémentation de notre algorithme BSO-UFSPM parallèle avec MapReduce, nous avons validé notre implémentation avec une série de tests sur un PC portable, puis nous avons interprété les résultats expérimentaux obtenus.

Nous arrivons à dire que Hadoop est un outil puissant pour faire des calculs simples d'immense données pour les entreprises mais pour les grands calculs scientifiques il n'est pas vraiment adapté car il est d'une architecture complexe et lourdes pas facile à la modifier suivant nos contraints. Il lui manque des outils pour supporter le calcul parallèle intensif.

CONCLUSION GÉNÉRALE

CONCLUSION GÉNÉRALE

Après l'engouement des travaux de recherche pour les règles d'association, les motifs séquentiels ont été très étudiés ces dernières années. Initialement, les premiers travaux ont consisté à améliorer les performances des propositions. Dans ce cadre, de nouvelles structures, représentations ou de nouvelles grandes et immenses bases de données que ça soit certaines ou bien incertaines ont été mises au point.

Même si les propositions ont permis de considérablement améliorer les temps de réponse, un premier constat a émergé.

Cette thèse intègre la mise au point de nouveaux dispositifs, de nouvelles technologies, de nouveaux algorithmes, méthodes, outils pour la visualisation et l'interaction avec des données de grand volume.

Concluons ce mémoire avec un résumé de notre travail de recherche et de nos principales contributions.

Dans ce mémoire, le travail qui nous a été confié consistait à concevoir et à réaliser un algorithme d'extraction de motifs séquentiels Fréquents à partir des données incertaines dans le contexte de big data.

Nous avons vu le premier chapitre, les différentes notions sur les technologies de big data, avec un algorithme qui existe réellement et utilisent le Hadoop.

Dans le deuxième chapitre de l'état de l'art, nous avons vu les différentes définitions et algorithmes qui existent dans le cadre d'extraction de motifs séquentiels à partir des données incertaines, 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. Ainsi, les algorithmes du type GSP seront très efficaces dans le cas de grandes bases de données avec des séquences moyennement longues.

Par contre, les algorithmes comme SPADE, SPAM ou PREFIXSPAN seront eux très efficaces dans le cas où un très grand nombre de candidats de même taille sont générés.

Dans le chapitre 2, nous avons présenté les différents algorithmes concernant les métas heuristiques.

Le troisième chapitre, propose notre méthode d'extraction de motifs séquentiels (ou bien algorithme) qui passe par 3 étapes principales.

CONCLUSION GÉNÉRALE

Nous nous sommes attaqués à notre problème de deux façons (cas) différentes : cas séquentielles et parallèles

Dans les deux cas, c'est le même principe et les mêmes étapes qui sont utilisé, la différence entre eux c'est le temps, il est primordial dans le cas parallèle dans les big data.

Nous avons prouvé la capacité de notre méthode par des évaluations (tant théoriques que pratiques) tout au long du mémoire.

Limites et Perspectives

Les perspectives de ce travail préliminaire sont nombreuses. Tout d'abord, nous devons valider la démarche proposée avec des bases de données réelles.

Nous pouvons prendre le problème dans l'autre sens : ne plus extraire les séquences fréquentes mais extraire des graphes dans le cas des big data avec les données incertaines.

Il est probablement intéressant de pouvoir créer des méthodes qui lance plusieurs machines sur plusieurs bases de données au même temps pour extraire des séquences fréquentes à partir des données incertaines dans le contexte de big data, c'est-à-dire de pouvoir réalisé une méthode qui peut lancer les machines en parallèles au niveau des données.

Les principales difficultés liées au traitement de l'extraction des séquences fréquentes à partir des données incertaines dans le contexte de big data sont au niveau de programmation avec la nouvelle technologie Haddop avec des bases de données incertaines.

Cependant, l'algorithme que nous propose est implémenté en Java aves la technologie de hadoop en utilisant les deux fonctions principales dans les big data : la fonction Map et la fonction Reduce pour évaluer la qualité de notre espace de recherche avec peu de temps.

Enfin le programme réalisé n'est pas encore optimisé au niveau des bases de données réelles sur le cluster IBN BADIS.

BIBLIOGRAPHIE

[1] From "The Big Data Long Tail" blog post by Jason Bloomberg (Jan 17, 2013).

[2] Ed Dumbill, program chair for the O'Reilly Strata Conference

[3] Source: wikipedia, op. cit.

[4] Cf. par exemple: The Great Disk Drive in the Sky: How Web giants store big—and we mean big—data. Ars Technica, janvier 2012.

[5] Guide de Big Data. 2013, 2014. pp.6

[6] Manyika, J., Chui,. M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H. (2011, June). Big data: The next frontier for innovation, competition, and productivity.

[7] R. Agrawal et R. Srikant: Mining sequential patterns. In Philip S. Yu et Arbee L. P. Chen, éditeurs : Proceedings of the Eleventh International Conference on Data Engineering (ICDE'95), March 6-10, 1995, Taipei, Taiwan, pages 3–14. IEEE Computer Society, 1995.

[8] Han J., KamberM., Data Mining : Concepts and Techniques, Morgan Kaufmann, septembre 2000.

[9] D. A. Zighed, Y. Kodratoff, and A Napoli, "Extraction de connaissance à partir d'une base de donnée," Bulletin AFIA'01, 2001.

[10] Y. Kodratoff, "techniques et outils de l'extraction de connaissances à partir des données," Signaux, vol. 92, pp 38–43, Mars 1998.

[11] Professeur Dr. Andreas Meier et Assistant Darius Zumstein. Le CRM analytique : Les outils d'analyse OLAP et le Data Mining, Dans le cadre du séminaire « Customer Relationship Management », Fribourg, le 26 avril 2008.

[12] M. Kantardzic, "Data Mining–Concepts, Models, Methods,and Algorithms,"IEEE Press, Piscataway, NJ, USA, 2003.

