# Cooperative Clustering Model and Its Applications

by

Rasha F. Kashef

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2008

## AUTHOR'S DECLARATION

Rasha F. Kashef

# Abstract

Data clustering plays an important role in many disciplines, including data mining, machine learning, bioinformatics, pattern recognition, and other fields, where there is a need to learn the inherent grouping structure of data in an unsupervised manner. There are many clustering approaches proposed in the literature with different quality/complexity tradeoffs. Each clustering algorithm works on its domain space with no optimum solution to all datasets of different properties, sizes, structures, and distributions. Challenges in data clustering include, identifying proper number of clusters, scalability of the clustering approach, robustness to noise, tackling distributed datasets, and handling clusters of different configurations. This thesis addresses some of these challenges through cooperation between multiple clustering approaches.

We introduce a Cooperative Clustering (CC) model that involves multiple clustering techniques; the goal of the cooperative model is to increase the homogeneity of objects within clusters through cooperation by developing two data structures, cooperative contingency graph and histogram representation of pair-wise similarities. The two data structures are designed to find the matching sub-clusters between different clusterings and to obtain the final set of cooperative clusters through a merging process. Obtaining the co-occurred objects from the different clusterings enables the cooperative model to group objects based on a multiple agreement between the invoked clustering techniques. In addition, merging this set of sub-clusters using histograms poses a new trend of grouping objects into more homogenous clusters. The cooperative model is consistent, reusable, and scalable in terms of the number of the adopted clustering approaches.

In order to deal with noisy data, a novel Cooperative Clustering Outliers Detection (CCOD) algorithm is implemented through the implication of the cooperation methodology for better detection of outliers in data. The new detection approach is designed in four phases, (1) Global non-cooperative Clustering, (2) Cooperative Clustering, (3) Possible outlier's Detection, and finally (4) Candidate Outliers Detection. The detection of outliers is established in a bottom-up scenario.

The thesis also addresses cooperative clustering in distributed Peer-to-Peer (P2P) networks. Mining large and inherently distributed datasets poses many challenges, one of which is the extraction of a global model as a global summary of the clustering solutions generated from all nodes for the purpose of interpreting the clustering quality of the distributed dataset as if it was located at one node. We developed distributed cooperative model and architecture that work on a two-tier super-peer P2P network. The model is called Distributed Cooperative Clustering in Super-peer P2P Networks *(*DCCP2P*).* This model aims at producing one clustering solution across the whole network. It specifically addresses scalability of network size, and consequently the distributed clustering complexity, by modeling the distributed clustering problem as two layers of *peer neighborhoods* and *super-peers*. Summarization of the global distributed clusters is achieved through a distributed version of the cooperative clustering model.

Three clustering algorithms, *k*-means (KM), Bisecting *k*-means (BKM) and Partitioning Around Medoids (PAM) are invoked in the cooperative model. Results on various gene expression and text documents datasets with different properties, configurations and different degree of outliers reveal that: (i) the cooperative clustering model achieves significant improvement in the quality of the clustering solutions compared to that of the non-cooperative individual approaches; (ii) the cooperative detection algorithm discovers the nonconforming objects in data with better accuracy than the contemporary approaches, and (iii) the distributed cooperative model attains the same quality or even better as the centralized approach and achieves decent speedup by increasing number of nodes. The distributed model offers high degree of flexibility, scalability, and interpretability of large distributed repositories. Achieving the same results using current methodologies requires polling the data first to one center location, which is sometimes not feasible.

# Acknowledgements

I want to start by expressing my deep gratitude to God for giving me the strength and faith to start this journey and the ability to finally complete this work.

This thesis would not be possible without the support of many individuals, to whom I would like to express my gratitude. First and foremost, I would like to thank my supervisor, Prof. Mohamed Kamel for the opportunity of working with him, and for his continuous guidance, motivations, and encouragement throughout my Ph.D. studies at the University of Waterloo. His invaluable suggestions and precious ideas have helped me to walk through various stages of my research, while his passion and extraordinary dedication to work have always inspired me and encouraged me to work harder. His trust and support in delegation have instilled in me great confidence and were key factors in my development as a person and as a researcher.

I would like also to thank many faculty members of the University of Waterloo, most notably my committee members, Prof. Otman Basir, Prof. Fakhri Karray, and Prof. Ali Ghodsi, for their valuable input and suggestions. I would like also to thank my external examiner, Prof. David Chiu for his discussions and ideas.
I am grateful to my colleagues in the PAMI lab especially Shady Shehata, Moataz El-Ayadi, and Mohamed El-Abd for valuable discussions and feedback. I would like also to thank former members of the PAMI group, especially Khaled Hammouda and Masoud Makrehchi for many useful discussions and insights. I also wish to express my gratitude to Hazem Shehata and Ahmed Yousef for their support and help during my graduate studies.

I would like to thank the administrative secretaries Heidi Campbell and Wendy Boles for their support and encouragement during the whole course of my PhD. program.

I would like to thank Wessam El-Tokhi, Noha Yousri, Reem Adel, Rania El-Sharkawy, Walaa ElShabrawy, Noran Magdi, Walaa Khaled, Safaa Mahmoud, and Hanan Saleet, for being such great friends.

Finally, words fail me to express my appreciation to my father Farouk, my mother Amal, my sisters, Reham, Rania, Randa, my brother Mostafa, and my brother-in-law Ahmed Hussein for their support and encouragement throughout my life. I also wish to thank my little niece Icel and nephew Eyad for encouraging me to finish this work with their few lovely words.

## Dedication

I would like to dedicate this thesis to my family for living with the thesis as well as me, and for countless ways they ensured that I would finish every bits and pieces. Without their patience, encouragement and understanding, this work would have not been possible.

*To my Father, Mother, Sisters, and Brother*

# Table of Contents

## List of Figures

# List of Tables

# List of Abbreviations

BKM: Bisecting *k*-means

BOINC: Berkeley Open Infrastructure for Network Computing

CBLOF: Cluster-based Local Outlier Factor

CC: Cooperative Clustering

CCG: Cooperative Contingency Graph

CCOD: Cooperative Clustering Outliers Detection

CD: Cluster Distance

CL: Complete Linkage

CLARA: Clustering Large

CLARANS: Clustering Large Applications based on Randomized Search

COF: Cooperative Outlier Factor

CSR: Compressed Sparse Row

DBSCAN: Density Based Spatial Clustering of Applications with Noise

DCCP2P: Cooperative Clustering in P2P networks

DCC: Distributed Collaborative Clustering

DDBC: Distributed Density-Based Clustering

DKM: Distributed *k*-means

DBKM: Distributed Bisecting k-means

DI: Dunn Index

DMBC: Distributed Model-based Clustering

FCM: Fuzzy c-means

KDEC: Kernel Density Estimation Based Clustering

KM: *k*-means

LOF: Local Outlier Factor

MDBT: Multidimensional Binary Tree

MPI: Message Passing Interface

MPMD: Multiple Program Multiple Data

OP: Ordinary Peer

ORS: Orthogonal Range Search

|  |  |
|---|---|
| OWSR: | Overall Weighted Similarity Ratio |
| PAM: | Partitioning Around Medoids |
| PDDP: | Principal Direction Divisive Partitioning |
| P2P: | Peer to Peer |
| RCSP: | Row-wise Cyclic Striped Partitioning |
| RMSSD: | Root Mean Square Standard Deviation |
| ROCK: | Robust Clustering for Categorical Attributes |
| SC: | Partition Index |
| SI: | Separation Index |
| SH: | Similarity Histogram |
| SHC: | Similarity Histogram Clustering |
| SL: | Single Linkage |
| SM: | Similarity Matrix |
| SODON: | Self-Organized Download Overlay Network |
| SOM: | Self Organizing Map |
| SPMD: | Single Program Multiple Data |
| SP: | Super Peer |
| VSM: | Vector Space Model |
| VoIP: | Voice over IP |

# Chapter 1
# Introduction

This thesis embodies research that aims at advancing the state of art in data clustering, clustering-based outlier detection, and the application of clustering in distributed environments. The first section gives an overview of the data clustering problem, clustering applications, and current challenges in clustering. The following sections give an overview of the main contributions of this thesis to address some of the identified challenges by developing the Cooperative Clustering (CC) model, Cooperative Clustering Outliers Detection (CCOD) algorithm, and finally the Cooperative Clustering model in Distributed super-peer P2P networks (DCCP2P).

## 1.1 Overview

Analysis of data can reveal interesting, and sometimes important, structures or trends in the data that reflect a natural phenomenon. Discovering regularities in data can be used to gain insight, interpret certain phenomena, and ultimately make appropriate decisions in various situations. Finding such inherent but invisible regularities in data is the main subject of research in data mining, machine learning, and pattern recognition.

### 1.1.1 Data Clustering

Data clustering is a data mining technique that enables the abstraction of large amounts of data by forming meaningful groups or categories of objects, formally known as clusters, such that objects in the same cluster are similar to each other, and those in different clusters are dissimilar. A cluster of objects indicates a level of similarity between objects such that we can consider them to be in the same category, this simplifying our reasoning about them considerably.

### 1.1.2 Applications of Data Clustering

Clustering is used in a wide range of applications, such as marketing, biology, psychology, astronomy, image processing, and text mining. For example, in biology it is used to form taxonomy of species based on their features and to group the set of co-expressed genes together into one group. In image processing it is used to segment texture in images to differentiate between various regions or objects. Clustering is also practically used in many statistical analysis software packages for general-purpose data analysis. A large number of clustering methods [1]-[27] have been developed in several different fields, with different definitions of clusters, methodologies, and similarity metrics between objects.

### 1.1.3 Challenges in Data Clustering

There are number of problems associated with clustering, some of these issues are:

- Determining number of clusters a priori
- Obtaining the natural grouping of data (i.e. proper number of clusters)
- Handling datasets of different properties, structures, and distributions
- Performing incremental update of clusters without re-clustering
- Dealing with noise and outliers
- Clustering large and high dimensional data objects (i.e. data scalability)
- Tackling distributed datasets
- Evaluating clustering quality
- Dealing with different types of attributes (features)
- Interpretability and usability

Much of the related work does not attempt to confront all the above mentioned issues directly; for example *k*-means is very simple and it is known for its convergence property, but on the other hand, it cannot handle clusters with different shapes, it is vulnerable to the existence of outliers and it needs number of clusters to be known a priori. In general, there is no one clustering technique that will work for all types of data and conditions. Thus most of the well known clustering algorithms work on their own problem space with their own criteria and methodology.

In this thesis, four of those challenges are addressed: handling datasets of different configurations and properties, achieving better detection of outliers, handling large datasets, and finally tackling distributed data. Dealing with datasets of different properties is addressed through developing a novel cooperative clustering model that uses two data structures, the pair-wise similarity histogram and the cooperative contingency graph. Achieving better detection of outliers than the traditional clustering-based outlier's detection approaches is obtained through the bottom-up detection algorithm using the cooperative clustering methodology. Finally, the data scalability and tackling distributed data are addressed through a novel distributed cooperative clustering model in two-tier super-peer P2P networks. Each of the above contributions is described in the following sections with a brief insight into each of the mentioned challenges that we need to solve.

## 1.2 Cooperative Clustering Model

It is well known that no clustering method can effectively deal with all kinds of cluster structures and configurations. In fact, the cluster structure produced by a clustering method is sometimes an artifact of the method itself. Combining clusterings invokes multiple clustering algorithms in the clustering process to benefit from each other to achieve global benefit (i.e. they cooperate together to attain better overall clustering quality).

*Ensemble clustering* is based on the idea of combining multiple clusterings of a given dataset to produce a superior aggregated solution based on aggregation function [28]-[30]. Ensemble clustering techniques have been shown to be effective in improving the quality. However, inherent drawbacks of these techniques are: (1) the computational cost of generating and combining multiple clusterings of the data, and (2) designing a proper cluster ensemble that addresses the problems associated with high dimensionality and parameter tuning.

Another form of combining multiple clusterings is *Hybrid Clustering*. Hybrid clustering assumes a set of cascaded clustering algorithms that cooperate together for the goal of refining the clustering solutions produced by a former clustering algorithm(s). However, in hybrid clustering one or more of the clustering algorithms stays idle till a former algorithm(s) finishes its clustering which causes a significant waste in the total computational time [31],[32].

The work presented in this thesis enables concurrent implementation of the multiple clustering algorithms and benefit from each other with better performance for datasets with different configurations by using cooperative clustering. The cooperative Clustering (CC) model achieves synchronous execution of the invoked techniques with no idle time and obtains clustering solutions with better homogeneity than those of the non-cooperative clustering algorithms. The cooperative clustering model is mainly based on four components (1) Co-occurred sub-clusters, (2) Histogram representation of the pair-wise similarities within sub-clusters, (3) The cooperative contingency graph, and (4) The coherent merging between the set of histograms. These components are developed to obtain a cooperative model that is capable of clustering data with better quality than that of the adopted non-cooperative techniques. Experimental results on various gene expression and document datasets in chapter 4 illustrate a significant improvement in the clustering quality using the cooperative models compared to that using the individual non-cooperative algorithms.

## 1.3 Outliers Detection Using Cooperative Clustering

Outlier detection refers to the problem of discovering objects that do not conform to expected behavior in a given dataset. These nonconforming objects are called *outliers*. A variety of techniques have been developed to detect outliers in several research applications including: bioinformatics and data mining [33]-[43]. Current clustering-based approaches for detecting outliers explore the relation of an outlier to the clusters in data. For example, in medical applications as gene expression analysis, the relation of unknown novel genes (outliers) to the gene clusters in data is important in studying the function of such novel genes. Traditional clustering-based outlier detection techniques are based only on the assumption that outliers either do not belong to any cluster or form very small-sized clusters.

In this thesis, a novel clustering-based outlier detection method is proposed and analyzed, it is called Cooperative Clustering Outliers Detection (CCOD) algorithm. It provides efficient outlier detection and data clustering capabilities in the presence of outliers. It uses the notion of cooperative clustering towards better discovery of outliers. The CCOD is mainly based on three assumptions:

- First, outliers form very small clusters,
- Second, outliers may exist in large clusters, and
- Third, outliers reduce the homogeneity of the clustering process.

Based on these assumptions, the algorithm of our outlier detection method first obtains a set of sub-clusters as an agreement between the multiple clusterings using the notion of cooperative clustering. Thus a large sub-cluster means strong agreement while a small sub-cluster indicates week agreement. The following stages on the CCOD involve an iterative identification of possible and candidate outliers of objects in a bottom-up fashion. The empirical results in chapter 6 indicate that the proposed method is successful in detecting outliers compared to the traditional clustering-based outlier's detection techniques.

## 1.4 Distributed Cooperative Clustering

The problem of clustering large, high dimensionality, and distributed data becomes more complex under the new emerged fields of text mining and bioinformatics. How can distributed objects across a large number of nodes be clustered in an efficient way? And can we interpret the results of such distributed clustering? The work presented in this thesis answers these questions. This section first discusses the problem of distributed clustering and then presents the new cooperative clustering model in distributed super-peer P2P networks to address the identified questions.

### 1.4.1 Distributed Clustering: An Overview

With the continuous growth of data in distributed networks, it is becoming increasingly important to perform clustering of distributed data in-place, without the need to pool it first into a central location. In general, centralized clustering usually implies high computational complexity, while distributed clustering usually aims for speedup but suffers from communication overhead. In general, distributed clustering achieves a level of speedup that outweighs communication overhead. The goal of distributed clustering can be either to produce globally or locally optimized clusters. Globally optimized clusters reflect the grouping of data across all nodes, as if data from all nodes were pooled into a central location for centralized clustering [44]. On the other hand, locally optimized clusters create a different set of clusters at each node, taking into consideration remote clustering information and data at other nodes. This implies exchange of data between nodes so that certain clusters appear only at specific nodes [45]. Locally optimized clusters are useful when the whole clusters are desired to be in one place rather than fragmented across many nodes. It is also only appropriate when data privacy is not a big concern.

In general, there are two architectures in distributed clustering: facilitator-workers and peer-to-peer (P2P). In the facilitator-workers architecture one node is designed as a facilitator, and all other nodes are considered as worker nodes. The facilitator is responsible for dividing the task among workers and aggregating their partial results. In the peer-to-peer architecture, all nodes perform the same task and exchange the necessary information to perform their clustering goals. P2P networks can be structured and unstructured. Unstructured networks are formed arbitrarily by establishing and dropping links over time, and they usually suffer from flooding of traffic to resolve certain requests. Structured networks, on the other hand, make an assumption about the network topology and implement a certain protocol that exploits such a topology. P2P networks are different from facilitator-workers architecture as there is no central control (i.e. no single point of failure), each peer has equal functionality: a peer is a facilitator and a worker; it is dynamic where each peer can join and leave the network. In P2P networks, nodes (peers) communicate directly with each other to perform the clustering task. On the other hand, communication in P2P networks can be very costly if care is not taken to localize traffic, instead of relying on flooding of control or data messages.

### 1.4.2 Cooperative Clustering Model in Distributed Super-Peer P2P Networks

In this thesis, we propose a new Distributed Cooperative Clustering model in super-peer P2P networks (DCCP2P). The proposed distributed architecture deviates from the standard definition of P2P networks, which typically involves loose structure (or no structure at all). The DCCP2P on the other hand, is based on a dynamic two-tier hierarchy structure that is designed up front, upon which the peer network is formed. The first layer of the network consists of a set of neighborhoods where a novel peer-clustering algorithm is applied, such that the closest peers are grouped together into one neighborhood. Then a super-peer is selected as a representative of the neighborhood using a super-peer selection algorithm. The second layer of the network is comprised of the selected super-peers from each neighborhood. All super-peers are connected to one root peer that is responsible for generating the global model. The designed two-tier super-peer network allows peers to join and leave the network by proposing two algorithms, the peer-join and peer-leave algorithms. Using the DCCP2P model, we can partition the problem into a modular way, solve each part individually, and then successively combines solutions to find a global solution. Using this approach, we avoid four main problems in the current state of art of distributed data clustering: (1) The high communication cost usually associated with a structured fully connected network, (2) The uncertainty in the network topology usually introduced by unstructured P2P networks, (3) The central control in the facilitator-workers architecture, and finally (4) the static structure of the network architecture. Experiments performed on the distributed cooperative clustering model show that we can achieve comparable results to centralized cooperative clustering with high gain in speedup.

## 1.5 Thesis Organization

The rest of this thesis is organized as follows: Chapter 2 provides a background and review of the research subjects related to the work herein. Chapters 3 and 4 introduce the cooperative clustering model and its empirical results, respectively. Chapters 5 and 6 present the novel cooperative clustering outliers detection algorithm and its experimental detection accuracy, respectively. Chapters 7 and 8 introduce the distributed cooperative clustering model in two tier super-peer P2P networks and its experimental setup and analysis, respectively. Finally, a thesis summary, conclusions, and future work are presented in chapter 9.

# Chapter 2
# Background and Related Work

In this chapter a relevant literature review of the various topics that fall under data clustering and distributed data clustering is discussed. The first section discusses classical clustering notations and formulations, different similarity criteria, internal and external quality measures to assess the clustering solutions, and finally some of the well known clustering algorithms along with their computational complexity. The following section discusses the current approaches in combining multiple clustering. The later two sections focus on parallel and distributed architectures and algorithms for performing the equivalent task of the centralized approaches in distributed environments with a brief insight into the different distributed performance measures that are used to evaluate the performance of the distributed approaches in distributed networks.

## 2.1 Data Clustering in General

The clustering task is to partition a dataset into meaningful groups (clusters) such that objects within a cluster are similar to one another (high intra-cluster similarity), but differ from objects in other clusters (low inter-cluster similarity) according to some similarity criteria.

The subject has been explored extensively under various disciplines in the past three decades. For example, in the context of text mining, clustering is a really powerful method for discovering interesting (inherent) grouping of documents, may be to form a computer-aided information hierarchy, such as Yahoo-like topic directory. Also in biology, co-expressed genes in the same cluster are likely to be involved in the same cellular processes, and a strong correlation of expression patterns between those genes indicates co-regulation. Clustering techniques have been proven to be helpful to understand gene function, gene regulation, cellular processes, and subtypes of cells.

A large number of clustering algorithms have been devised in statistics [1]-[4], data mining [6],[11],[13] pattern recognition [1],[7], bioinformatics[18]-[23] and other related fields. Some terminologies and notations are best presented at this point to pave the way for discussion of the different concepts and strategies of classical and distributed data clustering and also for the proposed models and algorithms defined in the next chapters. Table 2. 1 summarizes the notations and symbols that are used throughout this thesis.

**Table 2. 1:** Symbols and Notations

| **Symbol** | **Definition** |
|---|---|
| $c$ | Number of clustering algorithms |
| X | The whole dataset |
| $\boldsymbol{x}$ | Object or pattern or data vector or data point represented as a vector of features |
| $\boldsymbol{x}_i$ | The $i^{th}$ object |
| $d$ | Dimensionality of the object $\boldsymbol{x}$ |
| $x^i$ | The $i^{th}$ feature of the object $\boldsymbol{x}$ |
| $n$ | Number of objects |
| $k$ | Number of clusters |
| $S_j$ | The $j^{th}$ cluster |
| $R_j$ | The $j^{th}$ class (External labeling of objects) |
| $c_j$ | The centroid of cluster $S_j$ |
| $m_j$ | The medoid of cluster $S_j$ |
| $z_j$ | The prototype of cluster $S_j$ |
| $P$ | Number of distributed (or parallel) nodes |
| $N_p$ | The $p^{th}$ processing node or peer (processor, process or site) |
| $X_p$ | Local dataset at node $N_p$ |

***Definition***

The data clustering problem can be formulated as: given a dataset of $n$ objects, each having dimensionality $d$, the dataset is partitioned into subsets (clusters) $S_i$; $i$=0,1..,$k$-$1$, such that the *Intra-cluster* distance is minimized and the *Inter-cluster* distance is maximized. The quality of the produced clusters is evaluated using different external and internal quality measures.

Due to the large freedom of choices in the interpretation of the definition, particularly the notion of similarity, many clustering algorithms have been reported in the literature. Different notations to similarity, various types of clustering algorithms, and quality measures are defined in the following subsections.

