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

Université Abderrahmane Mira de Bejaia
Faculté des Sciences Exactes

Département d'Informatique

ECOLE DOCTORALE RESEAUX ET SYSTEMES DISTRIBUES

Mémoire de Magistère

En Informatique

Option : Réseaux et Systèmes Distribués

Thème

Proposition d'une Approche de Data Mining Distribuée Basée sur OPTICS

Présenté par

Souhila Ghanem

Devant le jury composé de :

Président	DAHMANI Abdelnasser	Professeur	Université de Bejaia
Rapporteur 1	KECHADI Mohand-Tahar	Professeur	University College Dublin
Rapporteur 2	TARI Abdelkamel	M. C. A	Université de Bejaia
Examinatrice1	BENATCHBA Karima	Professeur	ESI, Alger
Examinatrice2	NADER Fahima	M. C. A	ESI, Alger
Invité	DAHAMNI Foudil	M. A. A	ESI, Alger

Promotion 2008-2009

Remerciements

En premier lieu, je remercie le Dieu, de m'avoir donné la force et la patience pour pouvoir mener ce travail à terme.

Je tiens à exprimer ma profonde gratitude et mes vifs remerciements à mes encadrants, Pr M.Tahar KECHADI et Dr A.Kamel TARI, qui m'auront accueilli, accompagné et conseillé tout au long de ce parcours. Mon envie de persévérer dans l'enseignement et la recherche doit beaucoup à l'énergie qu'ils savent mettre quotidiennement dans ces deux tâches.

Je remercie sincèrement les membres de jury qui m'ont fait l'honneur de juger ce travail.

Mes remerciements les plus chaleureux vont vers toute ma famille en général, vers mes parents en particulier, qui me soutiennent depuis toujours, ainsi que vers toutes mes sœurs pour la patience dont elles ont fait part à mon égard.

Je ne pourrai oublier d'adresser ma reconnaissance, particulièrement, à Khlifa Hakim, Karima et Hamid pour leurs aides

Et un merci tout particulier à tous ceux qui mon apporté leur soutien.

Dédicaces

Je dédie ce travail à tous ceux qui sont chers à moi

A mes parents pour leur amour, leur soutien morale, et leur patience.

A mes très chères sœurs pour leur encouragement et leurs compréhension

A mes très chers frères pour leur aide et soutien

Aux maris de mes soeurs pour leur encouragement et leur aide.

A mes deux petites-nièces Marya et Sarah que j'aime beaucoup.

A mon adorable neveu Aimed.

A tous mes oncles et tentes

A tous mes cousins et cousines

A tous mes amis

Résumé

Les collections de données deviennent de plus en plus volumineuses et dans la majorité des cas ne résident pas dans un emplacement centralisé, ce qui complique l'application des techniques de Data Mining sur des données distribuées et souvent hétérogènes. La plupart des algorithmes distribués se basent sur l'agrégation des modèles produits de manière locale et notre approche rentre dans ce cadre. Nous présentons une nouvelle approche distribuée qui prend en compte certaines caractéristiques de l'algorithme OPTICS. Les données seront traitées localement sur chaque site pour produire des clusters à partir des données locales, ensuite nous construisons les clusters globaux de manière hiérarchique. Le but de cette approche est de minimiser les communications et maximiser le parallélisme. Cette technique est évaluée et comparée à la version séquentielle.

Mots clés: Data Mining, Data Mining distribué, clustering, OPTICS.

Abstract

The data collections are becoming more and more larger and in most cases do not reside in a centralized location. The latter complicates the application of traditional data mining techniques on the data sets, which are distributed and often heterogeneous. Most distributed algorithms are based on the aggregation of models produced locally. The approach we are proposing belongs to this category. Our approach is fully distributed and it takes into account certain characteristics of the OPTICS algorithm. The data will be processed locally on each node to produce clusters from local data, then we construct the global clusters hierarchically. The aim of this approach is to minimise the communications and maximise the parallelism and reduce the overhead due to extra processing while executing the hierarchical clustering. This technique is evaluated and compared to the sequential version using benchmark datasets and the results are very promising.

***Keywords*:** Data Mining, Distributed Data Mining, Clustering, OPTICS

Liste des Acronymes

DM	**D**ata **M**ining
DDM	**D**istributed **D**ata **M**ining
FDM	**F**ast **D**istributed algorithm for **M**ining association rules
GFM	**G**rid-based **F**requent itemsets **M**ining
DBSCAN	**D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise
OPTICS	**O**rdering **P**oints **T**o **I**dentify the **C**lustering **S**tructure
MPI	**M**essage **P**assing **I**nterface
PVM	**P**arallel **V**irtual **M**chine
CPU	**C**entral **P**rocessing **U**nit
DAPEB	**D**istante d'**A**ccessibilité du **P**oint le plus **E**levé dans la zone Basse
DAPEH	**D**istante d'**A**ccessibilité du **P**oint le plus **E**levé dans la zone **H**aute
MDP	**M**oyenne des **D**istances entre les **P**oints dans un cluster
DBDC	**D**ensity **B**ased **D**istributed **C**lustering
PSLB	**P**ositional **S**can **L**oad **B**alancing
BIRCH	**B**alanced **I**terative **R**educing and **C**lustering using **H**ierarchies

Table des matières

Chapitre 3 : Approche pour la distribution de l'algorithme OPTICS

Chapitre 4 : Evaluation et Expérimentation

Liste des figures

Liste des tableaux

Introduction générale

Les grandes quantités de données produites de nos jours posent un grand problème de stockage, de préservation et d'analyse. Par exemple, de nombreuses entreprises disposent d'entrepôts de données de l'ordre du téraoctets. Le Data Mining joue un rôle important dans l'extraction de la connaissance dans tous les domaines ; scientifique, ingénierie, médical, etc. Cependant, en raison de la taille de ces collections de données, les techniques centralisées et séquentielles de Data Mining présentent des limitations critiques, car elles nécessitent à la fois des capacités mémoires et des temps d'exécution importants. Les techniques centralisées de Data Mining généralement font l'hypothèse que toutes les données sont disponibles dans leur globalité. Dans un environnement distribué, cette hypothèse ne tient plus. La mise en œuvre des idées du Data Mining à haute performance et les environnements de calcul parallèle et distribué deviennent donc essentielles pour garantir l'évolutivité des systèmes de Data Mining.

Les algorithmes distribués traditionnels exigent qu'une grande quantité de données soit échangée entre les sites, ce qui engendre des coûts de communication très élevés. Ceci a motivé le développement de nouvelles approches distribuées basées sur l'agrégation des modèles locaux qui limite ainsi les coûts de communications en n'échangeant que les représentants des clusters aux sites d'agrégats. Parmi les algorithmes distribués, peu d'entre eux sont basés sur le clustering.

Les algorithmes connus de clustering nécessitent des paramètres d'entrées qui sont difficiles à déterminer, mais qui ont une influence significative sur le résultat de clustering. La détermination des paramètres d'entrées pour les algorithmes de partitionnement (le nombre de clusters k) nécessitent une certaine connaissance du domaine qui n'est malheureusement pas disponible pour de nombreuses applications. Contrairement aux algorithmes de partitionnement, les algorithmes hiérarchiques n'ont pas besoin de k comme une entrée. Toutefois, une condition de terminaison doit être définie en indiquant si le processus de fusion ou de division doit être arrêté. Un exemple d'une condition de terminaison dans l'approche ascendante est la distance D_{min} entre tous les clusters. Le principal problème est la difficulté de fixer la valeur appropriée pour cette condition de terminaison, par exemple une valeur de

D_{min} doit être assez petite pour séparer les clusters et en même temps suffisamment importante pour qu'aucun cluster ne soit divisé en deux parties. D'autres algorithmes de clustering tentent de remédier aux problèmes de définition des paramètres d'entrées, parmi, le DBSCAN **[1]** qui est un algorithme de partitionnement basé sur la notion de densité, les clusters dans l'ensemble de données ne peuvent pas être caractérisés par une seule valeur de densité, cependant le nombre de valeurs possibles pour le paramètre de densité est infini ; ce qui pose un autre problème **[2]**. Une manière de remédier à ce problème est de définir un algorithme qui permet de prendre en considération toutes les valeurs possibles pour ce paramètre de densité. L'algorithme OPTICS **[3]** est une extension de DBSCAN du fait qu'il prend en considération plusieurs paramètres d'entrées (toutes les valeurs possibles pour le paramètre de densité). Il fournit un ordre de tous les objets de la base de données à partir duquel nous pouvons extraire tous les niveaux de clustering où chaque niveau est caractérisé par un paramètre de densité différent. Nous avons opté pour l'utilisation de l'algorithme OPTICS dans notre approche distribuée, car, l'ordre fourni est plutôt insensible aux paramètres d'entrées.

Les algorithmes de clustering distribuées existants souffrent de plusieurs problèmes, car la plupart des versions parallèles et distribuées des algorithmes séquentiels se basent sur la version séquentielle qui génère beaucoup de communications et ainsi rendant inefficace leur utilisations, de plus la plupart de ces versions distribuées exige la connaissance du nombre de clusters qui n'est pas évident pour l'utilisateur et les techniques d'approximation qui l'estime sont généralement des problèmes NP-complet. Notre approche distribuée de l'algorithme OPTICS tente d'hériter des avantages des algorithmes distribués existants et apporte des solutions aux problèmes dont ils souffrent. Nous gardons la notion d'agrégation des modèles locaux qui évite l'échange de grandes quantités de données entre les sites comme dans l'approche multi-etage **[4]** et l'approche distribuée de DBSCAN **[5]**. Cependant, nous proposons un algorithme distribué sans tenir compte de la version séquentielle, contrairement à l'algorithme distribué de DBSCAN. De plus notre approche enlève la responsabilité à l'utilisateur pour le choix du nombre de clusters et évite l'utilisation des techniques d'approximation pour le déterminer contrairement à l'approche multi-etage.

Notre approche pour la distribution de l'algorithme OPTICS se compose de trois étapes principales : la première consiste à effectuer le clustering local (en chaque nœud) en exécutant la version séquentielle de l'OPTICS sur les données locale. La deuxième étape consiste à

extraire les représentants de chaque cluster, et les envoyer au site d'agrégat. Les représentants d'un cluster sont les points de sa bordure. Dans la dernière étape, les clusters sont régénérés afin de tester les recouvrements pour enfin construire les clusters globaux.

Ce mémoire est organisé en quatre chapitres : le premier chapitre présente un état de l'art sur le data mining distribué dans lequel nous présentons d'abord les motivations et les différents domaines d'applications des systèmes de Data Mining distribués (DDM), par la suite nous présentons quelques approches distribuées de certains algorithmes de DM, à partir desquels nous montrons les points faibles des méthodes distribuées traditionnels et comment les nouvelles approches distribuées ont pu surmonter ces problèmes. Dans le deuxième chapitre nous abordons la problématique des algorithmes de clustering et nous mettons l'accent sur une méthode d'analyse des clusters basée sur l'algorithme OPTICS. Le troisième chapitre est consacré à la description de notre approche distribuée et dans le dernier chapitre nous donnons les résultats expérimentaux afin d'évaluer notre approche par rapport à l'approche séquentielle et nous terminons ce mémoire par une conclusion générale.

Chapitre **1**

Techniques de Data Mining distribuées

1 Introduction

Dans ce chapitre, nous présentons les motivations et les différents domaines d'applications du Data Mining Distribué (DDM) et nous étudions quelques approches distribuées de DM. Nous commençons par la présentation des approches traditionnelles qui exigent le déplacement de grandes quantités de données vers un nœud central, ensuite nous discutons de nouvelles approches qui sont basées sur l'agrégation des modèles locaux. Dans les approches traditionnelles, nous étudions les performances d'extraction d'itemsets fréquents de l'algorithme Apriori distribué en se basant sur les algorithmes FDM (A fast distributed algorithm for mining association rules) **[6, 7]** et GFM (Grid-based Frequent itemsets Mining) **[8]**. Dans les nouvelles approches de clustering, nous étudions l'approche distribuée de clustering multi étage et l'approche basée sur la distribution de l'algorithme (DBSCAN). Nous concluons ce chapitre par une comparaison entres les approches de distributions étudiées.

1.2 Motivations et domaines d'application du DDM

Les progrès en calcul et en communication sur les systèmes distribués ont permis le développement de nombreux environnements informatiques distribués. Beaucoup de ces environnements ont des sources de données différentes, distribuées, volumineuses et plusieurs nœuds de calcul. Le Data Mining traditionnel doit être adapté pour permettre l'analyse de ces données distribuées. Nous soulignons l'existence d'un décalage entre l'architecture des systèmes de Data Mining centralisés et les besoins de ces systèmes pour les applications distribuées. La Figure 1.1 présente un schéma traditionnel basé sur une architecture centralisée; le temps de réponse long, le manque d'utilisation rationnelle des ressources distribuées, les coûts de communication élevés et parfois la nécessité de garantir la confidentialité des données font que les algorithmes du Data Mining centralisés sont inappropriés aux applications distribuées **[9]**. Une solution pour les applications distribuées nécessite un traitement distribué des données.

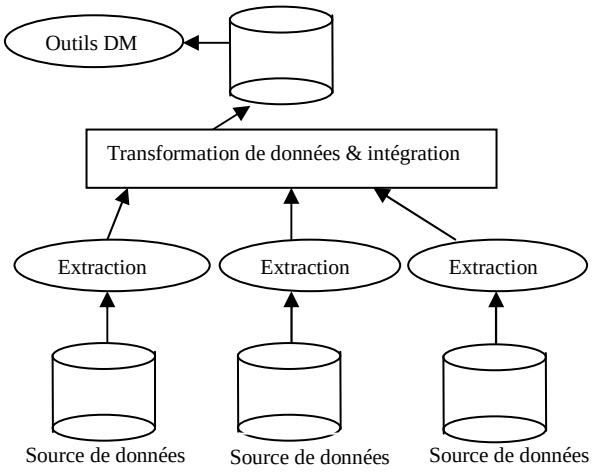


Figure 1.1 L'extraction dans une architecture de Data Mining centralisée.

Le traitement distribué des données se trouve dans plusieurs domaines, par exemple, dans les réseaux ad hoc sans fil, sur des appareils mobiles comme les téléphones cellulaires et les ordinateurs portables. Nous avons besoin d'une architecture qui porte une attention particulière aux ressources distribuées de données, de calcul, de communication et de leur consommation de façon optimale. La figure 1.2 illustre une architecture distribuée pour le Data Mining qui offre la possibilité d'effectuer les opérations basées sur la disponibilité des ressources distribuées.

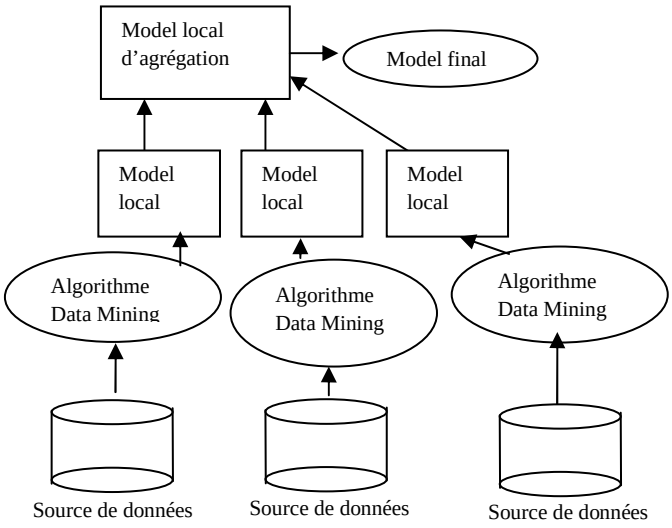


Figure. 1.2 Une architecture distribuée pour le Data Mining.

Le domaine sans fil n'est pas le seul exemple, le World Wide Web en est un autre. L'application des techniques de data mining à l'Internet est connue sous le nom de Web mining, D'une manière générale, le Web mining peu être défini comme la découverte et l'analyse d'informations utiles à travers le World Wide Web **[10]**. La recherche d'information, la consultation de données et l'achat en ligne, sont des exemples de l'utilisation du Web. Les sites Web représentent actuellement une véritable source de production de grands volumes d'informations produites à partir des quantités énormes de données. Cependant ces données et les ressources informatiques sont distribuées alors une approche distribuée pour l'analyse de ces données est susceptible d'être plus évolutive et plus pratique.

La protection de la vie privée joue un rôle de plus en plus important dans les applications émergentes de Data Mining. Si un consortium de banques différentes souhaite collaborer pour détecter les fraudes, alors un système de Data Mining centralisé peut exiger la collecte de toutes les données de chaque banque à un seul endroit. Les systèmes du DDM peuvent éviter cet échange de données. Cela permet à la fois la détection des fraudes et la préservation de la confidentialité de toutes les transactions bancaires pour les clients. La détection automatique de fraude peut s'appliquer à un téléphone mobile qui émet un appel d'une durée inhabituelle, de ou vers une zone inhabituelle. La détection en temps réel des transactions bancaires douteuses a aussi permis a la librairie sur le web Amazon de réduire son taux de fraudes de 50 % en 6 mois **[11]**.

1.3 Les techniques de Data Mining distribuées

Les approches de Data Mining distribuées peuvent être divisées en deux catégories : les approches traditionnelles et les nouvelles approches. Les approches de distribution traditionnelles exigent qu'un grand nombre de données soit transféré à un site central ; ce qui engendre des coûts de communication très élevés. Les nouvelles approches sont basées sur l'agrégation des modèles locaux, elles évitent le transport de grandes quantités de données à un site central ; ce qui réduit les coûts de communications.

Plusieurs algorithmes distribués de Data Mining on été développés. Parmi : une approche pour l'extraction de règles d'associations distribuée **[12]**, l'approche FDM (A fast distributed algorithm for mining association rules) **[6,7]**, L'approche GFM (Grid-based Frequent itemsets Mining) **[8, 13]** et l'approche parallèle de l'algorithme FP-Growth **[14]**. Ce domaine de recherche est très large, mais peu de travaux ont été consacrés au clustering distribué. Parmi les travaux existants, on trouve dans **[15, 16, 17]** La parallelisation de l'algorithme de clustering basé sur la densité DBSCAN. Dans **[18]** une version parallèle de BIRCH (balanced iterative reducing and clustering using hierarchies) et dans **[4]** l'approche de clustering distribuée multi-etage.

Dans la suite de ce chapitre, nous présentons quelques algorithmes distribués traditionnels. Nous étudions ceux basés sur l'extraction d'itemsets fréquents (l'approche FDM et deux variantes de l'approche GFM). Nous présentons aussi quelques nouvelles approches basées sur le clustering tels que l'approche multi-etage **[4]** et une version distribuée de DBSCAN **[5]**.

1.3.1 Techniques de distribution traditionnelles

1.3.1.1 Étude de performances de l'algorithme Apriori distribué

La découverte de motifs fréquents est l'un des domaines majeurs du DM. A l'origine de ce domaine se trouvent les travaux d'Agrawal et Srikant (1994) **[19]** sur la découverte d'itemsets (ensemble d'items) fréquents. Le problème de la découverte d'itemsets fréquents consiste à considérer les transactions dans une base de données comme des listes d'items, d'utiliser un seuil de support minimal, noté, minsup et de sélectionner tous les itemsets dont le nombre d'apparition dans la base de données est supérieur à minsup. Ce problème est le plus simple dans le domaine de la découverte de motifs fréquents, comparativement à la recherche de séquences, d'arbres ou de graphes fréquents. Toutefois, il se retrouve dans de nombreuses

applications, en particulier dans l'analyse de données de vente d'entreprise commerciales. De plus, de part sa simplicité, toutes les améliorations algorithmiques significatives en découverte de motifs fréquents ont d'abord été réalisées dans le cas des itemsets fréquents avant d'être adaptées à des motifs plus complexes, c'est en particulier le cas pour la fermeture **[20]** ou les méthodes sans génération de candidats **[21]**.

Nous allons nous concentrer sur les travaux qui traitent l'extraction d'itemsets fréquents dans un environnement distribué. L'approche GFM (Grid-based Frequent itemsets Mining) **[8]** à été proposée pour distribuer l'algorithme Apriori. Nous comparons cette approche avec l'approche distribuée classique FDM (A fast distributed algorithm for mining association rules). L'approche FDM est basée sur une stratégie d'élagage globale, contrairement à l'approche GFM qui est une amélioration de l'approche FDM du fait qu'elle est basée sur une stratégie d'élagage locale. Nous présentons ensuite une approche distribuée de l'algorithme qui traite l'hétérogénéité des données **[13]**. Avant de commencer l'étude de ces approches distribuées, nous présentons d'abord l'extraction d'itemsets fréquents par l'algorithme Apriori.

1.3.1.1.1 L'algorithme Apriori

L'algorithme Apriori proposé permet d'extraire les itemsets fréquents. La recherche pour trouver chaque groupe d'itemset nécessite une lecture complète de la base de données. Pour construire les ensembles d'items tout en réduisant l'espace de recherche nous utilisons la propriété Apriori **[19]**.

Propriété Apriori:

Les sous-ensembles d'un ensemble d'items fréquents sont aussi des ensembles d'items fréquents **[22].** Par exemple, si {*A, B*} est un itemset fréquents, alors {*A*} et {*B*} sont aussi des itemsets fréquents**.** Plus généralement, les sous-ensembles de k-1 items d'un ensemble de k items fréquent sont fréquents.

En utilisant la propriété Apriori, nous constatons que nous pouvons construire les ensembles d'items incrémentalement, nous commençons avec les ensembles à un item. Un k-itemset peut être construit par la jointure d'un ensemble de (k-1) itemsets avec lui-même, ensuite on applique l'élagage qui efface les éléments qui ne vérifient pas le support minimal.

L'élagage s'effectue de la manière suivante : Chaque (k-1) itemset qui n'est pas fréquent ne peut pas être un sous ensemble d'un k-itemset fréquent.

Pour mieux comprendre le processus de calcul des itemsets fréquents, nous présentant l'exemple suivant :

La Table 1.1 montre un exemple d'une base de données contenant 4 transactions (10, 20, 30, 40). Chaque transaction est constituée d'un ensemble d'items. Le seuil support minimal est égal à 2 dans cet exemple.

N°transaction	Item
10	1, 2, 5
20	2, 4
30	2, 3
40	1, 2, 4

Table 1.1: Exemple d'une base de données contenant 4 transactions

Nous allons calculer les itemsets fréquents d'ordre 2 à partir de la base de données de la Table 1.1. La première étape consiste à calculer le support de chaque itemset d'ordre 1 (Table 1.2). Le support d'un itemset est le nombre de fois où il apparaît dans la base de données.

Itemset	Support
{1}	2
{2}	4
{3}	1
{4}	2
{5}	1

Table 1.2: L'ensemble des items d'ordre 1 et leurs supports

A partir de la Table 1.2, la Table 1.3 est construite, qui contient que les itemsets qui vérifient le support minimal.

Itemset	Support
{1}	2
{2}	4
{4}	2

Table 1.3: Liste des itemsets qui vérifient le support minimal

Dans la Table 1.4 nous avons les itemsets d'ordre 2 tirés à partir de la Table 1.3. Le support de chaque itemset est calculé et le filtrage par rapport au seuil support minimal est effectué dans Table 1.4.

itemset	Support
{1,2}	2
{2,4}	2

Table 1.4 Les itemsets fréquents d'ordre 2 et leurs supports.