BIBLIOGRAPHIE

[13] S. Tufféry, Data mining et statistique décisionnelle, l'intelligence dans les bases de données, Groupe bancaire Français, Universités de Rennes 1 et paris- Dauphine, 2005.

[14] W. J. Frawley, G. Piatetsky-Shapiro and C. J. Matheus. Knowledge discovery in data bases: an overview. AI Magazine, vol.13, p.57-70, 1992.

[15] F. R. GREYC. Fouille de données orientée motifs, méthodes et usages. Université de Caen Basse-Normandie France.

[16] N.Pasquier. Data Mining : algorithmes d'extraction et de réduction des règles d'association dans les bases de données. Université Blaise Pascal - ClermontFerrand II, 2000. Français.

[17] W.J Frawley and G. Piatestsky-Shapiro and C. J. Matheus, Knowledge discovery in data base –an overview, Al Magazine, page 57-70, 1992.

[18] Sandy Maumus, Amedeo Napoli, Laszlo Szathmary, Yannick Toussaint; Réflexions sur l'extraction de motifs rares.

[19] Jiawei Han, Micheline Kamber, and Jian Pei University of Illinois at Urbana-Champaign & Simon Fraser University ©2013 Han, Kamber & Pei. All rights reserved; Data Mining: Concepts and Techniques (3rd ed.).

[20] Jonathan Marchand, Bruno Guillaume, Guy Perrier, Motifs de graphe pour le calcul de dépendances syntaxiques complètes, TALN 2010, Montréal, 19–23 juillet 2010

[21] Claude Pasquier, Jérémy Sanhes, Frédéric Flouvat , Nazha Selmaoui-Folcher, Extraction de motifs fréquents dans des arbres attribués, *Université de Nouvelle Calédonie PPME, BP R4, F-98851 Nouméa, Nouvelle Caledonie.

[22] R. Srikant et R. Agrawal : Mining sequential patterns : Generalizations and performance improvements. In Peter Apers, Mokrane Bouzeghoub 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.

[23] Simon Jaillet, Anne Laurent et Maguelonne Teisseire: Sequential patterns for text catégorization. Intell. Data Anal., 10(3):199–214, 2006.

[24] Agrawal R., Imielinski T., Swami A., « Mining Association Rules between sets of Items in Large Databases », Proc. ACM SIGMOD Int'l Conf. on Management of Data, mai 1993, p. 207-216.

[25] Z.Zhao, D.Yan and Wilfred. Mining Probabilistically Frequent Sequential Patterns in Uncertain Databases Ng *Department of Computer Science and Engineering*. The Hong Kong University of Science and Technology Hong Kong

[26] R. Srikant and R. Agrawal. "Mining Sequential Patterns: Generalizations and Performance Improvements". In EDBT, 1996.

[27] M. J. Zaki. "SPADE: An Efficient Algorithm for Mining Frequent Sequences". In Machine Learning, 42(1/2):31-60, 2001.

[28] Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal and M. C. Hsu. "PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth". In ICDE, 2001

[29] C.Tambellini, C.Berrut. Pondération des données incertaines dans les systèmes de recherche d'informations : une première approche expérimentale. Laboratoire CLIPS-IMAG. BP 53 38041 Grenoble cedex 9

[30] Sun X, Orlowska ME, Li X (2003) Introducing uncertainty into pattern discovery in temporal event sequences. In: ICDM. IEEE Computer Society, pp 299–306. ISBN 0-7695-1978-4

[31] Bernecker T, Kriegel H-P, RenzM, Verhein F, Züfle A. Probabilistic frequent itemset mining in uncertain databases. In: Elder et al. [32], pp 119–128. ISBN 978-1-60558-499

[32] Khoussainova, N., Balazinska, M., Suciu, D.: Probabilistic event extraction from RFID data. In: ICDE. pp. 1480–1482. IEEE (2008).

BIBLIOGRAPHIE

[33] M.Muzammal and R.Raman. On Probabilistic Models for Uncertain Sequential Pattern Mining. pp.4 -6.

[34] M.Muzammal. Mining sequential patterns from probabilistic databases, London 2014

[35] M.Muzammal, R. Raman.On Probabilistic Models for Uncertain Sequential Pattern Mining.

[36] J. Ge1(B), Y.Xia1, and J.Wang2. Mining Uncertain Sequential Patterns in Iterative MapReduce. Department of Computer and Information Science, Indiana University Purdue. University Indianapolis, Indianapolis, IN 46202, USA

[37] A. Gherboudj. Méthodes de résolution de problèmes difficiles académiques. 2012-2013. pp. 46-47

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

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

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

[41] J.-C, Boisson. Algorithmes génétiques et calcul haute performance. CReSTIC-SysCom – EA 3804pp 6-8

[42] Stéphane Gerbex, "Métaheuristique appliquées au placement optimal des dispositif FACTS dans un réseau électrique," Ecole Polytechnique Federal de Laussane Thèse de Doctorat Es Sciences Technique 2742, 2003.

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

BIBLIOGRAPHIE

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

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

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

[47] D.T. Pham, A. Ghanbarzadeh, E. KoÃğ, S. Otri, S. Rahim, and Zaidi M. The bees algorithm, a novel tool for complex optimization problems. Intelligent Production Machines and Systems, pages 454–459, 2006.

[48] J.H. Holland. Genetic Algorithms and the optimal allocation of trials, SIAM Journal of Computing. Vol. 2, N° 2, pp. 88-105, 1973.

[49] S.MOUASSA. Optimisation de l'écoulement de puissance par une méthode métaheuristique (technique des abeilles) en présence d'une source renouvelable (éolienne) et des dispositifs FACTS.