### 2.1.1 Similarity Measures

A key factor in the success of any clustering algorithm is the similarity measure adopted by the algorithm. In order to group similar data objects, proximity metric has to be used to find which objects (or clusters) are similar. There is a large number of similarity metrics reported in the literature, only most of the common ones are reviewed in this subsection. The calculation of the (*dis*) similarity between two objects is achieved through some *distance* function, sometimes also referred to as a *dissimilarity* function. Given two data vectors ***x*** and ***y*** representing two data points in the *d*-dimensional space, it is required to find the degree of *dis*(*similarity*) between them. A very common class of distances functions is known as the *family of Minkowski* distances [24], described as:

$$\| \boldsymbol{x} - \boldsymbol{y} \|_r = \sqrt[r]{\sum_{i=1}^{d} | x^i - y^i |^r} \qquad (2.\ 1)$$

This distance function actually describes an infinite number of distances indexed by *r*, which assumes values greater than or equal 1. Some of the common values of *r* and their respective distance functions are:

$$r = 1\text{: Manhattan Distance } \| \boldsymbol{x} - \boldsymbol{y} \|_1 = \sum_{i=1}^{d} | x^i - y^i | \qquad (2.\ 2)$$

$$r = 2\text{: Euclidian Distance } \| \boldsymbol{x} - \boldsymbol{y} \|_2 = \sqrt{\sum_{i=1}^{d} | x^i - y^i |^2} \qquad (2.\ 3)$$

$$\text{r} = \infty\text{: Tschebyshev Distance } \| \boldsymbol{x} - \boldsymbol{y} \|_\infty = \max_{i=1,2,\ldots,d} | x^i - y^i | \qquad (2.\ 4)$$

A more common similarity measure that is used specifically in document clustering is the *cosine correlation (Similarity)* measure (used by [6],[16]), defined as:

$$cosSim(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x} . \boldsymbol{y}}{\| \boldsymbol{x} \| \| \boldsymbol{y} \|} \qquad (2.\ 5)$$

Where (.) indicates the vector dot product and $\| . \|$ indicates the length of the vector. Another commonly used similarity measure is the *Jaccard* measure (used by [24],[25]), defined as:

$$Jaccard\ Sim(\boldsymbol{x}, \boldsymbol{y}) = \frac{\sum_{i=1}^{d} min(x^i, y^i)}{\sum_{i=1}^{d} max(x^i, y^i)} \tag{2. 6}$$

Which for the case of binary features vectors, could be simplified to:

$$Jaccard\ Sim(\boldsymbol{x}, \boldsymbol{y}) = \frac{|\boldsymbol{x} \cap \boldsymbol{y}|}{|\boldsymbol{x} \cup \boldsymbol{y}|} \tag{2. 7}$$

Many algorithms employ the distance function (or similarity function) to calculate the similarity between two clusters, a cluster and an object, or two objects. Calculating the distance between clusters (or clusters and objects) requires a representative feature vector of that cluster (sometimes referred to as prototype, e.g. centroid or medoid). Some clustering algorithms make use of a *similarity matrix.* A similarity matrix is an $n$ x $n$ matrix recording the distance (or degree of similarity) between each pair of objects. Obviously the similarity matrix is a positive definite symmetric matrix so we only need to store the upper right (or lower left) portion of the matrix.

### 2.1.2 Taxonomies of Data Clustering Algorithms

Clustering algorithms can be classified along different independent dimensions. For instance, different starting points, methodologies, algorithmic point of view, clustering criteria, and output representations, usually lead to different taxonomies of clustering algorithms. Different properties of clustering algorithms can be described as follows:

**Agglomerative *vs.* Divisive Clustering:** This concept relates to algorithmic structure and operation. An agglomerative approach begins with each object in a distinct (singleton) cluster, and starts merging clusters together until a stopping criterion is satisfied (bottom-up hierarchical clustering). On the other hand, a divisive method begins with all objects in a single cluster and iteratively performs splitting until a stopping criterion is met (top-down hierarchical clustering).

**Monothetic *vs.* Polythetic Clustering:** Both the monothetic and polythetic issues are related to the sequential or simultaneous use of features in the clustering algorithm. Most algorithms are polythetic; that is, all features enter into the computation of distances (or similarity functions) between objects, and decisions are based on those distances, whereas, a monothetic clustering algorithm uses the features one by one.

**Hard *vs.* Fuzzy Clustering:** A hard clustering algorithm allocates each object to a single cluster during its operation and outputs a Boolean membership function either 0 or 1. A fuzzy clustering method assigns degrees of membership for each input object to each cluster. A fuzzy clustering can be converted to a hard clustering by assigning each object to the cluster with the largest degree of membership.

**Distance *vs.* Density Clustering:** A distance-based clustering algorithm assigns an object to a cluster based on its *distance* from the cluster or its representative(s), whereas a density-based clustering grows a cluster as long as the *density* (or number of objects) in the neighborhood satisfies some threshold. It is not difficult to see that distance-based clustering algorithms can typically find only spherical-shaped clusters and encounter difficulty at discovering clusters of arbitrary shape, whereas density-based clustering algorithms are capable of finding arbitrary shape clusters.

**Partitional *vs.* Hierarchical Clustering**: A Partitional clustering algorithm obtains a single partition of the data instead of a clustering structure, such as the *dendrogram* produced by a hierarchical technique. Partitional methods have advantages in applications involving large data sets for which the construction of a *dendrogram* is computationally prohibitive. A problem accompanying the use of Partitional algorithms is the choice of the number of clusters.

**Deterministic *vs.* Stochastic Clustering:** This issue is most relevant to Partitional techniques designed to optimize a squared error function. Deterministic optimization can be accomplished using traditional techniques in a number of deterministic steps. Stochastic optimization randomly searches the state space consisting of all possible solutions.

**Incremental *vs.* Non-incremental Clustering:** This issue arises when the objects set to be clustered is large, and constraints on execution time or memory space need to be taken into consideration in the design of the clustering algorithm. Incremental clustering algorithms minimize the number of scans through the objects set, reduce the number of objects examined during execution, or reduce the size of data structures used in the algorithm's operations. Also incremental algorithms do not require the full data set to be available beforehand. New data can be introduced without the need for re-clustering.

**Intermediate *vs.* Original Representation Clustering:** Some clustering algorithms use an intermediate representation for dimensions reduction when clustering large and high dimensional datasets. It starts with an initial representation, considers each data object and modifies the representation. These classes of algorithms use one scan of the dataset and its structure occupies less space than the original representation of the dataset, so it may fit in the main memory.

### 2.1.3 Clustering Evaluation Criteria

The previous subsection has reviewed taxonomy of a number of clustering algorithms which partition the data set based on different clustering criteria. However, different clustering algorithms, or even a single clustering algorithm using different parameters, generally result in different sets of clusters. Therefore, it is important to compare various clustering results and select the one that best fits the "true" data distribution by using an informative quality measure that reflects the "goodness" of the resulting clusters.

***Definition***

Cluster validation is the process of assessing the quality and reliability of the cluster sets derived from various clustering processes.

Generally, cluster validity has two aspects: (1) first, the quality of clusters can be measured in terms of *homogeneity* and *separation* on the basis of the definition of a cluster: objects within one cluster are similar to each other, while objects in different clusters are dissimilar. Thus if the data is not previously classified, *internal quality* measures are used to compare different sets of clusters without reference to external knowledge, and (2) the second aspect relies on a given "ground truth" of the clusters. The "ground truth" could come from domain knowledge, such as known function families of objects, or from other knowledge repositories (e.g. such as the clinical diagnosis of normal or cancerous tissues for gene expression datasets). Thus, cluster validation is based on the agreement between clustering results and the "ground truth". Consequently, the evaluation depends on a prior knowledge about the classification of data objects, i.e. class labels. This labeling is used to compare the resulting clusters with the original classification; such measures are known as *external quality* measures.

***External Quality Measures***

Three external quality measures (used by [13],[16]) are reviewed, which assume that a prior knowledge about the data objects (i.e. class labels) is given.

**F-measure**

One external measure is the F-measure, a measure that combines the *Precision* and *Recall* ideas from the information retrieval literature. The precision and recall of a cluster $S_j$ with respect to a class $R_i$, $i$, $j$=0,1,..,$k-1$ are defined as:

$$Precision\left(R_i, S_j\right) = \frac{L_{ij}}{|S_j|} \tag{2. 8}$$

$$Recall\left(R_i, S_j\right) = \frac{L_{ij}}{|R_i|} \tag{2. 9}$$

Where $L_{ij}$ is the number of objects of class $R_i$ in cluster $S_j$, $|R_i|$ is the number of objects in class $R_i$ and $|S_j|$ is the number of objects in cluster $S_j$. The *F-measure* of a class $R_i$ is defined as:

$$F(R_i) = \max_j \frac{2 * Precision(R_i, S_j) * Recall(R_i, S_j)}{Precision(R_i, S_j) + Recall(R_i, S_j)} \tag{2. 10}$$

With respect to class $R_i$ we consider the cluster with the highest *F-measure* to be the cluster $S_j$ that is mapped to class $R_i$, and that *F-measure* becomes the score for class $R_i$. The overall *F-measure* for the clustering result of *k* clusters is the weighted average of the *F-measure* for each class $R_i$:

$$F-measure(k) = \frac{\sum_{i=0}^{k-1}(|R_i| \times F(R_i))}{\sum_{i=0}^{k-1}|R_i|} \tag{2. 11}$$

The higher the *F-measure* the better the clustering due to the higher accuracy of the resulting clusters mapped to the original classes.

**Entropy**

The second external quality measure is the *Entropy*, which provides a measure of "goodness" for un-nested clusters or for the clusters at one level of hierarchical clustering. Entropy tells us how *homogenous* a cluster is. The higher the *homogeneity* of a cluster, the lower the entropy is, and vice versa. The entropy of a cluster containing only one object (perfect homogeneity) is zero. Assume a partitioning result of a clustering algorithm consisting of *k* clusters. For every cluster $S_j$ we compute $pr_{ij}$, the probability that a member of cluster $S_j$ belongs to class $R_i$. The entropy of each cluster $S_j$ is calculated using the following standard formula, where the sum is taken over all classes:

$$E(S_j) = -\sum_{i=0}^{k-1} pr_{ij} \log(pr_{ij}),\ j=0, 1,\dots k-1 \tag{2. 12}$$

The overall entropy for a set of *k* clusters is calculated as the sum of entropies for each cluster weighted by the size of each cluster.

$$Entropy(k) = \sum_{j=0}^{k-1} \left( \frac{|S_j|}{n} \right) \times E(S_j) \qquad (2.\ 13)$$

Where $|S_j|$ is the size of cluster $S_j$, and $n$ is the total number of objects. As mentioned earlier, we would like to generate clusters of lower entropy, which is an indication of the *homogeneity* (or *similarity*) of objects within the clusters. The overall weighted entropy formula avoids favoring smaller clusters over larger clusters. The *F-measure* is a better quality measure than *Entropy* for evaluating the clustering quality. Normally the *Entropy* measure will report a perfect cluster if the *Entropy* of the cluster is zero (*i.e.* totally homogeneous). However, if a cluster contains all the objects from two different classes, its entropy will be zero as well. Hence *Entropy* does not tell us if a cluster maps totally to one class or more, but the *F-measure* does.

**Purity**

The *Purity* of a clustering solution is the average precision of the clusters relative to their best matching classes. For a single cluster $S_j$, *Purity* is defined as the ratio of the number of objects in the dominant cluster to the total number of objects in the cluster:

$$P\left(S_j\right) = \frac{1}{|S_j|} \max_{i=0,1..,k-1,\ j \neq i} (L_{ij}) \qquad (2.\ 14)$$

Where $L_{ij}$ is the number of objects from class $R_i$ into cluster $S_j$, and $|S_j|$ is the number of objects in cluster $S_j$. To evaluate the total purity for the entire $k$ clustering, the cluster-wise purities are weighted by the cluster size and the average value is calculated:

$$Purity\left(k\right) = \frac{1}{|n|} \sum_{j=0}^{k-1} \max_{i=0,1,..,k,\ j \neq i} (L_{ij}) \qquad (2.\ 15)$$

We are looking for higher values of the *Purity* measure which indicate better partitioning of objects.

***<u>Internal Quality Measures</u>***

Different scalar validity indices have been proposed in [46]-[48] as internal quality measures, none of them is perfect by itself, and therefore several indices should be used to evaluate the quality of the clustering algorithm. Some indices are used to assess the quality of un-nested clusters produced by hard clustering; others are used to evaluate the quality of fuzzy clusters generated by fuzzy clustering approaches. Another family of indices is applicable in the cases where the hierarchical clustering algorithms are used to cluster the data. In addition, we can use one index to assess two different partitions produced from two different clustering algorithms. Some of these internal indices are described next.

**Partition Index (*SC*)**

It is the ratio of the sum of compactness and separation of clusters. It is a sum of individual clusters internal quality normalized through division by the cardinality of each cluster. The *CS* index for a clustering solution of *k* clusters is defined as:

$$SC(k) = \sum_{i=0}^{k-1} \frac{\sum_{j=1}^{n} u_{ij}^{m} \| \boldsymbol{x}_j - c_i \|^2}{| S_i | \sum_{r=0}^{k-1} \| c_r - c_i \|^2} \tag{2. 16}$$

Where $m$ is the weighting exponent, $|S_i|$ is the size of cluster $S_i$, and $n$ is the total number of objects. *SC* is useful when comparing different partitions having equal number of clusters. A lower value of *SC* indicates better partitioning.

## Dunn Index (*DI*)

An internal quality index for crisp clustering that aims at the identification of "compact" and "well separated" clusters. The *Dunn Index* is defined in Eq. (2.17) for a specific number of *k* clusters.

$$DI(k) = \min_{i=0,1,\dots,k-1} \left\{ \min_{\substack{j=0,1\dots,k-1 \\ i\neq j}} \left\{ \frac{(\| S_i - S_j \|_2)^2}{\max\limits_{r=0,1,\dots,k-1} diam(S_r)} \right\} \right\} \tag{2. 17}$$

Where $\|S_i-S_j\|_2$ is the dissimilarity function (Euclidian distance) between two clusters $S_i$ and $S_j$ and *diam($S_r$)* is the diameter of cluster $S_r$, which may be considered as a measure of clusters' dispersion. If the dataset contains compact and well-separated clusters, the distance between clusters is expected to be large and the diameter of the cluster is expected to be small, thus large values of the index indicate the presence of compact and well-separated clusters. The problems with the *Dunn Index* are (1) its considerable time complexity for large *n*, and (2) its sensitivity to the presence of noise in the dataset, since these are likely to increase the values of the *diam(S)*.

**Separation Index (*SI*)**

Separation Index is a cluster validity measure that utilizes cluster prototypes to measure the dissimilarity between clusters, as well as between objects in a cluster to their respective cluster prototypes.

$$SI(k) = \frac{\sum_{i=0}^{k-1} \sum_{\forall \boldsymbol{x}_j \in S_i} \| \boldsymbol{x}_j - z_i \|_2}{n * \min\limits_{r,l=0,1,\dots,k-1, r\neq l} \{ \| z_r - z_l \|_2 \}} \tag{2. 18}$$

Where $\|\boldsymbol{x}-z_i\|_2$ is the Euclidian distance between object $\boldsymbol{x}$ and the cluster prototype $z_i$. For centroid-based clustering, $z_i$ is the corresponding centroid of the cluster $S_i$ while in medoid-based clustering, $z_i$ refers to the medoid of the cluster $S_i$. Clustering solutions with more compact clusters and larger separation have lower separation index, thus lower values indicates better solutions. The index is more computationally efficient than Dunn's index, and is less sensitive to noisy data.

### Root Mean Square Standard Deviation (*RMSSTD*) Index

The *RMSSTD* index measures the *homogeneity* of the formed clusters at each level of a hierarchical clustering algorithm. The *RMSSTD* of a new clustering solution defined at a level of a clustering hierarchy produced by a hierarchal clustering algorithm is the square root of the variances of all the variables (objects used in the clustering process). Since the objective of cluster analysis is to form homogenous groups, the *RMSSTD* should be as small as possible. In the case that the values of *RMSSTD* are higher than the ones of the previous step of a hierarchical clustering algorithm, we have an indication that the new clustering solution is worse. In the centroid hierarchical clustering where the centroids are used as clusters representatives, the *RMSSTD* takes the form:

$$RMSSTD(k) = \sqrt{\frac{\sum_{i=0}^{k-1} \sum_{\forall x_j \in S_i} (\boldsymbol{x}_j - c_i)^2}{\sum_{i=0}^{k-1} (|S_i| - 1)}} \qquad (2.\ 19)$$

### Cluster Distance Index (*CD*)

The *CD* index measures the distance between the two clusters that are merged in a given step of a hierarchical clustering algorithm. This distance depends on the selected representatives for the hierarchical clustering performed. For instance, for *centroid-based* hierarchical clustering the representatives of the formed clusters are the centroids of each cluster, so *CD* index is the distance between the centroids of the clusters as shown in equation Eq. (2.20).

$$CD(S_i, S_j) = \| c_i - c_j \|_2 \qquad (2.\ 20)$$

Where $S_i$ and $S_j$ are the merged clusters and $c_i$ and $c_j$ are the centroids of the clusters $S_i$ and $S_j$, respectively. In the *Single linkage* (SL) hierarchical clustering, the *CD* index is defined as the minimum Euclidian distance between all possible pairs of points.

$$CD(S_i, S_j) = \min_{\forall \boldsymbol{x} \in S_i, \forall \boldsymbol{y} \in S_j} \| \boldsymbol{x} - \boldsymbol{y} \|_2 \tag{2. 21}$$

The *Complete linkage* (CL) hierarchical clustering defines the *CD* index as the maximum Euclidian distance between all pairs of data points.

$$CD(S_i, S_j) = \max_{\forall \boldsymbol{x} \in S_i, \forall \boldsymbol{y} \in S_j} \| \boldsymbol{x} - \boldsymbol{y} \|_2 \tag{2. 22}$$

**Overall Similarity**

A common internal quality measure is the *overall similarity* and it is used in the absence of any external information such as class labels. The *overall similarity* measures cluster cohesiveness by using the weighted similarity of the internal cluster similarity.

$$Overall\ Similarity(S_i) = \frac{1}{|S_i|^2} \sum_{\forall \boldsymbol{x}, \boldsymbol{y} \in S_i} Sim(\boldsymbol{x}, \boldsymbol{y}) \tag{2. 23}$$

Where $S_i$ is the cluster under consideration and *Sim*($\boldsymbol{x}$,$\boldsymbol{y}$) is the similarity value between the two objects $\boldsymbol{x}$ and $\boldsymbol{y}$ in the cluster $S_i$. Different similarity measures can be used to express the internal cluster similarity (e.g. *cosine correlation*).

### 2.1.4 Data Clustering Algorithms

In this subsection, some of the methods that have been reported in the literature on data clustering are presented along with their complexity analysis. Jain and Murty [1] give a comprehensive account of clustering algorithms. By definition, clustering is an unsupervised learning technique, and that will be the focus of this subsection. Some of these techniques are employed in the experimental results for a comparison purpose.

***k-means (KM) Clustering Algorithm***

The classical *k*-means (KM) algorithm is considered as an effective clustering algorithm in producing good clustering results for many practical applications [1]-[4],[7]. The algorithm is an iterative procedure and requires the number of clusters *k* to be given a priori. The initial partitioning is randomly generated, that is, the centroids are randomly initialized to some points in the region of the space. *k*-means partitions the dataset into *k* non-overlapping regions identified by their centroids based on objective function criterion where objects are assigned to the closest centroid (*Calculation Step*). The most widely used objective function criterion is the distance criterion,

$$\boldsymbol{x} \in S_i \;\; iff \;\; distance(\boldsymbol{x}, c_i) < distance(\boldsymbol{x}, c_j) \;\; \forall j = 0,1,..,k-1; i \neq j \qquad (2.\ 24)$$

Where *distance* is the metric distance between any data vector and the corresponding cluster centroid where the vector belongs to. Thus this criterion minimizes the objective function *J* defined as:

$$J = \sum_{i=0}^{k-1} \sum_{\forall x \in S_i} distance(\boldsymbol{x}, c_i) \qquad (2.\ 25)$$

Both Euclidian distance, $\|\boldsymbol{x}-c_i\|_2$ (Eq. (2.3)) and the cosine correlation $cosSim(\boldsymbol{x},c_i)$ (Eq. (2.5)) are commonly used distance (or similarity) measures. The algorithm converges when re-computing the partitions (*Updating Step*) does not result in a change in the partitioning. For configurations where no data vector is equidistant to more than one centroid, the above convergence condition can always be reached. This convergence property along with its simplicity adds to the attractiveness of the *k*-means. KM often terminates at a local optimum. The global optimum maybe found using other techniques such as deterministic annealing and generic algorithms. On the other hand, *k*-means clustering is vulnerable to the existence of noise and cannot handle datasets with different shapes. It is biased to datasets of globular shapes. The description of the Partitional *k*-means algorithm is shown in Fig. 2. 1.

```
Algorithm: k-means Clustering : KM( X, k, $\varepsilon$)

Input: The dataset X, number of clusters k, and convergence threshold $\varepsilon$.
Output: Set of k clusters, $S=\{S_i, i=0,1,..,k-1\}$
Initialization: Select randomly a set of k initial cluster centroids $c_i$, $i=0,1,..,k-1$.
Begin
  Repeat
    Step1: For each data vector $\boldsymbol{x}_j$, $j=1,..,n$, compute its distance to each cluster centroid $c_i$, $i=0,1,..,k-1$
           and assign it to the cluster with the closest cluster centroid.
    Step2: Compute the objective function $J = \sum_{i=0}^{k-1} \sum_{\forall x \in S_i} (\|\boldsymbol{x}-c_i\|_2)^2$
    Step3: Re-compute cluster centroids $c_i$ for the k clusters; where the new centroid $c_i$ is the mean of all
           the data vectors in the cluster $S_i$.
  Until Convergence (Change in the objective function $\leq \varepsilon$)
Return S
End
```

**Fig. 2. 1.** The Partitional *k*-means Clustering Algorithm

The computational complexity of KM is determined by: number of objects ($n$), dimension of each vector ($d$), number of clusters ($k$), and number of loops ($Lops$). At each loop, the computational complexity of the *calculation step* is dominated by the clustering criterion function $J$, which has $f(n, k, d)$ operations. For the *updating step*, recalculating the centroids needs $kd$ operations. Thus, the time complexity of the KM is:

$$T^{KM}(k) = (f(n, k, d) + kd) * Lops \qquad (2.\ 26)$$

In this thesis, we use the cosine similarity as a measure of similarity between objects, clusters, objects and clusters. Thus for cosine similarity, $f(n, k, d) = 2nkd + nk + nd$. Then, the cost of a single iteration of the KM is of $O(kdn)$.