1.3.1.1.2 L'extraction d'itemsets fréquents distribués

L'algorithme Apriori distribué est basé sur deux propriétés principales

Propriété 1

Un itemset fréquent globalement doit être fréquent localement dans au moins un nœud.

Propriété 2

Si un itemset est fréquent globalement alors tous ses sous ensembles sont fréquents globalement. **[8]**.

Dans les deux algorithmes FDM et GFM, initialement l'ensemble des transactions est partitionné horizontalement[1] sur m nœud.

1.3.1.1.3 L'approche FDM

L'approche FDM (A fast distributed algorithm for mining association rules) consiste à distribuer l'algorithme Apriori. Elle est basée sur une stratégie d'élagage globale.

Les entrées de l'algorithme FDM sont le nombre de nœuds m sur lequel les transactions des données seront réparties, le seuil support global sw utilisé pour vérifier la fréquence des ensemble d'items et l'ordre des itemsets k qu'on souhaite générer. L'algorithme FDM est exécuté en plusieurs niveaux, le nombre de niveaux est égal à l'ordre des itemsets k qu'on souhaite générer. Dans un niveau i, chaque site local calcule le compte support[2] de chaque itemset et les itemsets fréquents de taille i en utilisant le seuil sw et envoie les comptes support calculés aux autres sites. Dans chaque site, le calcul des comptes support distants est effectué et les itemsets fréquents d'ordre i sont calculés, ensuite, chaque site diffuse les comptes support calculés. Nous refaisons les étapes précédentes en calculant à chaque fois les itemsets fréquents d'ordre i+1 jusqu'à atteindre l'ordre k désiré.

La Figure 1.3 résume cette approche :

[1] Le partitionnement horizontal consiste à diviser l'ensemble des transactions sur l'ensemble des sites

[2] Le compte support d'un item est le nombre de fois où il apparaît dans la base de données

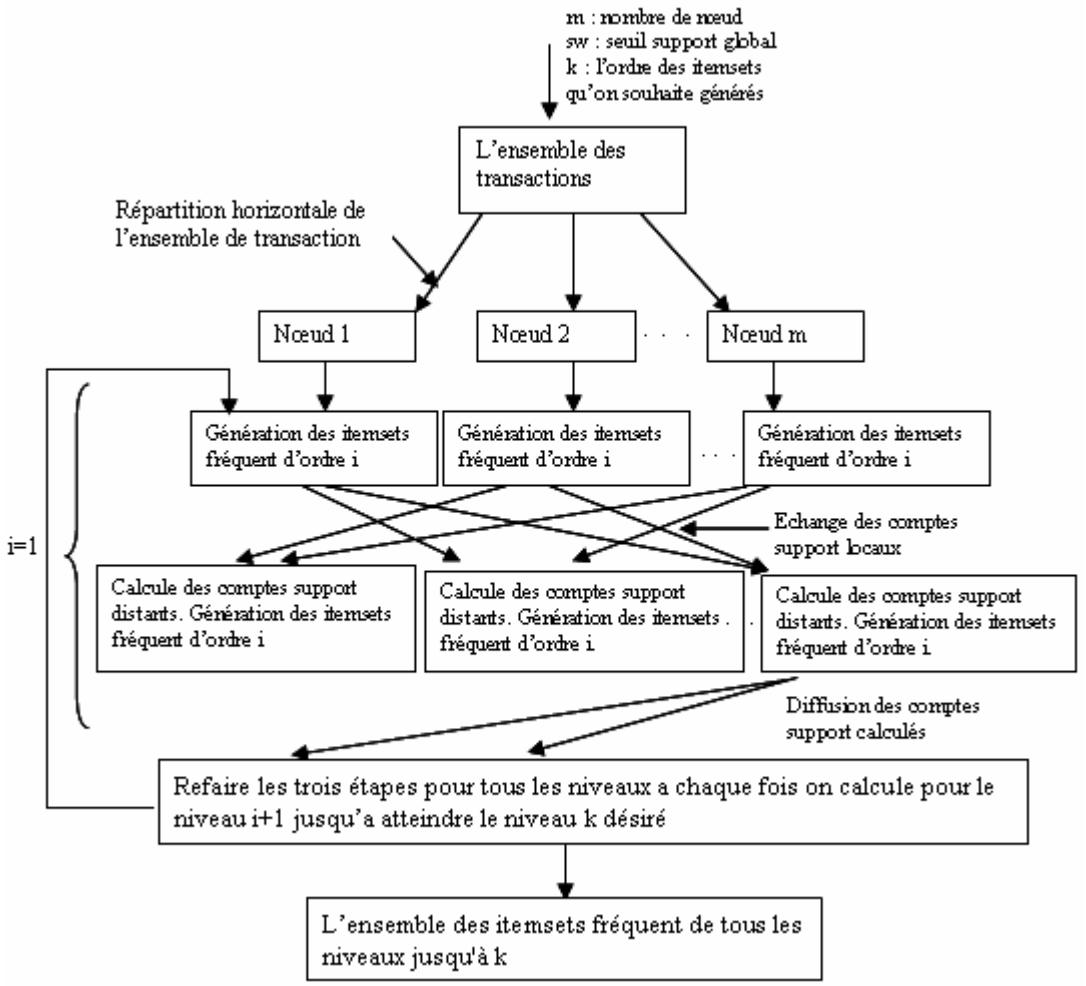


Figure 1.3 : schéma résumant l'approche FDM.

1.3.1.1.4 L'approche GFM (Grid-based Frequent itemsets Mining)

L'approche GFM est une amélioration de l'approche FDM du fait qu'elle est basée sur une stratégie d'élagage locale. Elle calcule la taille nécessaire k pour les itemsets fréquents localement dans chaque site sans l'échange des comptes support intermédiaire **[8]**. Tous les itemsets fréquents au niveau global sont déduits à la fin de la phase de calcul en tenant compte des :

- Comptes support global collectés à partir des itemsets fréquents de taille k et tous les itemsets fréquents maximaux dans chaque nœud qui ne sont pas des sous ensembles d'autres itemsets fréquents de taille plus grande.
- Sous ensembles d'itemsets fréquents localement qui échouent le test de fréquence global dans les étapes suivantes.

Les entrées de l'algorithme GFM sont le nombre de nœuds m, le seuil support global sw utilisé pour extraire les itemsets fréquents et l'ordre des itemsets k qu'on souhaite générer. Après avoir effectué le partitionnement horizontal de l'ensemble des transactions et l'affectation de chaque partition de données à un nœud local N_i {i=1,.., m}, chaque site local calcule les itemsets fréquents de taille k et envoie l'ensemble LLi aux autres sites. LLi représente les itemsets fréquents calculés localement dans le nœud N_i. L'ensemble LLi sera remplacé par l'ensemble des itemsets fréquents qui échouent le test global. Chaque site calcule les comptes support localement et les diffuse aux autres sites. Par la suite, chaque site va calculer l'ensemble Li qui représente l'ensemble des itemsets qui vérifient le test de fréquence global et aussi l'ensemble LLi. La sortie de l'algorithme est l'union des sous ensembles Li qui représente tous les itemsets fréquents d'ordre k désiré. La Figure 1.4 résume l'approche GFM :

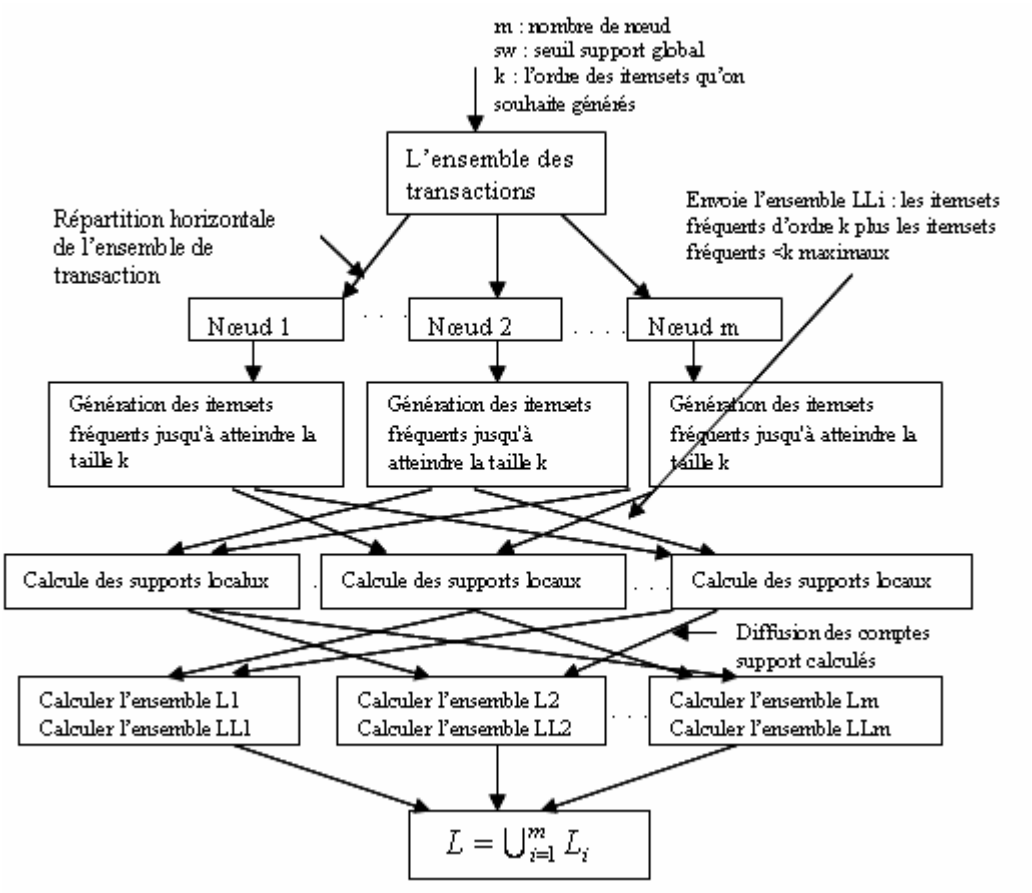


Figure 1.4 : schéma résumant l'approche GFM.

L'étude analytique faite pour les deux approches a montré que les étapes d'élagage global intermédiaires sont inefficaces et affectent les performances du système global **[8]**.

1.3.1.1.5 Evaluation et comparaison des deux approches

L'expérimentation de ces deux approches montre que les ensembles de candidats obtenus en utilisant une stratégie d'élagage globale sont très proches à ceux obtenus en utilisant la stratégie d'élagage locale indépendamment des seuils de support utilisés **[8]**. Cependant, l'approche GFM consomme moins en terme de communication. Elle n'a besoin que de deux passages de communication. Soit k l'ordre des itemsets que nous souhaitons générés. L'approche FDM adopte une stratégie d'élagage globale et nécessite k étapes de communication pour le calcul des comptes support et k+1 activités parallèles pour le processus de génération Apriori. Ainsi, le calcul des comptes support distants est effectué en k activités parallèles. Les sorties de l'algorithme sont les ensembles des itemsets fréquents de 1 jusqu'à k. Par contre, l'approche GFM adopte une stratégie d'élagage locale, le calcul du compte support s'effectue en une étape et la génération Apriori en deux autres étapes. Les sorties de l'algorithme sont les itemsets fréquents d'ordre k.

1.3.1.1.6 L'extraction d'itemsets fréquents distribuée dans les plateformes hétérogènes

L'extraction d'itemsets fréquents dans une implémentation distribuée est très coûteuse en terme de calcul et génère une charge de travail non équilibré. L'approche distribuée de l'algorithme GFM traite l'hétérogénéité et la distribution des données **[13]**. Cette approche est basée sur la gestion de la charge de travail et le partitionnement basé sur des blocks pour traiter les contraintes de mémoire.

Description de l'approche

Cette approche considère deux niveaux de partitionnement : le partitionnement de la base de données et le partitionnement de la mémoire. En premier lieu, l'ensemble des transactions est divisé sur m nœuds (l'unité la plus petite qui peut êtres assigné à un nœud est la transaction). Dans le deuxième niveau de partitionnement, chaque partition Ti est divisée en blocks. La taille d'un block est affectée selon la mémoire disponible dans le nœud correspondant et aussi selon les informations concernant l'ensemble de données comme le nombre d'items et le seuil support. La figure suivante montre les deux niveaux de partitionnement.

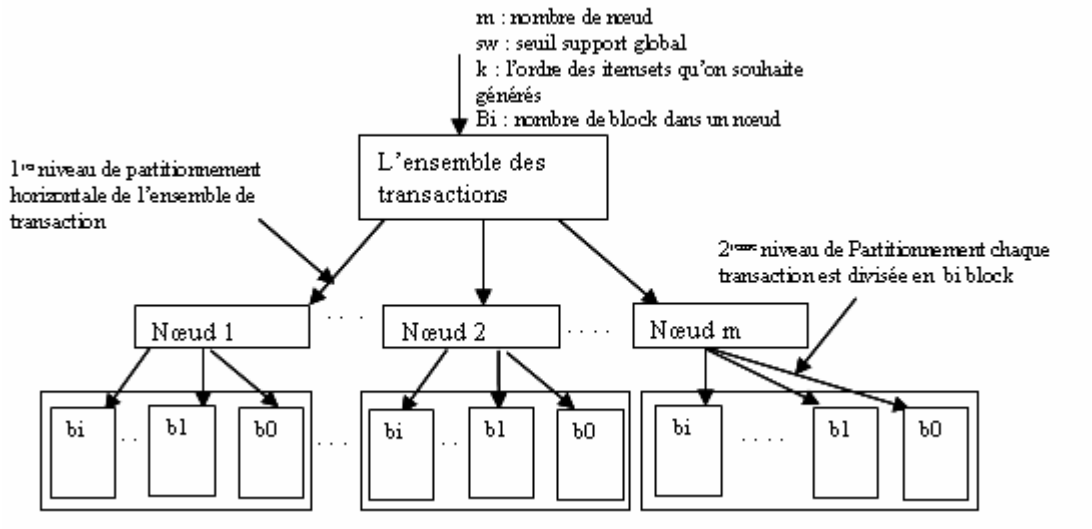


Figure 1.5 : Les deux niveaux de partitionnement.

Au début de la première phase, chaque nœud N_i {i=1,.., m}diffuse la taille de sa portion de données Di, son nombre de blocks bi et le vecteur de description de charge de travail WVi qui inclut la taille de la mémoire disponible et le taux de CPU. Dans cette phase, l'algorithme Apriori est exécuté et uniquement l'élagage local est considéré pour la génération des k itemsets fréquents sans étapes d'échange des comptes support intermédiaires. Les ensembles de candidats peuvent augmenter dans le processus de génération Apriori ; ce qui conduit à des demandes d'espace mémoire importantes qui engendrent un effet d'écrasement. Pour cela, si un nœud devient avide, c'est-à-dire, que le block ne peut pas contenir tous les itemsets générés, il envoie une demande de travail (requête) aux nœuds qui n'ont pas encore envoyé leurs notifications. Tous les nœuds qui ont un espace réservé suffisant pour la génération Apriori diffusent leurs notifications. La table *ThreadCheckRequest* est construite pour contenir l'ensemble des requêtes. Une mise à jour est faite à travers la table *TheadUpdate* qui collecte les informations à partir des blocks qui se déplacent entre les nœuds. A la fin de la première phase, chaque nœud demande le compte support distant de ces items fréquents locaux. Chaque nœud envoie l'ensemble LLi qui contient les items fréquents d'ordre k et les itemsets fréquents maximal inférieur à k. Chaque noeud reçoit ces ensembles à partir d'autres noeuds et calcule le compte support global et déduit les itemsets fréquents globaux dénotés par Li selon le seuil du compte support. Le résultat de cet algorithme est l'union des ensembles Li.

Evaluation de l'approche

Puisque le coup de génération dans FDM et GFM est presque le même alors dans l'approche avec gestion de charge de travail, le gain du temps de communication des comptes support est certain.

Les résultats obtenus par l'expérimentation faite montrent que la gestion de charge de travail est efficace dans le cas de distribution déséquilibrée et dans des plateformes hétérogènes **[13]**.

1.3.2 Les nouvelles méthodes distribuées

Les méthodes étudiées précédemment exigent qu'un grand nombre de données soient transférés à un site central ; ce qui engendre des coûts de communication très élevés. Pour remédier à cela, de nouvelles approches distribuées on été développées pour limiter les coûts de communication élevés en n'échangeant que les représentants des clusters aux sites d'agrégats. Nous allons étudier deux nouvelles approches distribuées des algorithmes de clustering. Pour cela, nous commençons par une brève présentation des algorithmes de clustering existants.

1.3.2.1 Le clustering

Les algorithmes de clustering sont classifiés en deux classes : les algorithmes de partitionnement et les algorithmes hiérarchiques **[23]**. Les algorithmes de partitionnement permettent de partitionner la base de données en un ensemble de k clusters. Le nombre de clusters k est un paramètre d'entrée pour ces algorithmes, une certaine connaissance du domaine est nécessaire qui n'est malheureusement pas disponible pour de nombreuses applications. Les algorithmes de partitionnement commencent généralement avec une partition initiale qui est l'ensemble de données, puis utilise une stratégie de contrôle itérative pour optimiser une fonction objectif. Chaque cluster est représenté par le centre de gravité du cluster (algorithmes k-means) ou par l'un des objets du cluster situé près de son centre (algorithmes k-medoid). Cette catégorie d'algorithmes utilise une procédure en deux étapes. La première étape consiste à déterminer les représentants k minimisant la fonction objectif. La seconde étape permet d'assigner chaque objet au cluster le plus proche. Les algorithmes hiérarchiques créent une décomposition hiérarchique de la base de données, qui est représentée par un dendrogramme, un arbre qui divise itérativement l'ensemble de données en plus petits sous-ensembles jusqu'à ce que chaque sous-ensemble se compose d'un seul objet.

Dans une telle hiérarchie, chaque nœud de l'arbre représente un cluster d'un ensemble de données. Le dendrogramme peut être créé à partir des feuilles jusqu'à la racine (approche ascendante) ou de la racine jusqu'aux feuilles (approche de division) en fusionnant ou en divisant les clusters à chaque étape. Contrairement aux algorithmes de partitionnement, les algorithmes hiérarchiques n'ont pas besoin de k comme une entrée. Toutefois, une condition de terminaison doit être définie en indiquant si le processus de fusion ou de division doit être clôturé. Un exemple d'une condition de terminaison dans l'approche ascendante est la distance D_{min} entre tous les clusters. Le principal problème avec les algorithmes de clustering hiérarchique est la difficulté de fixer les paramètres appropriés pour la condition de terminaison, par exemple une valeur de D_{min} doit être assez petite pour séparer les clusters et en même temps suffisamment importante pour qu'aucun cluster ne soit divisé en deux parties.

Une alternative pour la définition des paramètres d'entrées est d'utiliser un algorithme de partitionnement basé sur la notion de densité (DBSCAN) **[1]**, les clusters dans l'ensemble de données ne peuvent pas être caractérisés par une seule valeur de densité, cependant le nombre de valeurs possibles pour le paramètre de densité est infini ; ce qui pose un autre problème **[2]**. Une manière de remédier à ce problème est de définir un algorithme qui permet de prendre en considération toutes les valeurs possibles pour le paramètre de densité. L'algorithme OPTICS **[3]** est une extension de DBSCAN du fait qu'il prend en considération plusieurs paramètres d'entrées (toutes les valeurs possibles pour le paramètre de densité). Nous référons le lecteur aux travaux de A. K. JAIN et R. XU pour plus de détails sur ces algorithmes **[24, 25]**.

1.3.2.2 L'approche Multi Etage

La méthode de clustering distribuée multi-étage est une nouvelle approche distribuée **[4]**, elle est basée sur l'augmentation du critère de la variance **[26]** qui exige des flots de communication limités et utilise des méthodes d'approximation pour déterminer le nombre de clusters **[27, 28, 29]**. L'approche utilise aussi la notion de perturbation qui améliore la génération globale de clustering.

Description de l'approche : Elle est basée sur les étapes suivantes :

Etape1 :

Initialement, le clustering local peut être effectué en utilisant plusieurs algorithmes différents selon les caractéristiques de la base de données par exemple le k-means et ses variantes **[30, 31, 32]**. Le clustering local s'effectue indépendamment avec relativement un

grand nombre de sous clusters ou un nombre optimal de clusters qui est déterminé en utilisant des techniques d'approximation **[27, 28, 29]**.

Etape 2 :

Dans chaque site local les représentants des clusters seront déterminés et envoyés au site d'agrégat sélectionné. Les représentants sont : le nombre de clusters dans chaque site, la variance, la taille des données et le centre de chaque cluster dans chaque site local.

Etape 3 :

La méthode de pondération de l'écart statistique est appliquée sur les centres locaux pour déterminer le nombre de cluster global et aussi le critère de la variance maximum est calculé

Etape 4 :

On fusionne les sous clusters selon le critère de la variance maximale jusqu'à ce qu'ils correspondent à l'évaluation globale, c'est-à-dire, jusqu'à avoir un nombre de clusters égal au nombre estimé. Les candidats les plus prometteurs pour former les clusters globaux sont les sous clusters qui sont les plus proches dans l'espace des caractéristiques

La fusion est un processus de marquage, c'est-à-dire, chaque processus local peut générer des correspondances entre les sous clusters locaux sans construire nécessairement la sortie générale du clustering, car les clusters sont des centres, des tailles et des variances. Lors de la fusion, nous calculons les nouvelles caractéristiques du cluster (la variance ...)

Etape 5 :

Cette étape consiste à rechercher les limites bi de chaque cluster global. La limite d'un cluster global est représentée par les sous clusters locaux situés à sa bordure. Ces sous clusters peuvent être isolés et ajoutés à un autre cluster afin de contribuer à une amélioration du clustering global. La limite est déterminée en utilisant la mesure de distance euclidienne. Les sous clusters les plus lointains au sens de la distance sont appelés les candidats de perturbation. Le processus de perturbation est activé si l'action de fusion n'est plus appliquée. Ce processus déplace les candidats de perturbation à partir des clusters globaux voisins en essayant les plus proches tout en améliorant le critère de la variance.

Le critère le plus utilisé pour quantifier l'homogénéité du cluster est le critère de la variance qui est donnée par :

$$\sum_{i=1}^{M} E(C_i)$$

Où X = {x_1, x_2,…, x_N} une base de données de N éléments, C= {C_1, C_2, …, C_M} avec C_i (i=i, M) sont des sous ensemble de X. E(C_i) correspond à la variance et u(C_i) correspond à la moyenne.

$$E(C) = \sum_{x \in C} \|x - u(C)\|^2 \qquad u(C) = \frac{1}{|C|} \sum_{x \in C} x$$

Les contraintes traditionnelles utilisées pour minimiser un critère donné consistent à fixer le nombre de clusters M à un nombre connu à priori comme dans le k-mean et ses variantes, mais cette valeur n'est pas toujours connue dans la plupart des cas. La contrainte de l'approche proposée s'appuie sur le fait que l'augmentation de la variance de fusion de deux clusters SC_i et SC_j est inférieure à $\sigma_{i,j}^{\max}$ où cette valeur est définie pour chaque base de données en utilisant des informations globales d'évaluation.