WEBOGRAPHIE

[w1] http://www.gartner.com/it-glossary/big-data/

[w2] http://www.journaldunet.com/solutions/expert/51696/les-3-v-du-big-data---volume--vitesse-et-variete.shtml

[w3] http://insidebigdata.com/2013/09/12/beyond-volume-variety-velocity-issue-big-data--veracity

[w4] http://lebigdata.com/fr/les-3-v-du-big-data/

[w5] http://www.conseilwebsocial.com/index.php/2013/reseau-social-et-big-data/

[w6] Big Data Analytics. Prof. Dr. Lars Schmidt-Thieme docplayer.net819 × 608Recherche par image What is Big Data? Big Data Dimensions (the 4 Vs) 33

[12] http://www.piloter.org/business-intelligence/map-reduce.html

[w7] http://mon.univ-montp2.fr/claroline/backends/download.php

[w8] http://docplayer.fr/803852-Hadoop-une-plate-forme-d-execution-de-programme-mapreduce.html

[w9] http://www.piloter.org/business-intelligence/hadoop.html

[w10] http://hadoop.apache.org/index.pdf

[w11] http://www.informit.com/articles/article.aspx

[w12] http://www.intel.fr/content/dam/www/public/emea/fr/fr/documents/guides/bigdata/plannigguide-fr.pdf

[w13] http://en.wikipedia.org/wiki/NoSQL

[w14] http://www.digora.com/fr/blog/definition-base-nosql-datastax-mongodb

[w15] http://www.lebigdata.fr/definition-big-data

[w16] https://www.infoq.com/fr/articles/apache-spark-introduction

[w17] https://extractiondesconnaissances.files.wordpress.com/2012/03/schema.png

[w18] www.abdelhamid-djeffal.net

[w19] https://fr.wikipedia.org/wiki/Donn%C3%A9e_(informatique)

[w21] http://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=2678&context=cstech

[w20] http://Mining-Sequential-Patterns-from-Probabilistic.pdf

[52] Wikipedia. http://en.wikipedia.org/wiki/anpr—Wikipedia, the free encyclopedia, 2010. URL http://en.wikipedia.org/wiki/ANPR. [Online; accessed 30-April-2012

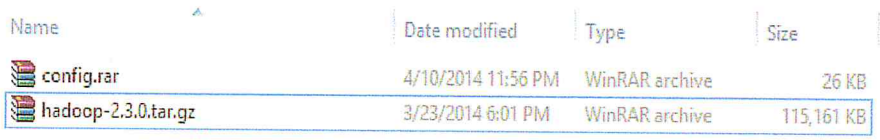
ANNEXE

Les étapes d'installation de Hadoop

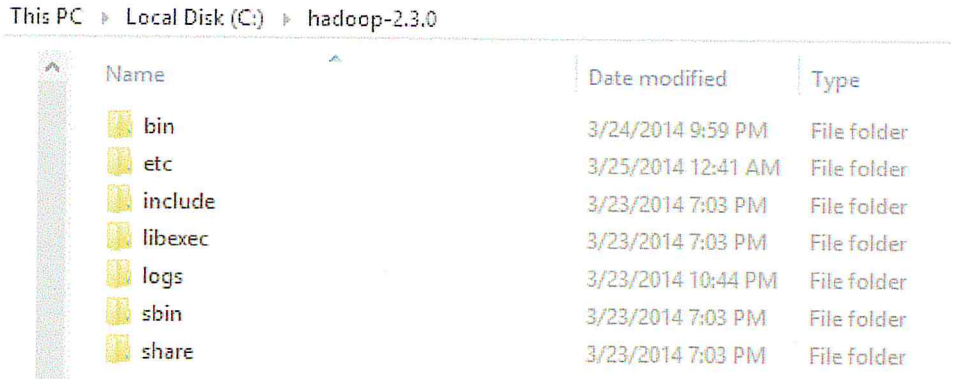
On a utilisé la version Hadoop 2.7.1 spécialement construit pour Windows 7, 64 bit.

La première étape

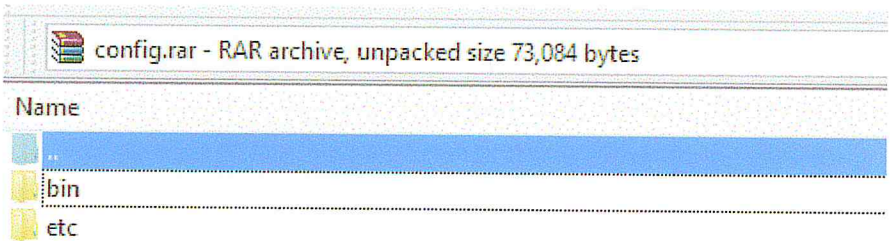
Au début, nous avons Téléchargé Hadoop 2.7.1 à partir du site web le suivant https://app.box.com/s/11fwozokqmc1ohttt117, Puits on a Télécharger un fichier winrar de configuration à partir du site web :https://github.com/prabaprakash/Hadoop-2.3-Config/archive/master.zip, le nom du fichier est : *2.3-Config/archive/master.zip*, l'affichage de ce téléchargements est présenté dans la figure suivante :



Dans l'ouverture du dossier *hadoop-2.3.0.tar.gz* avec winrar, dans le dossier de téléchargement, on trouve un ensemble de dossiers qui soit présenté dans la figure suivante :