***<u>Bisecting k-means (BKM) Clustering Algorithm</u>***

The basic bisecting $k$-means [6],[13] is a variant of the $k$-means algorithm. Bisecting $k$-means uses $k$-means to partition the data set into two clusters. Then one of the two clusters is selected and bisected further (*Bisecting Step*). This process is repeated until the desired number of clusters $k$ is obtained. There are a number of different ways (i.e. homogeneity criteria) to choose which cluster to split. For example, we can choose (1) the largest cluster at each step, (2) the one with the least overall similarity, or (3) a criterion based on both size and overall similarity. Note that by recursively using a divisive bisecting clustering procedure, the dataset can be partitioned into any given number of clusters. The bisecting $k$-means algorithm can produce either an un-nested (flat) clustering or a hierarchical clustering. Interestingly enough, the clusters obtained are structured as a hierarchical binary tree (or a binary taxonomy). The bisecting divisive approach is very attractive in many applications as document-retrieval/indexing problems. However, sometimes a "refinement" is needed to re-cluster the results as a fraction of the dataset is left behind with no way to re-cluster it again at each level. Recent studies [13] conclude that, the BKM is better than the standard $k$-means and as good as or better than the hierarchical approaches. The BKM is presented in Fig. 2. 2.

For the BKM algorithm, assume the largest remaining cluster is always split. The computational complexity of the BKM at each level of the hierarchical tree is determined by the size of the cluster $S_j$ at each bisecting step $|S_j|$, the dimension of the vector ($d$), the number of clusters ($k$), the number of loops of $k$-means in each bisecting step ($Loops$), and the number of iterations for each bisecting step ($ITER$) (which is usually specified in advance). In the bisecting step, $f(|S_j|, 2, d)$ operations are required for the $k$-

means calculation step, and 2*d* operations for the centroids updating step. The time complexity of the BKM at each level of the tree can be represented as:

$$T^{BKM} = (f(|S_j|, 2, d) + 2d) * Loops * ITER \quad (2.27)$$

```
Algorithm: Bisecting k-means Clustering: BKM (X, ITER, ζ, k)

Input: The dataset X, number of iterations ITER for the bisecting step, homogeneity criterion ζ, and the
       desired number of clusters k
Output: The set of k clusters $S=\{S_0,S_1,..,S_{k-1}\}$
Initialization: Let V = X, S={ }
Begin
  For number of clusters l =2 to k (Clustering Step)
        Step1: For i=1 to ITER (Bisecting Step)
                 - Select randomly two initial centroids $c_1$ and $c_2$ from the set V.
                 - Find two partitions from the set V using the basic k-means algorithm.
                   End
        Step2: Take the best of these splits as $V_1$ and $V_2$ with the corresponding centroids $c_1$ and $c_2$,
                 respectively.
        Step3: Select the cluster that satisfies the homogeneity criterion ζ as $V_1$
        Step4: Assign V to the remaining partition, $V=V_2$
        Step5: Add $V_1$ to the set of desired clusters $S=S \cup V_1$
  End
Add $V_2$ to the set of desired clusters $S=S \cup V_2$
Return S
End
```

**Fig. 2. 2.** The divisive Bisecting *k*-means Clustering Algorithm

***<u>Partitioning Around Medoids (PAM) Clustering Algorithm</u>***

Rather than calculating the mean of the objects in each cluster as in the *k*-means (KM) clustering, the Partition Around Medoids (PAM) algorithm [14] chooses a representative object, or medoid, for each cluster *at each iteration*. Medoids for each cluster are calculated by finding an object $m_i$ within the cluster that minimizes the objective function defined as the sum of distances of all objects within the clusters to the cluster medoid. PAM has the advantage of its robustness to noisy data and outliers compared to *k*-means. PAM works well for small datasets but cannot scale for large datasets. Thus, [14] also presents Clustering Large Applications (CLARA), which draws one or more random samples from the whole data set and runs PAM on the samples. Ng and Han [15] propose Clustering Large Applications based on Randomized Search (CLARANS) as an extension to PAM. Although CLARA and CLARANS are more scalable than PAM, they are inefficient for disk-resident datasets as they require multiple scans of the entire dataset and also a good clustering of a sample does not mean good clustering for the whole dataset.

```
Algorithm: Partitioning Around Medoids Clustering: PAM(X, $k$, $npass$)

Input: The dataset of objects X, number of clusters $k$, and number of iterations $npass$
Output: set of $k$ clusters, $S=\{S_0,S_1,..,S_{k-1}\}$
Initialization: Select $k$ objects randomly as medoids $(m_0,m_1,..,m_{k-1})$
Begin
        Repeat
            Step1: Assign each remaining non-medoid object to the cluster with the nearest medoid and
                   compute Total Distances of cluster $S_i$, $TD(S_i)$, $i=0,1,..,k-1$, as the sum of distances
                   from all objects to their nearest medoid $m_i$.
            Step2: For each pair of medoid $m_i$, $i=0,1,..,k-1$, and non-medoid $x_j$, $j=1,2,\ldots,|S_i|$,
                   - Compute the value $TD(S_i)(m_i \leftrightarrow x_j)$; i.e. the value of the compactness of cluster $S_i$
                     that results when swapping $m_i$ with $x_j$
                   - Select the non-medoid object $x \in S_i$ for which $TD(S_i)(m_i \leftrightarrow x)$ is minimal
                   - If $TD(S_i)(m_i \leftrightarrow x)$ is smaller than the current $TD(S_i)$,
                             Then swap $m_i$ with $x$ and set $TD(S_i) = TD(S_i)(m_i \leftrightarrow x)$
                   End
        Until (no Change in medoids or number of iterations <$npass$)
Return S
End
```

**Fig. 2. 3.** The Partitioning Around Medoids (PAM) Clustering Algorithm

PAM cannot recognize relatively small clusters in situations where good partitions around medoids clearly exist. Also PAM needs O($k(n-k)^2$) operations to cluster a given dataset, which is computationally prohibited for large $n$ and $k$. A new bisecting PAM algorithm is proposed in [49] that takes less computational time with comparable performance relative to the PAM algorithm.

***Fuzzy c-means (FCM) Clustering Algorithm***

A variant of $k$-means that allows overlapping clusters is known as Fuzzy $c$-means (FCM). The task of the traditional crisp clustering approach is to assign each data point to exactly one cluster. For fuzzy clustering, $k$ membership values are calculated for each data point $\boldsymbol{x}_j$, which are denoted by $u_{ij} \in [0,1]$, $i$=0,1…,$k$-1 and $j$=1,…,$n$. FCM allows for varying degrees of object memberships [50]. The fuzzy $c$-means clustering algorithm is shown in Fig. 2. 4. Krishnapurn *et al* [25] proposed a modified version of FCM called "Fuzzy $c$-medoids" (FCMdd) where the means are replaced with medoids. They claim that their algorithm converges very quickly and has a worst case of O($n^2$) and is an order of magnitude faster than the traditional FCM.

```
Algorithm: Fuzzy c-means Clustering Algorithm: FCM (X, m, k)
Input: The dataset X, the weighting exponent $m$, and number of clusters $k$
Output: Set of $k$ clusters, $S=\{S_0,S_1,..,S_{k-1}\}$
Initialization: - Initialize the membership matrix U with random numbers between 0 and 1 such
                  that $\sum_{i=0}^{k-1} u_{ij} = 1 \quad \forall \ j = 1,...,n$.
Begin
    Repeat
        Step1: Calculate $k$ fuzzy cluster centroids $c_i$ such that $c_i = \frac{\sum_{j=1}^{n} u_{ij}^m x_j}{\sum_{j=1}^{n} u_{ij}^m}$, $i$=0, 1,..,$k-1$
        Step2: Compute the objective function $J = \sum_{i=0}^{k-1} \sum_{j=1}^{n} u_{ij}^m (\| x_j - c_i \|_2)^2$
        Step3: Compute a new membership matrix U such that $u_{ij} = \left( \sum_{r=0}^{k-1} \left( \frac{\| x_j - c_i \|^2}{\| c_i - c_r \|^2} \right)^{1/(m-1)} \right)^{-1}$
    Until Convergence
End
```

**Fig. 2. 4.** The Fuzzy $c$-Means (FCM) Clustering Algorithm

As $m \to 1$, the partitions become increasingly crisp; and as $m$ increases the memberships become fuzzy. The value of $m=2$ is often chosen for computational convenience.

***Density Based Spatial Clustering of applications with Noise (DBSCAN) Algorithm***

The density-based clustering algorithm DBSCAN (Density Based Spatial Clustering of Applications with Noise) [51] is used because it yields the following advantages:

- DBSCAN is a very efficient and effective density-based clustering algorithm
- DBSCAN is rather robust concerning outliers
- DBSCAN can be easily implemented

In density-based clustering, clusters are regarded as regions in the data spaces in which objects are dense, and which are separated by regions of low object density (noise). The key idea of density-based clustering is that for each object of a cluster the neighborhood of a given radius (*Eps*) has to contain at least a minimum number of objects (*MinPts*), i.e. the cardinality of the neighborhood ≥ some threshold.

***Definition***

An object $\boldsymbol{x}$ is *directly-reachable* from an object $\boldsymbol{y}$ wrt. *Eps* and *MinPts* in a set of objects X, if $\boldsymbol{x}$ belongs to the neighborhood of $\boldsymbol{y}$ and number of objects in the neighborhood of $\boldsymbol{y} \geq MinPts$

***Definition***

An object $\boldsymbol{x}$ is *density-reachable* from an object $\boldsymbol{y}$ wrt. *Eps* and *MinPts* in the set of objects X, denoted as $\boldsymbol{x} >_{X} \boldsymbol{y}$, if there is a chain of objects $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_s$, $\boldsymbol{x}_1 = \boldsymbol{y}$ , $\boldsymbol{x}_s = \boldsymbol{x}$ such that $\boldsymbol{x}_i \in X$ and $\boldsymbol{x}_{i+1}$ is directly-reachable from $\boldsymbol{x}_i$ wrt *Eps* and *MinPts.*

***Definition***

An object $\boldsymbol{x}$ is *density-connected* from an object $\boldsymbol{y}$ wrt. *Eps* and *MinPts* in the set of objects X if there is an object $\boldsymbol{o} \in X$ such that both $\boldsymbol{x}$ and $\boldsymbol{y}$ are density reachable from $\boldsymbol{o}$ wrt *Eps* and *MinPts* in X.

***Definition***

A cluster is defined as a set of *density-connected* objects, which is maximal *wrt. density-reachability.*

There are different kinds of objects in density-based clustering: *core objects* (neighborhood cardinality ≥ *MinPts*) or *non-core* objects (neighborhood cardinality < *MinPts*). The non-core objects are either *border objects* (not a core object but density reachable from another core object) or *noise objects* (not a core object and not a density reachable from other objects). The procedure of finding a cluster is based on the fact that a cluster as defined is uniquely determined by any of its core objects. The DBSCAN algorithm can be described as in Fig. 2. 5.

```
Algorithm: DBSCAN (X, Eps, MinPts)
Input: The dataset X, radius Eps, and minimum neighborhood cardinality MinPts.
Output: The core objects of each cluster and the actual number of clusters k
Begin
    Repeat
        Step1: Start with an arbitrary object x which is not yet clustered
        Step2: Retrieve all objects y which are density-reachable from x wrt Eps and MinPts
        Step3: If x is a core object, a cluster S is formed wrt Eps and MinPts
        Step4: If x is a border point, no points are density-reachable from x; then visit the next point of X
    Until (All objects have been clustered)
Return (the core objects and the final set of k clusters)
End
```

**Fig. 2. 5.** The DBSCAN Algorithm

After clustering the data, representatives are needed to describe the clustering results accurately. We have to find an optimum trade-off between having small number of representatives and having an accurate description of the clusters. The core objects computed during the DBSCAN algorithm serve as good representatives. Unfortunately, the number of the produced core objects can become very large, especially in very dense areas of clusters. Thus a new set of core objects called a *complete set of specific core points* with less number of representatives is used.

***Definition***

Let $S \subseteq X$ be a cluster wrt. *Eps* and *MinPts*. Furthermore, let $Cor_S \subseteq S$ be the set of core points belonging to cluster *S*. Then $Scor_S \subseteq S$ is called a *complete set of specific core points* of *S iff* the following conditions are satisfied:

(1) $Scor_S \subseteq Cor_S$

(2) $\forall \boldsymbol{x}_i, \boldsymbol{x}_j \in Scor_S; \boldsymbol{x}_i \notin N_{Eps}(\boldsymbol{x}_j)$

(3) $\forall \boldsymbol{x} \in Cor_S\ \exists \boldsymbol{x}_i \in Scor_S; \boldsymbol{x} \in N_{Eps}(\boldsymbol{x}_i)$

***k*-Windows Clustering Algorithm**

The idea behind the *k*-windows algorithm [10] is the use of windows to determine clusters. A window is defined as an orthogonal range [52],[53] in *d*-dimensional Euclidian space. Therefore each window is a *d*-range of initial fixed area *a*. Every object that lies within a window is considered to belong to the corresponding cluster. The main idea is to construct a tree-like data structure with the properties that give the ability to perform a fast search of the set of objects. An orthogonal range search is based on this pre-processing where the tree is constructed. Thus objects that lie within a *d*-range can be found by traversing the tree. Let $X^s$ be a subset of the set X. The middle object $\boldsymbol{x}_h$ of $X^s$ with respect to the coordinate $i$ ($1 \leq i \leq d$) is defined as the object which divides the set $X^s$-$\{\boldsymbol{x}_h\}$ into two subsets $X^{s_1}$ and $X^{s_2}$, such that: $\forall x_g \in X^{s_1}$ and $\forall x_r \in X^{s_2}$, $x_g^i \leq x_h^i \leq x_r^i$ and both $X^{s_1}$ and $X^{s_2}$ have approximately equal number of vectors. The multidimensional binary tree (MDBT) *TR* that stores the data vectors of the set X is constructed as follows:

- Let $\boldsymbol{x}_r$ be the middle object of the given set X, with respect to the first coordinate $\boldsymbol{x}^1_r$. Let $X^1$ and $X^2$ be the corresponding partitions of the set X-$\{\boldsymbol{x}_r\}$. The object $\boldsymbol{x}_r$ is stored in the root of *TR*.
- Each node $\boldsymbol{x}$ of *TR* obtains a left child *left[**x**]* and a right child *right[**x**]* which are created by recursive construction of the MDBT for both the left and right nodes of the node $\boldsymbol{x}$ with respect to the next coordinate.

Let us consider a *d*-range query $Q = [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_d, b_d]$ specified by two points $(a_1, a_2, \ldots, a_d)$ and $(b_1, b_2, \ldots, b_d)$ with $a_j \leq b_j$; $j$=1,2,..,*d*. The search in the tree *TR* is affected by the orthogonal search algorithm, which accumulates the retrieved points in a set *V*. The Orthogonal Range Search (ORS) is shown in Fig. 2. 6.

```
Algorithm: Orthogonal Range Search: ORS(TR, Q, i)
Input: The tree TR, the d-range query Q and the coordinate i
Output: The set V
Begin
Step1: set V= { }, Let $x_r$ be the root of TR
Step2: Search TR, if $x_r \in Q$, then Add $x_r$ to V, else recursively search both the left and right trees of $x_r$
       with respect to coordinate $i$+1
End
```

**Fig. 2. 6.** The Orthogonal Range Search (ORS)

Iteratively, the *k*-windows algorithm moves each window in the Euclidian space by centering them on the mean of the included data vectors. This iterative process continues until no further movement results in an increase in the number of vectors that lie within each window. Subsequently the algorithm enlarges every window in order to contain as many vectors as possible from the corresponding cluster. The description of the *k*-windows algorithm is shown in Fig. 2. 7.

```
Algorithm: k-windows Algorithm(X, l, d, a)
Input: The dataset X and number of initial windows l, the d-ranges, and area a.
Output: Set of k clusters
Initialization: At first, l centroids are selected (possibly in a random way). Initial windows $W_i$, $i$=1,2,..,l
                in the d-ranges are centered on these initial centroids and each one is of area a.
Begin
Phase1 (Moving Phase):
    Repeat
      Step1: The vectors that lie within each window $W_i$ are found, using the Orthogonal Range Search
             technique of Computational Geometry [52],[53].
      Step2: The centroid of the vectors that lie within each window $W_i$ is calculated. Each centroid
             defines a new d-range, which is considered as a movement of the previous one.
    Until no window includes a significant increment of vectors
Phase2 (Enlarging Phase):
    Repeat
        The d-range windows $W_i$ are enlarged in order to include as many vectors of the corresponding
        cluster as possible.
    Until no further enlargement increases the number of vectors included in the window $W_i$
Phase3 (Checking Phase):
          The relative frequency of vectors assigned to a d-range window $W_i$ in the whole set of vectors is
          calculated. If the relative frequency is small with respect to a user specified threshold,
              Then both Phase1 and Phase2 are repeated.
Phase4 (Merging Phase): Any two overlapped windows $W_i$ and $W_j$ are merged, and the remaining set of k
        windows defines the final set of k clusters.
End
```

**Fig. 2. 7.** The *k*-windows Clustering Algorithm

***Principal Direction Divisive Partitioning (PDDP)***

The Principal Direction Divisive Partitioning (PDDP) [11] is an unsupervised clustering algorithm based on the principal component analysis [54]. The quality of the clustering solutions obtained and the cost of the computations are shown to be good. The PDDP algorithm starts with a root cluster comprising the entire set of documents. Then it splits the set into two parts using the principal directions. This process runs recursively. The result is a binary tree. To avoid generating clusters of unbalanced sizes, PDDP selects the next cluster with the largest scatter value to be split. The scatter value measures the cohesiveness of the objects within a cluster. The set of objects are represented by a features-objects matrix M, in which each column corresponds to an object and each row corresponds to a particular feature. The PDDP algorithm is shown in Fig. 2. 8 .

```
Algorithm: PDDP Clustering (X, k)
Input: The dataset X and number of clusters k
Output: Set of k clusters.
Initialization: - Start with the entire set of objects X as the initial matrix M
Begin
    Repeat
    Step1: Split the matrix M into two partitions $M_1$ and $M_2$ using the principal direction [54]
    Step2: Take the partition with largest scatter value as the new set M
    Until the desired number of clusters is reached
End
```

**Fig. 2. 8.** The PDDP Clustering Algorithm

## 2.2 Combining Multiple Clustering

Combining multiple clustering is considered as an example to further broaden and stimulate a new progress in the area of data clustering. Combining clusterings can be classified into two categories based on the level of cooperation between the clustering algorithms; either they cooperate on the *intermediate level* or at the *end-result* level. Examples of end-result cooperation are the ensemble clustering and the hybrid clustering [28]-[32]. Ensemble clustering is based on the idea of combining multiple clusterings of a given dataset X to produce a superior aggregated solution based on aggregation function. Ensemble clustering integrates a collection of "base clusterings" to produce a more accurate partition of a dataset.

Recent ensemble clustering techniques have been shown to be effective in improving the accuracy and stability of standard clustering algorithms and also it can provide novel, robust, and stable solutions. However, inherent drawbacks of these techniques are: (1) the computational cost of generating and combining multiple clusterings of the data, and (2) designing a proper cluster ensemble that addresses the problems associated with high dimensionality and parameter tuning. Hybrid clustering assumes a set of cascaded clustering algorithms that cooperate together for the goal of refining the clustering solutions produced by a former clustering algorithm(s) or to reduce the size of the input representatives to the next level of the cascaded model. Hybrid PDDP-*k*-means algorithm [32] starts by running the PDDP algorithm [11] and enhances the resulting clustering solutions using the *k*-means algorithm. Hybrid clustering violates the synchronous execution of the clustering algorithms at the same time, as one or more of the clustering algorithms stays idle till a former algorithm(s) finishes it clustering.

## 2.3 Parallel Data Clustering

Parallelization of data clustering algorithms to adapt the massive data sets requires dividing up the clustering task on multiple *P* nodes[1] so that each node can perform part of the clustering process in parallel with the other nodes. Thus each node is responsible for *n/P* vectors rather than the whole entire dataset X, where *n* is the total number of objects. The results are thus achieved faster than those obtained using single node architecture. In clustering parallelization, the computational task must be equally balanced among the processing nodes such that each node can access an equal portion of the data set, the balancing scheme is essential to minimize data communication cost during the clustering process, and to parallelize the access to the data set. Different strategies in parallelizing clustering algorithms can be adopted depending on the computational load at each node as follows [55]-[66]:

- **Independent-Parallel**: This strategy uses a single partitioning as a unit of parallelism. Each processing node has access to the whole data set and it performs a different partitioning based on different number of clusters used to partition the data set. At the end of the computation, all the resulting partitions performed in parallel by each node are collected and their accuracy is evaluated and compared by the user. The computational load of each node depends on the number of clusters used to partition the data set, thus the larger the number of cluster, the higher the computational load.

[1] The term "node" is used in this thesis to denote a processor, a process, a computer or a site.