Contrairement à plusieurs autres algorithmes distribués, cette approche utilise une contrainte globale simple, un flot de communication limité et la structure de données peut ne pas être connu à priori. Cet algorithme est efficace dans la recherche des clusters.

1.3.2.3 Distribution de l'algorithme DBSCAN

L'algorithme de clustering DBSCAN distribué (Density-Based Spatial Clustering of Applications with Noise) **[5]** consiste d'abord à effectuer le clustering séquentiel localement. Ensuite, les informations agrégées sur les clusters crées localement sont extraites et envoyées à un site central. Le coût de transmission est minime car les représentants ne sont qu'une fraction des données d'origine. Sur le site central, le clustering global fondé sur les représentants est effectué ensuite les résultats sont envoyés aux sites locaux. Les sites locaux mettent à jour leur clustering basé sur le modèle global. Cette approche suit les étapes suivantes :

- Le clustering local.
- Détermination des modèles locaux.
- Détermination du modèle globale.
- Mise à jour du clustering global.

Le clustering local

Les données résident initialement dans des sites différents. Ils sont clustérisées indépendamment sur les différents sites locaux. L'algorithme de clustering utilisé est le DBSCAN qui est un algorithme de clustering basé sur la densité **[1]**. Dans la suite, nous décrivons en détail le clustering à base de densité et l'algorithme DBSCAN.

Le clustering à base de densité

Le clustering basé sur la densité considère les clusters comme des régions denses d'objets dans l'espace de données qui sont séparées par des régions de faible densité qui représentent le bruit. Il permet de découvrir des clusters avec des formes arbitraires et utilise des paramètres de densité ε et *MinPts* où ε représente le rayon maximal du voisinage et le voisinage dans un rayon ε d'un objet donné s'appelle le *ε-voisinage* de l'objet. Le *MinPts* représente le nombre minimum de points dans un *ε-voisinage* d'un point donné.

Définition 1 *La densité d'accessibilité directe*

L'objet *p* est *directement densité accessible* à partir de l'objet *q* avec ε et *MinPts* dans un ensemble d'objets *D* si :

1. $p \in N_\varepsilon(q)$ où $N_\varepsilon(q)$ est un sous ensemble de *D* contenu dans le ε voisinage de *q*.
2. $Card\ (N_\varepsilon\ (q)) \geq MinPts$. (*Card (N)* dénote la cardinalité de l'ensemble *N*)

La condition $Card(N_\varepsilon\ (q)) \geq MinPts$ est appelée la condition du noyau principal. Si cette condition est vérifiée pour un objet *p*, alors *p* est appelé le *noyau principal*.

Définition 2 *Densité d'accessibilité*

L'objet *p* est *densité accessible* à l'objet *q* avec ε et *MinPts* dans un ensemble d'objets D s'il existe une chaîne d'objets $p_1, \ldots, p_n$, $p_1=q$, $p_n=p$ où $p_i \in D$ et p_{i+1} est *directement densité accessible* à partir de p_i avec le ε et le *MinPts*.

La densité d'accessibilité est la transitivité de la densité d'accessibilité directe. En général, cette relation n'est pas symétrique.

Définition 3 : La *densité de connectivité*

Un objet *p* est *densité connectée* à l'objet *q* avec ε et *MinPts* dans l'ensemble d'objets *D* s'il existe un objet $o \in D$ tel que *p* et *q* sont *densités accessibles* à partir de *o* avec ε et *MinPts*. La densité de connectivité est une relation symétrique.

La *densité basée cluster* est définie comme un ensemble maximal d'objets qui sont *densités connectés*.

Définition 4 : *clusters*

Soit *D* un ensemble d'objets. Un cluster $C \subset D$ avec ε et *MinPts* est un sous ensemble non vide qui satisfait les conditions suivantes :

- Maximalité : *p, q* $\in$ *D :* si $p \in C$ et *q* est densité accessible à partir de *p* avec ε et *MinPts,* alors $q \in C$.
- Connectivité : *p, q* $\in C$ *: p* est *densité connectée* à *q* avec ε et *MinPts.*

Le cluster ne contient pas uniquement des objets principaux mais également ceux qui ne satisfont pas la condition du noyau principal. Ces objets sont appelés *Objets de bordures* du cluster. Cependant, ils sont densités accessibles directement à partir au moins d'un noyau principal du cluster (contrairement au objet bruit).

Définition 5 : *bruit*

Soit *C1,..., Ck* un ensemble de clusters dans D, avec les paramètres ε et MinPts, le bruit est définit comme un ensemble d'objets dans D qui n'appartient à aucun cluster C_i, c'est-à-dire bruit=$\{p \in D | \forall i : p \notin C_i\}$

Pour illustrer ces notions, considérons l'exemple suivant : Les valeurs des paramètres d'entrées utilisées dans cet exemple sont : *MinPts* =3 et ε = 2 cm. Dans la figure. 1.6, nous pouvons constater que :

L est *directement densité accessible* à partir de Q, S est *densité accessible* à partir de Q et P est *densité connectée* à l'objet Q car il existe l'objet L tel que P et Q sont *densités accessibles* à partir de L.

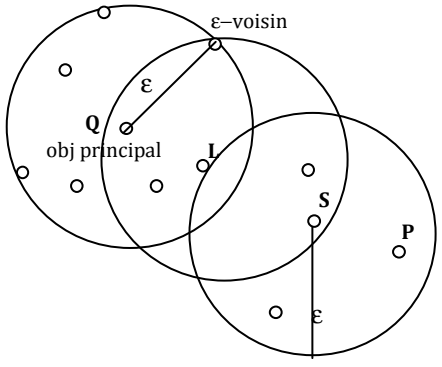


Figure 1.6: Exemple de densité d'accessibilité directe, densité d'accessibilité et densité de connectivité.

DBSCAN (densité basée sur le clustering spatial des applications avec bruit)

DBSCAN cherche les clusters par la vérification des *ε–voisins* pour chaque objet dans la base de données. Si le *ε–voisin* d'un objet *p* contient plus d'un *MinPts* d'objets, un nouveau cluster avec *p* comme objet noyau est créé. DBSCAN collecte itérativement les objets densités accessibles directes à partir de cet objet noyau qui peut générer la fusion de clusters. Le processus se termine lorsque aucun nouveau point ne peut être ajouté aux clusters retenus. La complexité de DBSCAN est de O (n log n) où n est le nombre d'objets dans la base de données. L'algorithme permet de gérer les données bruitées et cherche des formes arbitraires de clusters.

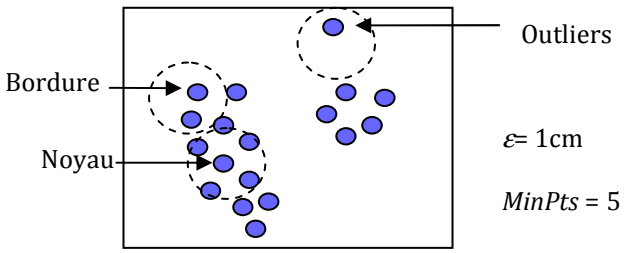


Figure 1.7: Exemple pour l'algorithme DBSCAN

Détermination du modèle local

Après avoir effectuer le clustering localement, nous avons besoin d'un petit nombre de représentants qui décrivent le résultat du clustering local. Les points noyaux calculés pendant l'exécution de DBSCAN peuvent servir de bons représentants. Malheureusement, leur nombre peut devenir très élevé, particulièrement dans les secteurs très denses du cluster. Pour cela, une nouvelle approche est développée pour déterminer les représentants appropriés qui sont basés sur le concept de point noyau spécifique.

***Définition* :** points noyau spécifique

Soit $C \subseteq$ D un cluster avec les paramètres ε et MinPts. Soit $Cor_C \subseteq C$ l'ensemble des points noyaux appartenant au cluster. Alors $ScorC \subseteq C$ est appelé ensemble complet des points noyaux spécifiques de *C* si les conditions suivantes sont vérifiées.

- $ScorC \subseteq CorC$
- $\forall s_i, s_j \in ScorC: s_i \notin N_\varepsilon(s_j)$ où $N_\varepsilon(s_j)$: le voisinage de s_j à une distance ε.
- $\forall c \in Cor_C \; \exists s \in Scor_C: c \in N_\varepsilon(s)$

Il peut exister plusieurs ensembles différents de $Scor_C$ qui vérifient cette définition. Chacun de ces ensembles $Scor_C$ est constitué de plusieurs points noyaux spécifiques qui peuvent être utilisés pour décrire le cluster C.

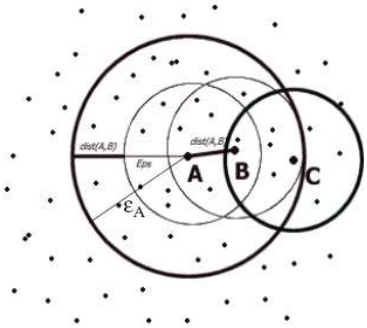


Figure 1.8: les points noyaux spécifique et le ε-intervalle.

La Figure 1.9 montre que si A est un élément de l'ensemble de points noyaux spécifiques *Scor* alors tous les points noyaux localisés dans le ε-voisinage de A ne peuvent pas appartenir à *Scor* tel que le point B.

Le point C peut appartenir à Scor comme il n'est pas dans le ε-voisinage de A. D'autre part, si B est dans *Scor*, alors A et C ne peuvent pas appartenir à *Scor* car ils sont tous les deux dans l'ε-voisinage de B.

Dans le modèle local, noté *REPScor*, chaque cluster local C_i est représenté par un ensemble complet de noyau spécifique $Scor_{Ci}$. Supposons que nous avons trouver n clusters C_i,.. C_n dans le site local k, le modèle local est formé par l'union des différents $Scor_{Ci}$.

D'après la Figure 1.9, le point A ne représente pas seulement les objets dans son propre voisinage, mais aussi les objets dans le voisinage de B, il est donc un représentatif des points $N_\varepsilon(A)$ et $N_\varepsilon(B)$. Nous allons assigné un nouveau ε à A qui est égal à ε+dist (A,B) (voir Figure 1.9). Par conséquent, un tel epsilon est assigné à tous les points noyaux spécifiques donnés par la définition suivante.

Définition ε-intervalle spécifique

Soit $C \subseteq D$ un cluster avec les paramètres ε et *MinPts*. En outre, soit $Scor \subseteq C$ est un ensemble complet de points noyaux spécifiques. Alors, nous assignons pour chaque s ∈ *Scor* un ε_s -intervalle indiquant la zone représentée par s $\varepsilon_s = \varepsilon + \max \{dist(s,s_i) \mid s_i \in Cor \wedge s_i \in N_\varepsilon(s)\}$

La valeur spécifique de ε- intervalle est une partie du modèle local et elle est évaluée dans le site serveur pour développer un modèle global précis. En outre, il est très important pour le processus de mise à jour des objets locaux. La valeur spécifique de ε- intervalle est intégrée dans le modèle local du site k comme suit :

$$LocalModel_k = \bigcup_{i \in 1..n} \{(s, \varepsilon_s) | s \in Scor_{Ci}\}$$

Chaque modèle local noté $LocalModel_k$ consiste en un ensemble de m_k pair (r,ε), donné par le représentatif r et par l'intervalle de ε. Le nombre de pairs m transmises à partir de chaque site k est déterminé par le nombre n de cluster C_i dans le site k, et le nombre de point noyau spécifique $|Scor_{Ci}|$ pour chaque cluster Ci comme suit :

$$m = \sum_{i=1..n} |Scor_{C_i}|$$

Définition du modèle global :

Pour trouver un modèle global, l'algorithme de clustering DBSCAN basé sur la densité est utilisé à nouveau avec le réglage des paramètres locaux dans le but de créer un clustering similaire à celui produit par DBSCAN quant il est appliqué à l'ensemble de données complet. Comme nous avons un accès à l'ensemble des représentants locaux seulement, le paramétrage global doit être adapté à cette information locale agrégée. On dit que deux clusters provenant du même site ou des sites différents appartiennent au même cluster global si les deux clusters sont densités connectées.

Le ε global doit être plus grand que le paramètre ε local utilisé dans le clustering dans les sites locaux. Pour de grandes valeurs de ε global, nous prenons le risque de fusionner des clusters qui ne devrait pas l'être. Dans le cas où la valeur de ε global est petite, il n'est pas possible de détecter les clusters susceptibles d'être fusionnés. Le paramètre ε global doit être réglable par l'utilisateur dépendant des valeurs de ε_r de tous les représentants locaux r. La valeur proposée par défaut est généralement proche de 2*ε local.

Mise à jour du clustering local

Apres avoir effectué le clustering global, le modèle global complet est envoyé à tous les sites locaux. Sur un site local, deux clusters indépendants peuvent être fusionnés en raison de ce nouveau réétiquetage (nouvelle valeur de ε ...). En outre, les points qui ont été autrefois assignés au bruit local peuvent faire partie d'un cluster global.

Dans **[5]** une évaluation expérimentale de cette approche de clustering distribuée est présentée qui montrent qu'elle offre presque le même résultat qu'un clustering central sur toutes les données. D'autre part, un énorme avantage d'efficacité comparée à un clustering classique effectuée sur toutes les données.

1.4 Conclusion

Dans ce chapitre, nous avons présenté quelques approches de distribution du DM. Les nouvelles approches apportent beaucoup d'avantage par rapport aux méthodes distribuées traditionnelles où uniquement les représentants des clusters sont transportés aux sites d'agrégats; ce qui évite le déplacement des données qui engendre des coûts de communication très élevés, contrairement aux méthodes traditionnelles qui exigent le déplacement de grandes quantités de données vers un seul nœud central. Les nouvelles approches distribuées utilisent des techniques plus efficaces pour la détermination du nombre de clusters locaux. Le choix de ce nombre initialement n'est pas évident pour l'utilisateur.

Chapitre 2

Etude de l'algorithme de clustering OPTICS

2.1 Introduction

Dans ce chapitre, nous présentons un algorithme de clustering basé sur la densité qui définit un ordre pour les objets de la base de données. Cet ordre permet d'identifier les clusters des différentes formes et aussi les structures de clusters intrinsèques. Nous abordons dans ce chapitre les motivations du développement de l'algorithme OPTICS. Nous présentons l'ordre des clusters à base de densité ainsi que la méthode d'identification de la structure de clustering et enfin nous étudions l'algorithme qui permet de déterminer les clusters à partir de l'ordre obtenus.

2.2 Motivation

La plupart des recherches récentes relatives au problème de clustering se concentrent sur l'efficacité à savoir la qualité ou l'utilité du résultat. Il est difficile d'assurer l'efficacité des algorithmes de clustering car presque tous ces algorithmes exigent des valeurs pour les paramètres d'entrées qui sont difficiles à déterminer surtout pour les données issues du monde réel. Par ailleurs, les algorithmes sont très sensibles à ces valeurs d'entrées et les ensembles de données réelles de dimension complexe ont souvent une distribution très inégale qui ne peut être révélée par un algorithme de clustering en utilisant un seul paramètre global. Par exemple, dans l'ensemble de données représenté dans la Figure 2.1, il n'est pas possible de détecter les clusters C1 et C2 avec les clusters C3, C4 et C5 simultanément en utilisant un seul paramètre de densité global. Nous pouvons avoir soit les deux clusters C1 et C2 ou bien les clusters : C3, C4 et C5. Dans le deuxième cas, les clusters C1 et C2 sont considérés comme un bruit.

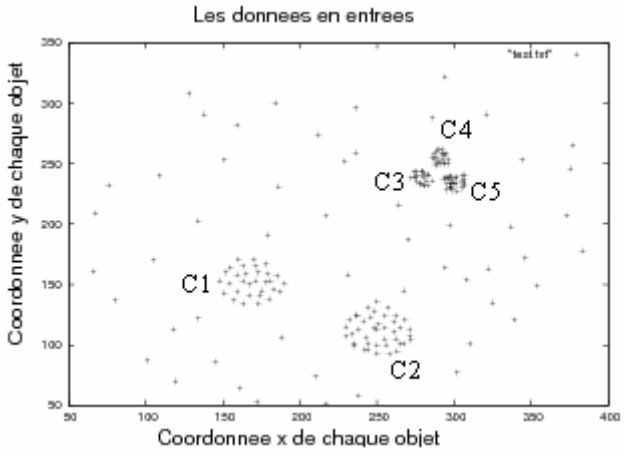


Figure 2.1 : Clusters avec des paramètres de densité différente.

Pour cela, une alternative consiste à employer un algorithme basé sur la densité qui permet de produire un ordre spécial pour les objets de la base de données. Cet ordre contient des informations sur chaque niveau de clustering de l'ensemble de données. L'algorithme OPTICS permet de créer cet ordre **[3]**.

2.3 L'ordre des clusters à base de densité

L'idée principale du clustering basé sur la densité est que le voisinage par rapport à un rayon donnée ε de chaque objet d'un cluster doit avoir au moins un nombre minimum d'objets (MinPts), c'est-à-dire, la cardinalité du voisinage doit dépasser un seuil donné. La définition formelle de cette notion du clustering basé sur la densité est donnée dans le chapitre 1.

Avant de présenter la notion d'ordre du cluster basé sur la densité, nous faisons la remarque suivante: Pour une valeur constante de MinPts, les clusters à base de densité avec une densité élevée (c'est-à-dire une petite valeur de ε) sont complètement contenus dans des ensembles avec des densités plus faibles (une grande valeur de ε). Ce fait est illustré dans la Figure 2.2, où C1 et C2 sont des clusters à base de densité avec un paramètre ε2. C : est un cluster à base de densité avec le paramètre ε1 contenant complètement les ensembles C1 et C2. (ε2 < ε1)

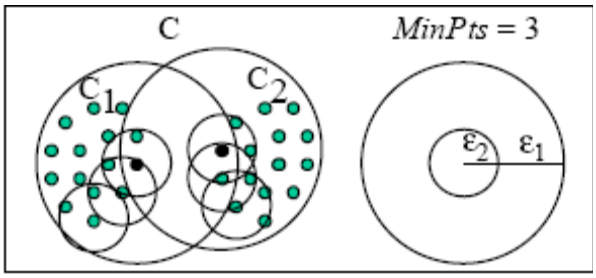


Figure 2.2 Illustration des clusters imbriqués fondés sur la densité

Par conséquent, une extension de l'algorithme DBSCAN **[1]** peut être effectuée de sorte que plusieurs paramètres de distance sont pris en considération. Pour produire un résultat cohérent, nous devrions obéir à un ordre spécifique dans lequel les objets seront traités lors de la construction d'un cluster. Nous devons toujours choisir un objet qui est « densité accessible » avec la plus petite valeur de ε pour garantir la construction de clusters avec une grande densité en priorité.

L'information utilisée pour effectuer l'ordre fourni par l'algorithme OPTICS est composée de deux valeurs pour chaque objet : la distance noyau et la distance d'accessibilité. Ces deux concepts sont définis comme suit :

La distance noyau d'un objet p (core distance)

Soient *p* un objet de la base de données *D*, ε une valeur de distance, $N_\varepsilon(p)$ le ε*–voisinage* de *p*, *MinPts* un nombre naturel et Distance-MinPts : la distance entre *p* et ces *MinPts* voisins. La distance noyau de l'objet *p* est défini comme suit :

$$\text{distance - noyau}_{\varepsilon,\text{MinPts}}(p) = \begin{cases} \textit{Indéfini, si} \quad \textit{Card}(N_\varepsilon(q) < \textit{MinPts}) \\ \text{Distance - MinPts, autrement} \end{cases}$$

La distance d'accessibilité de l'objet *p* par rapport à l'objet *o*.

Soient *p* et *o* deux objets de la base de données *D*, soit $N_\varepsilon(o)$ le *ε-voisinage* de *o* et *MinPts* un nombre naturel. La distance d'accessibilité de *p* par rapport à *o* est définit comme :

$$\text{distance-d'accessibilité}_{\varepsilon,\text{MinPts}}(p,o) = \begin{cases} \text{Indéfini, si } |N_\varepsilon(o)| < MinPts \\ \text{Max(distance-noyau(o), distance(o,p)) autrement} \end{cases}$$

La distance d'accessibilité d'un objet *p* par rapport à un objet noyau *o* est égale à la distance entre l'objet *p* et l'objet *o* si cette distance est supérieure à la distance noyau sinon elle est égale à la distance noyau. Si *o* n'est pas un objet noyau alors la distance d'accessibilité de *p* par rapport à *o* n'est pas définie. La Figure 2.3 illustre les notions de distance noyau et de distance d'accessibilité.

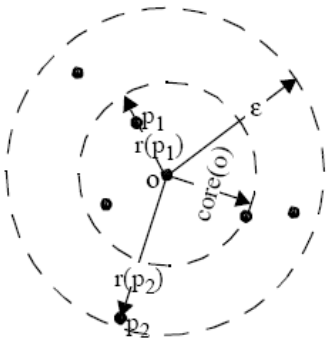


Figure 2.3: distance noyau de o Core (o), distance d'accessibilité r (*p1*,*o*), r(*p2*,*o*) pour *MinPts*=4

Pour chaque objet de la base de données, nous stockons la distance noyau et la distance d'accessibilité correspondante. Ces informations sont suffisantes pour extraire tous les clusters à base de densité. La Figure 2.4 illustre la boucle principale de l'algorithme OPTICS. Chaque objet de la base de données **EnsDeObjets** est pris en charge par une procédure **EtendreClusterOrdre** si l'objet n'est pas encore traité.

```
OPTICS (EnsDeObjets, ε, MinPts, FichierOrdre)
    FichierOrdre.ouvrir() ;
    Pour i de 1 à EnsDeObjets.taille faire
        Objet= EnsDeObjets.obtenir(i) ;
        Si non Objet.traite alors
            EtendreClusterOrdre(EnsDeObjets, Objets, ε, MinPts, FichierOrdre)
    FichierOrdre.ferme() ;
Fin : // OPTICS
```

Figure 2.4 L'algorithme OPTICS

La procédure **EtendreClusterOrdre** est représentée dans la Figure 2.5. Premièrement, cette procédure recherche les ε-voisins de l'objet **«Objet »** passé dans la boucle principale

OPTICS, place sa distance d'accessibilité à indéfinie et détermine sa distance noyau. L'objet est écrit dans **FichierOrdre.** La condition *Si* contrôle les propriétés noyau de l'objet, s'il n'est pas un noyau principal avec la distance ε, le contrôle est simplement retourné à la boucle principale de OPTICS qui sélectionne le prochain objet non traité dans la base de données. Autrement, si **Objet** est un objet noyau à une distance inférieure ou égale à ε, nous collectons itérativement les objets avec la densité d'accessibilité directe avec ε et MinPts. Les objets qui sont directement densités accessibles à partir de l'objet noyau courant sont insérés dans la liste **OrderSeeds**. Les objets contenus dans **OrderSeeds** sont triés par leurs distances d'accessibilité à l'objet noyau le plus proche qui ont été directement densités accessibles. Dans chaque étape de la boucle *Tant que***,** un objet **ObjectCourant** ayant la plus petite distance d'accessibilité dans la liste **OrdreSeeds** est sélectionné par la méthode **OrderSeeds.suivant ()**. Les ε–voisinage de cet objet et sa distance noyau sont déterminés. L'objet est simplement écrit dans le fichier **FichierOrdre** avec sa distance noyau et sa distance d'accessibilité.

```
EtendreClusterOrdre(EnsDeObjets, Objets, ε, MinPts, FichierOrdre) ;
    Voisinage= EnsDeObjets.voisinage(Objets, ε) ;
    Objet.traite=vrai ;
    Objet.accessibilite_distance=INDEFINIE ;
    Objet.EnsDistanceNoyau(voisins, ε, MinPts) ;
    FichierOrdre.écrire(Objet) ;
    Si Objet.noyau_distance <> INDEFINIE alors
          OrdreSeeds.Mise-à-jours(voisins, Objets) ;
          Tant que non OrdreSeeds.vide( ) faire
               ObjetCourant .traite=OrderSeeds.suivant( ) ;
               Voisins=EnsDeObjets.voisins(ObjetCourant, ε) ;
               ObjetCourant.traite=vrai
               ObjetCourant.EnsDistanceNoyau(voisins, ε, MinPts) ;
               FichierOrdre.écrire( ObjetCourant) ;
               Si ObjetCourant.noyau_distance <> INDEFINIE alors
                    OrdreSeeds.Mise-à-jours(voisins, ObjetsCourant) ;
Fin ; // EtendreClusterOrdre
```

Figure 2.5 La procédure ExpandClusterOrder.