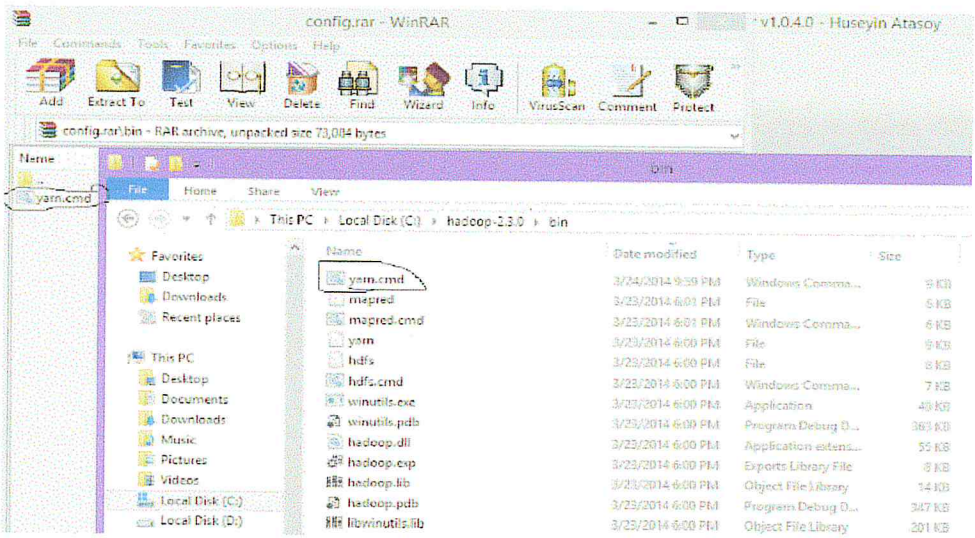
Dans l'ouverture du dossier *config.rar* à partir de winrar :



ANNEXE

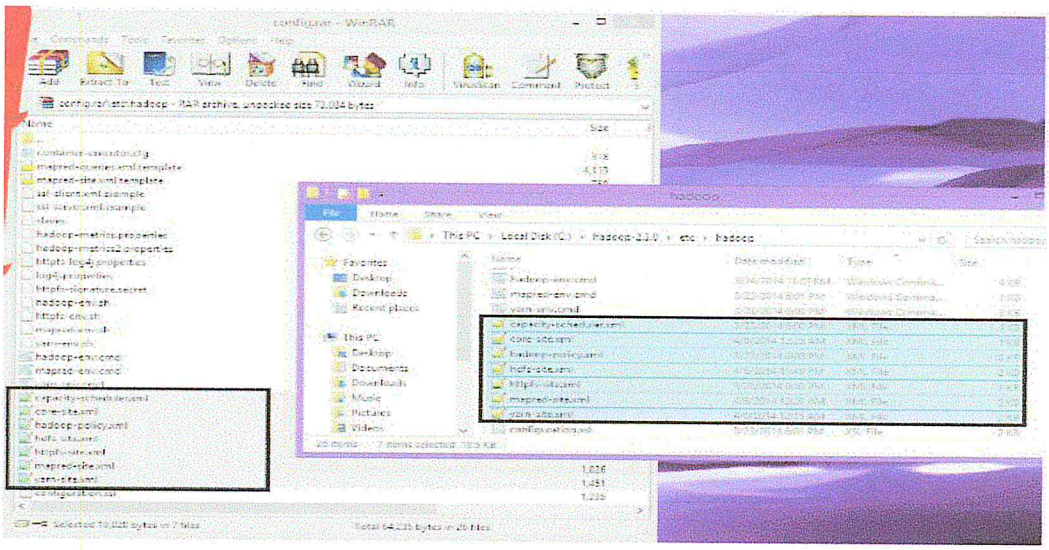
Après on passe par les étapes suivantes :

- Ouvrir le répertoire *bin* dans winrar. Ensuite on copie le fichier *yarn.cmd* dans le chemin *c:\\hadoop-2.3.0 dossier \ bin*



- Ouvrir le dossier *config\etc\hadoop,* puits les fichiers suivants :
    1. yarn-site.xml
    2. mapred.xml
    3. https-site.xml
    4. hdfs-site.xml
    5. hadoop-policy.xml
    6. core-site.xml
    7. capacity-scheduler.xml
- Ouvrir le chemin *c:\\ hadoop-2.3.0 \ etc \ hadoop* et on remplace les fichiers qui sont dans le chemin *c:\\ hadoop-2.3.0 \ etc \ hadoop* par les fichiers précédent.

ANNEXE



La deuxième étape

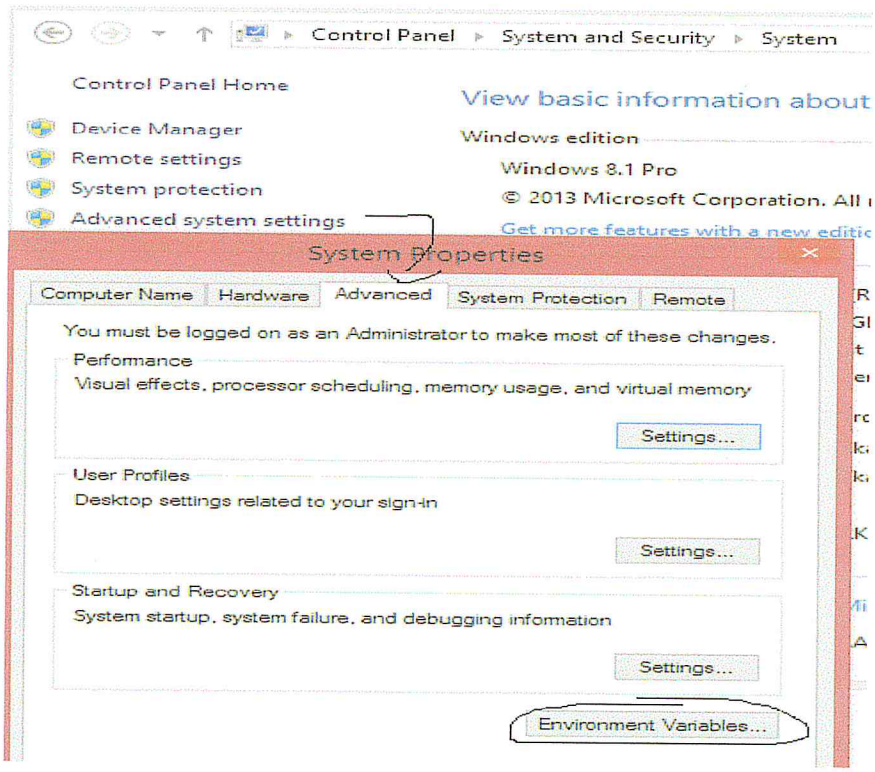
Pour installer java 1.8

1. Télécharger Java SDK 1.8.0_31 à partir du site web le suivant et l'installer http://download.oracle.com/otn/java/jdk/6u31-b05/jdk-6u31-windows-x64.exe