- **Task-Parallel**: This strategy would involve getting each processing node to run a complete version of the clustering algorithm over the entire data set X, but dividing up the search space into a set of regions and assigning each single region to a single node. This parallel strategy could initially require a significant startup cost for partitioning the search space among the nodes; however the nodes would then be independent until the gathering of results at the end of the clustering process. This strategy allows a good balance of computational costs of each node.
- **Single Program Multiple Data (SPMD) Parallelization**: SPMD parallelism typically involves mapping a problem that manipulates a data structure of size $n$ into $P$ instances of a program so that each instance manipulates an $n/P$-sized block of the original domain. In other words, in the single program multiple data parallelization, each processing node executes the same code, but runs through different data. Communication occurs among the nodes when data, local (and/or) global representatives are exchanged between two or more nodes that compose the SPMD computations.
- **Multiple Program Multiple Data (MPMD) Parallelization**: MPMD parallelism can be divided into multiple levels, where different cluster algorithms are used at different levels to enhance the clustering quality of the former algorithm(s) or to reduce the size of the representatives as an input to the next level.

Most of the parallel clustering algorithms presented in the literature follow the Single Program Multiple Data parallelization strategy; some of these algorithms are: parallel *k*-means [62], parallel fuzzy *c*-means [63], parallel AutoClass [66], and many others.

### 2.3.1 Parallel Hybrid Approaches

The parallel hybrid [32] clustering assumes multiple parallel clustering algorithms are used in *cascade* (i.e. the clustering algorithms cooperate with each other at the *end-result* level) to cluster the dataset. This cascading model is mainly used to enhance the clustering quality produced by a former algorithm(s). The parallel hybrid clustering is carried out by either communication between all the different nodes or through the interaction between only facilitator (master) nodes.

***Parallel Hybrid Principal Direction Divisive Partitioning and k-means Algorithm***

For parallel PDDP, the matrix M (features-object matrix) is stored in the Compressed Sparse Row (CSR) format. Each of the P nodes has a data structure consisting of the three arrays:

- MM: A real number array containing the real values of the nonzero elements of M (features-object matrix) stored on each node row by row
- CM: An integer array for the column indices corresponding to the array MM
- RM: An integer number array containing the pointers to the beginning of each row in the arrays MM and CM

Each data vector is broken into many parts, and different parts are distributed to different nodes. Each row of the nonzero entries is distributed to its corresponding node in a row-wise cyclic striped partitioning (RCSP) form [60]. To compute the mean vector of the global cluster, all nodes can compute its own part of the mean vector in parallel. The mean vector is also distributed among the nodes, in the same way as for each vector. A high level description of the parallel PDDP algorithm is illustrated in Fig. 2. 9.

```
Algorithm: Parallel PDDP Clustering (M, $h$, P)
Input: The matrix M, the tree height $h$, and number of nodes P
Output: Set of global $k$ centroids
Initialization: - Let M be the root of the global tree and divide M on the P nodes
Begin
Step1: For each level $i$ =1 to $h$
        For each cluster $S_i$ at level $i$
            If $S_i$ is singleton, (A singleton means that its object set is exactly the same as that of its parent cluster) then process the next cluster, else
              - Each node $N_p$ evaluates local part of the mean vector $c^p_i$ of the cluster $S_i$ and local part of the leading eigenvector (the eigenvector of the covariance matrix with maximum variance) $\boldsymbol{u}_i^p$ using local matrices in a CSR format.
              - Global centroid $c_i$ of cluster $S_i$ and global leading eigenvector $\boldsymbol{u}$ are produced in a CSR representation using the local parts at each node.
            For each document $\boldsymbol{x}$ in the cluster $S_i$
                If dot product $(\boldsymbol{u},\boldsymbol{x}) \geq 0$, then assign $\boldsymbol{x}$ to the left child of cluster $S_i$
                Else assign $\boldsymbol{x}$ to the right child of cluster $S_i$
            End
Step2: The leaf nodes of the tree comprise the final set of $k$ clusters
End
```

**Fig. 2. 9.** Parallel PDDP Clustering Algorithm

The parallel PDDP algorithm is fast in clustering web documents datasets. However, the quality of its clustering for some cases may not be good as that produced by the parallel *k*-means clustering [62] algorithm. Thus the parallel *k*-means is used to refine the clustering solutions that results from the parallel PDDP algorithm. This hybrid combination is based on *end-result* cooperation between both the parallel PDDP and parallel *k*-means, this strategy of cooperation is mainly to enhance the clusters produced by the PDDP algorithm and to give the *k*-means algorithm a good initial points. The parallel hybrid PDDP and *k*-means clustering algorithm is shown in Fig. 2. 10.

```
Algorithm: Parallel Hybrid PDDP and PKM Clustering(M, h, P, k)
Input: The matrix M, the tree height h, number of nodes P, and number of clusters k
Output: The set of k clusters
Initialization: Divide the matrix M on the P nodes
Begin
    Step1: Run the Parallel PDDP and generate centroids for the k clusters
    Step2: Use these k centroids as initial centroids to the parallel k-means algorithm
    Step3: The parallel k-means generates the final set of clusters k
End
```

**Fig. 2. 10.** Parallel Hybrid PDDP and *k*-means Clustering Algorithm

## 2.4 Distributed Data Clustering

Advances in computing and communications over wired and wireless networks have resulted in many pervasive distributed computing environments. The internet, intranets, local area networks, ad hoc wireless networks, and sensor networks are some examples. These environments often come with different distributed sources of data and computations. The transmission of huge amounts of data from nodes to another central node is in some application areas almost impossible. Distributed data clustering is a generalization of parallel data clustering where data is equally partitioned among nodes. In astronomy, for instance, several highly sophisticated space telescopes are spread all over the world. These telescopes gather data unceasingly. Each of them is able to collect 1GB of data per hour [67] which can only, with difficulty, be transmitted to a central node to be analyzed centrally there. On the other hand, it is possible to analyze the data locally where it has been generated and stored. Aggregated information of different local nodes are combined and analyzed. The result of the central analysis may be returned to the local nodes, so that the local nodes are able to put their data into a global context. Also, centralized clustering can hardly scale to magnitude of the data e.g. the Web. Google for example, is able to index the web daily and responds to millions of queries per day because it employs a farm of distributed computing nodes that apply distributed algorithms for content indexing and query processing. Also a major thrust of gene expression analysis over the last twenty years has been the acquisition of enormous amount of various distributed sources of gene expression datasets. Thus, it is becoming increasingly important to perform clustering of distributed data in-place, without the need to pool it first into a central node. Applications of distributed clustering are numerous. They often try to solve problems in mathematics and science. Specific areas and examples include:

- Specific projects include: astronomy (SETI@home), biology (Folding@home, Predictor@home), climate change (CPDN), physics (LHC@home), cryptography (distributed.net), and biomedicine (grid.org). Those projects are usually built on top of a common platform providing low level services for distributed or grid computing. Examples of those platforms include: Berkeley Open Infrastructure for Network Computing (BOINC), Grid.org, World Community Grid, and Data Mining Grid.
- Supermarket chains where check-out scanners that located at different stores gather data unremittingly.
- Furthermore, international companies such as DaimlerChrysler[2] have some data which is located in Europe and some data in the US, those companies have various reasons why the data cannot be transmitted to a central site, e.g. limited bandwidth or security aspects.

### 2.4.1 Distributed Clustering Definition and Goals

Distributed clustering in general deals with the problem of finding patterns in an environment where data is either naturally distributed, or could be artificially partitioned across computing nodes. It implies distribution of one or more of: users, data, hardware, or software [68],[69]. Centralized data clustering systems do no address some of the requirements of the distributed environments, such as data distribution across nodes, scalability and changing information between nodes. The main assumption in distributed clustering is that data is distributed over a number of nodes, and that is desirable through distributed clustering techniques a global model that reflects the characteristics of clustering the whole data set. In general, the attributes ascribed to the distributed clustering system can be identified in terms of its local models, a global model of the clustering process, and the clustering algorithm itself.

[2] http://www.chryslercanada.ca/en/

***Local Models***

Each node is required to build and maintain a local model of the problem space it is responsible for. In terms of data clustering this is the cluster prototype. Let $N_0$, $N_1$, .. , $N_{P-1}$ be the set of nodes in the system running the clustering algorithm A. Let $Z_0$, $Z_1$,..,$Z_{P-1}$ be the corresponding set of local models. Each local data model $Z_p$; $p$=0,1,..,P-1 consists of the prototype representation (e.g. centroids) of the collection of local data objects $X_p$ the node $N_p$ owns.

***Global Model***

The global model is a representation of the whole cluster sets produced by all nodes, it is constructed based on the local models generated at each node and it is considered as a common repository of knowledge that is accessible to all nodes. Global model is what makes individual nodes aware of the big picture and serves as a catalog which nodes consult to find out if the desired clustering quality is achieved or not and then update their local models based on the generated global model. At the level of generating the global model, nodes are communicating through sharing and negotiating by using only the local models while data are preserved for privacy issues. Distributed data clustering can be illustrated as in Fig. 2. 11. The architecture design of the distributed system relies on the presence of multiple nodes, not necessarily running on the same machine, that are able to communicate and negotiate among themselves to achieve better performance. The nodes cooperate by sharing resources, such as data objects and information about their local (and/or) global models such as clusters prototypes.

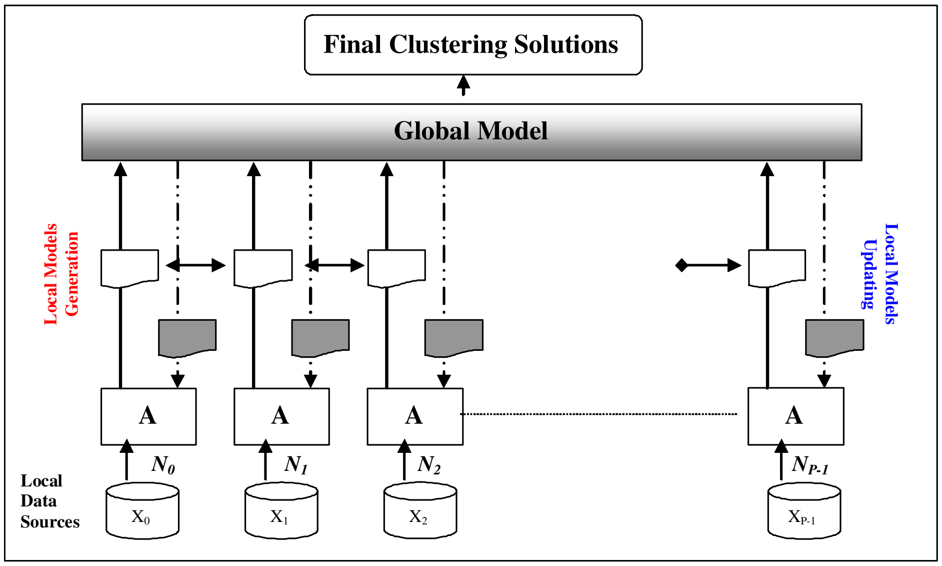


**Fig. 2. 11.** Distributed Data Clustering

### 2.4.2 Challenges in Distributed Data Clustering

A number of challenges (often conflicting) arise when developing distributed clustering approaches:

- **Communication model and complexity**: It is desirable to develop methods that have low communication complexity, especially in mobile applications such as sensor networks, where communication consumes battery power. In addition, although many distributed systems are designed for sharing large data files (e.g. music, movies), a distributed clustering system that involves analyzing such data, may not have the luxury to frequently exchange large volume of data among the nodes in the distributed network just for data clustering purposes. Thus distributed clustering environment should be "light-weight"; it should be able to perform distributed data analysis with minimal communication overhead.
- **Quality of the global model**: in distributed systems, data is clustered locally at each node. Afterwards only the information (summary) about the local clusters is aggregated from all nodes to construct a global model of data. Quality of the global model derived from the data should be either equal or comparable to a model derived by a centralized method.
- **Privacy of local data***:* in some situations when local data is sensitive and not easily shared, it is desirable to achieve a certain level of privacy of local data while deriving the global model. Although not yet proven, usually deriving high quality models requires sharing as much data as possible, thus incurring higher communication cost and sacrificing privacy at the same time.

### 2.4.3 Distributed Clustering Architectures

There are two architectures in distributed clustering: Peer-to-Peer and facilitator-workers. In the Peer-to-Peer model, all peers (nodes) perform the same task and exchange the necessary information to perform their clustering goals. In the facilitator-workers model, one node is designed as a facilitator (master node), and all other nodes are considered as worker (slave) nodes. The facilitator is responsible for partitioning the task among the workers and aggregating their partial results.

***<u>Peer-to-Peer (P2P) Architecture</u>***

P2P networks are networks where peers communicate and transport information directly with each other. All nodes communicate with each other either to generate the global model or to enhance the performance of each other either by broadcasting messages or exchanging the local models (and/or) global models. An example of communication between peers through exchanging of local models is illustrated in Fig. 2. 12.

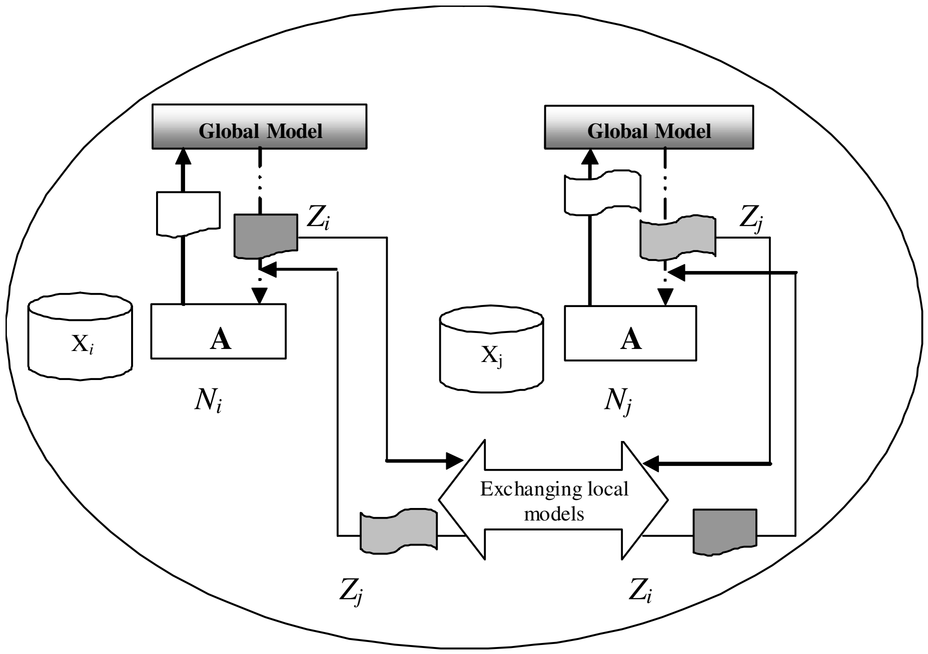


**Fig. 2. 12.** Peer-to-Peer Communication through Exchanging of Local Models

P2P networks are different from facilitator-workers model as there is no central control, each peer has equal functionality: a peer is a facilitator and a worker; it is dynamic where each peer can join and leave the network. Benefits of P2P networks include:

- Efficient use of resources
- Scalability
  - Consumers of resources also donate resources
  - Aggregate resources grow naturally with utilization
- Reliability
  - Geographic distribution
  - No single point of failure
- Second generation P2P overlay networks have the following characteristics
  - Ease of administration
  - Nodes self organize
  - No need to deploy servers to satisfy demand (i.e. peer-scalability)
  - Built-in fault tolerance, replication, and load balancing

The P2P networks have been applied in many applications where the datasets are inherently distributed, some of these applications include:

- **Bioinformatics:** P2P networks have begun to attract attention from scientists in many disciplines, especially those that deal with large datasets such as bioinformatics. P2P networks can be used to run large programs designed to carry out tests to identify drug candidates. The first such program was begun in 2001 by the Centre for Computational Drug Discovery[3] at Oxford University[4] in cooperation with the National Foundation for Cancer Research[5].
- **Education and Academia:** Due to the fast distribution and large storage space features, many organizations are trying to apply P2P networks for educational and academic purposes. For instance, Pennsylvania State University[6], MIT[7], and Simon Fraser University[8] are carrying on a project called *LionShare*[9] designed for facilitating file sharing among educational institutions globally.
- **Business:** P2P networks have already been used in business areas, but it is still in the beginning stages. Currently, Kato et al's studies [10] indicate over 200 companies with approximately $400 million USD are investing in P2P network. Besides file sharing, companies are also interested in distributing computing, distributed clustering, content distribution, e-marketplace, distributed search engines, groupware and office automation via P2P networks. There are several reasons why companies prefer P2P sometimes, such as: Real-time collaboration, a server cannot scale well with increasing volume of content; a process requires strong computing power; a process needs high-speed communications, etc. At the same time, P2P is not fully used as it still faces a lot of security issues.

[3] http://www.chem.ox.ac.uk/ccdd/ccdd.html

[4] www.ox.ac.uk/

[5] http://www.nfcr.org/

[6] www.psu.edu/

[7] http://web.mit.edu/

[8] www.sfu.ca/

[9] www.lionshare.its.psu.edu/

[10] www.en.wikipedia.org/wiki/Peer-to-peer

- Additional applications of peer-to-peer networks include:
    - VoIP (Voice over IP)
    - Streaming media
    - Instant messaging
    - Software publication and distribution
    - Media publication and distribution.

***Facilitator-Workers Architecture***

A facilitator node is used to collect local models from all nodes. Its job then is constructing the global model and sending this global model back to the worker nodes to update their local models. A facilitator-workers network has the advantage over P2P network as it consumes lower communication cost compared that of a P2P network. However, if the facilitator fails or decides to leave the network, all the other nodes will remain in an idle state until a new facilitator joins the system. Limitations of the facilitator-workers architecture that the peer-to-peer networks try to solve are:

- Presents a single point of failure
- Requires administration
- Provides unused resources at the network edge
- Scalability is hard to achieve

### 2.4.4 Locally and Globally Optimized Distributed Clustering

The objective of locally-optimized distributed clustering is to employ collaboration between nodes to improve the quality of local clustering solutions [45]. This implies that there is no common set of clusters across nodes, but rather local clusters that reflect the characteristics of local data sets. However, by strategically moving (or copying) certain data objects from one node to another the quality of local clusters is boosted. This effectively creates a network where certain nodes have authority over certain clusters, which can simplify query-answering in P2P networks by routing queries to relevant nodes only, instead of flooding the query throughout the network. Collaboration between nodes is achieved through sharing of summarized local clustering information across the network.

The goal of globally-optimized distributed clustering is to compute one set of clusters over all local data sets, as if the data were pooled into one location and a centralized clustering algorithm was applied to it. As a result, at the end all nodes require the same clustering solution, but local data stays the same. Globally optimized clustering is suitable for speeding up clustering of large datasets by partitioning the data among nodes. Both the Peer-to-Peer [44],[70],[71], and the facilitator-workers [72]-[76] architectures can be used to achieve globally optimized clustering.

### 2.4.5 Communication models

Communication between nodes in distributed clustering can be categorized into three classes (in increasing order of communication cost):

- **Communication Prototypes**, which involves calculating local models that are then sent to peers or a central node. These models often are comprised of cluster prototypes, e.g. centroids in P2P *k*-means [44], cluster dendrogram in RACHET [77], or generative models as in DMBC [78].
- **Communication Representatives**, in which nodes select a number of representative samples of the local data to be sent to a central node for global model generation, such as the case in the KDEC distributed clustering algorithm [76] and the DBDC algorithm [79].
- **Communication Data**, in which nodes exchange actual data objects, i.e. data objects can change their nodes to facilitate construction of clusters that exist in certain nodes only, such as the case in collaborative clustering scheme in [80], and the distributed signature-based clustering in [81].

### 2.4.6 Exact *vs*. Approximate Distributed Clustering Algorithms

A distributed clustering algorithm can be described as exact or approximate. Exact algorithms produce a final global model identical to a hypothetical model generated by a centralized approach having access to the full data set. The exact algorithm works as if the local data sets at each node were bought together into one data set first, then a centralized clustering algorithm had been performed on the whole data set. The clustering solutions are then distributed again by intersecting the local data sets with the global clustering solutions. Approximate algorithms on the other hand, produce a model that closely approximates a centrally-generated model. Most distributed data clustering research focuses on approximate algorithms as they tend to produce comparable results to exact algorithms with far less complexity [82].

### 2.4.7 Distributed Clustering Algorithms

To summarize the state of art of distributed clustering, In [83], they illustrate the various distributed clustering algorithms and their taxonomies according to the time frames and the different classification of the original centralized approaches as shown in Fig. 2. **13**. We updated the taxonomy with the recent research work in distributed clustering during the last year. Some of these distributed clustering algorithms are described next.