L'insertion dans la liste **OrdreSeeds** et le traitement de la distance d'accessibilité sont gérés par la méthode **OrdreSeeds ::Mise-à-jour** présenté dans la Figure 2.6. La distance d'accessibilité pour chaque objet dans l'ensemble des voisins est déterminée par rapport à l'objet **ObjectCentre.** Les objets qui ne sont pas encore dans la liste **OrderSeeds** sont simplement insérés avec leurs distances d'accessibilité. Les distances d'accessibilité des objets qui sont dans la liste sont remplacées par leurs nouvelles distances d'accessibilités si ces nouvelles distances sont plus petites que leurs distances d'accessibilité précédentes

```
OrdreSeeds ::Mise-à-jour(voisins, ObjetCentre) ;
    c_dist=ObjetCentre.distance_noyau ;
    Pour tous Objet DE voisinage FAIRE
        Si non Objet.traite alors
        nouveau_r_dist=max(c_dist, ObjetCentre.dist(Objet));
        Si Objet.distance_accessibilité=INDEFINIE alors
            Objet.distance_accessibilité=nouveau_r_dist ;
                Insérer (Objet, nouveau_r_dist)
        Sinon
            Si nouveau_r_dist < Objet.distance_accessibilité alors
                Objet.distance_accessibilité=nouveau_r_dist ;
                Décroître (Objet, nouveau_r_dist ) ;
Fin ; //  OrdreSeeds ::Mise-à-jours
```

Figure 2.6 La méthode OrderSeeds::mise-à-jour ()

En raison de son équivalence structurale à l'algorithme DBSCAN, le temps d'exécution pour OPTICS est fortement dominé par le temps d'exécution des requête des *ε-voisinage* qui doivent être exécutées pour chaque objet dans la base de données, c-à-d le temps d'exécution pour les deux algorithmes est *O (n*t)* où *t* est le temps d'exécution d'une requête de ε–voisinage et *n* le nombre d'objet dans la base de données.

2.4 Identification de la structure de clustering

L'ordre des clusters de la base de données peut être représenté graphiquement. Nous pouvons voir la structure de clustering d'un ensemble de données si les valeurs des distances d'accessibilité r sont tracées pour chaque objet o de la base de données dans l'ordre des clusters. La Figure 2.7 représente le graphe des distances d'accessibilité pour un ensemble de données à deux dimensions. La visualisation de l'ordre des clusters est indépendante de la dimension des données.

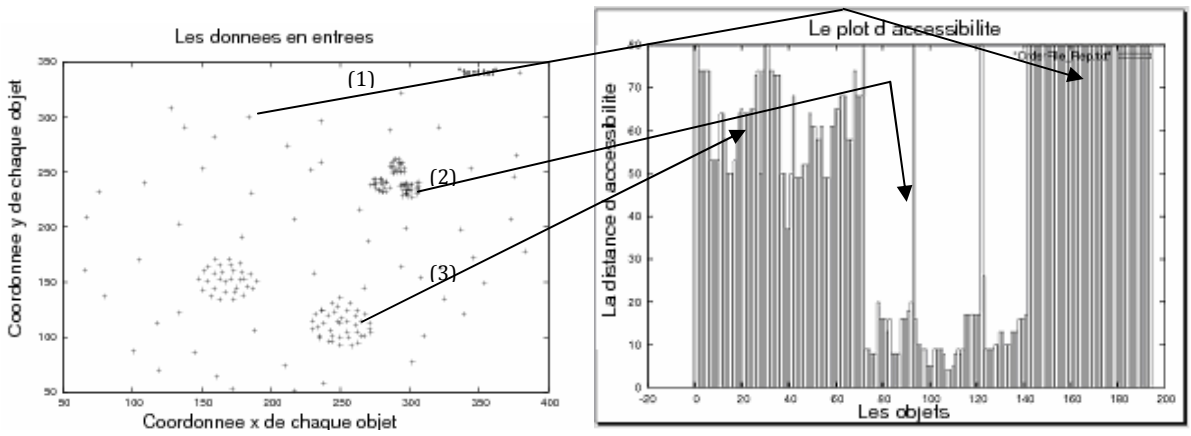


Figure 2.7. Le graphe des distances d'accessibilité pour un ensemble de données avec des clusters hiérarchique de différentes densités. La flèche (1) représente le bruit, (2) les clusters avec une grande densité, (3) les clusters moins dense

Un autre avantage de l'ordre des clusters de l'ensemble de données par rapport aux autres méthodes de clustering est que le graphe des distances d'accessibilité est peu sensible aux paramètres d'entrées de la méthode (ε et MinPts). Les valeurs doivent seulement être assez importantes pour obtenir un bon résultat. Les vraies valeurs ne sont pas essentielles, car il y a un large éventail de valeurs possibles pour lesquelles nous pouvons toujours identifier la structure de clustering d'un ensemble de données.

2.5 Définition formelle d'un cluster

Afin d'identifier les clusters dans une base de données, nous utilisons le graphe des distances d'accessibilités. La valeur d'accessibilité d'un point correspond à la distance de ce point à l'ensemble de ses prédécesseurs. A titre d'exemple, dans la Figure 2.8 un cluster commence à partir du point n°3 et se termine au point n°16. Le point 3 qui est le dernier point avec une valeur d'accessibilité élevée fait partie du cluster, car cette valeur d'accessibilité élevée indique qu'il est loin des points 1 et 2. Il doit être proche du point 4, parce que le point 4 a une valeur d'accessibilité faible indiquant qu'il est proche de l'un des points 1, 2 ou 3, cependant l'algorithme OPTICS choisit le point suivant dans l'ordre des clusters, pour cela le point 4 doit être proche du point 3. Un argument similaire s'applique pour le point 16 qui est le dernier point avec une faible valeur d'accessibilité et il fait partie également du cluster.

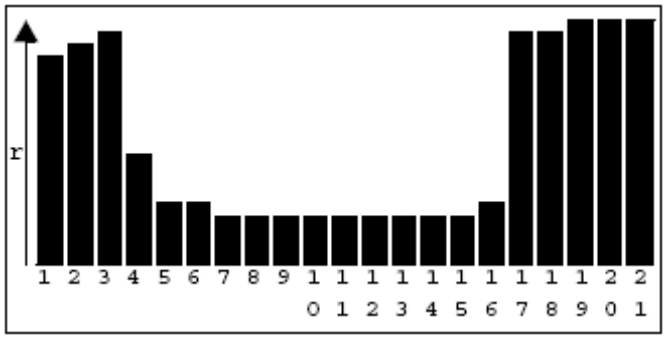


Figure 2.8 Un cluster.

Les clusters dans les ensembles de données réelles ne commencent pas et ne se terminent pas souvent avec des points extrêmes élevés. Par exemple, dans la Figure 2.9, nous observons trois clusters qui sont très différents. Le premier commence avec le plus grand point A et se termine avec un point élevé B. Le deuxième cluster se termine avec un nombre d'étapes non rapides, mais élevées au point D. Pour saisir ces différents degrés de hauteur, le paramètre ξ est introduit, il est choisit par expérimentation.

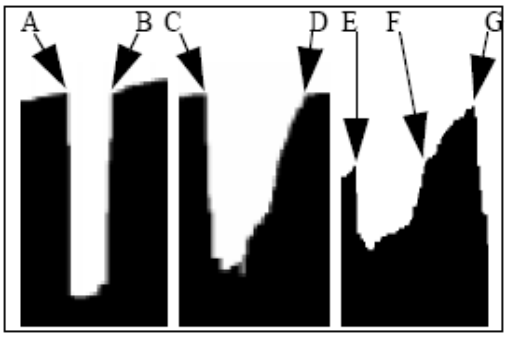


Figure 2.9 Clusters réels

Définition 1 **(**ξ-points raide)

Soit n : le nombre d'objets dans la base de données.

Un point p ∈ {1…n-1} est appelé ξ-point montant (ξ-steep upward) s'il est ξ% plus petit que son successeur : $UpPoint_{\xi}(p) \Leftrightarrow r(p) \leq r(p+1)*(1-\xi)$.

Un point p ∈ {1…n-1} est appelé ξ-point descendant (ξ-steep downward) si son successeur est ξ% plus petit : $DownPoint_{\xi}(p) \Leftrightarrow r(p)*(1-\xi) \leq r(p+1)$.

En regardant de près la Figure 2.9, nous voyons que tous les clusters commencent et se terminent à un nombre de point raide (steep point). Ces régions commencent par des points ξ-raide suivie d'autres points. Une telle région est appelée une zone raide. Plus précisément, une zone raide haute commence et se termine à un point ξ-raide montant. En outre, il ne doit pas contenir trop de points non-raides consécutifs en raison de la condition de base utilisée dans OPTICS qui stipule : MinPts points non-raide consécutives peuvent être considérés comme un cluster distinct et ne devraient pas faire partie d'une zone raide. La définition d'une zone raide (ξ-steep area) est donnée comme suit :

Définition 2 **(**ξ-steep area**)**

Un intervalle I=[s,e] est appelé une zone ξ-steep upward (ξ-raide montante) noté $UpArea_{\xi}(I)$ s'il satisfait les conditions suivantes:

- s est un point ξ-steep upward : $UpArea_{\xi}(s)$
- e est un point ξ-steep upward : $UpArea_{\xi}(e)$
- chaque point entre s et e est un peu plus haut que son prédécesseur :
  $\forall x, s < x \leq e : r(x) \geq r(x-1)$

- I ne contient pas plus de MinPts points consécutifs qui ne sont pas ξ-steep upward
  $\forall [s',e'] \subseteq [s,e] :((\forall x \in [s',e'] : UpArea_{\xi}(x)) \Rightarrow e'-s' < MinPts)$

- I est maximal: $\forall J :(I \subseteq J, UpArea_{\xi}(J) \Rightarrow I=J)$

Une zone ξ-steep downward (ξ-raide descendante) est définit de manière analogique ($UpArea_{\xi}(I)$).

Définition 3 **(**ξ-cluster**)**

Un intervalle $C = [s,e] \subseteq [1,n]$ est appelé ξ-cluster s'il satisfait les conditions de 1 à 4 :

ξ-cluster $\Leftrightarrow \exists D = [s_D,e_D], U=[s_U,e_U]$ avec

1. $DownArea_{\xi}(D) \wedge s \in D$
2. $UpArea_{\xi}(U) \wedge e \in U$
3. a) $e\text{-}s \geq MinPts$

   b) $\forall x, s_D < x \leq e_U : (r(x) \leq \min(r(s_D), r(e_U)) \times (1-\xi))$
4. $(s,e) =$

$$\begin{cases} (\max\{x \in D \mid r(x) > (e_U+1)\}, e_U) \text{ si } r(s_D) \times (1-\xi) \geq (e_U+1) & \text{(b)} \\ (s_D, \min\{x \in U \mid r(x) < r(s_D)\}) \text{ si } r(e_U+1) \times (1-\xi) \geq (s_D) & \text{(c)} \\ (s_D, e_U) \quad \text{autrement} & \text{(a)} \end{cases}$$

Les conditions 1 et 2 indiquent simplement que le début d'un cluster est contenu dans une zone ξ-raide descendante D et la fin d'un cluster est contenue dans une zone ξ-raide montante U. La condition 3a) indique que le cluster doit être constitué au moins d'un MinPts de points. La condition 3b) montre que les valeurs d'accessibilités de tous les points du cluster doivent être au moins ξ% inférieure à la valeur d'accessibilité du premier point de D et de la valeur d'accessibilité du premier point après la fin de U. La condition 4) détermine le point début et le point fin du cluster selon les valeurs d'accessibilité du premier point de D (appelé *ReachStart*) et la valeur d'accessibilité du premier point après la fin de U (apelé ReachEnd). On distingue trois cas (voir Figure 2.10) :

- Si ces deux valeurs sont au même niveau à une valeur prés de ξ%, le cluster commence au début de D et se termine à la fin de U (Figure 2.10 a).
- Si ReachStart est ξ% plus élevé que ReachEnd, le cluster se termine à la fin de U, mais commence à un point dans D qui a approximativement la même valeur d'accessibilité que ReachEnd (Figure 2.10 b), le cluster commence à SoC.

- Si ReachEnd est ξ% plus élevé que ReachStart, le cluster commencera au premier point de D et se terminera à un point dans U, qui a approximativement la même valeur d'accessibilité comme ReachStart (Figure 2.10 c, le cluster ce termine à EoC).

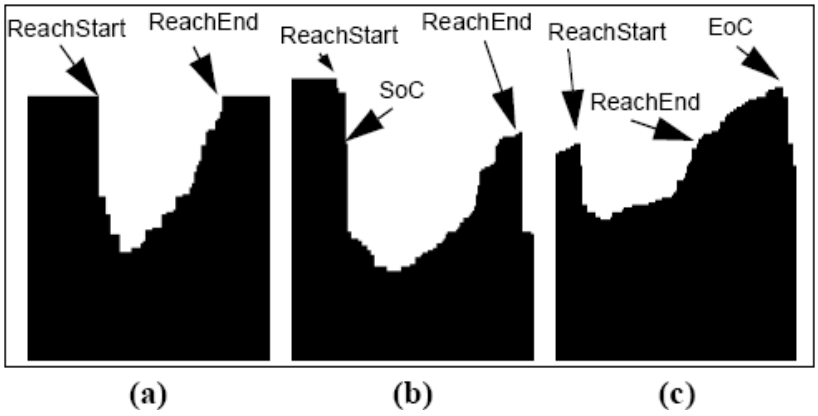


Figure 2.10 : Trois types différents de clusters pris d'un jeu de données artificiel

2.6. Algorithme pour la détermination des clusters

Nous allons présenter un algorithme pour calculer tous les clusters en utilisant un seul passage sur les valeurs d'accessibilité de tous les points. L'idée de base est d'identifier toutes les zones basses et hautes et les combiner en clusters. Nous allons commencer avec une implémentation naïve de la définition du cluster. L'algorithme qui en résulte est inefficace, pour cela il sera transformé pour trouver les clusters réels.

2.6.1 L'algorithme naïf

L'algorithme naïf consiste à faire un seul passage sur toutes les valeurs d'accessibilité dans l'ordre calculé par l'algorithme OPTICS.

SDASet est un ensemble de régions raides basses qui contiennent pour chacune d'elle un indice de début et de fin.

Nous commençons par indice=0 avec un SDASet vide.

Tant que indice < n faire.

(1) Si une nouvelle région basse commence à indice, elle est ajoutée à SDASet et on continue à parcourir les prochains points.

(2) Si une nouvelle zone haute commence à indice, elle est combinée avec les zones basses dans SDASet. On vérifie chaque combinaison pour remplir les conditions du cluster et on calcule la valeur de s et de e selon la condition 4). Nous continuons à parcourir les points après cette région raide haute.

(3) Sinon incrémenter l'indice.

Cet algorithme trouve tous les clusters. Il fait un seul passage sur les valeurs d'accessibilité et aussi une boucle pour chaque cluster pour vérifier la condition 3b dans l'étape (2). La plupart des combinaisons ne sont pas réellement des clusters. Pour cela, nous employons un autre algorithme pour surmonter cette difficulté.

2.6.2 Amélioration de l'algorithme naif

L'amélioration de l'algorithme naif consiste en deux techniques, dans la première nous filtrons la plupart des clusters qui n'entraînent pas des clusters réels et dans la seconde étape nous nous débarrassons de la boucle sur tous les points dans le cluster. La condition 3b est donnée par : $\forall x, s_D < x \leq e_U : (r(x) \leq \min(r(s_D), r(e_U))\times(1-\xi))$. Cette condition peut être écrite de la manière suivante :

(sc1) $\forall x, s_D < x \leq e_U : (r(x) \leq r(s_D)\times(1-\xi))$

(sc2) $\forall x, s_D < x \leq e_U : (r(x) \leq r(e_U)\times(1-\xi))$

La condition 3b est divisée en (sc1) et (sc2). Pour cela, nous vérifions les sous conditions (sc1) (sc2) séparément. Nous pouvons transformer ces conditions aux conditions équivalentes suivantes :

(sc1*) $\max\{x \mid s_D < x \leq e_U\} \leq r(s_D)\times(1-\xi)$

(sc2*) $\max\{x \mid s_D < x \leq e_U\} \leq r(e_U)\times(1-\xi)$

Afin d'utiliser les conditions (sc1*) (sc2*), nous avons besoin d'introduire le concept de maximum entre valeurs noté mib, contenant la valeur maximum entre un certain point dans le graphe d'accessibilité (début où fin de zone basse où haute) et l'indice courant (indice de la zone basse ou haute courante). Nous garderons la trace de la valeur de mib pour chaque région basse dans SDASet, qui représente la valeur maximum entre la fin de la région basse et l'indice courant et une valeur de mib global contenant le maximum entre la fin de la dernière zone raide (haute ou basse) trouvé et l'indice courant. L'algorithme est modifié pour garder la trace de toutes les valeurs de mib. Il est représenté dans la Figure 2.11.

```
EnsDeZoneRaidBasse=∅
EnsDeClusters= ∅
Indice = 0, mib= 0
Tant que (indice<n)
      Mib=max(mib,r(index))
      Si (début d'une zone basse D à indice)
            Mettre à jours des valeurs de mib et filtrer EnsDeZoneRaidBasse (*)
            Ens D.mib = 0 ;
            Ajouter D à EnsDeZoneRaidBasse
            Indice=fin de D+1 ; mib=r(indice)
      Sinon Si (début d'une zone haute U à indice)
            Mettre à jours des valeurs de mib et filtrer EnsDeZoneRaidBasse
            Indice=fin de U+1 ; mib=r(indice)
            Pour chaque D dans EnsDeZoneRaidBasse faire
                Si (la combinaison de D et U est valide ET (**))
                Satisfaire les conditions du cluster 1, 2, 3a)
                Calculer [s, e] ajouter cluster à EnsDeClusters
      Sinon indice=indice+1
Retourner(EnsDeClusters)
```

Figure 2.11 Algorithme d'extraction de clusters.

Cet algorithme consiste à filtrer toutes les zones basses à partir de SDASet qui ont début de zone basse $\times(1-\xi)$ est inférieur à la valeur de mib-global, cela réduit le nombre de clusters significativement et satisfait en même temps la condition (sc1*) (ligne * dans la Figure 2.11). Dans la ligne (**) de la Figure 2.11, nous comparons la fin de la zone haute $U\times (1-\xi)$ à la valeur mib de la zone basse D. Ainsi, la condition (sc*) est satisfaite. La valeur de mib permet de satisfaire les conditions (sc1*) (sc2*) ; ce qui implique la condition 3b est remplie. Ainsi nous réduisons le nombre de clusters significativement et par conséquent économisons la boucle sur tous les points dans chaque cluster.

2.7. Conclusion

Dans ce chapitre, nous avons présenté une méthode d'analyse des clusters, basée sur l'algorithme OPTICS qui calcule l'ordre des objets dans la base de données. Le principal avantage de l'algorithme OPTICS par rapport à la classification des algorithmes proposés dans la littérature est qu'il ne se limite pas à un seul paramètre global. Il permet d'extraire automatiquement et efficacement non seulement les informations traditionnelles de clustering tels que les formes arbitraires des clusters, mais aussi les clusters chevauchés.

Nous avons présenté dans ce chapitre l'ordre des clusters graphiquement et également un algorithme pour extraire automatiquement les clusters à partir de l'ordre des objets dans la base de données.

Chapitre 3

Approche pour la distribution de l'algorithme OPTICS

3.1 Introduction

Dans ce chapitre, nous présentons une nouvelle approche de clustering distribuée, qui tente d'hériter des avantages des algorithmes distribués existants et apporte des solutions à leurs limitations. Elle évite le transport de grandes quantités de données à un site central en acheminant que les représentants des clusters aux sites d'agrégat et elle est indépendante de la version séquentielle.

Ce chapitre est organisé de la manière suivante : Nous proposons d'abord une méthode pour le calcul des clusters à partir des graphes des distances d'accessibilités après avoir exécuté l'algorithme OPTICS dans chaque site local, par la suite nous définissons les modèles locaux (les représentants des clusters) qui consiste principalement à déterminer les bordures de chaque cluster. Nous proposons également un algorithme pour reproduire la frontière originale de tous les clusters dans les sites d'agrégats à partir des bordures calculés dans les sites locaux. Nous expliquons le processus de régénération des clusters à partir des nouvelles bordures calculées, nous définissons le modèle de distribution et les modèles globaux et enfin nous estimons la complexité de notre l'approche.

3.2 Le clustering local

Le clustering local consiste à exécuter l'algorithme OPTICS sur chaque site local et à déterminer les clusters à partir des graphes d'accessibilité obtenus. Nous utilisons l'algorithme OPTICS pour les raisons suivantes : Premièrement, le graphe des distances d'accessibilité fournit par l'algorithme OPTICS est plutôt insensible aux paramètres d'entrée (ε et MinPts). En général, les valeurs de ces paramètres doivent être suffisamment grandes pour conduire à un bon résultat. Deuxièmement, la visualisation graphique de l'ordre produit est indépendante de la dimension de l'ensemble des données. L'algorithme OPTICS permet de donner un ordre pour tous les objets dans la base de données où nous pouvons extraire les clusters de différentes densités et formes. Cependant, la méthode existante pour la détermination des clusters à partir de cet ordre des clusters n'est pas efficace **[3]**. Elle consiste à déterminer les zones hautes et les zones basses. La première étape de cette méthode suppose que toute combinaison de zones hautes et basses forme un cluster et la deuxième étape essaye de filtrer les clusters. Les limites de cette méthode résident dans le fait qu'elle n'élimine que certains clusters et dans le cas des graphes des distances d'accessibilité avec plusieurs pics, (voir Figure 3.1), il reste toujours plusieurs groupes qui ne sont pas réellement des clusters.