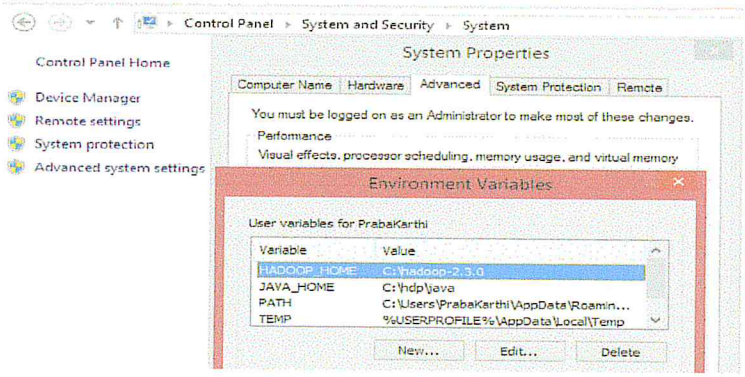
2. Définissez les variables d'environnement

- Control Panel \ Système et sécurité \ System
- Paramètres système Advanced Open

ANNEXE

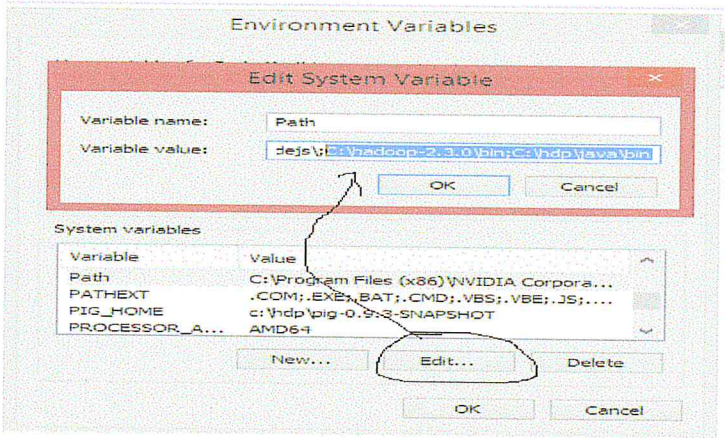


- ajouter le nouveau "HADOOP_HOME" variable - valeur c : \\hadoop-2.3.0 et la nouvelle variable «JAVA_HOME" – valeur.

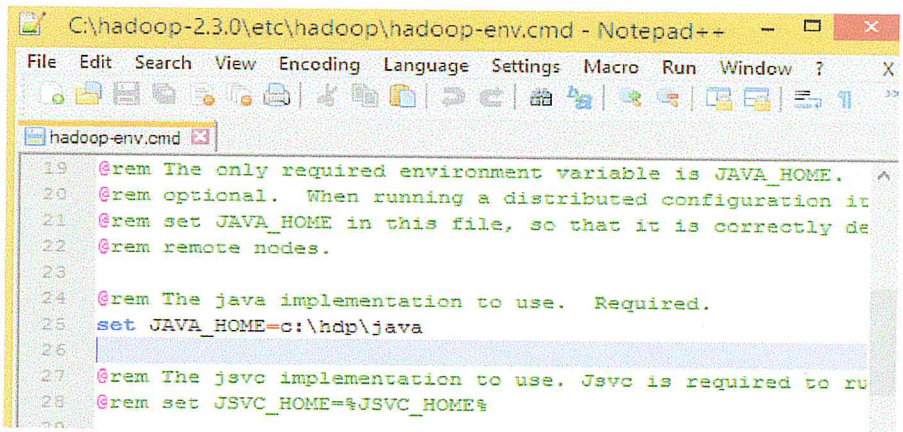


- Variables système -> Path -> Modifier
- Ajouter Hadoop chemin bin, Java 8 chemin bin -> cliquez sur ok

ANNEXE



- Ensuite ouvrir *hadoop-env.cmd* dans WordPad situé dans *C : \\ hadoop-2.3.0 \ etc \ hadoop \ hadoop-env.cmd*
- Définissez le chemin JAVA_HOME dans la ligne 25.



- Laissez Jouer avec Apache Hadoop 2.7 et ouvrir cmd en tant qu'administrateur
- Création de HDFS dans le système :