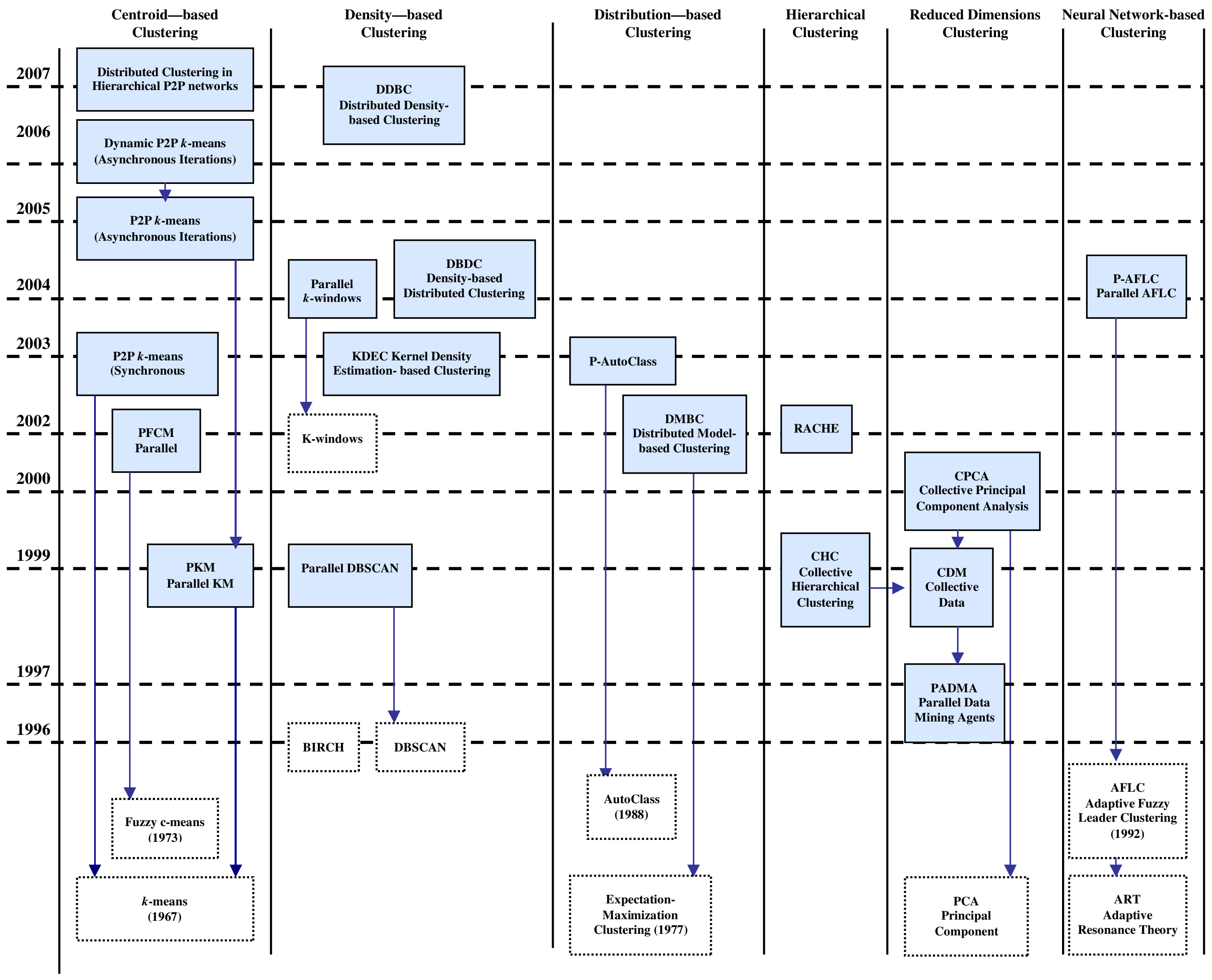

Centroid—based Clustering
Density—based Clustering
Distribution—based Clustering
Hierarchical Clustering
Reduced Dimensions Clustering
Neural Network-based Clustering
2007
2006
2005
2004
2003
2002
2000
1999
1997
1996
Distributed Clustering in Hierarchical P2P networks
Dynamic P2P k-means (Asynchronous Iterations)
P2P k-means (Asynchronous Iterations)
P2P k-means (Synchronous
PFCM Parallel
PKM Parallel KM
Fuzzy c-means (1973)
k-means (1967)
DDBC Distributed Density-based Clustering
Parallel k-windows
DBDC Density-based Distributed Clustering
KDEC Kernel Density Estimation- based Clustering
K-windows
Parallel DBSCAN
BIRCH
DBSCAN
P-AutoClass
DMBC Distributed Model-based Clustering
AutoClass (1988)
Expectation-Maximization Clustering (1977)
RACHE
CHC Collective Hierarchical Clustering
CPCA Collective Principal Component Analysis
CDM Collective Data
PADMA Parallel Data Mining Agents
PCA Principal Component
P-AFLC Parallel AFLC
AFLC Adaptive Fuzzy Leader Clustering (1992)
ART Adaptive Resonance Theory

**Fig. 2. 13.** Taxonomies of Distributed Clustering Algorithms in [83]

***<u>Distributed k-Means (DKM) Clustering Algorithm</u>***

The classical *k*-means [7] needs to perform a large number of "nearest-neighbor" queries for the data vectors in the dataset. As one would have to run several iterations, it is generally not feasible to run the naïve *k*-means for large datasets. Because of this drawback of the KM clustering, a distributed scalable version is developed. In the DKM [84],[85], initially each node will receive the entire centroid list, but only the "root node", $N_0$ ,will compute the initial centroids and then broadcasts these *k* initial centroids to all other nodes. Thus each node will compute the distances between its local vectors to each of the centroids. A series of assignments are generated mapping vectors to the closest centroid. Each node then gathers the sum of all vectors allocated to a given cluster and computes the mean of the vectors assigned to a particular cluster using Message Passing Interface (MPI) reduction routine [86]. The MPI reduction routine sums all the local copies of centroids from all nodes (*reduction operation*) and broadcasts the sum to all the nodes (*broadcast operation*). This is repeated for every cluster and a new set of centroids is available at each node. Then the local data vectors can be reassigned with respect to the newly calculated centroids. The objective function *J* (Eq.(2.25)) is used as the quality measure of clustering. Each node computes the local *J* for the portion of the dataset over which it is working. Then, a simple reduction (by summing all values of local *J* and broadcasting the global value to all nodes) of local *J* values among all nodes will determine the overall performance of clustering. The DKM is shown in Fig. 2. 14.

```
Algorithm: Distributed k-means Clustering: DKM(P, $X_p$, k)
Input: Number of nodes P, local datasets $X_p$, and number of clusters $k$
Output: Set of global $k$ centroids
Initialization: The root node $N_0$ selects randomly the initial global centroids $c_i$, $i$=0,1,..,$k$-1 and broadcasts
                (replicates) these initial centroids to the P-1 nodes.
Begin
  Repeat
    Step1: Each node $N_p$, $p$=0,1,..P-1 computes the distance of each local vector to the $k$ global centroids.
    Step2: The local objects are assigned to the closest centroid and the local $J$ is computed at node $N_p$.
    Step3: A reduction of the local centroids and local objective functions is performed to produce the
           global $k$ centroids and global $J$ using MPI reduction routine.
  Until Convergence
End
```

**Fig. 2. 14.** The Distributed *k*-means (DKM) Clustering Algorithm

***Distributed Bisecting k-means (DBKM) Clustering Algorithm***

In the distributed bisecting *k*-means [65], the root node $N_0$ selects randomly the two initial centroids $c_1$ and $c_2$ as centroids for the initial partitions $V_1$ and $V_2$, and then broadcasts the selected initial centroids to all other nodes. Each node applies the basic *k*-means algorithm and finds two local sub clusters $My\_V_1$ and $My\_V_2$ from the local objects. This step is called the *Bisecting step*. This step is repeated for every number of iterations *ITER* until the best global splits $V_1$ and $V_2$ are found. The global partition ($V_1$ or $V_2$) with the lowest similarity (or any other splitting criterion) is chosen to be split next; this step is called the *Picking step* and it is repeated until the desired number of clusters *k* is reached. The local partitioning at each node is updated after the picking step is performed. Fig. 2. 15 outlines the DBKM clustering.

```
Algorithm: Distributed Bisecting k-means Clustering : DBKM(P, $X_p$, ITER, $k$)
Input: Number of nodes P, local datasets $X_p$, number of iteration ITER, and number of clusters $k$
Output: Set of global $k$ centroids
Initialization: - At each node $N_p$ let $My\_V^p = X_p$
Begin
For level =1 to $k$-1 (The Picking Step)
    Step1: For I=1 to ITER (The Bisecting Step)
        - $N_0$ selects randomly the initial global centroids $c_1$ and $c_2$ and broadcasts them to the P-1 nodes.
        - Each node $N_p$ $p$=0,1,..,P-1, finds two local sub-clusters $My\_V_1^p$ and $My\_V_2^p$ from the local
          dataset $X_p$ using the basic $k$-means clustering algorithm
        - Global $V_1$ and $V_2$ are found such that, $V_1=\bigcup_{p=0}^{P-1} My\_V_1^p$ and $V_2=\bigcup_{p=0}^{P-1} My\_V_2^p$
    Step2: Take the best of these splits as $V_1$ and $V_2$ and update $My\_V_1^p$ and $My\_V_2^p$
    Step3: The global centroids $c_1$ and $c_2$ are updated using MPI reduction routine.
    Step4: Each node $N_p$ evaluates the similarity of the local partitions $My\_V_1^p$ and $My\_V_2^p$, and global
           internal similarity of the partitions $V_1$ and $V_2$ are found using MPI reduction routine.
    Step5: Take the split $V_1$ or $V_2$ that produces the clustering with the highest overall similarity (or any
           other homogeneity criterion) and set V to the remaining partition.
    Step6: Each node $N_p$ updates the local $My\_V^p$ based on the new set V found in step5.
End
```

**Fig. 2. 15.** The Distributed Bisecting *k*-means Clustering Algorithm

The communication between nodes to is facilitated using the Message Passing Interface (MPI), more details about the MPI routines are defined in Appendix A.

***<u>Distributed Collaborative Clustering (DCC) Algorithm</u>***

The Distributed Collaborative Clustering (DCC) algorithm [45] belongs to the family of distributed homogenous collaborative clustering. In this model the goal is not to achieve one global clustering solution, but rather maximizing the quality of the local clustering solutions at each node through *collaboration*. This is achieved through augmenting and enriching the local clustering solutions with recommendations from peer nodes after exchanging summarized cluster information.

The (DCC) algorithm assumes a set of P nodes, where the data at each node is clustered independently to form an “initial local clustering.” The goal is to enhance the local clustering by gaining access to summarized cluster information from other nodes. Each node broadcasts its summarized cluster information to all other nodes. Each other node collects this information from its peers, calculates similarity values between local data and the peer cluster summaries, then it sends a list of recommended local objects to be merged with the peer clusters. The original node receives recommendations from all its peers, and based upon its own judgment it either merges the recommended objects or rejects them. This process results in an expanded local clustering that is both of higher quality and of more coverage than the initial local clustering.

Algorithms in Fig. 2. 16 and Fig. 2. 17 describe the DCC algorithm. In Fig. 2. 16 the algorithm recommends objects to peers based on the received peer cluster prototypes. It starts by exchanging cluster prototypes between peers. Each node receives *P-1* peer summaries; each peer summary is a list of cluster prototypes $c_i$. The receiving node compares the cluster prototypes to its own local dataset and builds a similarity matrix *SM* between its local objects and peer cluster prototypes. The node then sends to each peer those objects that are most similar (above similarity threshold $r_T$) to the cluster prototype summaries; those objects are called “peer-positive” objects.

In Fig. 2. 17 the algorithm merges the recommendations of each peer node. It starts by receiving recommended objects from each peer. It then follows the same logic of the SHC [16] clustering algorithm, except that it does not create new clusters for objects that were not assigned to any cluster.

```
Algorithm: DCC: (Recommend to Peers)($X_p$, peer cluster prototypes, P)
Input: local objects $X_p$, peer cluster prototypes and number of nodes P
Output: Recommendation {$X^+$} of local objects to peer node
Initialization: Initial clustering done at each node and cluster prototypes are calculated
Begin
    Step1: Each node $N_p$ receives cluster prototypes from a peer node.
    Step2: The node $N_p$ initializes similarity matrix $SM$ to hold similarity values between peer clusters
           and local data objects.
    Step3: The node $N_p$ updates $SM$ by calculating the similarity between each cluster prototype $c_i$ and
           each local object $\boldsymbol{x}_j$.
    Step4: At each node $N_p$,
               For each cluster prototype
                   Calculate a set of recommended objects $X_i^+$ for peer cluster $S_i$, such that the objects
                   in $X_i^+$ have $SM(\boldsymbol{x}_j, c_i) > r_T$ (similarity threshold)
    Step5: $N_p$ sends recommended sets {$X^+$} to peer to be merged with its local clusters
End
```

**Fig. 2. 16.** DCC Algorithm: Collecting Prototypes and Recommending Merge of Objects

```
Algorithm: DCC (Merge Peer Recommendations)({X+})
Input: Recommendation {X+} from peer node
Output: Modified local clusters after merging peer recommendation
Begin
    For each peer
        Step1: Receive a set of recommendations {$X^+$} from peer
        Step2: For each recommendation set $X_i^+$
                   Consider recommendation set $X_i^+$ (corresponding to cluster $S_i$) as a partial data set to
                   be clustered with the existing local clusters and apply the SHC algorithm on it
End
```

**Fig. 2. 17.** DCC Algorithm: Collecting Recommendations and Merging Peer Objects

In Fig. 2. 17, to avoid the side effect of creating small clusters of objects that did not fit into any existing cluster, do not allow the creation of new clusters in *Step 2*. Those objects that do not fit into existing clusters are simply dropped.

***Distributed Density-Based Clustering (DDBC) Algorithm***

The DDBC [87] algorithm first clusters the data locally and extracts suitable representation out of these clusters. These representatives are sent to a facilitator node where the complete clustering is stored based on the local clustering. For both the local and global clustering the DBSCAN [51] algorithm is used. Two suitable local models called $REP_{scor}$ and $REP_{k\text{-}means}$ are designed to create the local model based on the definition of the *complete set of specific core points, $Scor_{Si}$*. In $REP_{scor}$, each local cluster $S_i$ is represented by the set $Scor_{Si}$. If we assumed that we have found $k$ clusters on a local node $N_p$, the local model $LocalModel_p$ is formed by the union of the different sets $Scor_{Si}$. A specific ε-range is assigned to all the specific core points $\boldsymbol{x}$ in the set $Scor_{Si}$, which indicates the representing area of each point. This ε-range value is a part of the local model and is evaluated on the facilitator node to develop an accurate global model. If we assume that $k$ clusters are found at each node, then the local model on node $N_p$ is defined as:

$$LocalModel_p = \bigcup_{i=1,..,k} \{(\boldsymbol{x}, \varepsilon_x) \mid \boldsymbol{x} \in Scor_{Si}\} \tag{2. 28}$$

The $REP_{k\text{-}means}$ approach is also based on the *complete set of specific core points*. In contrast to the foregoing approach, the specific core points are not directly used to describe a cluster. Instead, the number $|Scor_{Si}|$ and the elements of $Scor_{Si}$ are used as input parameters for further "clustering step" using the $k$-means clustering algorithm. For each cluster $S_i$, found by DBSCAN, $k$-means yields $|Scor_{Si}|$ centroids within $S_i$. These centroids are used as representatives. The local model $REP_{k\text{-}means}$ at each node $N_p$ is found using the $k$-means algorithm as described in Fig. 2. 18.

```
Algorithm: Further Local DBSCAN Clustering using k-means(S_i)
Input: Each local cluster $S_i$ , $i$=0,1,..,$k-1$, which was found throughout the DBSCAN at the local node $N_p$
Output: Local model $REP_{k\text{-}means}$
Begin
    Step1: Each local cluster $S_i$ on the local node $N_p$ is again re-clustered using the k-means algorithm,
           then a set of $|Scor_{Si}|$ centroids $c_{i,1}, c_{i,2}, .., c_{i,|Scor_{S_i}|}$ is produced.
    Step2: Each centroid $c_{i,j}$, $j$=1,2,..,$|Scor_{Si}|$ is assigned with a ε-range value which indicates the
           represented area by $c_{ij}$.
    Step3: The local model describing the set of $k$ clusters on the node $N_p$ is
           $LocalModel_p = \bigcup_{i=0,1,..,k-1} \bigcup_{j=1,..,|Scor_{S_i}|} \{(c_{i,j}, \varepsilon_{c_{i,j}})\}$
End
```

**Fig. 2. 18.** Further Local DBSCAN Clustering using *k*-means Algorithm

In the DBSCAN algorithm, each specific local representative forms a cluster on its own. After the local models have been constructed on each node, where each local model consists of a set of *k* local clusters with the corresponding representatives; a global model based on the created local models is created at the facilitator node. These local models are sent to the facilitator node and the facilitator nodes checks whether it is possible to merge two or more of these clusters together. These merged local representatives together with the unmerged local representatives form the global model at the facilitator node. Thus the global model consists of clusters containing one or several local representatives. To find such a global model, the density-based clustering algorithm DBSCAN is used again. We would like to create a clustering similar to the one produced by DBSCAN if applied to the complete dataset with the central parameters settings. As we have access to the set of all local representatives, the global parameter setting for both the $MinPts_{global}$ and $Eps_{global}$ values has to be adapted to this aggregated local information. Suitable values for $MinPts_{global}$ and $Eps_{global}$ can be found in [87] such that $Eps_{global}= 2Eps_{local}$ and $MinPts_{global} = 2$. After having created a global clustering, the facilitator node sends the complete global model to all other nodes. Each node reassigns all locally objects independently from each other. On each node, two former independent clusters may be merged together due to this new re-labeling. Furthermore, points, which were formerly assigned to local noise, are now part of a global cluster. The DDBC algorithm is presented in Fig. 2. 19.

```
Algorithm: Distributed Density-Based Clustering : DDBC($X_p$, $Eps_{local}$, $MinPts_{Local}$)
Input: Local datasets $X_p$, $p$=0,1,..,P-1, local $Eps_{local}$ , and local $MinPts_{Local}$
Output: Set of global $k$ clusters
Begin
    Step1: Each node $N_p$ runs the DBSCAN algorithm with input parameters $Eps_{local}$ and $MinPts_{Local}$
           locally, and generates the local model $LocalModel_p$
    Step2: Each node $N_p$ sends its local model $LocalModel_p$ to the facilitator node.
    Step3: The facilitator node runs DBSCAN algorithm globally with global parameters $MinPts_{global}$ and
           $Eps_{global}$ such that $Eps_{global}= 2Eps_{local}$ and $MinPts_{global} =2$
    Step4: The facilitator node sends the global model to each node $N_p$.
    Step5: Each node $N_p$ re-labels all local objects, such that two former independent clusters are merged
           together due to this new re-labeling.
End
```

**Fig. 2. 19**. The DDBC Algorithm

### 2.4.8 Distributed Clustering Performance Evaluation

The performance of the distributed algorithm in comparison with the corresponding centralized algorithm is evaluated by different measures of performance; some of these measures are illustrated next.

***Distributed Time ($T_d$)***

The time taken by a clustering algorithm to execute on a single node is called the *centralized execution time* and is denoted by $T_c$. The execution time of the corresponding distributed clustering algorithm on $P$ identical nodes is called the *distributed execution time* and is denoted by $T_d$. A distributed clustering algorithm incurs several overheads during execution. These include overheads due to idling, communication, and contention over shard data structure. The sum total of time spent by all nodes doing work, which is not done by the centralized technique, is termed as the total overhead $T_o$. Since the sum total of time spent by all nodes is $PT_d$, and the total overhead is $T_o$, we can see that,

$$PT_d = T_c + T_o \qquad (2.\ 29)$$

Or

$$T_d = \frac{T_c + T_0}{P} \qquad (2.\ 30)$$

***Speedup***

The *Speedup* performance measure is defined as the ratio of the execution time for clustering a dataset into $k$ clusters on one node to the execution time for identically clustering the same dataset on P nodes. *Speedup* is a summary of the efficiency of the distributed clustering algorithm. The *speedup* measure is defined as:

$$Speedup = \frac{Running\ time\ with\ 1\ node}{Running\ time\ with\ P\ nodes} = \frac{T_c}{T_d} \qquad (2.\ 31)$$

***Efficiency***

The *Efficiency* of a distributed clustering algorithm is defined as the ratio of the speedup obtained to the number of nodes used. Therefore,

$$Efficiency = \frac{Speedup}{P} = \frac{T_c}{T_c + T_o}\ \% \qquad (2.\ 32)$$

## 2.5 Discussions

The various fields relevant to the research in this thesis have been reviewed so as to give the reader the necessary background to follow the work introduced in later chapters, specifically: intensive background on data clustering similarity and evaluation criteria, clustering algorithms, and distributed data clustering paradigm. The background on clustering in general will be of benefit throughout this thesis, since it is the backbone of the work introduced herein. Specifically, in chapter 3, where the cooperative model invokes multiple clustering techniques into one model of cooperation. The background on distributed data clustering will be of benefit when discussing chapter 7, where the cooperative clustering model is applied in distributed environments.

# Chapter 3
# Cooperative Clustering

It is well known that no clustering method can adequately handle all sorts of cluster structures and properties (e.g. overlapping, shape, size and density). In fact, the cluster structure produced by a clustering method is sometimes an artifact of the method itself that is actually imposed on the data rather than discovered about its true structure. Combining clusterings invokes multiple clustering algorithms in the clustering process to benefit from each other to achieve global benefit (i.e. they cooperate together to attain better overall clustering quality). One way to enable concurrent implementation of the multiple clustering algorithms and benefit from each other with better performance synchronously is by using cooperative clustering. The cooperative clustering model is mainly based on four components (1) co-occurred sub-clusters, (2) histogram representation of the pair-wise similarities within the sub-clusters, (3) cooperative contingency graph, and (4) coherent merging of sub-clusters. These components are developed to obtain a cooperative model that is capable of clustering data with better quality than that of the adopted individual techniques.

This chapter is organized as follows: Section 3.1 gives an overview of the proposed model. Sections 3.2 and 3.3 illustrate the set of inputs to the cooperative model and the preprocessing steps of data, respectively. The cooperative clustering model and its complexity are illustrated in section 3.4. The *Overall Weighted Similarity Ratio (OWSR) measure* and the *Scatter F-measure* are presented in sections 3.5 and 3.6, respectively. The scalability of the cooperative model in terms of number of clustering techniques is discussed in section 3.7. Section 3.8 discusses the cooperation on the intermediate levels, and finally some discussions and conclusions are shown in section 3.9.

## 3.1 An Overview

The cooperative clustering (CC) model is mainly based on a cooperative methodology between multiple clustering approaches for the goal of achieving better clustering quality than that of the non-cooperative approaches. The cooperative model takes first the dataset and a set of clustering algorithms as inputs. A number of preprocessing steps is performed on the dataset before entering to the model. Each clustering algorithm generates a set of *k* clusters. The cooperative model employs an agreement strategy between the multiple clustering algorithms to find the set of intersections between the different clusterings informs of sub-clusters.

The extracted sub-clusters are then represented by similarity histograms, then each sub-cluster acts as a node in a cooperative contingency graph (*CCG*). Edges of the *CCG* are weighted by a cohesiveness factor for merging two sub-clusters into one cluster. Finally, a coherent merging of sub-clusters is performed to attain the original number of clusters. Fig. 3. 1 illustrates the different components of the cooperative clustering model.