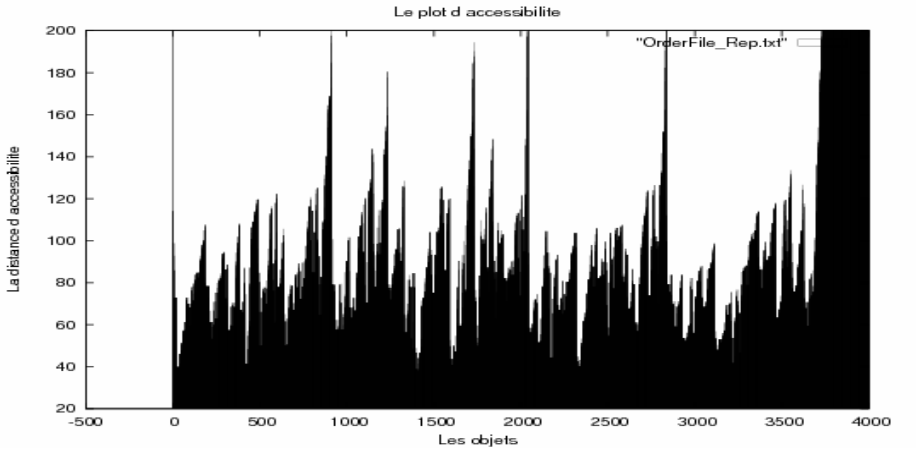


Figure 3.1 : Graphe des distances d'accessibilité.

3.2.1 Nouvelle méthode pour le calcul des clusters

Notre méthode consiste à calculer les zones basses et les zones hautes mais en définissent un seuil par expérimentation égal à 80 % de la plus grande zone haute ou basse. Ce qui revient à tracer une droite à une hauteur égale à 80% de la plus grande zone haute ou basse. Toute combinaison de zones qui commence par une zone basse et se termine par une zone haute et dont la hauteur est supérieure ou égale au seuil est considérée comme un cluster. Les points qui se situent entre une zone basse et haute qui vérifient la condition du seuil

appartiennent sûrement au cluster, mais, les points des zones basses et hautes n'appartiennent pas forcément au cluster. Pour cela, nous allons sélectionner parmi ces points ceux appartenant au cluster.

Dans la Figure 3.2, un cluster commence à partir du point n°2 et se termine au point n°16. Le point 2 fait partie du cluster, car sa valeur d'accessibilité élevée indique qu'il est loin du point 1. Il doit être proche du point 3, parce que le point 3 a une valeur d'accessibilité faible indiquant qu'il est proche du point 2. Un argument similaire s'applique pour le point 16. Pour automatiser ce processus, nous avons besoin de définir une hauteur à partir de laquelle nous pouvons dire qu'un point d'une zone haute ou basse est plus proche des points du clusters ou bien le contraire. Par expérimentation un point d'une zone basse appartient au cluster si son successeur est deux fois inférieur à DAPEB tel que DAPEB: représente la distante d'accessibilité du point le plus élevé dans la zone basse du Cluster. Un point d'une zone haute appartient au cluster, s'il est deux fois inférieur à DAPEH, tel que DAPEH: représente la distante d'accessibilité du premier point (élevé) après la fin d'une zone haute du Cluster. L'algorithme est donné comme suit :

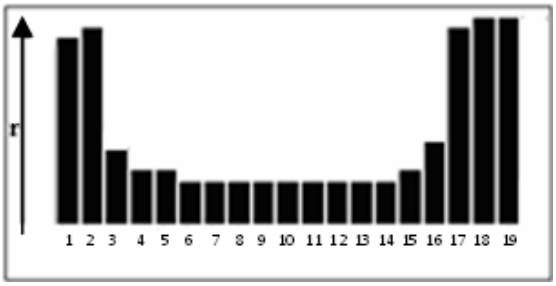


Figure 3.2 : Les objet dans un cluster.

EnsZoneRaide : contient le début de chaque zone basse et la fin de chaque région haute. **EnsDebutFin** contient les valeurs d'accessibilité du début de chaque zone basse et du premier point après la fin de chaque zone haute qui forme le cluster.
n : le nombre d'objet dans la base de données.

Etape 1

indice =0, EnsZoneRaide=∅, EnsDebutFin= ∅ ;
r (p): la distance d'accessibilité du point p
Tant que indice < n faire.

(1) Si une région basse commence à indice, elle sera ajoutée à EnsZoneRaide
Debut =r (indice) ; on parcourt les points suivants dans l'ordre des objets.

(2) Si une nouvelle zone haute commence à indice, elle sera combinée avec les zones basses dans EnsZoneRaide. On vérifie chaque combinaison pour remplir les trois premières conditions du cluster **[3]**.

Si cela est vérifié alors

Fin=r(indice+1) ;

Ajouter la valeur de Début et de Fin à EnsDebutFin et continuer à droite.

(3) Sinon incrémenter indice.

Etape 2

max = 80% du plus grand pic dans EnsDebutFin.

Comparaison des valeurs dans EnsDebutFin à max et suppression des valeurs inférieures à max et aussi suppression des zones basses et hautes correspondantes dans EnsZoneRaide.

Après le filtrage de EnsZoneRaide, les valeurs restantes correspondent réellement aux points appartenant à des zones hautes et basses qui forment les clusters.

Etape 3

Ajout des points entres deux zones basse et haute dans EnsZoneRaide aux clusters et filtrage des points des zones basses et hautes pour sélectionner que ceux qui appartiennent aux clusters.

Le filtrage s'effectue de la manière suivante :

Pour tous point x_i d'une zone haute dans EnsZoneRaide

Si $2*x_i \leq$ DAPEB **alors** Ajout x_i au cluster

Pour tous point x_i d'une zone basse dans EnsZoneRaide

Si $2*x_{i+1} \leq$ DAPEH **alors** Ajout point xi au cluster

3.3 Définition des modèles locaux (les représentants des clusters)

Après avoir effectué le clustering sur chaque site, nous avons besoin de représentants qui décrivent le résultat du clustering local. Nous devons trouver un compromis entre les deux contraintes suivantes : avoir un petit nombre de représentants et avoir une description précise d'un cluster local. Un modèle local ne devrait pas être complexe, parce que nous voulons former des modèles globaux de ces modèles locaux.

Notre solution consiste à calculer les bordures de chaque cluster local. Les bordures peuvent servir de très bons représentants. Chaque cluster dans un site local est représenté dans

un site d'agrégat par sa bordure. Dans les sites d'agrégats, nous avons besoin de régénérer les clusters à partir des bordures reçues. Pour cela, nous allons introduire d'autres concepts tel que la moyenne des distances entre les points de chaque cluster, la moyenne des distances noyaux dans chaque cluster et le signe du vecteur d'équilibre de chaque point bordure que nous définissons par la suite.

3.4 Détermination des bordures

La recherche des frontières d'un cluster est un problème difficile, car les clusters peuvent avoir des formes concaves avec des trous comme l'illustre la Figure 3.3. Les données peuvent avoir aussi plusieurs dimensions. Un cluster ne contient pas uniquement des objets principaux, mais aussi ceux qui ne satisfont pas la condition du noyau principal. Ces objets sont appelés *Objets bordures* du cluster. En premier lieu, nous nous sommes basés sur cette définition du cluster avec d'autres manipulations pour le calcul des bordures. La Figure. 3.4 représente les clusters avec leurs bordures calculées en utilisant cette définition. Les points en noir représentent les bordures. Nous remarquons que l'on peut trouver des points bordures même à l'intérieur du cluster, cela revient à la différence de densité entre les régions (présence des trous) et aussi les points qui représentent les frontières sont plus denses quant les données initiales sont plus denses.

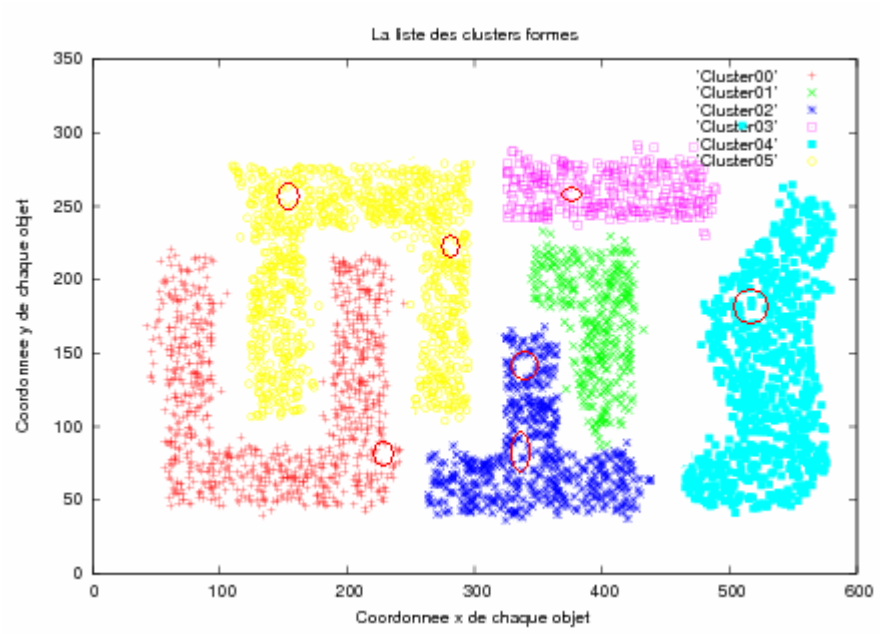


Figure 3.3 : Clusters avec des trous.

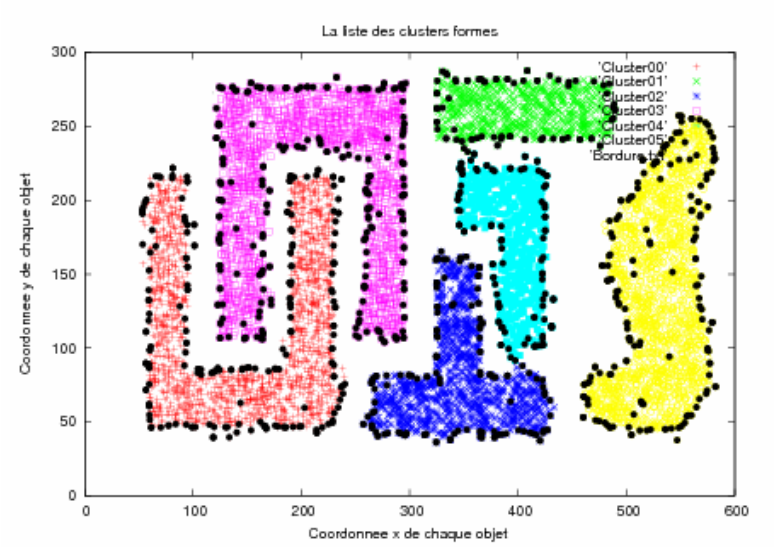


Fig. 3.4 : représentation des bordures des clusters avec la première méthode.

Pour remédier à tous ces problèmes, nous avons opté pour la méthode **[33]**. Cette méthode est basée sur la création du vecteur d'équilibre.

Le vecteur d'équilibre

Soit p_i le point dont nous souhaitons avoir le vecteur d'équilibre. Soit p_j j={1..n} tous les points aux voisinages de p_i, n : est le nombre d'objets dans le cluster. $N_\varepsilon(p_i)$: représente le voisinage a une distance ε du point p_i. Le vecteur d'équilibre est défini comme la moyenne de tous les vecteurs allant du point p_j au point p_i. Le vecteur d'équilibre est représenté dans la Figure.3.5 avec une flèche rouge.

$$\vec{v}_i = \frac{\sum_{p_j \in N_\varepsilon(p_i)} p_i - p_j}{\left|N_\varepsilon(p_i)\right|}$$

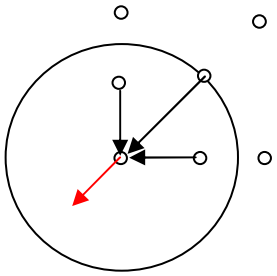


Fig.3.5 : Le vecteur d'équilibre.

Le voisinage d'un point p_i est donné par le minimum de points les plus proches de ce point, c'est-à-dire, les voisins à une distance inférieure ou égale à la distance noyau du point

p_i. Le minimum de points est une valeur d'entrée de l'algorithme. Le vecteur d'équilibre est souvent orienté vers l'extérieur du cluster. Si un point est un point bordure, alors il ne doit pas y avoir des points dans son entourage dans la direction du vecteur d'équilibre.

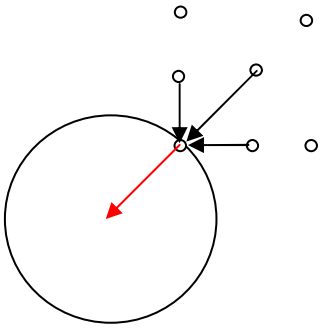


Figure 3.6 : Représentation des alentours dans la direction du vecteur d'équilibre.

Le code de l'algorithme :

Soit C un cluster, B(C) les points bordures du cluster C, ρ>0 la distance noyau et n_p la norme du vecteur d'équilibre du point p.

Pour tout point $p_i \in$ C faire
  $p_i' = p_i + \rho n_i$
  Pour tous point $p_j \neq p_i$ dans C faire
    Si $dist(p_i', p_j) < \rho$ alors
      Enlever le point p_j dans B(C)
    Fin si
  Fin pour (1)
Fin pour

Figure 3.7 : Première partie de l'algorithme de calcule de bordures.

Cet algorithme consiste à calculer le vecteur d'équilibre de chaque point dans le cluster, pour chaque point nous calculons sa distance par rapport aux autres points du cluster. Les points avec des distances inférieures à la distance noyau sont supprimés dans l'ensemble des bordures qui est initialisé à l'ensemble de tous les points du cluster.

Cet algorithme donne de très bons résultats même pour les clusters avec des régions de différentes densités et avec des trous, mais, ils ont remarqué que les bordures sont plus denses lorsque les données initiales sont plus denses. Pour cela, ils appliquent aussi la deuxième partie de l'algorithme qui permet d'éliminer certains points dans les régions denses **[33]**:

```
Soit B(C)  les points bordures du cluster
Pour tout points p_i dans B(C)  faire
    Pour tout point p_j ≠ p_i dans C faire
        Si dist(p_i,p_j)< ρ alors
            Enlever le point p_j dans B(C)
        Fin si
    Fin pour                    (2)
Fin pour
```

Figure 3.8 : deuxième partie de l'algorithme de calcule de bordures.

Cet algorithme calcule la distance de chaque point bordure par rapport à tous les autres points et élimine ceux qui ont leur distance inférieure à la distance noyau.
Cet algorithme nous donne de très bons résultats que nous allons représenter dans le chapitre suivant.

3.5 Régénération des Clusters

La régénération des clusters est effectuée dans les sites d'agrégats, à partir des représentants des clusters calculés localement. Elle doit produire des clusters identiques à ceux formés dans les sites locaux. Les bordures calculées ne suffissent pas pour la représentation des clusters dans les sites d'agrégats, car, elles ne nous donnent aucune information sur l'état du cluster (s'il est dense ou pas) si on les utilise toutes seules. Nous avons besoin d'introduire d'autres notions qui peuvent nous fournir ces informations de manière précise. Il s'agit de calculer la moyenne des distances entre les points dans un cluster, qui sera utilisée dans les sites d'agrégats comme la distance entre deux points régénérés. La régénération des clusters peut être un processus très compliqué surtout si les clusters ont des formes compliquées. Pour remédier à ce problème, nous avons introduit la notion du signe du vecteur d'équilibre qui permet de montrer le début et la fin du cluster tel que soit sa forme. Le signe du vecteur d'équilibre est calculé pour touts points bordures. Il est défini comme suit : le signe du vecteur d'équilibre en un point **i** est calculé par rapport à un nouveau repère définit au point **i**. Si le vecteur est orienté à gauche du point **i** alors le signe est négatif sinon il est positif. Le calcul de la distance noyau et du signe du vecteur d'équilibre peut être effectué lors de calcul des bordures. L'algorithme utilisé pour le calcul des bordures peut être modifié de la manière suivante :

Moy-Dist-MinPts: représente la moyenne des distances entre un point i et ses MinPts voisins dans un cluster, MDP : représente la moyenne des distances entre les points dans un cluster. Moy-Dist-Noy : la moyenne des distances noyaux dans un cluster.
Som-Dist-Pts=0 ;
Moy-Dist-Noy=0 ;
Pour tout point *i* dans C faire
    Recherche des MinPts voisin de i et calcule de la distance de i à tous ses MinPts
    Moy-Dist-MinPts =∑ dist(i, MinPts voisin)/MinPts
    Calculer la distance noyau ρ de i
    $i'=i+\rho n_i$
    Som-Dist-Pts = Som-Dist-Pts+ Moy-Dist-MinPts
    Moy-Dist-Noy= Moy-Dist-Noy+ρ ;
    nb=nb+1, var=0
    Pour tout point *j != i* dans C faire
        Si *dist(i, j)*< ρ alors, var=1
    Fin pour
    Si (var= =0)
        Ajouté i a B(C)
        // Calcule du signe du vecteur d'équilibre.
        Si (i_x<0), // i_x: coordonné x du point i.
            Signe(i)='-'
        Sinon,
            Signe(i)='+'
        Fin si
    Fin si
Fin pour
MDP= Som-Dist-Pts/nb

Moy-Dist-Noy= Moy-Dist-Noy /nb

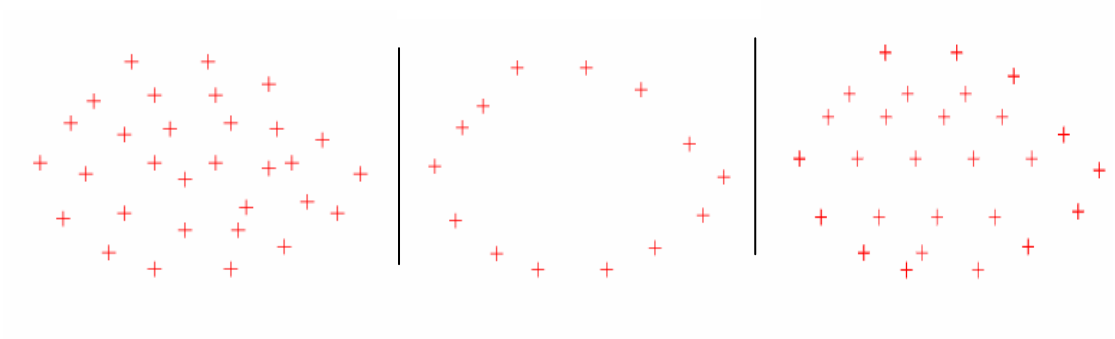
Figure 3.9 : Algorithme de calcule des représentants.

Cet algorithme calcule la moyenne des distances entres les points de chaque cluster noté MDP, la moyenne des distances noyaux Moy-Dist-Noy et le signe du vecteur d'équilibre de chaque point bordure. Ces informations aident dans la régénération des clusters.

La Figure 3.10 représente les bordures d'un cluster, le signe du vecteur d'équilibre des points en rouge est négatif et les points en vert ont le signe du vecteur d'équilibre positif.

Figure 3.10. Spécification du signe du vecteur d'équilibre pour chaque point bordure
(+) Le signe du vecteur d'équilibre est négatif. (x) Le signe du vecteur d'équilibre est positif.

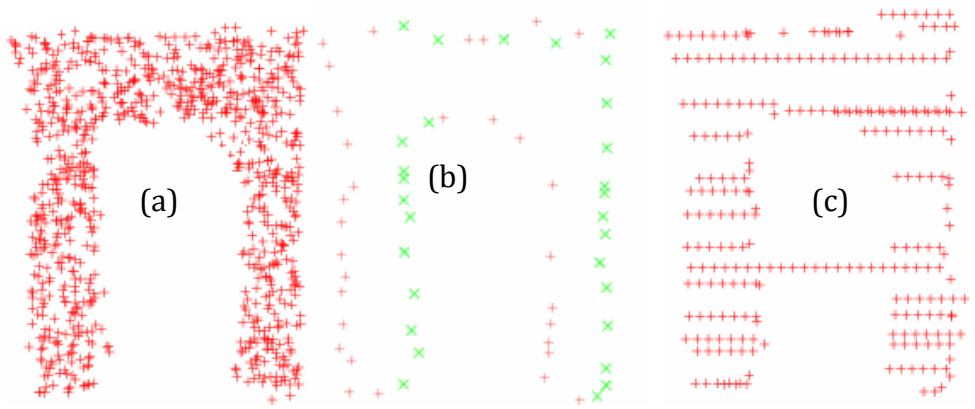
Pour régénérer un cluster similaire au cluster original à partir des bordures, nous sélectionnons tous les points bordures dont le signe du vecteur d'équilibre est négatif. A partir de chacun de ces points, d'autres points seront régénérés. La distance entre un point bordure et un point régénéré ou bien entre les points régénérés est égale à MDP (la moyenne des distances entre les points dans un cluster). Nous régénérons des points à partir de chaque point bordure dont le signe du vecteur d'équilibre est négatif jusqu'à ce que nous rencontrions au moins un point bordure ayant le signe du vecteur d'équilibre positif. Dans ce cas, nous régénérons des points à partir du prochain point bordure avec le signe du vecteur d'équilibre négatif. Ce processus est répété jusqu'à ce que nous parcourions tous les points bordures dont le signe du vecteur d'équilibre est négatif. La Figure. 3.11 (a) montre le cluster original, la Figure. 3.11 (b) représente les bordures du cluster et la Figure. 3.11 (c) représente le cluster régénéré.



(a) Cluster original (b) Bordure du cluster (c) Cluster régénéré

Figure 3.11 Etapes de régénération d'un cluster.

La méthode de régénération que nous venons de présenter ne donne pas souvent de bons résultats parce que nous régénérons les clusters qu'à partir les points bordures reçus et dans le cas des clusters denses le nombre de points constituant la bordure est petit par rapport au nombre de point réel à la frontière. Cela permet de minimiser les communications, mais la régénération à partir de ces points ne donne pas un cluster avec la même densité que le cluster original, parce que nous ignorons des points qui doivent être entre ceux des bordures et à partir desquels, nous devons aussi régénérer d'autres points. D'autre part, pour arrêter le processus de régénération, nous devons rencontrer un point positif et si nous n'avons pas tous les points positifs de la frontière, nous risquons de régénérer des points qui ne font pas partie du cluster et que le processus de régénération risque de boucler infiniment, la Figure.3.12 (c) montre ces difficultés.



(a) Cluster original (b) Bordures (c) Le cluster régénéré.

Figure 3.12 Limites de la régénération des clusters avec la méthode 1.

Pour remédier à tous ces problèmes, nous avons besoin de régénérer d'abord tous les points bordures. A partir de ces nouvelles bordures nous régénérons des clusters similaires aux clusters originaux. La distance entre un point bordure et un point régénéré ou bien entre les points régénérés est égale à MDP (la moyenne des distances entre les points dans un cluster).