ANNEXE

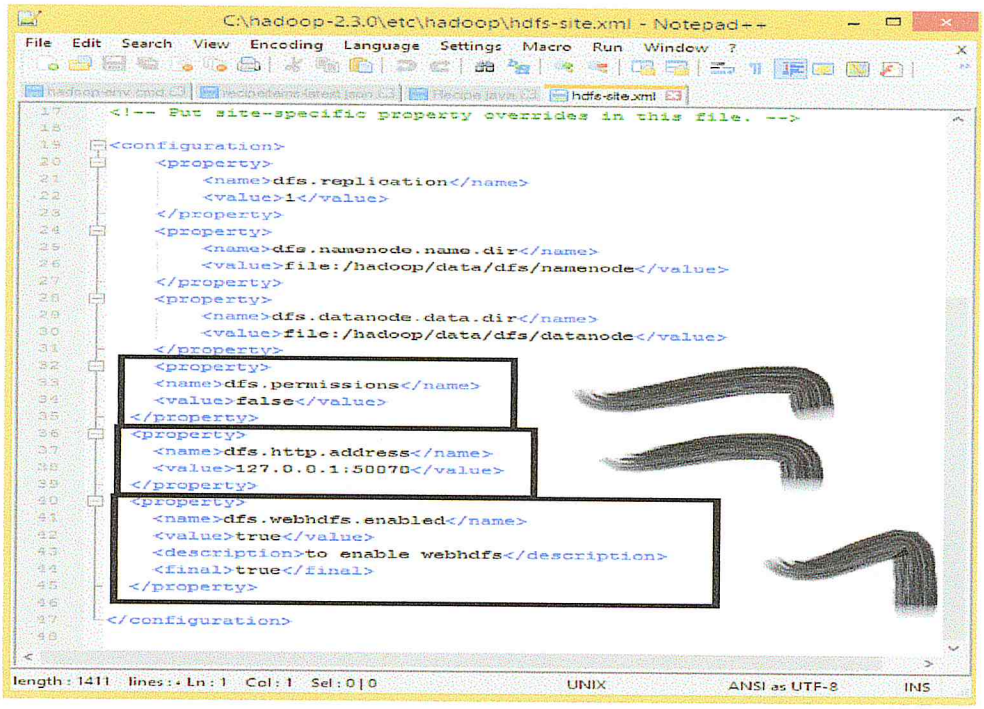
A la fin de ces étapes en vérifiez si Apache NameNode et DataNode, Apache File NodeManager et File Resource Manager sont exécuté en même temps.

La troisième étape

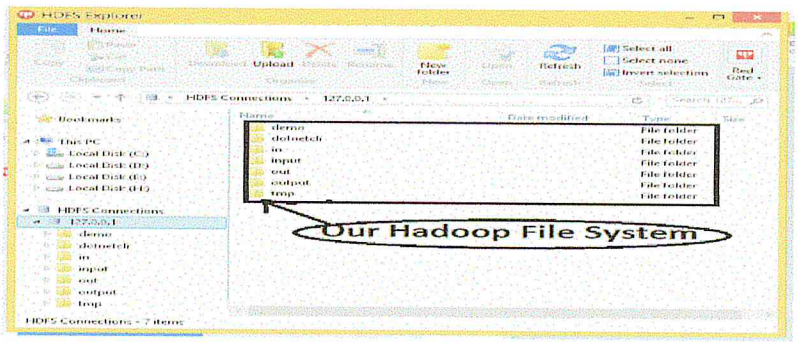
En utilise une Commande spécial pour copier les fichiers qui se trouve dans HDFS, après télécharger le dossier qui se trouve dans le site suivant (2,5 Mo)
http://bigdatainstallers.azurewebsites.net/files/HDFS%20Explorer/beta/1/HDFS%20Explorer%20-%20beta.application

1. Installez-le, nous avons déjà copié les fichiers de configuration pour Hadoop 2.7, notre HDFS sera accessible à distance, en utilisant également web client en Java

ANNEXE



2. Ouvrir HDFS Explorer et ajouter File-> Connection
3. Parcourir le système Hadoop de fichier dans l'explorateur de fichiers graphique.
4. Copiez le fichier d'entrée à partir du disque local et le coller dans HDFS, également
5. copier la forme de sortie HDFS et le coller dans votre disque local.



La quatrième étape

Dans cette étape en fait l'installation d'Eclipse Plugin pour Hadoop MapReduce Jobs avec Simple HDFS Explorer

ANNEXE

En télécharge Eclipse à partir du site suivant http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/kepler/SR2/eclipse-jee-kepler-SR2-win32-x86_64.zip

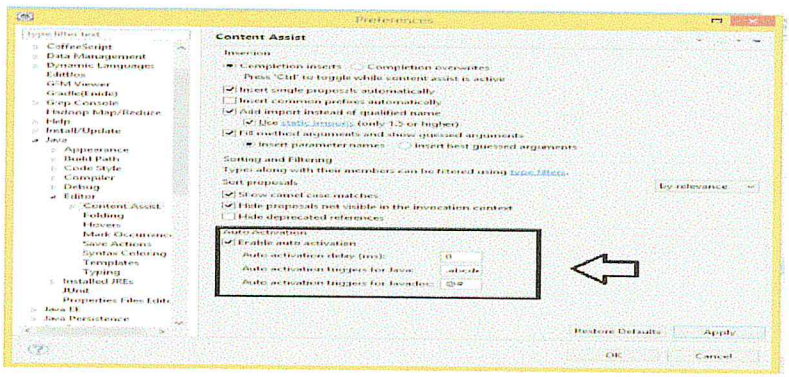
Et en télécharge aussi le Hadoop MapReduce Plugin pour Eclipse (23 MB) à partir de :

https://github.com/winghc/hadoop2x-eclipse-plugin/archive/master.zip

2. Ouvrir le hadoop2x-*eclipse-plugin-master.zip* :

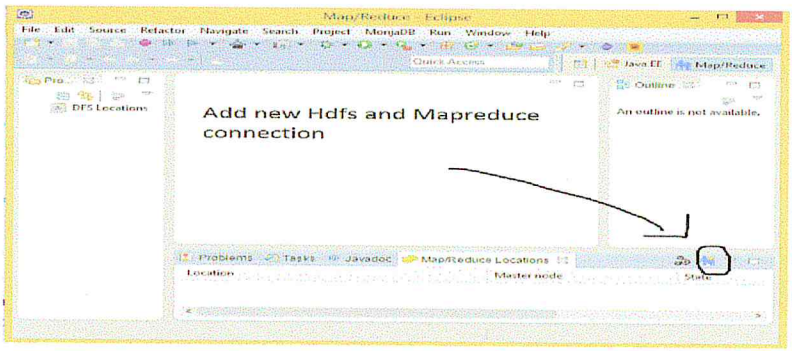
3. Ouvrir Eclipse IDE (Exécuter en tant qu'administrateur)