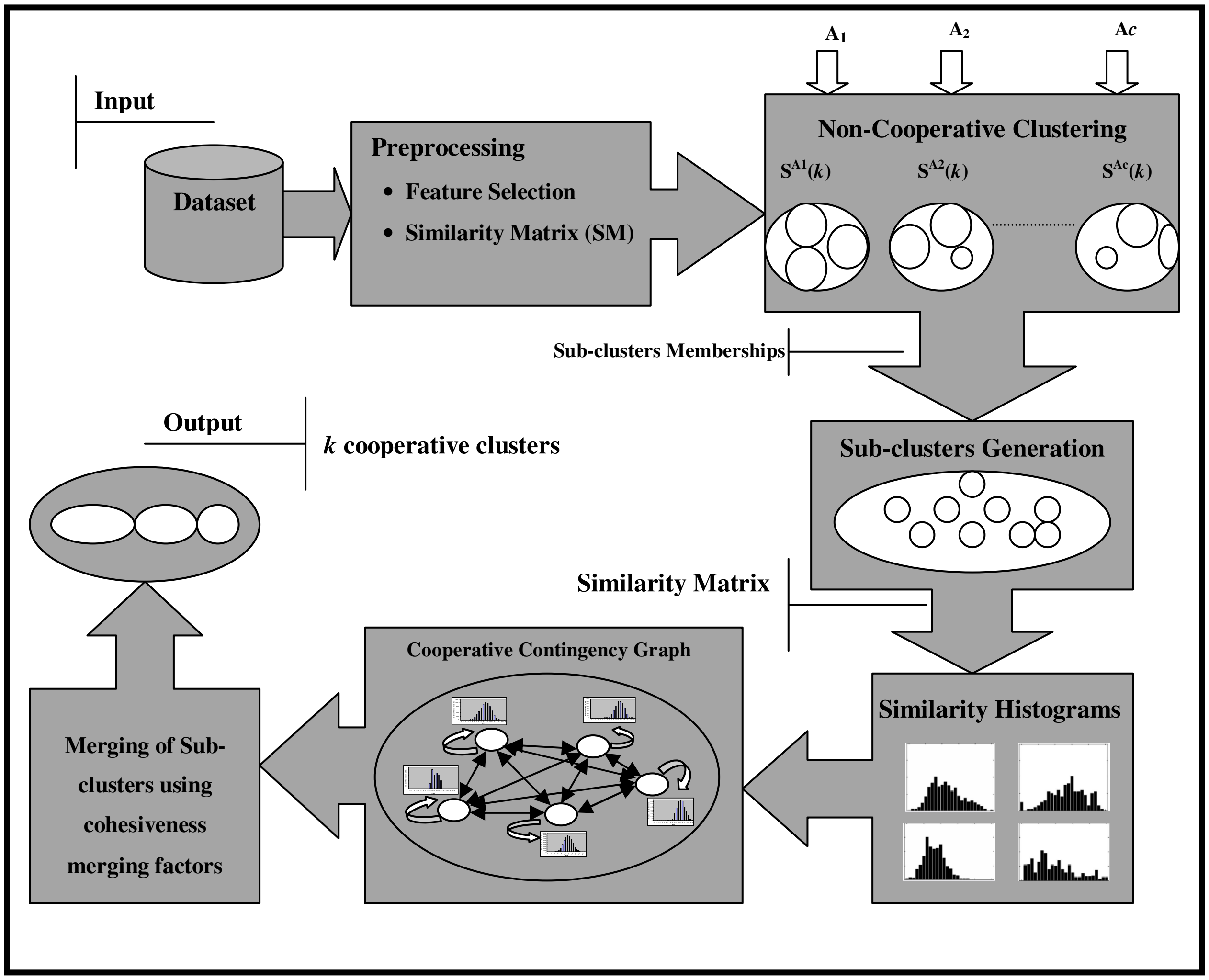


**Fig. 3. 1.** Cooperative Clustering Model

We shall use all notations and definitions discussed in chapter 2 in our model and also introduce some new concepts that add an advantage to the proposed model. Thus, at this point, some terminologies and notations are best presented to pave the way for discussion of the different concepts of the cooperative clustering model. Table 3. 1 summarizes the notations and symbols that are used throughout this chapter.

**Table 3. 1:** Cooperative Clustering Symbols and Notations

| Symbol | Definition |
|---|---|
| $A_i$ | A clustering algorithm |
| $\zeta_i$ | Parameters setting for the clustering algorithm $A_i$ |
| $c$ | Number of clustering algorithms |
| $\delta$ | Similarity threshold |
| *NumBins* | Number of bins in a similarity histogram |
| *BinSize* | The size of the histogram's bins |
| *Sb* | Set of sub-clusters |
| $Sb_i$ | The $i^{\text{th}}$ sub-cluster |
| $n_{sb}$ | Number of sub-clusters |
| $H_i$ | Histogram of the sub-cluster $Sb_i$ |
| $n_{sim}(Sb_i)$ | Number of pair-wise similarities in a sub-cluster $Sb_i$ |

## 3.2 Inputs

The cooperative clustering model takes a set of resources as inputs; the input set includes the dataset of $d$-dimensional vectors represented by a $n \times d$ matrix $X = \{\boldsymbol{x}_i\}, i=1,..n$, where $n$ is the number of objects and the row vector $\boldsymbol{x}_i$ represents the $i^{th}$ object, a pool of $c$ clustering algorithms $(A_1, A_2, .., A_c)$, and set of parameters $\zeta_i$ that are associated with each clustering technique $A_i$.

## 3.3 Preprocessing Stage

The dataset X is first passed through a number of preprocessing steps before being entered into the cooperative clustering model. These steps include:

1) **Feature Selection**: This step is applied in order to (1) assure the selection of the most important features, (2) eliminate any redundant features, and (3) reduce the dimensionality of the selected dataset. In the experimental results we used a simple selection technique as will be illustrated in chapter 4.

2) **Proximity Calculations**: In this step, the pair-wise similarities between objects are stored in a two dimensional $n$ x $n$ similarity (distance) matrix, *SM*. The similarity matrix is a symmetric

matrix, so we store only ($n$ x ($n$-1)/2) elements. The cosine similarity (Eq. (2.5)) is adopted to calculate the similarity between objects, such that $Sim(\boldsymbol{x}, \boldsymbol{y}) \in [-1,1]$, $\boldsymbol{x}$ and $\boldsymbol{y} \in$ X.

## 3.4 The Cooperative Clustering Model

The cooperative clustering model relies on four main components: co-occurred sub-clusters, similarity histograms, a cooperative contingency graph and a coherent merging of the sub-clusters histograms. Each of those components is discussed in the following sub-sections.

### 3.4.1 Generation of Sub-clusters

In general, let A= {$A_1$, $A_2$,.., $A_c$} be a set of $c$ clustering techniques in the model. Assume $S^{A_i}(k)$={$S_j^{A_i}$, $0 \leq j \leq k-1$} is a set of $k$ clusters generated by a clustering technique $A_i$. We assume that number of clusters $k$ is the same for each clustering algorithm. For each object $\boldsymbol{x} \in$ X, a cluster membership value, $mem(\boldsymbol{x})|_{A_i}$ is assigned by each clustering algorithm $A_i$ such that $mem(\boldsymbol{x})|_{A_i} \in \{0,1,..,k\text{-}1\}$.

In order to find the co-occurrence of objects between the multiple $c$ clusterings, a new set of disjoint sub-clusters $Sb$ is generated. The maximum number of disjoint sub-clusters $n_{sb}$ is $k^c$. If number of clusters is different from one partitioning to another, for example $A_1$ generates $k_1$ clusters, $A_2$ generates $k_2$ clusters,…, and $A_c$ generates $k_c$ clusters, then the upper bound of number of sub-clusters is $k_1*k_2*..*k_c$. In order to find the association of objects in the corresponding set of sub-clusters, a new sub-cluster membership is assigned to each object. This clusterings-mapping recognizes the set of disjoint sub-clusters $Sb = \{Sb_i\}_{i=0}^{n_{sb}-1}$, generated by the intersection of the $c$ clusterings. Thus, the underlying model indicates the agreement between the various clustering techniques on clustering the data into a set of clusters. The new cooperative sub-cluster membership is defined as:

$$mem(\boldsymbol{x})|_{A_1, A_2,..,A_c} = mem\ (\boldsymbol{x})|_{A_1} + mem(\boldsymbol{x})|_{A_2} *k + mem(\boldsymbol{x})|_{A_3} *k^2 + \ldots + mem(\boldsymbol{x})|_{A_c} *k^{c-1} \quad (3.\ 1)$$

***Definition***

For any two objects $\boldsymbol{x}$ and $\boldsymbol{y} \in$ X, if $mem(\boldsymbol{x}) = mem(\boldsymbol{y})$, then $\boldsymbol{x}$ and $\boldsymbol{y}$ belong to the same cluster (or sub-cluster)

For simplicity, assume that only two clustering techniques $A_1$ and $A_2$ are available in the model. Let $S^{A_1}(k)$={$S_j^{A_1}$, $0 \leq j \leq k-1$} and $S^{A_2}(k)$ ={$S_j^{A_2}$, $0 \leq j \leq k$-1} be the set of $k$ clusters, generated by $A_1$ and

$A_2$, respectively. Each sub-cluster $Sb(S_i, S_j)$ contains the set of objects from cluster $S_i$ ($S_i \in S^{A_1}(k)$) that are co-occurred into cluster $S_j$ ($S_j \in S^{A_2}(k)$), $i$=0,1,..,$k$-1 and $j$=0,1,..,$k$-1. This set of disjoint sub-clusters, $Sb$ is generated by the $A_1$'s clusterings that are co-occurred in the $A_2$'s clusterings (and vice versa).

In the formulation of sub-cluster memberships, we assume that the number of generated clusters is the same for each clustering algorithm. This assumption is made based on the assigned membership values to each object which involves the parameter $k$ as a factor of evaluating the cooperative memberships and also for further benefits in the cooperative model. A future work involves applying the same cooperative methodology if number of the generated clusters is different from one partitioning to another and employing a new membership function to find the intersection between the $c$ clustering solutions.

### 3.4.2 Similarity-Histogram (SH)

Each sub-cluster is represented as concise statistical representation called *Similarity Histogram* [16].

***Definition***

Similarity Histogram $H$ is a concise statistical representation of the set of pair-wise similarities distribution in a collection of objects. Number of bins in the histogram corresponds to fixed similarity value intervals. Each bin contains the count of pair-wise similarities in the corresponding interval.

Regardless of which similarity function is chosen, the similarity histogram concept remains neutral to our choice. The only requirement is that the similarity measure constitutes a metric on the vector space. *Euclidian* distance (Eq.(2.3)), *cosine* similarity (Eq.(2.5)), and *Jaccard* coefficient (Eq.(2.6)) are the commonly used similarity measures. In this thesis, we calculate the similarity between any pair of objects using the widely used *cosine* coefficient. Thus the similarity histogram in our model is built over the interval [-1, 1] with fixed size of bins, *BinSize*. The number of bins in the histogram is *NumBins* (a user input parameter), thus *BinSize* equals 2/*NumBins*.

A coherent cluster should have high pair-wise similarities. A *typical* cluster has a histogram where the distribution of similarities is almost a normal distribution, while an *ideal* cluster would have a histogram where all similarities are of maximum values, and a *loose* similarity histogram is a histogram where similarities in the cluster are all of minimum values. A typical histogram of a sub-cluster with *NumBins*=20 is illustrated in Fig. 3. 2. For a fixed bin size, *BinSize*, the $binId^{th}$ bin in the histogram contains the count of similarities that fall in the interval $\left](binId-(NumBins/2))*BinSize,\ (binId-(NumBins/2))*BinSize+BinSize\right]$. The first bin (i.e. bin with index =0) also contains similarities equal to -1. In general, the *Build-Histogram* utility is shown in Fig. 3. 3.

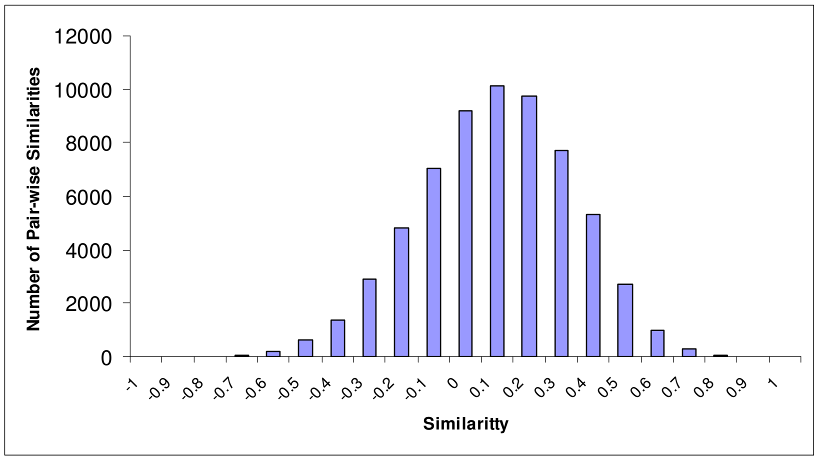


**Fig. 3. 2.** Similarity Histogram of a Sub-cluster (*NumBins*=20)

```
Algorithm: Build-Histogram ($Sb_i$, NumBins, SM)

Input: Sub-cluster $Sb_i$, number of bins NumBins, and the pair-wise similarity matrix SM.
Output: Similarity histogram $H_i$ of size NumBins
Initializations: Let $H_i(bin)=0,\ \forall\ bin=0,..,NumBins-1$
Begin
As similarities range from -1 to 1 then $BinSize=2/NumBins$
        For each pair of objects x and y in $Sb_i$
            $Sim(\mathbf{x},\mathbf{y}) = SM(\mathbf{x},\mathbf{y})$
            If ($Sim(\mathbf{x},\mathbf{y})=-1$) then $binId=0$
            Else If( $Sim(\mathbf{x},\mathbf{y})=1$) then $binId=NumBins-1$
                Else $binId=-1+\lceil Sim(\mathbf{x},\mathbf{y})/BinSize \rceil+(NumBins/2)$
            Increment $H_i(binId)$ by one
            End
Return $H_i$
End
```

**Fig. 3. 3.** Build-Histogram

### 3.4.3 Cooperative Contingency Graph (*CCG*)

The cooperative clustering model is primarily based on the construction of the Cooperative Contingency Graph (*CCG*).

***Definition***

The *CCG* is an undirected graph $G=\{Sb,E\}$ where the co-occurred sub-clusters represent the vertices *Sb* of the graph. The relationships among sub-clusters are represented by the set *E*; *E* is the set of edges in the graph such that each edge $e_{ij}$ represents the relationship between any pair of nodes $Sb_i$ and $Sb_j$ in the graph.

Based on the fact that a coherent cluster should have a similarity histogram where most similarities fall close to the maximum range of the similarity interval; while a loose cluster will have most similarities lie on the minimum range of the similarity interval, each edge in the graph is assigned a weight, we will refer to this weight as a merging factor. This factor represents the coherency (quality) of merging two sub-clusters into a new coherent cluster. We will call this factor the *merging cohesiveness factor (mcf),* defined as the ratio of the similarities above a certain similarity threshold $\delta$ to the total count of similarities in the sub-cluster. The quality of merging two sub-clusters is calculated by the coherency of merging the corresponding histograms. We assume that number of bins is the same in each sub-cluster's histogram. The process of merging two sub-clusters obtains a new histogram; this histogram is constructed by adding the corresponding counts of each bin from the two merged histograms, and also by adding the additional pair-wise similarities that are obtained during merging of the two sub-clusters that were not calculated in each individual histogram. The new histogram is constructed as in Eq.(3.2).

$$H_{ij} = \left( \begin{array}{l} \left\{ \sum_{bin=0}^{NumBins-1} H_i(bin) + H_j(bin) + |Sim(\boldsymbol{x},\boldsymbol{y})| \right\};\ \forall \boldsymbol{x} \in Sb_i, \boldsymbol{y} \in Sb_j, \\ Such\ \ that\{((bin-(NumBins/2))*BinSize) < Sim(\boldsymbol{x},\boldsymbol{y}) \le ((bin-(NumBins/2))*BinSize+BinSize)\} \end{array} \right) \quad (3.\ 2)$$

Where $H_{ij}$ is the histogram of the new cluster and $H_i(bin)$ is the count of similarities in the *bin*th bin of the similarity histogram $H_i$,. $|Sim(\boldsymbol{x},\boldsymbol{y})|$ refers to the number of the additional pair-wise similarities due to the merging. Let $|Sb_i|$ and $|Sb_j|$ be the number of objects in sub-clusters $Sb_i$, $Sb_i$, respectively. The number of pair-wise similarities in each sub-cluster $Sb_i$ and $Sb_i$, are $n_{sim}(Sb_i) = |Sb_i|*(|Sb_i|-1)/2$, and $n_{sim}(Sb_j) = |Sb_j|*(|Sb_j|-1)/2$, respectively. The number of similarities for merging the two sub-clusters together is $n_{sim}(Sb_i,Sb_j) = (|Sb_i|+|Sb_j|)*(|Sb_i|+|Sb_j|-1)/2$.

The *merging cohesiveness factor (mcf)* between any two sub-clusters is computed by calculating the ratio of the count of similarities weighted by the bin similarity above a certain similarity threshold $\delta$ to the total count of similarities in the new merged histogram. The, *mcf* ($Sb_i$, $Sb_j$) is calculated by the following formula:

$$mcf\left(Sb_i, Sb_j\right) = \frac{\left(\sum_{bin=binThreshold}^{numBins-1} ((bin * binSize) - 1 + (binSize/2)) * H_{ij}(bin)\right)}{n_{Sim}(Sb_i, Sb_j)} \qquad (3.\ 3)$$

Where *binThreshold* is the bin corresponding to the similarity threshold $\delta$. The higher the *mcf,* the more coherent the new generated cluster. The *mcf*($Sb_i$, $Sb_j$) corresponds to the weight of the edge $e_{ij}$ between two sub-clusters ($Sb_i$, $Sb_j$) in the cooperative contingency graph (*CCG*). The *CCG* is illustrated in Fig. 3.4 and the pseudo code of constructing the *CCG* graph using the concepts of sub-clusters and histograms is illustrated in Fig. 3.5.

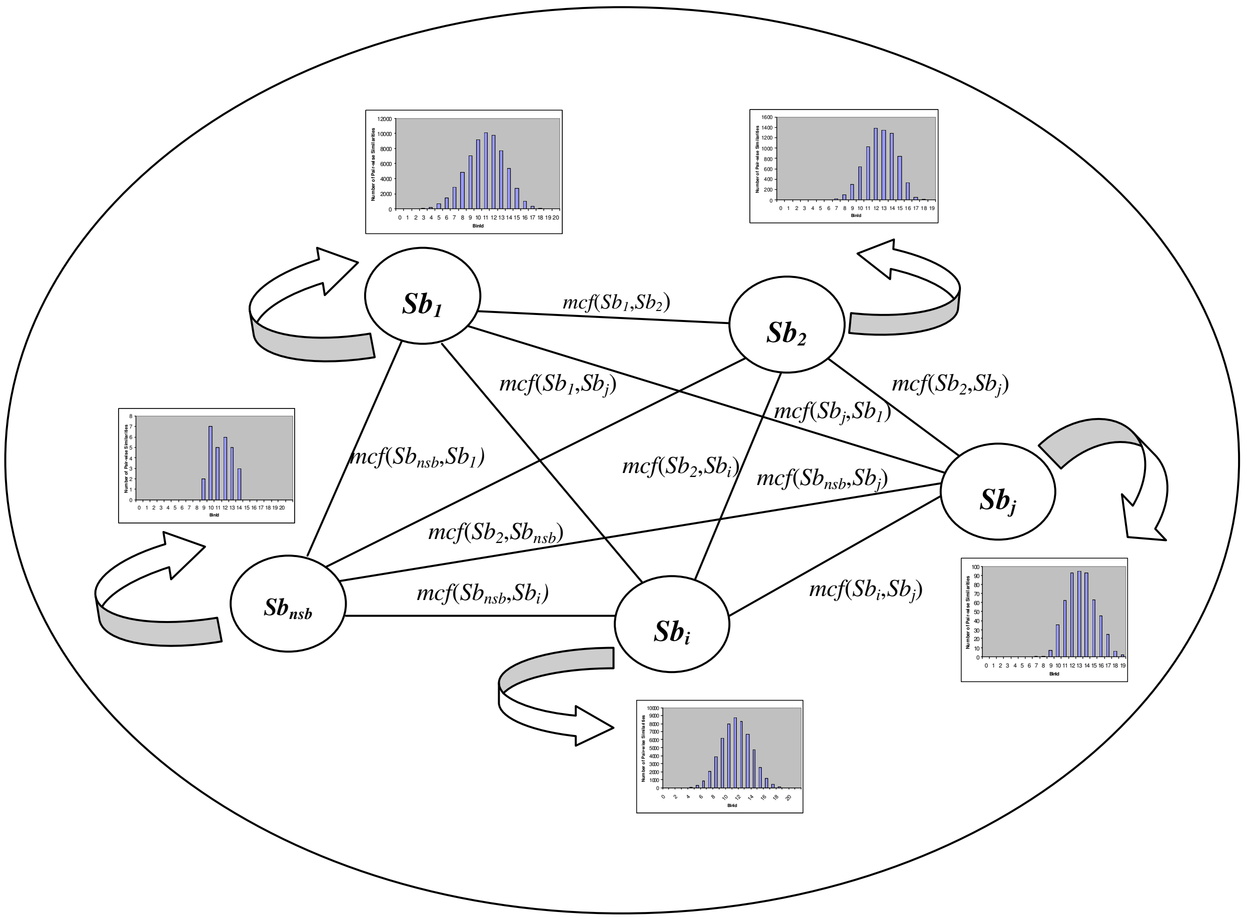


**Fig. 3. 4.** The *CCG* Graph

```
Algorithm: Build-CCG Graph ($S^{Ai}(k)=\{S_j^{Ai}, 0\le j \le k-1\}, SM, \delta, NumBins$)
Input: A set of $c$ clusterings, each clustering solution consists of $k$ clusters $S_0,S_1,..,S_{k-1}$, similarity Matrix
        $SM$, similarity threshold $\delta$, and number of bins in each histogram, $NumBins$.
Output: The Cooperative Contingency Graph ($CCG$)
Initialization: $Sb=\{\}$, $n_{sb}=0$
Begin
        Step1: For all $x \in X$
                $mem(x)|_{A1, A2,.., Ac} = mem(x)|_{A1} + mem(x)|_{A2}*k + mem(x)|_{A3}*k^2 + \dots + mem(x)|_{Ac}*k^{c-1}$
                Assign $index = mem(x)|_{A1, A2,.., Ac}$
                If $Sb_{index}$ is empty, then
                    Create new sub-cluster $Sb_{index}$, insert $x$ to it, add $Sb_{index}$ to $Sb$, and increment $n_{sb}$ by 1
                Else add $x$ directly to the sub-cluster $Sb_{index}$
                End
        PS: The set $Sb$ contains $n_{sb}$ disjoint sub-clusters, $Sb=\{Sb_i, i=0,1,\dots,n_{sb}-1 \le k^c\}$
        Step2: For each sub-cluster $Sb_i \in Sb$
                Build-Histogram ($Sb_i$, $NumBins$, $SM$)
                End
        Step3: Create the cooperative contingency graph $CCG=G(Sb,E)$, where $Sb=\{Sb_i.i=0,..,n_{sb}-1\}$,
                $E=\{e_{ij}(Sb_i,Sb_j)\}$ where each edge $e_{ij}$ is assigned a weight = $mcf(Sb_i,Sb_j)$(Eq. (3.3))
Return CCG
End
```