Régénération des bordures

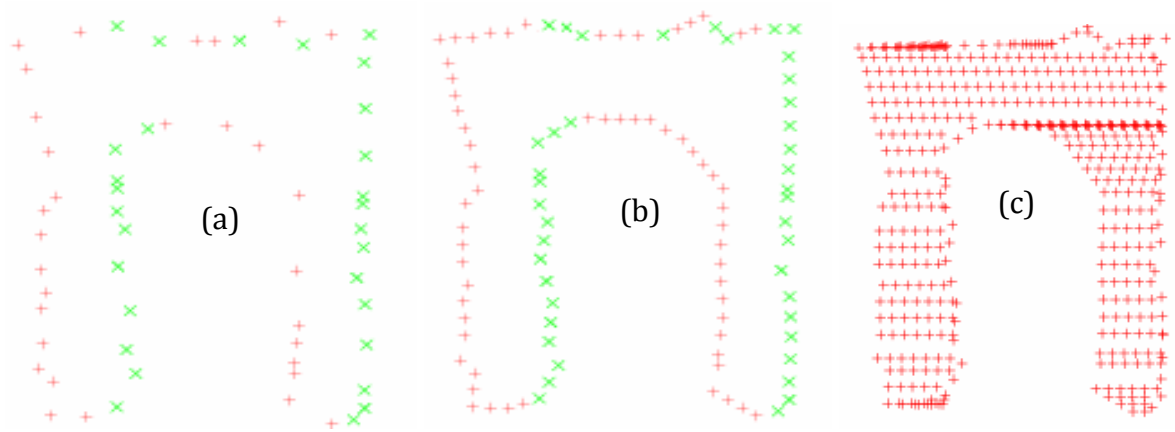
Pour avoir une bordure complète qui permet de régénérer des clusters similaires aux clusters originaux, nous allons procéder comme suit :

Initialement tous les points bordures sont marqués comme non traités. Pour chaque point p non traité nous cherchons le point le plus proche non traité noté p1, nous marquons le point p comme traité et nous allons régénérer des points entre ces deux points (p, p1). Si le signe du vecteur d'équilibre du point p est négatif alors le signe des points régénérés est négatif aussi, sinon si le signe est positif alors le signe des points régénérés est positif. La distance entre les points régénérés est égale à MDP. Nous allons continuer la régénération à partir du point p1 en lui cherchons son voisin le plus proche. La procédure est continuée jusqu'à ce que tous les points soient traités. Dans la procédure suivante nous avons marqué le premier point traité avec un 1 et les autres points traités par 2 pour permettre de continuer la régénération jusqu'au point de départ.

Régénération (B(C)) // B(C) : Bordures du cluster C,
Traite[i] : tableau indiquant si les objets du cluster sont traités ou pas, initialisé à 0, i={1,.. , nb}, nb : nombre de points bordures dans B(C).
Tant que ∃ un point dans B(C) !=traité
Sélectionné un point p non traité dans B(C) (c-a-d traite[p]==0)
Traite[p]=1 ;
Recherche du point le plus proche de p non traité noté p1 (p !=p1)
Tant que (traite[p1] !=2)
Si p1 est (-)
Régénération des points (-) entre les points (p, p1) à une distance MDP
Sinon si p1est (+)
Régénération des points (+) entre les points (p, p1) à une distance MDP
Fin si
Si (traité[p1] ==0)
p=p1 ;
Recherche du point le plus proche pour p noté p1 (où traite[p1] !=2)
traite[p1]=2
Else Si (traite[p1] ==1)
traite[p1]=2
Fin si
Fin tant que
Fin tant que

Figure 3.13 : Algorithme de régénération des bordures.

Pour régénérer un cluster similaire au cluster original à partir des nouvelles bordures, nous sélectionnons tous les points bordures dont le signe du vecteur d'équilibre est négatif. A partir de chacun de ces points, d'autres points seront régénérés jusqu'à ce que nous rencontrions des points positifs au voisinage. La distance entre un point bordure et un point régénéré ou bien entre les points régénérés est égale à MDP. Durant la régénération, nous récupérons les points régénérés et nous les représentons juste pour voir que cette opération s'effectue réellement que dans la surface délimitée par les bordures calculées (Figure.3.14 (c))



(a) Cluster original (b) Bordure régénéré (c) Cluster régénéré

Figure.3.14. Régénération des clusters après la régénération des bordures.

6 Modèle de distribution

Chaque site qui termine son exécution séquentielle doit pouvoir décider avec celui qu'il effectuera son agrégation. Pour des contraintes de programmation, nous avons utilisé une topologie en anneau pour les processeurs. Les processeurs sont disposés en anneau où chacun possède deux voisins : l'un à sa gauche et l'autre à sa droite. Chaque processeur doit avoir les informations du voisin qui est à gauche. Cela veut dire que chaque processeur à gauche doit envoyer ses représentants (les bordures, …) à son voisin de droite. Si un processeur reçoit les clusters de son voisin de gauche, il regarde s'il y a un recouvrement avec ses clusters formés localement. Si tel est le cas, les nouveaux représentants de clusters seront formés et seront envoyés au voisin de droite. Dans le cas contraire, le processus envoie ces propres représentants et ceux reçus des voisins de gauche au processeur voisin de droite. Ce processus sera répété jusqu'à ce qu'une condition de terminaison est satisfaite. La condition de terminaison est atteinte si l'agrégation de tous les sites est effectuée. La procédure d'agrégation permet de tester les chevauchements entre les clusters de deux sites différents. Elle sera détaillée dans la section suivante. L'algorithme suivant résume le modèle de distribution.

Si le site S_i a terminé son exécution séquentielle **alors**
  Si S_i reçoit des représentants à partir de S_{i-1} **alors**
    Si (S_j n'a pas déjà effectué son agrégation) **alors**
      Agrégation (représentants de S_i, représentants reçus à partir de S_{i-1})
      Envoyer les nouveaux représentants au site S_{i+1}
    Sinon
      S_i envoie les représentants reçus de S_{i-1} à S_{i+1}
    Fin si
  Sinon
    Envoi les représentants de S_i au site S_{i+1}.
  Fin si
Sinon
  Si un site reçoit les représentants et il n'a pas encore terminé son exécution **alors**
    Envoyer les représentants reçus de S_{i-1} au site S_{i+1}
  Fin si
Fin si

Figure 3.15 : Algorithme de distribution d'OPTICS

3.7 Modèles globaux

Chaque cluster dans un site d'agrégat est représenté par ses bordures et par le signe du vecteur d'équilibre de chaque point bordure et aussi par la moyenne des distances entre les points du cluster. Une fois les représentants sont transférés aux sites d'agrégat, les bordures sont régénérées en faisant appel à la procédure **Régénération ( )** et par la suite nous testons s'il existe des clusters qui se chevauchent en appelant la procédure **test_chevauchement ( )**.

Le test de chevauchement peut se faire en même temps que la régénération des clusters. Lors de ce test nous parcourons d'abord le cluster le plus à gauche, car la régénération se fait de gauche à droite, alors nous régénérons le cluster de gauche et nous testons s'il existe des chevauchements avec un autre cluster situé à droite. Dans la procédure suivante, nous effectuons l'agrégation entre les clusters du site S_1 et ceux du site S_2. Nous régénérons les bordures des clusters des deux sites et nous testons les chevauchements entres eux.
Soient S_1 et S_2 deux sites, Cluster1 : cluster du site S1, Cluster2 : un cluster du site S2.

```
Agrégation (Représentants de S1, Représentants de S2)
Pour tout cluster appelé Cluster1 du site S1 faire
    Régénération (Bordures de Cluster1)
    Pour tout cluster appelé Cluster2 du site S2 faire
        Régénération (Bordures de Cluster2)
        Si le Cluster1 est plus à gauche que le Cluster2 alors
            test_chevauchement (Bordures-Cluster1,Bordure-Cluster2)
        Sinon
            test_chevauchement (Bordures-Cluster2,Bordures-Cluster1)
        Fin si
    Fin pour
Fin pour
```

Figure 3.16 : La procédure Agrégation

On dit que deux clusters se chevauchent, s'il y a un recouvrement entre deux clusters, ou bien s'il existe plus de MinPts points bordures d'un cluster à une distance inférieure à MDP d'un autre cluster. Les bordures des nouveaux clusters formés sont calculées lors du test de chevauchement des clusters. Le nombre de bordure formé est élevé. Pour minimiser les communications, nous appliquons la partie (2) de l'algorithme de calcul des bordures (section 3.4), mais au lieu de filtrer par rapport à la moyenne des distances noyaux de chaque objet bordure, nous le faisons par rapport à la moyenne des distances noyaux pour chaque cluster. L'application de cet algorithme permet de laisser qu'un petit nombre de bordure qui représente bien le nouveau cluster dans un autre site d'agrégat. La moyenne des distances noyaux correspondante pour le nouveau cluster formé est égale à la moyenne des distances noyaux des clusters fusionnés. La moyenne des distances entre les points dans le nouveau cluster est égale à la moyenne des moyennes des distances entre les points des clusters fusionnés

La procédure **test_chevauchement ( )** Figure 3.17, permet de régénérer les clusters, de tester les recouvrements entres eux et de former les nouvelles bordures des nouveaux clusters dans un site d'agrégat. Les différents cas possible de chevauchements sont représentés dans la Figure 3.18.

B1, B2 : bordures des clusters C1 et C2. NC : nouveau cluster, NB : nouvelles bordures, MDP (NC) : la moyenne des distances entres les points dans le cluster NC, MDN : moyenne des distance noyaux dans le cluster NC

```
test_chevauchement (B1, B2)
fusion=0 ;
pour tout point i aux bordures du cluster C1 faire
    variable =vrai ; nb=0 ;
    tant que (variable==vrais) alors
        recherche des voisins de i dans B2
        si l'ensemble des voisins de i noté j dans B2 != ∅ faire
            si (signe (i) ==’+’) et ( signe (j) ==’-’) alors // test des frontières
                nb=nb+1 ;
                Sauvegarder i et j dans tab1.
            Sinon //---------- recouvrement ----------
                NB= B1+B2. // NB l'ensemble des nouvelles bordures formés
                Variable=faux ; fusion=1 ;
                si ((signe (i) ==’-’) et (signe (j) ==’-’))
                    Éliminer les points d'intersection j dans NB.
                    Recherche des voisins de j dans B2
                    tant que l'ensemble de voisin de j== ∅ alors
                        Recherche les voisins l de j dans B1
                        si l'ensemble de voisin l != ∅ alors
                            Supprimer les points l dans NB ;
                        sinon Régénération d'un point j
                            Recherche les voisins de j dans B2
                    fin tant que
                    si l'ensemble k de voisin de j != ∅ alors
                        Eliminer les voisins de k dans NB
                fin si
                Si (tab1 != ∅)
                    Eliminer les éléments de tab1 dans NB
            fin si
        sinon Si signe(i==’-’)
                régénération d'un point i
                Recherche les voisins de i dans B1.
                si (V(i)==∅) alors Variable=vrai.
                else Variable=false.
        fin si
        si (signe(i)==’+’) alors variable=faux;
    fin tant que
fin pour
Si ((nb>=MinPts) et (fusion=0)) alors // fusionner les deux clusters
    NB= B1+B2, fusion=1 ;
    Eliminer les éléments de tab1 dans NB
Fin si
Si (fusion==1)
    MDP(NC)=MDP(C1)+ MDP(C2)/2
    MDN(NC)=MDN(C1)+ MDN(C2)/2
    Filtrage (NB)// appliquer la partie 2 de l'algorithme du calcule des bordure (section 3.4)
Fin si
```

Figure 3.17 : La procédure **test_chevauchement**.

Pour chaque cluster du site 1 appelé « cluster1 », nous vérifions s'il se chevauche avec chacun des clusters du site 2 noté « cluster 2 ». On commence par régénérer le cluster le plus à gauche et nous testons s'il se chevauche avec un autre appartenant à un site différent. Nous supposons que le cluster1 est le plus à gauche, alors nous allons commencer par parcourir les points bordures du cluster1.
Si le signe du vecteur d'équilibre du point à traiter est positif alors on est à la fin du cluster 1, on vérifie s'il y a des points du cluster 2 dans son voisinage. Si tel est le cas alors

- Si le point au voisinage est négatif (on est à la fin du premier cluster et au début du deuxième cluster). Dans ce cas, nous sauvegardons les points d'intersection et si le nombre de points d'intersection dépasse MinPts alors les deux clusters sont fusionnés et les points d'intersections sont éliminés dans le nouveau cluster formé (Figure 3.18 (c)).

Si le signe du vecteur d'équilibre du point à traiter est négatif (on est au début du cluster), alors tant que nous ne trouvons pas de points positifs du même cluster au voisinage de ce dernier alors nous régénérons d'autres points pour ce cluster.

- Si dans le voisinage du point négatif ou du point régénéré, nous trouvons des points du cluster 2, alors nous éliminons tous ces points d'intersections dans le nouveau cluster formé. Si le point au voisinage est négatif alors nous continuons la régénération à partir de ce dernier (le point du cluster 2 qui est au voisinage). Nous régénérons des points dans le cluster 2 tant que nous ne trouvons pas au voisinage de ce dernier des points du cluster 2 ou du cluster 1. La distance entre les points régénérés est égale à MDP qui est calculé dans le cluster 2. Si nous rencontrons un point du premier cluster au voisinage du point régénéré alors nous le supprimons (Figure 3.18 (a)), sinon si nous ne rencontrons aucun point du premier cluster alors, nous éliminons les points du deuxième cluster rencontré (Figure 3.18 (b))

Les schémas suivants représentent les différents cas possibles pour les chevauchements de deux clusters appartenant à deux sites différents.

+ Bordure dans le site1 × Bordure dans le site2

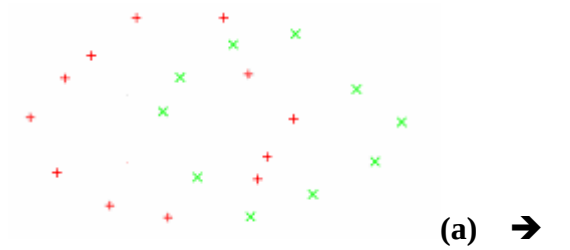
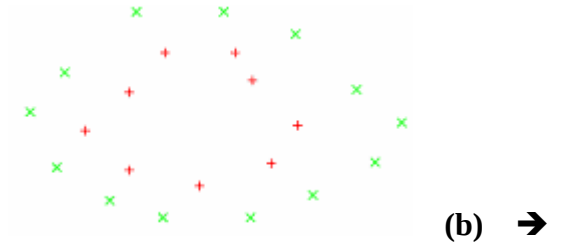
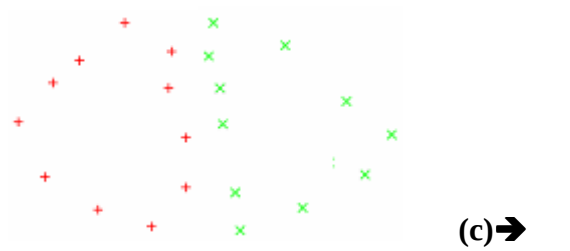
Deux clusters qui se chevauchent	Résultat après le processus d'agrégation
 **(a)** ➔	
 **(b)** ➔	
 **(c)**➔	

Figure.3.18 les différents cas possible pour les chevauchements de deux clusters.

3.8 Complexité de l'approche

Notre approche consiste à exécuter d'abord l'algorithme OPTICS dans chaque site, ensuite nous calculons les représentants locaux. Ces derniers seront transférés aux sites d'agrégats selon un modèle distribué bien défini. Au niveau de chaque site d'agrégat les clusters seront régénérés pour tester s'il existe des recouvrements avec d'autres clusters. Le temps d'exécution de notre approche est donné comme suit :

Le temps d'exécution de l'algorithme OPTICS séquentiel dans un site + le temps de calcul des bordures + le temps de régénération des bordures + le temps du test des recouvrements et de génération des clusters + le temps des communications.

La complexité de l'algorithme OPTICS séquentiel est donnée par : *O (N*temps d'exécution d'une requête de ε–voisinage dans un site)* où N est le nombre d'objets dans le

site qui contient un plus grand nombre d'objets. La requête de *ε–voisinage* est calculée par rapport au nombre d'objet N.

La complexité de l'algorithme du calcul des bordures est donnée par : *O (M*L*temps d'exécution d'une requête de ε–voisinage dans un cluster)* où M est le nombre de clusters dans un site, L est le nombre d'objet dans le cluster qui contient le plus d'objets. La requête de calcule de *ε–voisinage* est donnée par rapport à L.

Le temps de régénération des bordures est négligeable devant le temps d'exécution des requêtes de *ε–voisinage* parce que le nombre de point bordure est inférieur à 3 % du nombre de points dans un cluster **[33]**.

Le temps de communication est aussi négligeable devant le temps d'exécution des requêtes de *ε–voisinage* parce que nous transportons qu'un petit nombre de représentants qui est donnée en fonction du nombre de représentants et de la vitesse de communication entre les différents sites.

Le test de recouvrement et de régénération des clusters est effectué en même temps. Le temps d'exécution est négligeable devant le temps de calcul des requêtes de *ε–voisinage* parce que la régénération et faite qu'à partir des points bordures. Les voisinages de chaque point sont calculés par rapport aux points bordures.

La complexité de notre approche est polynomiale, elle est donnée par : *O (N*temps d'exécution d'une requête de ε–voisinage dans un site) + O (M*L*temps d'exécution d'une requête de ε–voisinage dans un cluster).*

3.9 Conclusion

Dans ce chapitre, nous avons proposé une approche distribuée pour l'algorithme de clustering OPTICS. Notre approche met en valeur les points forts de l'algorithme séquentiel et corrige ses faiblesses. Elle est basée sur trois étapes principales : Dans la première étape, nous avons défini le clustering local qui consiste à exécuter l'algorithme OPTICS séquentiel dans chaque site et à déterminer les clusters à partir des graphes des distances d'accessibilités. La deuxième étape consiste à extraire les représentants de chaque cluster qui sont envoyés au site d'agrégat. La dernière étape sert à régénérer les clusters afin de tester les recouvrements pour construire les clusters globaux. Enfin, nous avons estimé la complexité de notre

approche qui est polynomiale. Notre approche sera évaluée et comparée à la version séquentielle dans le chapitre suivant.

Chapitre 4

Evaluation et Expérimentation

4.1 Introduction

Dans ce chapitre, nous représentons les résultats expérimentaux de notre approche et ceux de l'approche séquentielle. Nous évaluons les deux approches selon les résultats obtenus. L'implémentation de notre approche distribuée est faite en utilisant le langage de programmation MPI.

Le chapitre est organisé de la manière suivante : Nous décrivons d'abord l'environnement de programmation, ensuite nous présentons les résultats de l'implémentation de l'algorithme OPTICS séquentiel sur différents jeux de données. Nous donnerons les résultats de l'implémentation de l'algorithme des bordures, leurs reproductions, la régénération des clusters ainsi que l'exécution du processus d'agrégation sur trois sites. Enfin, nous évaluons notre approche par rapport à l'approche séquentielle.

4.2 L'environnement de programmation

La première question qui vient à l'idée avant de faire la parallélisation d'une application séquentielle est de savoir si cette dernière est vraiment nécessaire, car la conception d'un logiciel séquentiel est une tâche difficile en elle-même; la parallélisation la rendra encore plus dur. L'application séquentielle utilisée consomme trop de ressources en temps d'exécution ou en mémoire. La seule solution, pour des raisons techniques ou économiques, reste la parallélisation. Pour un programmeur, la différence la plus importante entre les superordinateurs est la question de la mémoire partagée ou distribuée. Évidemment, il existe aussi des effets créés par l'architecture du processeur, le système d'exploitation, etc. Cependant, toutes ces considérations ne changent pas grand chose dans l'écriture du code lui-même. Avec un système à mémoire partagée, un programme parallèle consiste en un groupe de files d'exécution (thread). Ces files d'exécution partageront le même espace d'adressage dans la mémoire. Nous n'avons pas besoin de nous occuper des besoins de la communication entre ces files d'exécutions parce qu'ils ont tous des accès à la même mémoire. Avec un système à mémoire distribuée, par contre, il n'y a plus des files d'exécution partageant un seul espace de mémoire. Chaque processus est indépendant avec sa propre mémoire. Cela implique que le programmeur doit ajouter des fonctions explicites pour que les données d'un processus soient accessibles aux autres. Il y a plusieurs manières de programmer un système à mémoire distribuée: MPI, PVM et autres. La norme la plus répandue de nos jours est MPI **[34]**. Avec MPI, nous utilisons un échange de messages entre les noeuds pour accomplir la transmission des données, d'où son nom en anglais, *Message Passing Interface*. Une application MPI est typiquement du genre SPMD (*single program, multiple data*). Cela veut dire que chaque processus tourne le même programme que les autres. Comme le programmeur est responsable de cet échange de messages et de la synchronisation des données, il est clair que la parallélisation par MPI est plus exigeante. Cela dit, une application MPI offre l'avantage de la portabilité.

4.3. Implémentation

Nous avons exécuté notre approche en utilisant plusieurs jeux de données artificiels à deux dimensions. Certains jeux de données contiennent des milliers d'objets et d'autres avec des centaines d'objets. Un objet dans un jeu de données est représenté par ses coordonnées. Dans la suite de ce chapitre, nous montrons les résultats d'implémentation de notre approche.

4.3.1 Implémentation de l'approche séquentielle

L'algorithme OPTICS génère un ordre sur les objets de l'ensemble de données: les points sont ordonnés suivant leurs densités par rapport aux deux paramètres ε et MinPts. La structure de clustering peut être visualisée en représentant graphiquement les valeurs des distances d'accessibilité r_i en fonction des points ordonnés o_i. Ces valeurs d'accessibilité sont stockées dans un fichier. La représentation graphique des distances d'accessibilité dans le fichier est faite en utilisant le logiciel 'Gnuplot'. Nous présentons quelques résultats d'exécution de l'algorithme OPTICS. Les Figures (Figure 4.1 (a), Figure 4.2 (a) et Figure 4.3 (a)) montrent les données initiales pour plusieurs jeux de données. Les Figures (Figure 4.1 (b), Figure 4.2 (b) et Figure4.3 (b)) montrent une représentation graphique de l'ordre fournit par l'algorithme OPTICS pour chaque jeu de données.