4. Choisissez votre propre WorkPlace Situation -> Cliquez sur OK
5. Menu -> Window -> Ouvrir la perspective -> Autre ->Map/Reduce.
6. Menu-> Window-> Preference-> Java-> Editor-> Contenu Assistent -> "Activation automatique"
    i. vérifier la permutation de l'activation automatique.
    ii. retard d'activation automatique (ms) : 0.
    iii. déclenchement d'activation automatique pour java
    iv. activation automatique pour java doc :
    v. Appliquer-> Ok

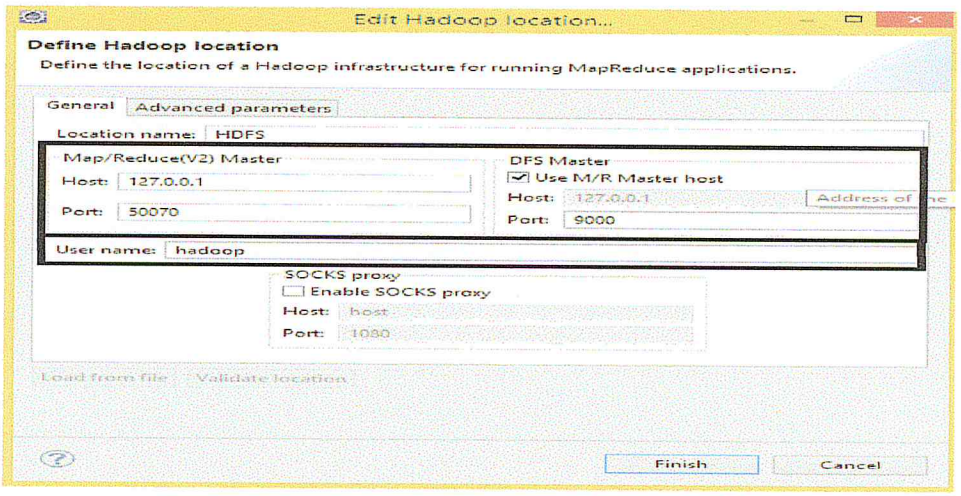


7. Configuré la connexion de HDFS et MapReduce : Map/reduce location->new hadoop location

ANNEXE

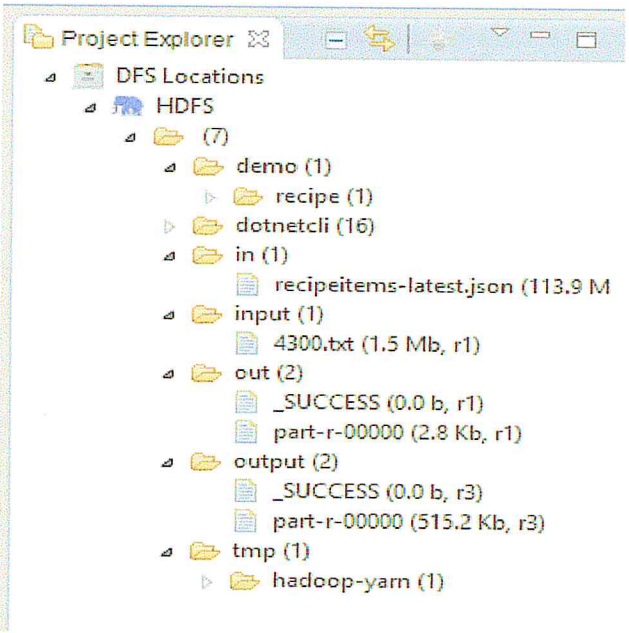


8. Après avoir transité vers la perspective Map/Reduce. Sélectionnez l'onglet d'emplacements de Map/Reduce, situé à la partie inférieure de votre environnement Eclipse. Faites un clic droit sur l'espace vide dans cet onglet, puis sélectionnez « Nouvel emplacement Hadoop... » dans le menu contextuel. Vous verrez la boîte de dialogue similaire à celui illustré ci-dessous.



Exemple de HDFS :

ANNEXE



La cinquième étape

Dans cette partie en créer d'un projet Map/Reduce sous Eclipse

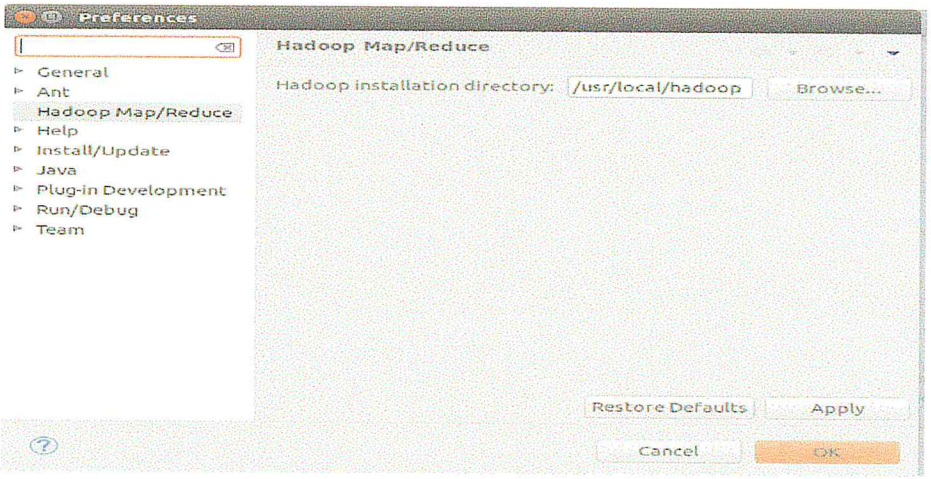
1. Clic droit sur "Window" dans Explorateur de projets, puis sélectionnez New-> Projet... pour créer un nouveau projet.