**Fig. 3. 5.** Building the Cooperative Contingency Graph (*CCG*)

### 3.4.4 Coherent Merging of Sub-Clusters

The cooperative clustering model CC($A_1,A_2,\dots,A_c$) is comprised of two main phases *Phase 1* and *Phase 2*, the first phase includes building the Cooperative Contingency Graph (*CCG*) and associating edges with the corresponding cohesiveness factors. The second phase includes attaining the same number of clusters *k* as the original designed clustering problem "finding *k*-clusters" through merging of sub-clusters within the *CCG*. The first phase has been discussed in details in sub-section 3.4.3 where the *c* clusterings solutions are obtained synchronously and the *CCG* is built. In the second phase, the *CCG* graph has $n_{sb}$ sub-clusters and the goal is to obtain *k* homogenous clusters. The best combined sub-clusters (most similar sub-clusters) are defined as the ones that have maximal *mcf* value. Thus, the two most similar sub-clusters are merged first into a new cluster; i.e. cluster with better homogeneity than the two sub-clusters; and then both the vertices and edges in the *CCG* are updated based on the new added cluster. This step is repeated until the desired number of clusters $k$ is reached. As clustering is unsupervised classification of objects, thus number of clusters is not known, then in the cooperative clustering model both *Phase 1* and *Phase 2* are repeated for different number of clusters $l \geq 2$, then the cooperative model reveals a homogenous clustering solution at number of clusters $l=k$ with the maximum quality value. In the experimental results, we rely on the *SI* index (as internal quality measure) defined in section 2.1.3 to assess finding the proper number of clusters ($k$). The multi-level cooperative clustering model using different number of clusters $l$ is described in Fig. 3. 6.

```
Algorithm: Multi-Level Cooperative Clustering: CC(X, SM, $A_1,A_2,..,A_c$, $\zeta$, $\delta$, NumBins)
Input: Dataset X, similarity matrix SM, set of clustering algorithms, $A_1,A_2,..,A_c$, a set of input parameters
        $\zeta=\{\zeta_i\}$ for each clustering technique $A_i$, similarity threshold $\delta$, and number of bins, NumBins.
Output: Set of Cooperative Clusters, $S^{Cooperative}(k)=\{S_0,S_1,..,S_{k-1}\}$
Initializations: Let $k^{initial}=2$
Begin
    For number of partitions $l = k^{initial}$ to $k^{final}$ (Non-cooperative Clustering Step)
        Phase 1:
            Step1: Synchronously generate the $c$ clusterings sets $S^{A1}(l)$, $S^{A2}(l)$ ,..,and $S^{Ac}(l)$ where $S^{Ai}(l)=$
                    $A_i(X, l, \zeta_i)=\{S_j^{Ai}, 0 \leq j \leq l-1\}$
            Step2: Build_CCG($S^{Ai}(l)$, SM, $\delta$, NumBins) (Cooperation Step)
        Phase 2: Repeat (Merging Step)
                        Step 1: Merge the two most similar sub-clusters into one cluster, i.e. two sub-
                                clusters with the highest mcf in the graph and update the CCG.
                        Step2: Reduce the number of sub-clusters $n_{sb}$ by one
                  Until (number of sub-clusters $n_{sb} = l$)
    $S^{cooperative}(l)$= final set of the merged $l$ sub-clusters
    End
Return the final set of $k$ clusters $S^{cooperative}(k)$ with the maximum quality
End
```

**Fig. 3. 6.** The Multi-Level Cooperative Clustering Model

In Fig. 3. 6, if external information about the dataset is given (i.e. class labels) then the CC model will be performed at the given number of clusters $k$ such that $k^{initial} = k^{final} = k$.

### 3.4.5 Complexity Analysis

All the basic operations are assumed to have the same unit operation time. Assume $T^{A_1}(l)$, $T^{A_2}(l)$,..,$T^{A_c}(l)$ are the computational time complexity of the clustering algorithms A$_1$,A$_2$,..,A$_c$, respectively, at a given number of clusters $l$=2,3,..,$k$. The analysis of the cooperative model can be divided in two stages based on the processing of each individual phase of the cooperative model:

- *Phase 1*: Complexity of constructing the contingency cooperative graph, ($T^{Phase1}$)
    a. Finding the set of sub-clusters takes $n$ operations where $n$ is the total number of objects.
    b. Building a histogram of a sub-cluster $Sb_i$ needs ($|Sb_i|$*($|Sb_i|$-1))/2 operations. Thus $\sum_{i=0}^{n_{sb}-1} \frac{|Sb_i| * | (|Sb_i|-1)}{2}$ operations are required to construct the $n_{sb}$ histograms, where $|Sb_i|$ is the size of the sub-cluster $Sb_i$.
    c. Calculating the *mcf* for each pair of sub-clusters $Sb_i$ and $Sb_j$ in the *CCG* takes (*NumBins-binThreshold)* +$|Sb_i|$*$|Sb_j|$ operations. Where *binThreshold* is the bin corresponds to similarity threshold $\delta$.
    d. Thus *Phase 1* is of order O($n$+$|Sb_i|^2$), $\forall i = 0,1,..,n_{sb}-1$ <<< O($n^2$).

The number of sub-clusters $n_{sb} \leq l^c$, and the size of sub-clusters determine the cost of generating the *CCG.*

- *Phase 2:* Complexity of merging histograms, ($T^{Phase2}$)
    e. Finding the two most homogenous sub-clusters to be merged generate a new cluster $S_i$ is of order O ($n_{sb}^2$) operations, $n_{sb} \leq l^c$.
    f. Updating the *CCG* with the new added cluster takes ($NumBins * \sum_{j=0}^{n_{sb}-3} |S_i| * |Sb_j|$) operations.
    g. Thus *Phase 2* is of order O($n_{sb}^2$+$|S_i|$*$|Sb_j|$) $\forall i, j = 0,..,n_{sb}-1$ <<< O($n^2$).

The time complexity of the Cooperative Clustering (*CC*) model for $l$ partitions is computed as:

$$T^{CC}(l)=max(T^{Ai}(l)))+ T^{Phase1} + T^{Phase2} \qquad (3.\ 4)$$

The time complexity of the *CC* model is based on the clustering algorithm with the maximum running time, and the additional computational costs of both phases that is mainly based on the number of sub-clusters and the size of each sub-cluster which is much lower than $n^2$.

## 3.5 Overall Weighted Similarity Ratio (*OWSR*)

We developed a new measure called the Overall Weighted Similarity Ratio (*OWSR*) that monitors the quality of the set of sub-clusters, such that for any sub-cluster $Sb_i$, we define the *SimRatio* measure as:

$$SimRatio\left(Sb_i\right)=\left\{\begin{array}{ll}\dfrac{\sum\limits_{bin=BinThreshold}^{NumBins-1} H_i(bin)*((bin*binSize)-1+(binSize/2))}{|Sb_i|*(|Sb_i|-1)/2} & if\ |Sb_i|>1\\ 0 \quad if\ |Sb_i|=1 & \end{array}\right\} \tag{3. 5}$$

Where $H_i$ is the histogram representation of the sub-cluster $Sb_i$, $|Sb_i|$ is the number of objects in the sub-cluster $Sb_i$, and *binThreshold* is the bin corresponding to the similarity threshold $\delta$. The value of the *SimRatio*($Sb_i$) increases if objects within the sub-cluster $Sb_i$ are of maximum similarity above the similarity threshold $\delta$. Sub-clusters of only one object have the lowest similarity ratio. The *Overall Weighted Similarity Ratio (OWSR)* for a set of $n_{sb}$ sub-clusters is calculated as the average of the similarity ratio of each sub-cluster weighted by the size of each sub-cluster.

$$Overall\,Weighted\ Similarity\ Ratio(n_{sb})=OWSR(n_{sb})=\frac{\sum_{i=0}^{n_{sb}-1} SimRatio(Sb_i)*|Sb_i|}{n} \tag{3. 6}$$

Where $n$ is the total number of objects. This measure is used to compare two partitioning having different number of sub-clusters.

## 3.6 Scatter F-measure

The traditional *F-measure* measures the difference between the original labeling of the dataset (i.e. class labels) and the resulting clustering of the data. The proposed *scatter F-measure* measures the diversity of the clustering solutions obtained from two clustering algorithms. Given two clustering algorithms $A_1$ and $A_2$, each algorithm generates a clustering set of $k$ clusters $S^{A1}(k)=\{S_i^{A1}, 0\leq i \leq k-1\}$, and $S^{A2}(k)=\{S_j^{A2}, 0\leq j \leq k-1\}$, respectively. Assume $|S_i^{A1}|$ is the number of objects in cluster $S_i^{A1}$ ($S_i^{A1} \in S^{A1}(k)$), and $|S_j^{A2}|$ is the number of objects in cluster $S_j^{A2}$ ($S_j^{A2} \in S^{A2}(k)$). The *F-score* of a cluster $S_i^{A1}$ is defined as:

$$F\text{-}score\,(S_i^{A_1})=\max_j \frac{2*n_{ij}}{|S_i^{A_1}|+|S_j^{A_2}|} \tag{3. 7}$$

Where $n_{ij}$ is the number of objects of cluster $S_i^{A1}$ that co-occurred in the cluster $S_j^{A2}$. With respect to cluster $S_j^{A1}$ we consider the cluster with the highest value of *F-score* to be the cluster $S_j^{A2}$ that is mapped to cluster $S_i^{A1}$, and that value becomes the score for cluster $S_i^{A1}$. The overall s*catter F-measure* for the clustering result of *k* clusters is the weighted average of the *F-score* for each cluster $S_i^{A1}$:

$$Scatter\ F\text{-}measure = \frac{\sum_{i=0}^{k-1} (|S_i^{A_1}| \times F(S_i^{A_1}))}{\sum_{i=0}^{k-1} |S_i^{A_1}|} \tag{3. 8}$$

The higher the overall *Scatter F-measure*, the close solution both $A_1$ and $A_2$ generate due to the higher accuracy of the resulting clusters of $A_2$ mapping to the clusters generated by $A_1$. In the cooperative clustering model, we seek lower values of the *scatter F-measure* between the adopted clustering approaches in order to obtain significant improvement in the clustering performance. As if the clustering solutions of the invoked algorithms are almost the same, then the generated set of sub-clusters will be the same as the original non-cooperative clusters of the adopted techniques. Thus, in turn no additional information is obtained within the set of sub-clusters. We rely on the *Overall Weighted Similarity Ratio* (*OWSR*) as an internal quality measure to evaluate the homogeneity of sub-clusters at different values of the *Scatter F-measure*.

## 3.7 Scalability of the Cooperative Model

Let B be the clustering technique that will be added to the cooperative model (that already contains *c* clustering algorithms). If the set of sub-clusters *Sb* remains the same, i.e. $\exists\ A_i \in \{A_1, A_2,.., A_c\}$ such that the *Scatter F-measure* between B and $A_i$ is of maximum value, then the resulting *c*+1 cooperation is almost the same as the *c* cooperation. However, if adding B to the model will generate a new set of sub-clusters with better homogeneity than the old sub-clusters then the new set of sub-clusters acts as incremental agreement between the *c* clustering techniques and the additional approach B. Thus adding the new technique B to the system was beneficial and it moved the clustering process into a more homogenous clustering process. The homogeneity is evaluated using the *OWSR* measure. In general, increasing number of algorithms in the model will in turn increase the number of sub-clusters $n_{sb}$; the upper bound of number of sub-clusters is $k^c$, where *k* is number of clusters and *c* is number of clustering techniques. Thus if *c* is large enough with different clustering solutions then the number of the generated sub-clusters $n_{sb} \rightarrow n$, which extremely increases the computational complexity of the cooperative model. In this case, each sub-cluster will be a singleton sub-cluster with a maximum of one object and with a similarity ratio of value equals zero (Eq. (3.5)) then the quality of the sub-clusters will be of a minimum value. Thus after a specific value of *c*, $c^*$, the cooperative clustering quality degrades rapidly. Then after

$c^*$, no more techniques can be added to the model. The value of $c^*$ was determined experimentally as will be illustrated in chapter 4. Future work involves evaluating the value of $c^*$ theoretically.

## 3.8 Intermediate and End-results Cooperation

The cooperative model is applied at the end-result level; that means cooperation is established after the clustering process is performed for each clustering algorithm and the final $k$ clusters are obtained. An additional advantage of the cooperative model is that it could be applied at the intermediate steps, e.g. intermediate *iterations* for iterative clustering approaches as $k$-means [7] or intermediate *levels* of the hierarchical tree (or *dendrogram*) for hierarchical techniques as bisecting $k$-means [13]. This sort of cooperation can be used to enhance either the time (e.g. fast convergence) (and/or) quality performance (i.e. clustering quality) as will be illustrated in the experimental results.

### 3.8.1 Example of Cooperation at Intermediate Levels of Hierarchical Clustering

The hierarchical bisecting $k$-means (BKM) [13] is better than the standard $k$-means (KM) [7] and as good as or better than the hierarchical approaches. However, in some scenarios when a fraction of the dataset is left behind with no other way to re-cluster it again at each level of the binary tree, a "refinement" is needed to re-cluster the resulting solutions. Current approaches for enhancing the performance of the BKM use end-result cooperation [13], such that the final set of centroids obtained from the BKM are used as initial seeds for further refinement using $k$-means clustering. These approaches involve wasting time due to the idle status that KM is performing until BKM finishes it clustering. Enhancing the performance of the BKM synchronously can be achieved by intermediate cooperation at each level $i$ of the binary tree with another clustering technique, e.g. $k$-means such that it provides the BKM with better clusterings obtained from the cooperative model. These results replace the current solutions of the BKM and consequently better selection and splitting criteria will be achieved at the next level $i$+1. These enhanced solutions guide the BKM towards better clustering along the tree. Undertaken experimental results in chapter 4 show that the BKM algorithm with intermediate cooperation outperforms the traditional bisecting $k$-means.

### 3.8.2 Example of Cooperation at Intermediate Iterations of Partitional Clustering

Another example of cooperation at the intermediate steps is illustrated through cooperation between two well known iterative clustering approaches, $k$-means and fuzzy $c$-means (FCM) *at each iteration.* In fuzzy clustering, each data point gets multiple and non-dichotomous cluster memberships. FCM can be seen as an improvement and generalization of KM and produces better results than KM when the clusters are overlapped. However, it suffers from the high computational load and similarity to KM. Both KM and FCM attempt to reduce the objective function in every step, and may terminate at a solution that is locally optimal. A bad choice of initial centroids can have a great impact on both the performance and quality of the clustering. Also a good choice of initial centroids reduces the number of iterations that are required for the algorithm to converge. So when KM or FCM are fed by good cluster centroids through cooperation, they will result in a better clustering quality and thus faster convergence to the desired solutions will be achieved. The intermediate cooperation between KM and BKM aims at reducing the total computational time and achieving faster convergence to solutions. Fig. 3.7 outlines the cooperation at the intermediate iterations between KM and FCM through replacing their centroids with the newly received set of cooperative centroids at each iteration. Undertaken experimental results in chapter 4 show that intermediate cooperation provides faster convergence.

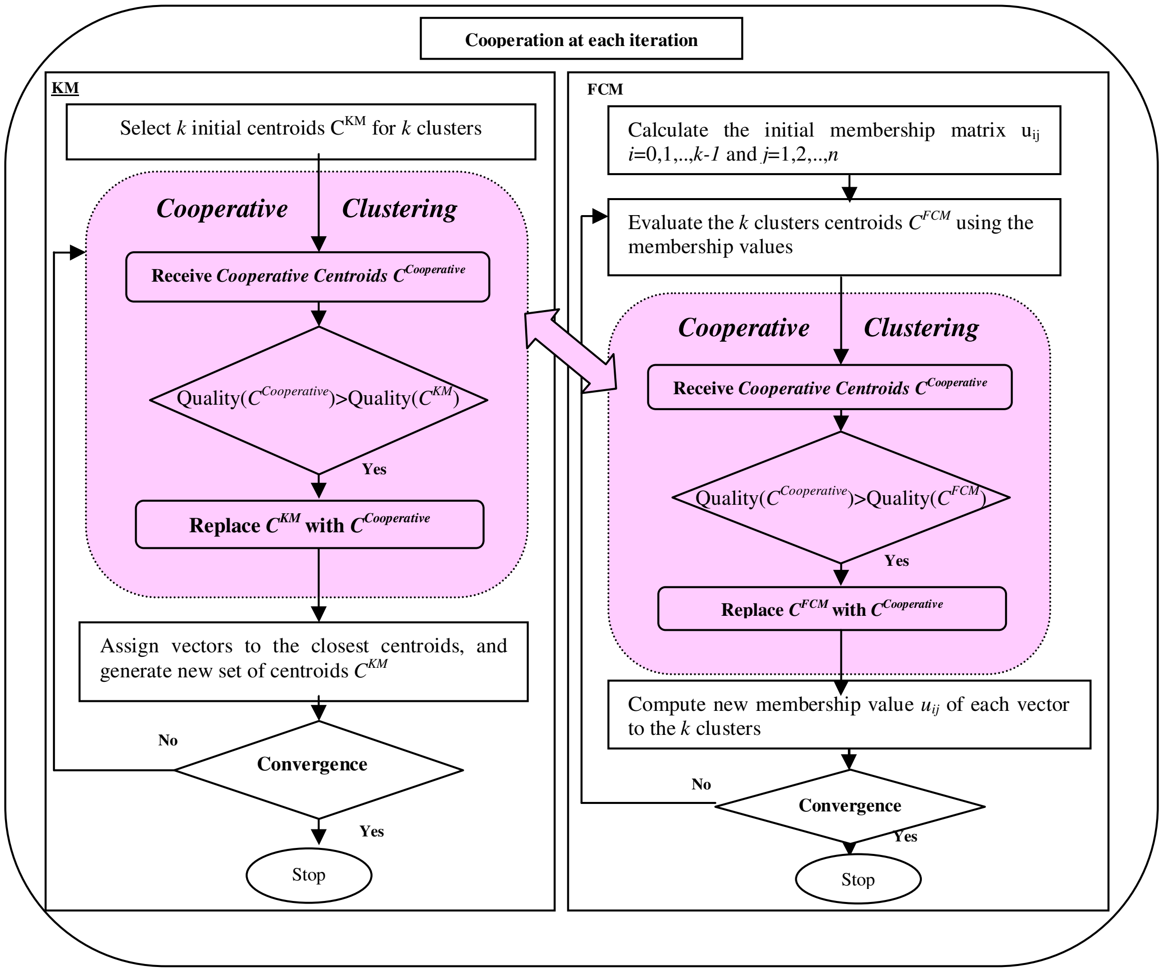

Cooperation at each iteration
KM
Select $k$ initial centroids $C^{KM}$ for $k$ clusters
Cooperative Clustering
Receive Cooperative Centroids $C^{Cooperative}$
Quality($C^{Cooperative}$)>Quality($C^{KM}$)
Yes
Replace $C^{KM}$ with $C^{Cooperative}$
Assign vectors to the closest centroids, and generate new set of centroids $C^{KM}$
No
Convergence
Yes
Stop
FCM
Calculate the initial membership matrix $u_{ij}$ $i$=0,1,..,$k$-1 and $j$=1,2,..,$n$
Evaluate the $k$ clusters centroids $C^{FCM}$ using the membership values
Cooperative Clustering
Receive Cooperative Centroids $C^{Cooperative}$
Quality($C^{Cooperative}$)>Quality($C^{FCM}$)
Yes
Replace $C^{FCM}$ with $C^{Cooperative}$
Compute new membership value $u_{ij}$ of each vector to the $k$ clusters
No
Convergence
Yes
Stop

**Fig. 3. 7**. Cooperation at Intermediate Iterations between KM and FCM

## 3.9 Discussions

In this chapter, a new cooperative clustering (CC) model was presented targeting better clustering solutions than the traditional non-cooperative techniques. The CC model is primary based on finding the intersection between the multiple clusterings in terms of a set of sub-clusters. Each sub-cluster is represented by a similarity histogram. By carefully monitoring the pair-wise similarities between objects in the sub-clusters, the CC model applies a homogeneous merging procedure on the cooperative contingency graph to attain the same number of clusters. The complexity analysis of the cooperative model was presented and analyzed. Also the notion of the *Scatter F-measure* was presented to show the scattering in clustering solutions between two clustering approaches. Also, a new internal quality measure named, *Overall Weighted Similarity Ratio* was formally defined and proposed to assess the quality of the generated sub-clusters. Finally, the advantage of the cooperative model for enhancing the time (and/or) quality performances at the intermediate steps was presented.