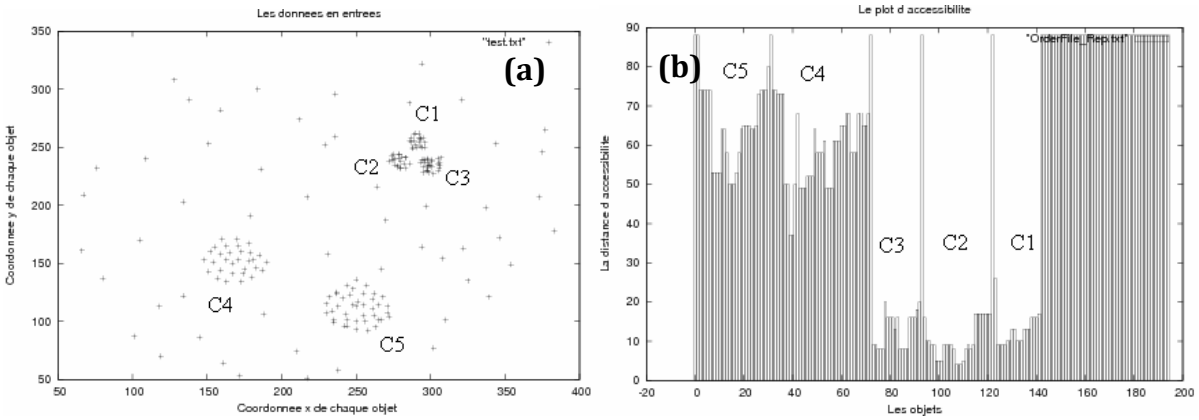


Figure 4.1 : Jeux de données 1 (a) les données initiales (b): Le graphe d'accessibilité

Dans la Figure.4.1(a), nous remarquons des clusters avec des densités différentes. En effet, les clusters C1, C2 et C3 sont plus denses que les clusters C4 et C5. Les clusters denses sont représentés dans la Figure 4.1(b) par l'ensemble de points qui ont des distances d'accessibilité petites. Les clusters sont séparés par des points avec des distances d'accessibilité élevés. Les clusters moins denses sont représentés par des points avec des distances d'accessibilité plus élevés. Les points après le clusters C1 dans la Figure.4.1(b) représente un bruit. A partir de ce graphe, nous pouvons facilement identifier les clusters C1, C2, C3, C4, C5.

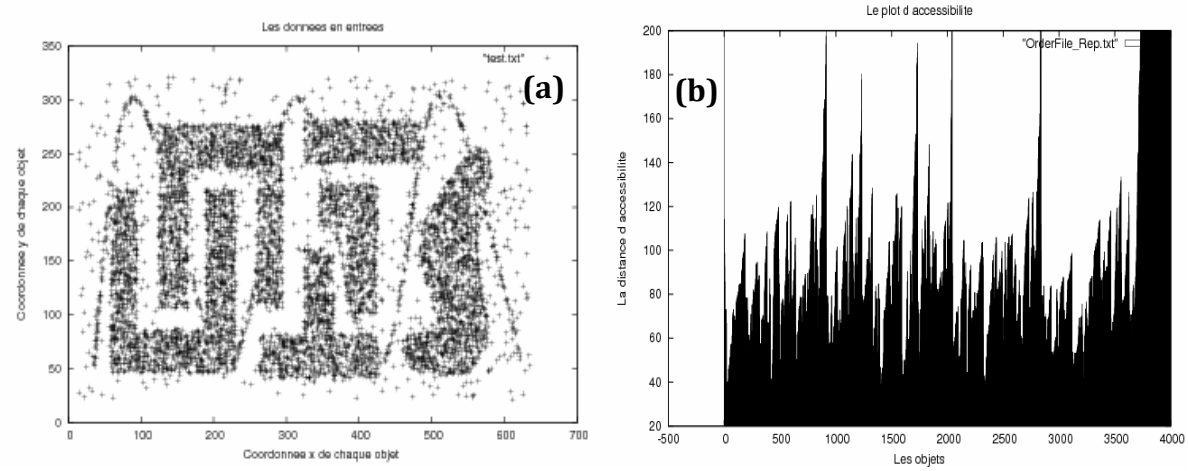


Figure 4.2 : Jeux de données 2 (a) les données initiales, (b): Le graphe d'accessibilité

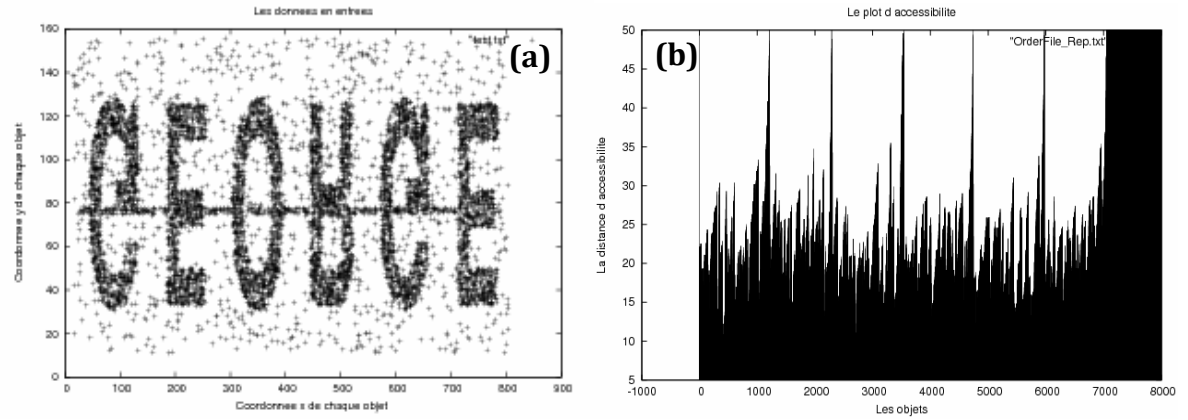


Figure 4.3 : Jeux de données 3 (a) les données initiales, (b): Le graphe d'accessibilité

Les Figures (4.2 et 4.3) (a) représentent les données initiales pour deux jeux de données contenant des milliers d'objets. Dans les Figures (4.2 et 4.3) (b), nous remarquons aussi la présence de plusieurs distances d'accessibilité contrairement à la Figure 4.1 (b). Dans les deux Figures (4.2 et 4.3) (b), nous pouvons remarquer qu'il existe six clusters parce qu'ils sont séparés par des points avec des distances d'accessibilité élèves.

Le graphe des distances d'accessibilité est plutôt insensible aux paramètres ε et MinPts. Ces paramètres sont introduits par l'utilisateur. En général, les valeurs de ces paramètres doivent être juste assez grandes pour conduire à un bon résultat. Les valeurs concrètes de ces paramètres ne sont pas cruciales : la structure en clusters des données est toujours visible dans le graphe des distances d'accessibilité pour un grand intervalle de valeurs possibles et il est indépendant de la dimensionnalité des données. Ceci permet à notre approche d'être valable quelque soit la quantité et la dimension des données.

4.3.2 Détermination des clusters

Après avoir déterminé l'ordre des clusters des jeux de données, nous avons implémenté l'algorithme qui permet d'extraire les clusters à partir de ces graphes des distances d'accessibilité. Nous avons exécuté cet algorithme sur les jeux de données des Figures (4.1, 4.2, 4.3). Les résultats d'exécution sont montrés dans les figures suivantes :

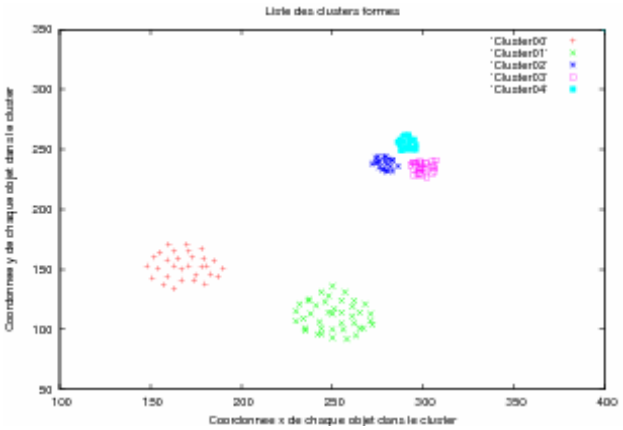


Figure 4.4 Les clusters formés à partir du jeu de données de la Figure 4.1.

La Figure 4.4 représente l'ensemble des clusters calculés à partir du graphe d'accessibilité de la Figure 4.1. Nous remarquons que les cinq clusters (C1, C2, C3, C4, C5) sont bien identifiés et le bruit est éliminé.

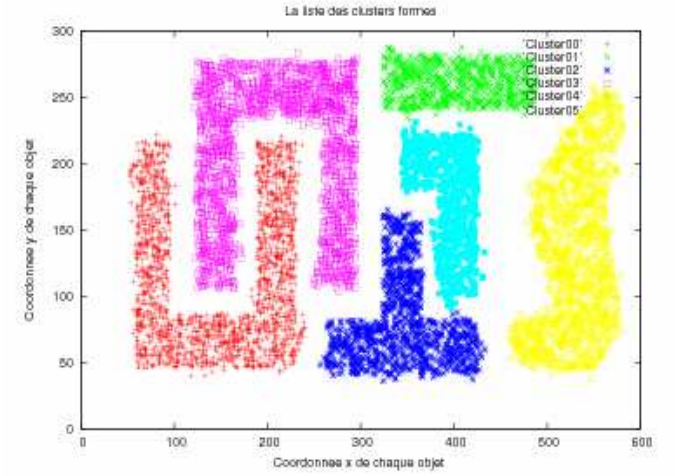


Figure 4.5 Les clusters formés à partir du jeu de données de la Figure 4.2.

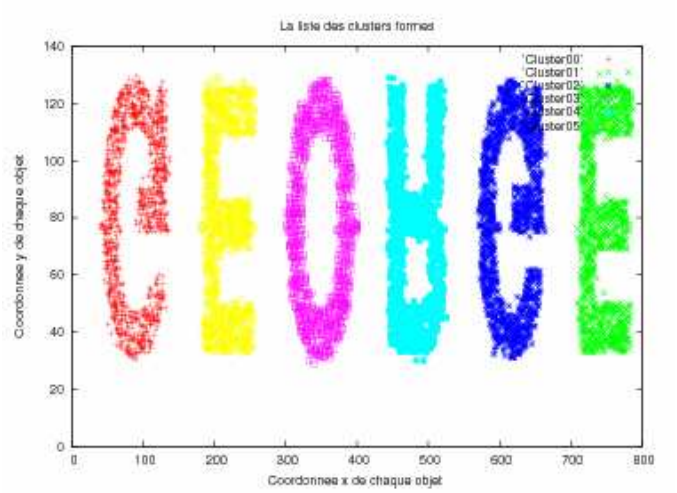


Figure 4.6 Les clusters formés à partir du jeu de données de la Figure 4.3.

Les Figures (4.5, 4.6) représentent l'ensemble des clusters calculés à partir des graphes d'accessibilité des Figures (4.2, 4.3) (b). Nous remarquons que les six clusters dans les deux figures sont bien identifiés et le bruit est éliminé.

Les graphes des distances d'accessibilité dans les deux Figures (4.2, 4.3) (b) contiennent plusieurs pics ; ce qui complique la détermination des clusters. Cependant, la méthode que nous avons proposée pour la détermination des clusters à partir des graphes des distances d'accessibilité basée sur la définition du seuil donne de très bons résultats comme le montre les deux Figures (4.5, 4.6).

4.3.3 Détermination des bordures

Après avoir effectué le clustering sur chaque site, nous avons besoin de représentants qui décrivent le résultat du clustering local. Notre solution consiste à calculer les bordures de chaque cluster local. L'implémentation de la méthode décrite dans **[14]** sur plusieurs jeux de données nous a donné de très bons résultats. Voici quelques exemples de calcul des bordures pour les clusters des Figures (4.4, 4.5, 4.6).

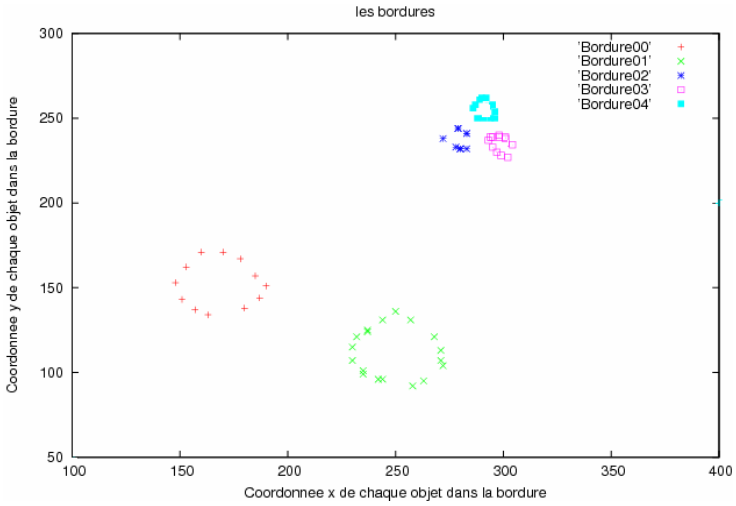


Figure 4.7: Les bordures formées à partir des clusters de la Figure 4.4.

La Figure 4.7 représente l'ensemble des bordures des clusters calculés à partir des clusters de la Figure 4.4. Nous remarquons que les bordures des cinq clusters sont bien identifiées.

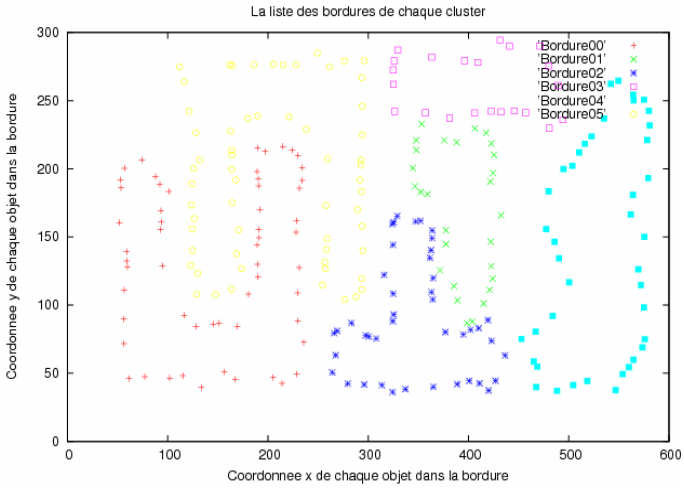


Figure 4.8: Les bordures formées à partir des clusters de la Figure 4.5.

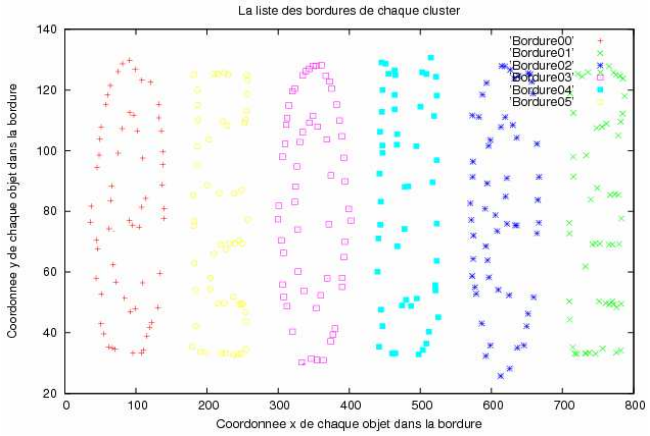


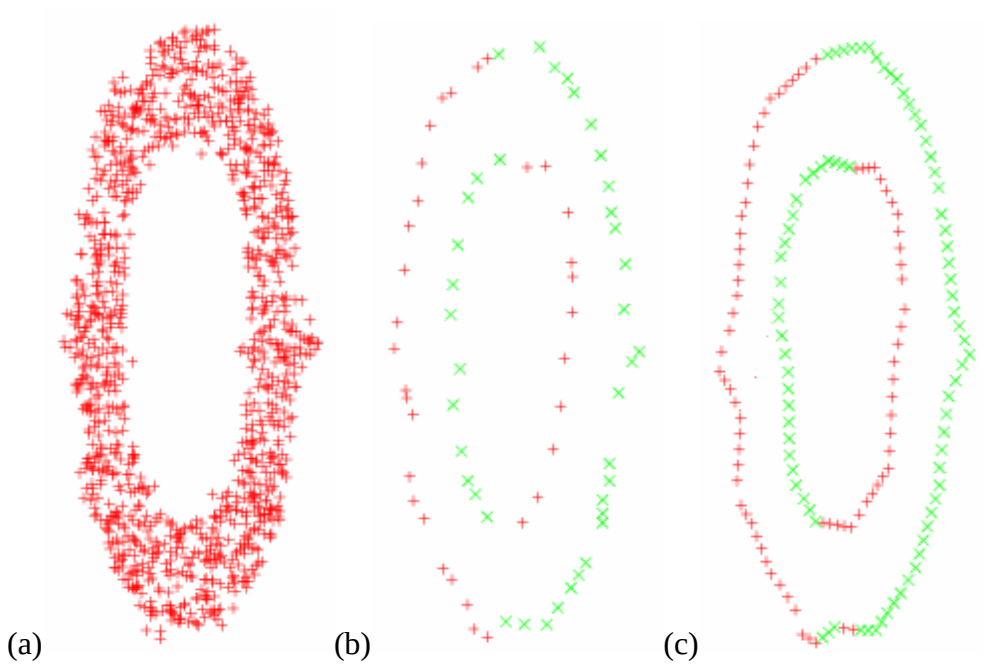
Figure 4.9: Les bordures formées à partir des clusters de la Figure 4.6.

Les Figures (4.8, 4.9) représentent l'ensemble des bordures des clusters calculés à partir des clusters des Figures (4.5, 4.6). Nous remarquons que les bordures des six clusters dans les deux figures sont bien identifiées telles que soit la forme des clusters. Cet algorithme de calcul des bordures présentent l'avantage d'avoir un petit nombre de représentants qui représente de manière efficace les clusters quelque soit leurs formes.

4.3.4. La régénération des bordures

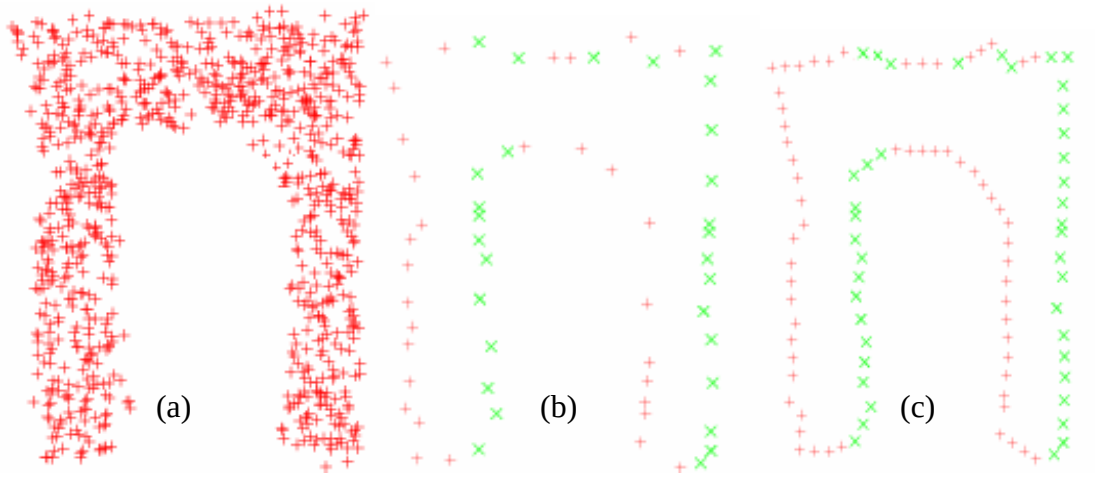
A partir des résultats obtenus pour le calcul des bordures, nous avons remarqué que le nombre de points constituant la bordure est petit par rapport au nombre de point réel à la frontière. Cela permet de minimiser les communications, mais la régénération des clusters à partir de ces points ne donne pas un cluster régénéré avec la même densité que le cluster original, parce que nous ignorons des points qui devraient être entre les points bordures et à partir desquels nous devons aussi régénérer d'autres points.

Nous avons implémenté l'algorithme de régénération des bordures sur plusieurs jeux de données. Dans ces deux exemples, nous montrons les résultats de la régénération des bordures et nous représentons le signe du vecteur d'équilibre pour chaque point bordure. Le signe du vecteur d'équilibre des points en rouge est négatif et ceux en vert est positif.



(a) Cluster original (b) les bordures. (c) régénération des bordures.

Figure.4.10. régénération de bordures pour le premier cluster.



(a) Cluster original (b) les bordures. (c) régénération des bordures

Figure.4.11. Régénération de bordures pour le deuxième cluster.

(+) Le signe du vecteur d'équilibre est négatif. (x) Le signe du vecteur d'équilibre est positif.

Les Figures (4.10, 4.11) (b) représentent l'ensemble des bordures des clusters calculés à partir des clusters des Figures (4.10, 4.11) (a). Les Figures (4.10, 4.11) (c) représentent l'ensemble des bordures régénérées à partir des bordures des Figures (4.10, 4.11)(b). Nous remarquons que les bordures régénérées représentent mieux les bordures des clusters tel que soit leurs formes et densité.

4.3.5. La régénération des clusters

La régénération est effectuée en parallèle avec le test de chevauchements des clusters. A chaque point régénéré d'un cluster, nous vérifions s'il existe des points d'un autre cluster à l'entourage de chaque point. Durant la régénération, nous récupérons les points régénérés et nous les représentons pour vérifier si la régénération s'effectue réellement dans la surface délimitée par les bordures calculées.

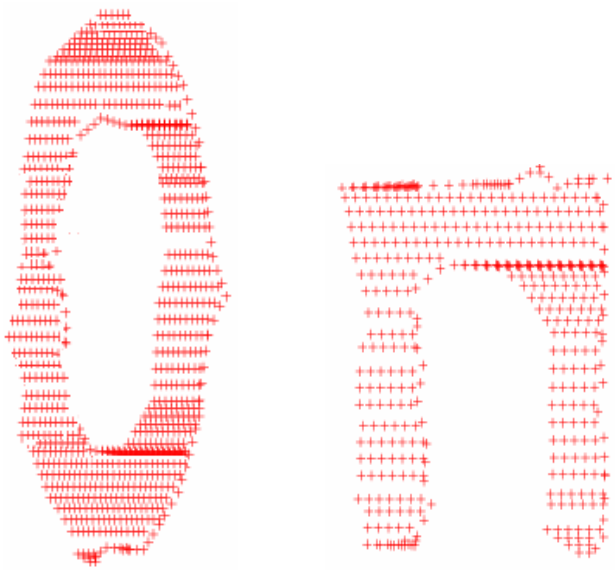


Figure. 4.12. Clusters régénérés

4.3.6 Résultats de l'exécution de l'algorithme OPTICS distribué

Nous avons évalué notre approche sur des jeux de données à deux dimensions. Nous avons effectué le clustering local. Nous avons construit les représentants de chaque cluster dans chaque site local, ensuite nous avons défini un modèle distribué selon lequel l'agrégation est effectuée. Nous avons exécuté notre approche sur trois machines avec plusieurs jeux de données. Voici deux exemples d'exécution :

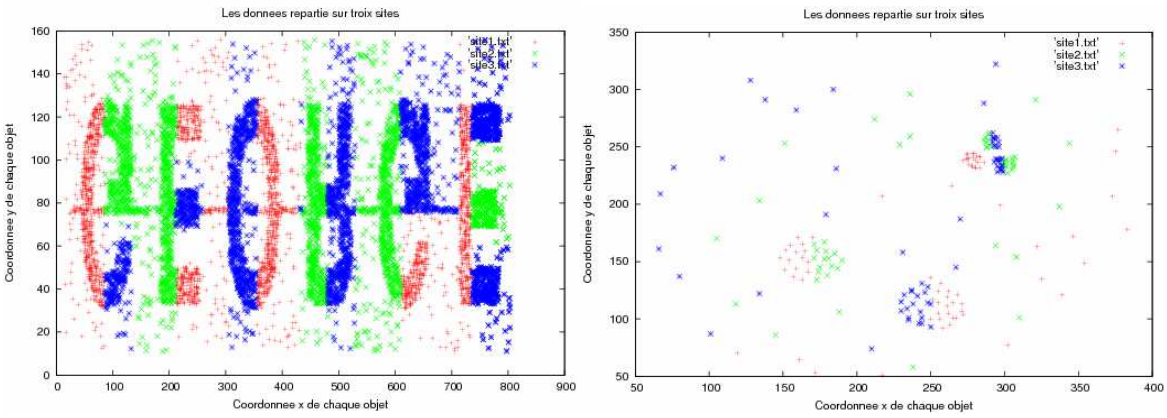


Figure. 4.13 Deux jeux de données, chacun est réparti sur trois machines

La Figure 4.13 représente les données initiales pour deux jeux de données: les points en rouge dans les deux figures résident dans le site 1, ceux en bleus dans le site 2 et ceux en vert dans le site 3.