2. Sélectionnez le projet de Map/Reduce dans la liste des types de projets comme le montre la figure ci-dessous.

3. Appuyez sur le bouton suivant, la fenêtre de propriétés de projet s'affichera.

4. Cliquez sur " configure Hadoop install directory ... «, dans la fenêtre de préférences de projet, entrez l'emplacement du répertoire Hadoop dans le champ Répertoire

ANNEXE



5. Après avoir entré l'emplacement, fermez la fenêtre de préférences en appuyant sur le bouton OK. Puis fermez la fenêtre de projet avec le bouton Terminer.

6. Vous avez maintenant créé votre premier projet Hadoop sous Eclipse, dont le nom s'affichera dans l'onglet Explorateur de projets.

Création de classes Map/Reduce

1. Clic droit sur le projet Hadoop nouvellement créé dans l'onglet Explorateur de projets, puis sélectionnez New-> autres dans le menu contextuel.

2. Allez dans le dossier de Map/Reduce, sélectionnez MapReduceDriver, puis appuyez sur le bouton suivant.

3. Lorsque l'Assistant de MapReduce s'affiche, entrez un nom pour votre classe et appuyez sur le bouton Terminer.

4. De même pour les 2 classes "Map" et "Reduce".

Exécution d'un exemple Map/Reduce : "Word-Count" sous Eclipse

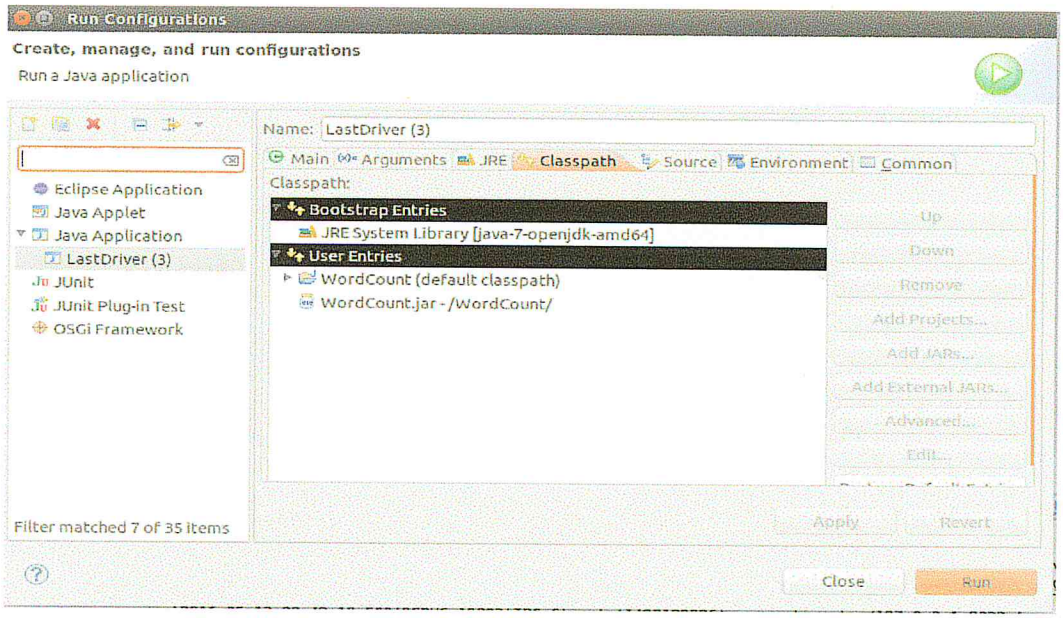
1. Copier le code du WordCount fourni par le tutoriel d'Apache ou une autre source.

2. Rajouter dans le dossier " src " de votre projet : les fichiers "core-site.xml","hdfssite. xml"," log4j.propreties " et " pom.xml ".

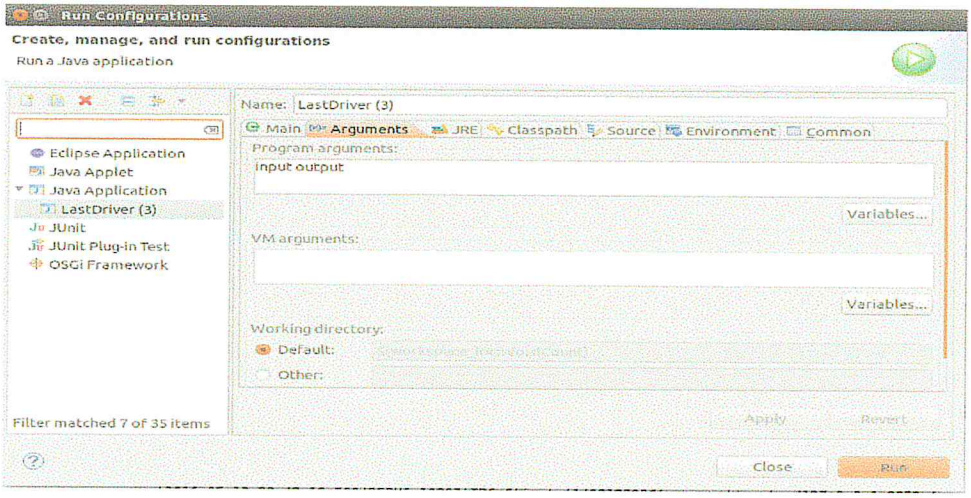
3. Exporter le projet sous forme de .jar, File => Export.

4. Rajouter le chemin vers le .jar dans le Classpath, Run => Run Configurations, la fenêtre suivante s'affiche puis " Add Jars ", indiquer le chemin du jar.

ANNEXE



5. Dans la même fenêtre, cliquez sur l'ongle "Arguments" spécifié le dossier d'entrée "Input" contenant les données et le nom du dossier de sortie pour le résultat dans le HDFS. Comme la montre la figure ci-dessous, puis "Run" exécution.



6. Enfin l'exécution, le résultat sera stocké sur le HDFS dans le dossier donné comme Argument, comme illustré par la figure.