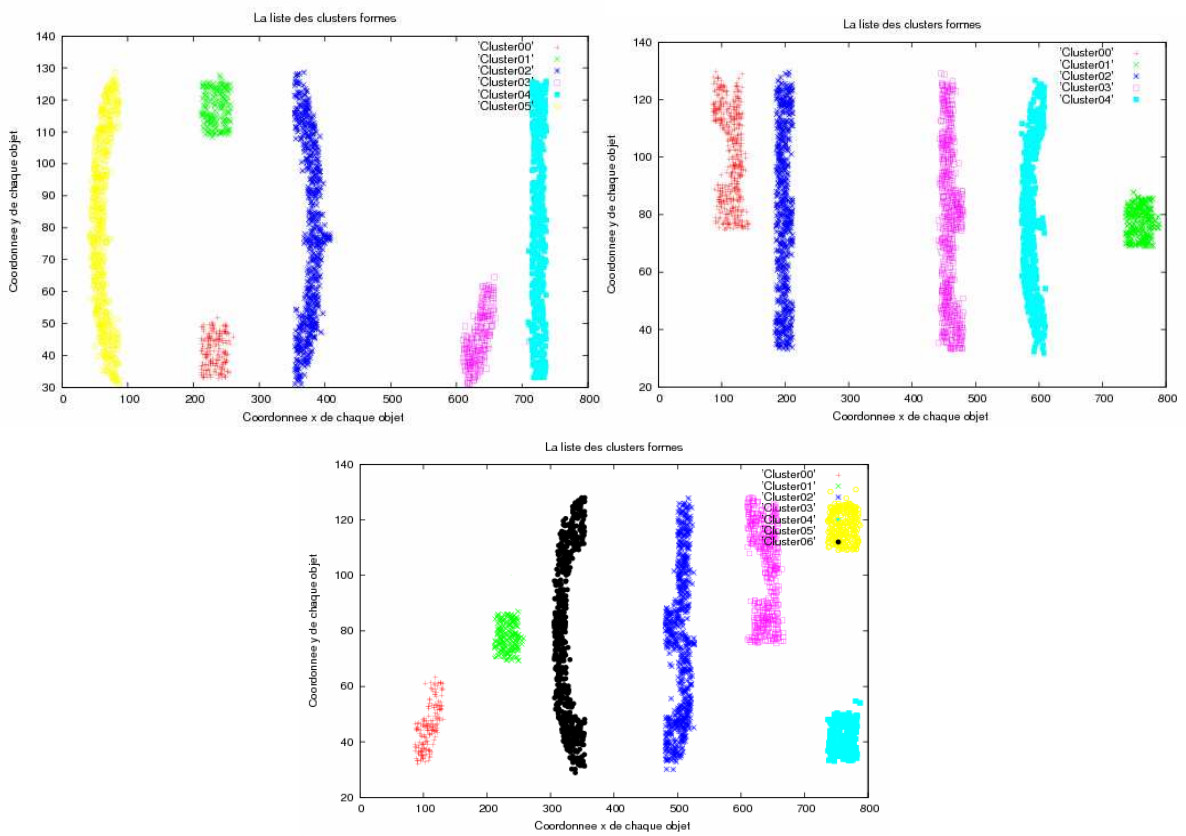


Figure.4.14 Les clusters formés dans chaque site pour le premier jeu de données

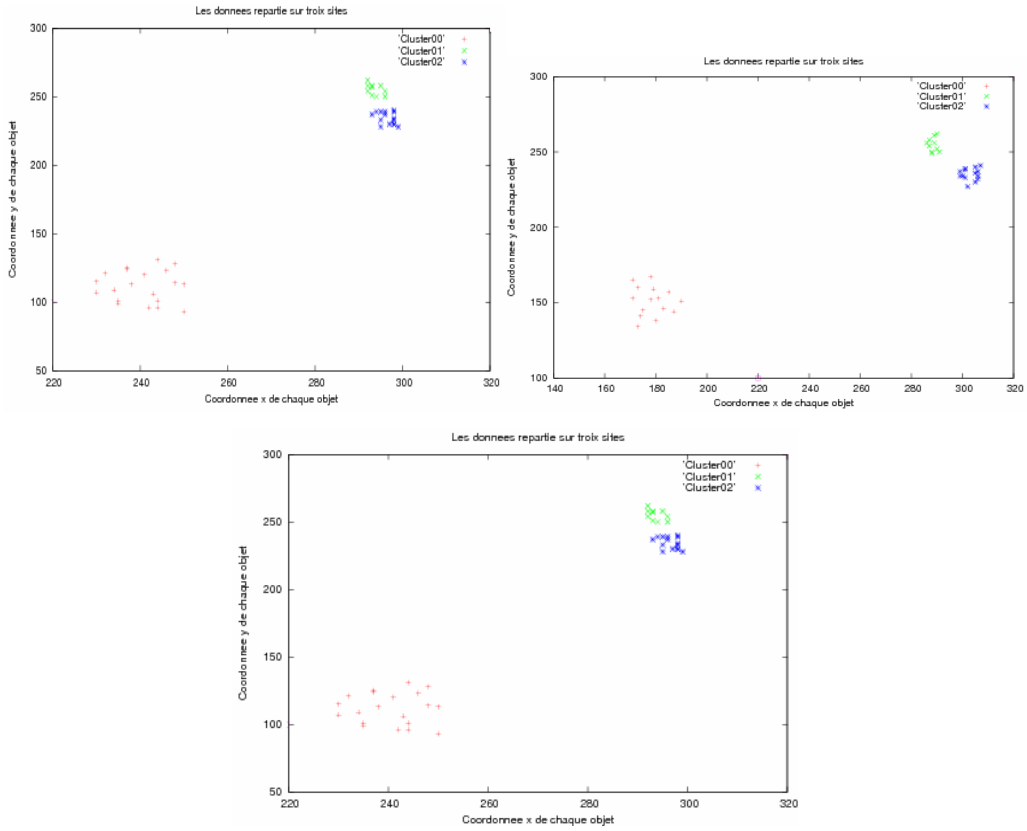


Figure 4.15 Les clusters formés dans chaque site pour le deuxième jeu de données

Les Figures (4.14, 4.15) représentent les clusters formés dans chacun des trois sites pour les deux jeux de données de la Figure4.13. Nous remarquons que les clusters sont bien identifiés dans chaque site.

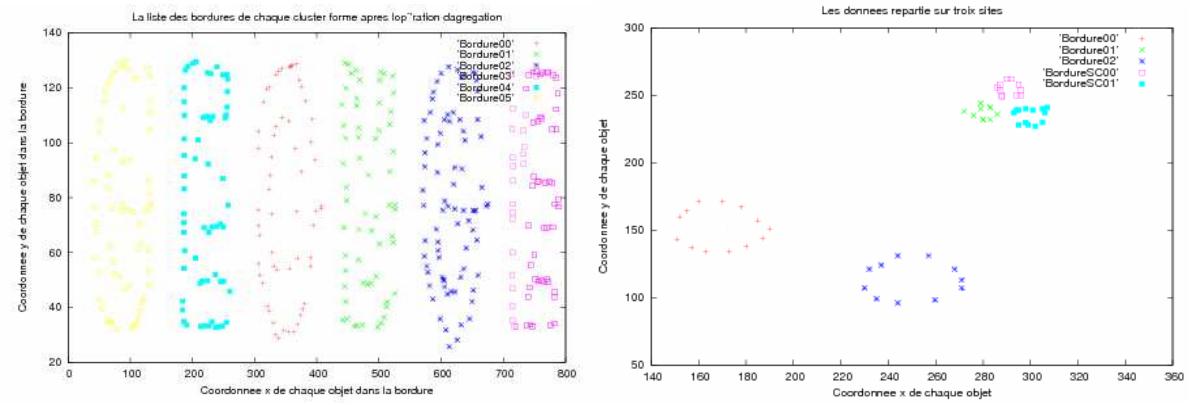


Figure. 4.16 Résultat d'agrégation des trois sites pour les deux jeux de données.

La Figure 4.16 représente les résultats d'agrégation des clusters formés dans chacun des trois sites pour les deux jeux de données de la Figure 4.13.

4.4. Evaluation des résultats

La comparaison des résultats obtenus par notre approche distribuée (Figure 4.16) avec ceux obtenus en exécutant l'algorithme OPTICS séquentiel sur toutes les données (Figures (4.7, 4.9)), nous suggère des résultats identiques pour les deux exécutions séquentielle et distribuée. Les bordures obtenues après l'agrégation des modèles locaux représentent bien les bordures des clusters initiaux avant la répartition.

Notre approche permet de gagner beaucoup en temps d'exécution par rapport à l'algorithme séquentiel. En effet, le temps d'exécution pour OPTICS est fortement dominé par le temps d'exécution des requêtes des *ε-voisinage* qui doivent être exécutées pour chaque objet dans la base de données, c'est-à-dire, le temps d'exécution pour l'algorithmes séquentiel est *O(n*t)* où *t* est le temps d'exécution d'une requête de *ε–voisinage*. Dans notre approche distribuée, le nombre d'objets est divisé sur l'ensemble des sites ; ce qui réduit considérablement le temps d'exécution d'une requête de *ε–voisinage* et qui permet de gagner beaucoup en temps d'exécution de l'algorithme OPTICS effectué localement et de manière parallèle. L'étape suivante est la formation des bordures qui est aussi dominé par le temps d'exécution d'une requête de calcul de voisinage, qui est calculé que pour les points bordures et le nombre de ces points est inférieur à 3 % du nombre total des points du cluster **[33]**. Le

temps d'exécution de l'algorithme d'agrégation qui est négligeable devant celui de l'algorithme OPTICS et celui du calcul des bordures. Le temps de communication dépend du nombre de points bordures transférés d'un site à un autre site d'agrégat. Ce nombre est trop petit et représente de manière efficace les clusters locaux ; ce qui rend notre approche très intéressante.

De plus, l'algorithme OPTICS séquentiel donne généralement de bons résultats lorsque la méthode d'extraction des clusters à partir du graphe des distances d'accessibilité est efficace. Dans certains jeux de données particuliers, cette méthode peut offrir des résultats qui ne correspondent pas aux résultats attendus. Notre approche remédie à ce problème car dans le cas où nous effectuons l'agrégation nous testons les chevauchements entre les clusters et si nous rencontrons deux clusters dans un site local alors que réellement ils forment qu'un seul, les deux clusters sont ainsi fusionnés.

4.5. Conclusion

Dans notre évaluation expérimentale, nous avons montré que la nouvelle approche de clustering distribué donne les mêmes résultats qu'une exécution séquentielle sur toutes les données. De plus, elle permet de régler les clusters mal formés dans les sites locaux. Nous avons aussi montré que la nouvelle approche a un énorme avantage d'efficacité comparée au clustering séquentiel, du fait que nous gagnons beaucoup en temps d'exécution. Pour des raisons économiques ou de sécurité, il n'est pas souvent possible de transférer toutes les données à un site central; ce qui rend les approches centralisées impraticables.

Conclusion générale & Perspectives

Dans ce mémoire, notre motivation est portée d'abord sur la nécessité de distribuer les algorithmes de clustering. De plus, la plupart des versions parallèles et distribuées de ces algorithmes se basent sur les versions séquentielles qui génèrent beaucoup de communications et trouvent des difficultés pour le choix du nombre de cluster, qui n'est pas évident pour l'utilisateur et les méthodes d'approximation qui l'estiment sont des problèmes NP complet. Nous avons opté pour l'utilisation de l'algorithme de clustering OPTICS car il fournit un ordre pour les objets de la base de données, qui est insensible aux paramètres d'entrées et permet d'identifier la structure de clustering tel que soit la forme arbitraire des clusters et même dans le cas des clusters chevauchés.

L'objectif principal de ce projet est de proposer un algorithme distribué sans tenir compte de la version séquentielle. Notre approche est basée sur l'agrégation des modèles locaux. Elle évite l'acheminement de grandes quantités de données à un site central en transportant que les représentants des clusters aux sites d'agrégat. Notre approche distribuée est basée sur trois étapes principales suivantes :

La première étape consiste à implémenter l'algorithme OPTICS séquentiel et à proposer une nouvelle méthode pour la détermination des clusters à partir des graphes des distances d'accessibilités.

La deuxième étape consiste à extraire les représentants de chaque cluster. Les représentants d'un cluster sont les points de sa bordure. Les bordures des clusters sont de très bons représentants. A partir desquels, nous régénérons les clusters originaux. Pour cela, nous avons défini d'autres concepts qui ont un rôle important dans la régénération des clusters. Nous avons introduit la notion du signe du vecteur d'équilibre, la moyenne des distances entre les points dans un cluster et la moyenne des distances noyaux. Avec ces informations, nous pouvons régénérer un cluster à partir des bordures quelque soit sa forme et sa densité. Les différents représentants sont envoyés au site d'agrégat selon un modèle distribué bien défini.

La dernière étape consiste à tester les recouvrements des clusters des différents sites en même temps que la régénération des clusters. Avant de commencer le processus de régénération des clusters, nous avons besoin de régénérer la bordure entière de chaque cluster car, les points bordures pour un cluster calculés initialement sont dispersés ; ce qui permet de minimiser les communications. Cependant pour mieux reproduire les clusters, nous avons proposé un algorithme qui permet de régénérer les bordures. La régénération des clusters est faite à partir des nouvelles bordures calculées. Les nouvelles bordures de nouveaux clusters formées lors du test de recouvrement sont formées à partir de celles des clusters qui se chevauchent.

Nous avons réalisé une solution pour la distribution de l'algorithme OPTICS indépendamment de la version séquentiel. Dans l'évaluation expérimentale, nous avons montré que notre approche de clustering distribué donne les mêmes résultats qu'une exécution séquentielle dans un site central sur toutes les données, mais en un temps d'exécution réduit. Elle permet également de régler les clusters mal formés dans les sites locaux. De plus, pour des considérations économiques et de sécurisation des données, il n'est pas souvent possible de les transmettre d'un site local à un autre puis faire le clustering.

Comme perspectives de notre travail, nous proposons tout d'abord d'exécuter cette approche sur plusieurs machines pour mieux voir l'efficacité du modèle de distribution proposé, du processus de régénération et celui de formation des nouvelles bordures pour les nouveaux clusters calculés.

Naturellement les données sont réparties d'une manière aléatoire sur les différents sites et les processeurs ont des capacités différentes. Dans notre implémentation, nous n'avons pas pris cela en considération. Cependant, nous proposons de faire l'équilibre de charge avant d'exécuter l'algorithme séquentiel dans chaque site en utilisant la méthode PSLB (positional Scan Load Balancing) **[35]** basée sur l'opérateur SCAN. Pour utiliser la méthode PSLB nous devons définir premièrement l'unité de travail. Dans notre cas elle peut correspondre à un ensemble d'objets. Le nombre d'unité de travail dans chaque site initialement ne prend pas en considération des capacités des processeurs. Notre but est d'équilibrer la charge des unités de travail sur les différents sites et d'affecter plus d'unités de travail aux processeurs les plus puissants. L'équilibre de charge entre les différents sites permet à tous les sites de terminer leurs exécutions séquentielles en même temps, ce qui facilite la définition du modèle de

distribution du fait que chaque site peut savoir à l'avance avec qu'elle autre site il va effectuer son agrégation.

L'efficacité de notre approche dépend énormément de la précision du processus de calcule des représentants des clusters, chaque cluster est représenté principalement par sa bordure, l'algorithme de calcul des bordures donne généralement de bons résultats, mais nous pouvons avoir des points de la frontière qui sont très dispersés alors que ce n'est pas le cas, car cet algorithme ne donne pas forcément tous les points de la frontière et comme nous allons régénérer les clusters, alors nous avons besoin de plus de points que le nombre renvoyé par l'algorithme **[33]**. Pour remédier à ce problème, nous devons modifier cet de sorte qu'il donne la frontière entière du cluster et si nous constatons que le nombre de points calculé est important ce qui peut causer un coût élevé lors des communications alors nous pouvons appliquer la deuxième partie de l'algorithme**[33]**.

Références Bibliographiques

[8] Lamine M. Aouad, Nhien-An Le-Khac, M. Tahar Kechadi, 'Performance study of distributed Apriori-like frequent itemsets", 2009

[13] Lamine M. Aouad, Nhien-An Le-Khac and M. Tahar Kechadi, "Distributed Frequent Itemsets Mining in Heterogeneous Platforms", journal of Engineering, Computing and Architecture 2007.

[26] Lamine M. Aouad, Nhien-An Le-Khac, and M. Tahar Kechadi, "Variance- based clustering technique for distributed data mining applications", In Pro- ceedings of the 2007 International Conference on Data Mining, DMIN 2007, Las Vegas, Nevada, USA, June 2007.

[4] Lamine M. Aouad, Nhien-An Le-Khac and M. Tahar Kechadi, "A Multi-Stage Clustering Algorithm for Distributed Data Mining Environments", School of Computer Science and Informatics University College Dublin – Ireland, 2008

[27] Y. Mingjin and Y. Keying, "Determining the number of clusters using the weighted gap statistic", Biometrics, 63(4), 2007.

[28] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the Number of Clusters In a Data Set Via the Gap Statistic", Journal of Royal Statistical Society, 2001

[5] Eshref Januzaj, Hans-Peter Kriegel, Martin Pfeifle, "Towards Effective and Efficient Distributed Clustering", Institute for Computer ScienceUniversity of Munich Germany, 2003

[23] Kaufman L and Rousseeuw P.J., "Finding Groups in Data: an Introduction to Cluster Analysis", John Wiley & Sons, 1990

[1] Ester M., Kriegel H.-P., Sander J., Xu X, "A Density- Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD'96), Portland, OR, AAAI Press, pp.226-231,1996.

[22] Jiawei Han and Micheline Kamber, “Data Mining: Concept and Techniques”. University of Illinois at Urbana-Champaign. Elsevier, 2006.

[9] Byung-Hoon Park and Hillol Kargupta, “Distributed Data Mining: Algorithme, Systems, and Applications”, University of Maryland Baltimore, 2001

[3] Mihael Ankerst. Markus M. Breunig. Hans-Peter Kriegel, Jörg Sander, “OPTICS: Ordering Points To Identify the Clustering Structure”. Institute for Computer Science, University of Munich Oettingenstr. 67, D-80538 Munich, Germany: 1999.

[33] A. Piras and M-T. Kechadi, “Distributed Clustering – Hierarchical Clustering on Grids”, Journal of Parallel Computing, (in press) 2010.

[34] Rohit Chandra, Leonardo Dagum, Dave Kohr, Dror Mydan, Jeff McDonald, Ramesh Menon, “Parallel Programming in OpenMP”, 2001

[2] Jean-Pierre Nakache, Josiane Conais, “Approche pragmatique de la classification”, Editions Technip, Paris, 2005.

[6] CHEUNG, D. W., HAN, J., NG, V. T., FU, A. W., AND FU, Y, “A fast distributed algorithm for mining association rules”, In PDIS: International Conference on Parallel and Distributed Information Systems, IEEE Computer Society Technical Committee on Data Engineering, and ACM SIGMOD, 1996.

[19] Agrawal, R. et R. Srikant, “Fast algorithms for mining association rules”, In Proceedings of the 20th VLDB Conference, pp. 487–499, 1994.

[21] Han, J., J. Pei, et Y. Y, “Mining frequent patterns without candidate generation”, In SIGMOD’00 : Proceedings of the International Conference on Management of Data, pp.1–12, 2000.

[20] Pasquier, N., Yves, Y. Bastide, R. Taouil, et L. Lakhal, “Efficient mining of association rules using closed itemset lattices”, Information Systems 24, 25–46, 1999.

[7] Cheung DW, Ng VT, Fu AW, Fu Y, “Efficient mining of association rules in distributed databases”, IEEE Trans Knowl Data Eng 8(6):911–922, 1996

[24] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data Clustering: A Review. ACM Computing Surveys”, Sep 1999.

[25] R. Xu and D.Wunsch, “Survey of Clustering Algorithms”, IEEE Transactions on Neural Networks, 16, May 2005.

[15] M. Ester, H.-P Kriegel, J. Sander, and X. Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”, In 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD), 1996.

[18] A. Garg, A. Mangla, V. Bhatnagar, and N. Gupta. PBIRCH, “A Scalable Parallel Clustering algorithm for Incremental Data”. In 10th International Database Engineering and Applications Symposium, IDEAS'06, 2006.

[16] X. Xu, J. Jager, and H.-P. Kriegel, “A Fast Parallel Clustering Algorithm for Large Spatial Databases”. Journal of Data Mining and Knowledge Discovery,3, 1999.

[29] R. B. Calinski and J. Harabasz, “A dendrite method for cluster analysis”. Communication in statistics, 3, 1974

[30] R. T. Ng and J. Han. E_cient and E_ective, “Clustering Methods for Spatial Data Mining”. In VLDB, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile, 1994.

[31] R. Xu and D.Wunsch, “Survey of Clustering Algorithms”. IEEE Transactions on Neural Networks, 16, May 2005.

[32] B. Zhang, M. Hsu, and U. Dayal. K-Harmonic Means, “A Data Clustering Algorithm”, Technical report, HP Labs, 1999.

[35] David F. Hegarty, M.T. Kechadi, “Topology Preserving Dynamic Load Balancing for Parallel Molecular Simulation”, August 15, 1997

[10] Robert Cooley, Bamshad Mobasher, Jaideep Srivastava, “Web mining: information and pattern discovery on the World Wide Web”, Department of Computer Science University of Minnesota, Minneapolis, MN 55455, USA

[11] Stéphane TUFFERY, “Data Minig et statistique décisionnelle. L’intelligence des données” edition Technip, Paris, 2007.

[12] Ashrafi MZ, Taniar D, Smith KA ODAM: “An optimized distributed association rule mining algorithm”. IEEE Distrib Syst Online 5(3), 2004

[14] Pramudiono I, Kitsuregawa M “Parallel FP-Growth on PC cluster”, In: Proceedings of the 7th Pacific–Asia conference of knowledge discovery and data mining (PAKDD03), 2003