# Chapter 4
# Cooperative Clustering Experimental Analysis

In this chapter, evidence in support of the three major contributions of the cooperative clustering model is presented, namely: attaining better clustering quality, achieving scalability in terms of the number of clustering techniques and enhancing the performance of individual algorithms using intermediate cooperation. Evaluation of the algorithms and the cooperative models presented in this chapter is done through conducting a set of experiments using various gene expression and document data sets. The main measures of evaluation are the external and internal quality of the output clusters generated by the non-cooperative algorithms and the cooperative models. The evaluation measures discussed in section 2.1.3 are used for this purpose, with an emphasis on using *F-measure*, *Entropy,* and *Purity* as external quality measures and *SI-Index* and *Overall Weighted Similarity Ratio (OWSR)* as internal quality measures. This chapter is organized as follows. Section 4.1 describes the different clustering techniques that are adopted in our experiments. Different datasets that are used are presented in section 4.2. Section 4.3 discusses the statistical *t*-test. External and internal quality measures are discussed in section 4.4, the performance of the non-cooperative algorithms and the cooperative models is illustrated in section 4.5. The scalability of the cooperative model is presented in section 4.6. Some experiments for showing the performance of the cooperative model with variable number of clusters are illustrated in section 4.7. The intermediate cooperation results are shown in section 4.8, and finally discussions and conclusions are illustrated in section 4.9.

## 4.1 Adopted Clustering Algorithms

Three well known clustering algorithms, KM, BKM, and PAM are invoked in the cooperative model. The following table describes the parameter setting of each of the clustering techniques.

**Table 4. 1:** Parameters Settings of the Adopted Clustering Techniques

| Algorithm | Parameters Setting |
|---|---|
| **KM** | Convergence Threshold=0.001 |
| **BKM** | ITER=3 |
| **PAM** | *npass*=10 |

## 4.2 Data Sets

Experiments were performed on a number of gene expression and documents datasets with various characteristics, dimensions, and sizes. There are four gene expression datasets and three document datasets. The gene expression datasets are *Leukemia*, *Yeast*, *Breast Cancer*, and *Serum* and the three document datasets are *UW, SN,* and *Yahoo*.

### 4.2.1 Gene Expression Datasets

Four gene expression datasets were used to test the performance of the cooperative models as well as the KM, BKM, and PAM algorithms. The data sets are: *Leukemia* dataset [88], *Yeast* gene expression dataset [89], *Breast Cancer* data set [90], and *Serum* dataset [91]. The classification model (i.e. class labels) for both the *Leukemia* and the *Breast Cancer* datasets is discovered using the same approach as in [92]. The characteristics of the gene expression datasets are depicted in Table 4. 2. The SVD (Singular Value Decomposition) representation of each dataset is illustrated in Fig. 4. 1.

**Table 4. 2:** Summary of the Gene Expression Datasets

| **Dataset** | ***n*** | ***k*** | ***d*** |
|---|---|---|---|
| *Leukemia(Leuk)* | 999 | 3 | 38 |
| *Yeast* | 703 | 5 | 73 |
| *Breast Cancer* | 7129 | 4 | 49 |
| *Serum* | 517 | *No External Classification* | 12 |

***<u>Leukemia(Leuk)</u>***

*Leukemia (leuk)* dataset is an example of a non-temporal gene expression set. The dataset contains the expression of 999 genes along 38 samples obtained from ALL (Acute Lympboblastic Leukemia) (27 samples) and AML (Acute Myeloblastic Leukemia) (11 samples). Furthermore, the ALL samples are arranged in 18 B lineage and 9 T lineage samples. The order of samples along the data set columns is: ALL-B lineage, ALL-T lineage and AML.

***<u>Yeast</u>***

The *yeast* cell cycle time series dataset contains the expression levels of 6,218 gene transcripts (identified as ORFs) measured at 10-minutes intervals over two cell cycles (160 minutes). The same filtering and data normalization procedures of [93] are applied resulting in a set of 703 genes. Based on the analysis conducted by Spellman et al [94], the five main clusters are G1-peaking genes, S-peaking genes, G2-peaking genes, M -peaking genes, and M/G1-peaking genes.

***<u>Breast Cancer (BC)</u>***

The *Breast Cancer* dataset was primary used in the work done by [90]. The dataset contains 7129 gene expressions and 49 tumors. The tumor samples are labeled according to the examination of Estrogen Receptor (ER) and Lymph Nodes (LN). The tumor samples are given labels ER+/LN+, ER+/LN-, ER-/LN+, and ER-/LN-.

***<u>Serum</u>***

The *Serum* dataset is a time series gene expression dataset contains 12 time point expressions for about 500 genes; it is obtained from [91].

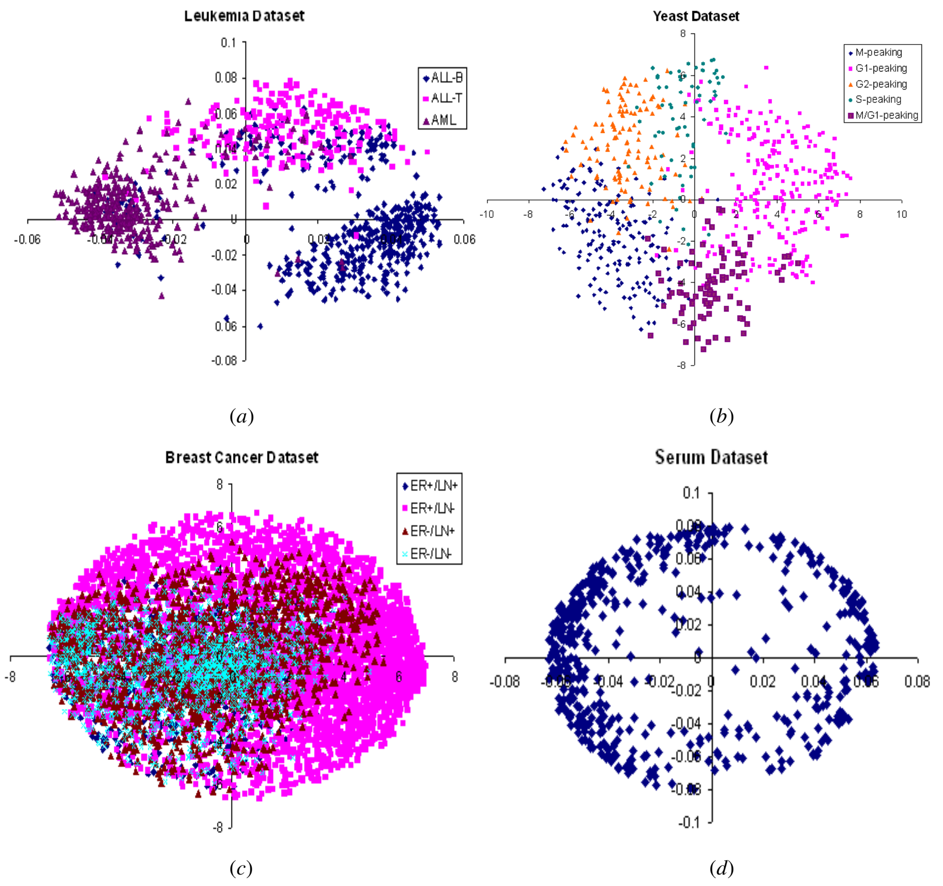


**Fig. 4. 1.** Coefficients of SVD modes for (*a*) *Leukemia* dataset, (*b*) *Yeast* dataset, (*c*) *Breast Cancer* dataset, and (*d*) *Serum* dataset

### 4.2.2 Document Datasets

Experiments were also performed on a number of document datasets with different configurations and sizes. The datasets are: *UW*, *SN*, and *Yahoo*. The *Yahoo* is a standard text mining data set, while *UW* was manually collected and labeled; *SN* was manually collected but was labeled. Below is a brief description of each data set. Table 4. 3 lists the document datasets that are used for the evaluation.

**Table 4. 3:** Documents Datasets

| ***Dataset*** | **Name** | **Type** | **#Documents(*n*)** | **#Terms(*d*)** | **#Categories(*k*)** | **Average terms/document** |
|---|---|---|---|---|---|---|
| *UW* | UW | HTML | 314 | 10 | 15,134 | 469 |
| *SN* | SchoolNet | Metadata | 2,371 | 17 | 7,166 | 145 |
| *Yahoo* | Yahoo News! | HTML | 2,340 | 20 | 28,298 | 289 |

***<u>UW dataset</u>***

The *UW* dataset contains manually collected documents from the University of Waterloo (www. uwaterloo.ca) various web sites, such as the Graduate Studies Office, Information Systems and Technology, Career Services, Cooperative Education, Health Services, and others. The dataset also contains various documents from other Canadian web sites. The total number of documents in this set is 314 documents, categorized into 10 different categories (with some relevancy between the categories.) and the average number of words per document is 469. The *UW* dataset was primarily used for the work presented in [16],[17].

***<u>SN dataset</u>***

The *SN* dataset is a data set of 2371 metadata records collected from the Canada's SchoolNet learning resources web site (http://www.schoolnet.ca/). Specifically, the data was collected from the "Curriculum Area" of the web site. The fields containing text from the metadata records (title, description, and keywords) are extracted and combined to form one document per metadata record. The 17 top-level categories from the *SN* data set are used.

***<u>Yahoo dataset</u>***

The *Yahoo* dataset is a collection of news articles from the Yahoo! News website. The set contains 2340 documents classified into 20 different categories (such as health, entertainment, etc), which have rather unbalanced distributions. There is some relevancy between the categories as well. The average number of words per document is 289 words. The degree of overlapping between classes is quite low in this dataset. The *Yahoo*[11] dataset was used in document clustering-related research conducted by Boley *et al* in [11],[95],[96].

***Text Preprocessing***

For text data, we used the *Vector Space Model* (VSM) which is the most common document representation model used in text mining. In VSM each document is represented by a vector $\boldsymbol{x}$, in the term space, $\boldsymbol{x} = [tf_1, tf_2 \ldots tf_d]$, where $tf_i$, $i = 1, \ldots, d$ is the term frequency in the document, or the number of occurrences of the term $t_i$ in a document $\boldsymbol{x}$. To represent every document with the same set of terms, we have to extract all the terms found in the documents and use them as our feature vector[12]. Sometimes another method is used which combines the term frequency with the inverse document frequency (*TF-IDF*) [97],[98]. The document frequency $df_i$ is the number of documents in a collection of $n$ documents in which the term $t_i$ occurs. A typical inverse document frequency (*idf*) factor of this type is given by $\log(n/df_i)$. The *weight* of a term $t_i$ in a document is given by:

$$weight_i = tf_i \times \log\left(\frac{n}{df_i}\right) \tag{4. 1}$$

The words are tokenized in the *UW*, *SN*, and *Yahoo* datasets in the following way:

- Words consisting of numbers only are removed,
- All words are converted to lower case letters,
- Stop words[13] are removed,
- Words of length less than 3 are removed, and finally
- The remaining words were stemmed using the popular Porter stemmer algorithm [99].

[11] The dataset is available at: http//ftp.cs.umn.edu/dept/users/boley/.

[12] Obviously the dimensionality of the feature vector is always very high, in the range of hundreds and sometimes thousands.

[13] Stop-words are very common words that have no significance for capturing relevant information about a document (such as “the”,”and”,”a”,..,etc).

## 4.3 Significance Testing

To back the claim of clustering quality improvement, statistical significance testing is presented here, where the average values of a variable taken by any two approaches are compared. Assume $q$ (e.g. *F-measure* or *Entropy* or *Purity* or *SI*) is the quality measure used for comparison between any two clustering techniques $A_1$ and $A_2$. Let $q_1$ and $q_2$ be the two samples of the quality measure $q$ for the clustering results of both $A_1$ and $A_2$, respectively. Our Null hypothesis (which we will argue to be rejected in favor of the alternate hypothesis) is that the average values of $q$ for $A_1$ and $A_2$ are the same (i.e. no significance difference).

$$H_0: \ \overline{q}_1 = \overline{q}_2 \ \text{(No significant improvement in } q \text{ using } A_1\text{)} \tag{4. 2}$$

Where $\overline{q}_1$ is the average $q$ value for $A_1$ clustering over $n_1$ samples, and $\overline{q}_2$ is the corresponding average value of $q$ for $A_2$ clustering over $n_2$ samples. The alternative hypothesis is:

$$H_1: \ \overline{q}_1 \neq \overline{q}_2 \ \text{(Better improvement in } q \text{ using } A_1\text{)} \tag{4. 3}$$

For directional difference, for example for *F-measure*, the null hypothesis $H_0$ is $\overline{F}_1 = \overline{F}_2$ and the alternative hypothesis is $\overline{F}_1 > \overline{F}_2$, where $\overline{F}_1$ is the average *F-measure* of the cooperative model clustering over all runs and $\overline{F}_2$ is the corresponding average *F-measure* obtained from the non-cooperative clustering algorithm. Since the actual underlying means and standard deviations are not known, we are going to use a two-sample $t$-statistic, in which the population standard deviations are estimated by the calculated standard deviations $sd_1$ and $sd_2$ from the samples. The $t$-statistic [100],[101] is given by:

$$t = \frac{(\overline{q}_1 - \overline{q}_2)}{\sqrt{\frac{sd_1^2}{n_1} + \frac{sd_2^2}{n_2}}} \tag{4. 4}$$

Where $sd_1$ and $sd_2$ are the calculated standard deviations and $n_1$ and $n_2$ are the sample sizes from the two populations. To compute a (1-$\alpha$) % confidence interval (usually 95%) for the difference between the two means,

1. First, find the $t^{critcial}$ value from the $t$-distribution table[14] at degree of freedom $df$ and confidence interval α. Where $t^{critical}$ is the upper $(1-\alpha)/2$ critical value for the $t$ distribution with degree of freedom $df$ equals $(n_1+n_2-2)$.
2. If the calculated $t$ value < $t^{critcial}$ then the null hypothesis $H_0$ is accepted otherwise the "significance difference "hypothesis $H_1$ is accepted and $H_0$ is rejected.

In our experiments, we obtained 20 runs of each algorithm. Thus at $df$=38 and confidence interval 95%, the critical value of $t$ (cut off), $t^{critcial}$ equals 2.024. Thus, in each experiment we evaluate the $t$-test value of the evaluation measure and compare this value to the assigned critical $t$-value at 95% confidence interval.

## 4.4 Quality Measures

In order to evaluate the quality of the clustering, we adopted four quality measures widely used in the data clustering literature. The first is the *F-measure*, which combines the *Precision* and *Recall* ideas from the information retrieval literature. The second is *Entropy*, which provides a measure of "goodness" for un-nested clusters or for the clusters at one level of a hierarchical clustering. *Entropy* tells us how homogeneous a cluster is. The third is the *Purity* measure, and finally the internal measure, *SI* index (See section 2.1.3 for details about these evaluation measures.) Basically we would like to maximize the *F-measure*, minimize the *Entropy* of clusters, maximize the *Purity* of solutions, and minimize the separation index of the obtained clusters to achieve high quality clustering. By using both external and internal measures we have confidence that our evaluation of both the cooperative and non-cooperative approaches will be justified.

## 4.5 Cooperative Clustering Performance Evaluation

In this section, the performance of the cooperative models is presented. Initially, let $c$=2, thus we assume only two clustering techniques are available in the system. We will refer to the cooperation between KM and BKM by CC(KM,BKM), cooperation between KM and PAM by CC(KM,PAM), and finally cooperation between BKM and PAM by CC(BKM,PAM).

[14] http://www.medcalc.be/manual/t-distribution.php

### 4.5.1 Clustering Quality

Tables 4.4 - 4.9 show the average performance of 20 runs of the non-cooperative KM, BKM, and PAM algorithms as well the three cooperative models. The value of the similarity threshold $\delta \in [0.1, 0.25]$ for the gene expression datasets and $\delta \in [0.2, 0.3]$ for the document datasets. In each table, there are two types of cells, one that describes the non-cooperative algorithm and another one that describes the performance of any model of the cooperative models. Assume a random variable $q$, where $q$ is any measure of the quality measures described above.

For the non-cooperative KM, BKM, and PAM, each cell contains two entities:

- $\overline{q}$ : The average value of the variable $q$ over 20 runs
- $\pm sd$: The standard deviation of the variable $q$ over the calculated 20 runs

For any cooperative model CC($A_1$, $A_2$), where $A_1$ and $A_2 \in$ {KM, BKM, PAM}, each cell contains 5 components that describe the random variable $q$, each cell can be described by the tuple ( $\overline{q}$ , $sd$, $t_1$, $t_2$, $+q\%$ ); the description of the each component is as follow:

- $\overline{q}$ : The average value of the variable $q$ over 20 runs
- $\pm sd$: The standard deviation of the variable $q$ over the 20 runs
- $t_1$: the $t$-test value between the results of the cooperative model CC($A_1$, $A_2$) and the results of $A_1$
- $t_2$: the $t$-test value between the results of the cooperative model CC($A_1$, $A_2$) and the results of $A_2$
- $+q\%$: the percentage in improvement in $q$ using the cooperative model CC($A_1$, $A_2$) compared to the value of $q$ that is calculated by $A_1$ and $A_2$

In each table, $t_1$ and $t_2$ >2.024, thus the Null hypothesis $H_0$ is rejected and that means the obtained clustering results from the cooperative models are better that those obtained from the individual approaches using the $t$-test values, $t_1$, $t_2$, respectively.

**Table 4. 4**: Performance Evaluation of the Cooperative and Non-cooperative Approaches [*Leuk*]

| ($k$=3) | ***F-measure*** | ***Entropy*** | ***Purity*** | ***SI*** |
|---|---|---|---|---|
| **KM** | 0.8366± (0.021) | 0.4086± (0.032) | 0.8328±(0.036) | 0.3921± (0.034) |
| **BKM** | 0.8073± (0.022) | 0.4738± (0.011) | 0.8048±(0.028) | 0.4502± (0.025) |
| **PAM** | 0.8754± (0.033) | 0.3971± (0.026) | 0.8478±(0.020) | 0.3615± (0.024) |
| **CC(KM,BKM)** | **0.9375**± (0.019) | **0.3109**± (0.026) | **0.9096**± (0.019) | **0.2647**± (0.018) |
| | $t_1$=15.93 | $t_1$=10.59 | $t_1$=8.43 | $t_1$=15.16 |
| | $t_2$=20.03 | $t_2$=25.81 | $t_2$=13.85 | $t_2$=26.23 |
| **Improvement (%)** | **(%12)** | **(%24)** | **(%9)** | **(%32)** |
| **CC(KM,PAM)** | **0.9485**± (0.013) | **0.2966**± (0.017) | **0.9381**± (0.011) | **0.2385**± (0.017) |
| | $t_1$=20.26 | $t_1$=13.82 | $t_1$=12.51 | $t_1$=18.50 |
| | $t_2$=9.21 | $t_2$=14.46 | $t_2$=17.69 | $t_2$=18.70 |
| **Improvement (%)** | **(%8)** | **(%25)** | **(%11)** | **(%34)** |
| **CC(BKM,PAM)** | **0.9630**± (0.010) | **0.2673**± (0.028) | **0.9565**± (0.035) | **0.2071**± (0.029) |
| | $t_1$=24.65 | $t_1$=30.69 | $t_1$=15.13 | $t_1$=27.91 |
| | $t_2$=11.36 | $t_2$=15.19 | $t_2$=12.05 | $t_2$=18.34 |
| **Improvement (%)** | **(%10)** | **(%32)** | **(%13)** | **(%42)** |

**Table 4. 5:** Performance Evaluation of the Cooperative and Non-cooperative Approaches [*Yeast*]

| (*k*=5) | ***F-measure*** | ***Entropy*** | ***Purity*** | ***SI*** |
|---|---|---|---|---|
| **KM** | 0.6301± (0.040) | 0.4351± (0.032) | 0.6715± (0.031) | 1.6303± (0.206) |
| **BKM** | 0.6784± (0.011) | 0.4136± (0.024) | 0.7496± (0.020) | 1.2991± (0.185) |
| **PAM** | 0.6922± (0.035) | 0.4032± (0.031) | 0.7667± (0.028) | 1.0433± (0.176) |
| **CC(KM,BKM)** | **0.8175**± (0.041) | **0.2748**± (0.022) | **0.8946**± (0.019) | **0.6162**± (0.092) |
| | $t_1$=14.63 | $t_1$=18.46 | $t_1$=27.44 | $t_1$=20.10 |
| | $t_2$=14.65 | $t_2$=19.07 | $t_2$=23.50 | $t_2$=15.38 |
| **Improvement (%)** | **(+21%)** | **(+34%)** | **(+19%)** | **(+52%)** |
| **CC(KM,PAM)** | **0.8586**± (0.026) | **0.2463**± (0.020) | **0.9326**± (0.017) | **0.5815**± (0.066) |
| | $t_1$=21.42 | $t_1$=22.37 | $t_1$=33.02 | $t_1$=21.68 |
| | $t_2$=17.07 | $t_2$=19.02 | $t_2$=22.64 | $t_2$=10.98 |
| **Improvement (%)** | **(+24%)** | **(+39%)** | **(+22%)** | **(+44%)** |
| **CC(BKM,PAM)** | **0.7812**± (0.012) | **0.3047**± (0.030) | **0. 8792**± (0.011) | **0.7757**± (0.057) |
| | $t_1$=28.24 | $t_1$=12.67 | $t_1$=25.39 | $t_1$=12.09 |
| | $t_2$=10.75 | $t_2$=10.21 | $t_2$=16.72 | $t_2$=6.46 |
| **Improvement (%)** | **(+13%)** | **(+24%)** | **(+15%)** | (+**26**%) |

**Table 4. 6:** Performance Evaluation of the Cooperative and Non-cooperative Approaches [*BC*]

| ($k$=4) | ***F-measure*** | ***Entropy*** | ***Purity*** | ***SI*** |
|---|---|---|---|---|
| **KM** | 0.4271± (0.011) | 0.8031± (0.032) | 0.5734± (0.018) | 0.7578± (0.024) |
| **BKM** | 0.4355± (0.020) | 0.7948± (0.041) | 0.5825± (0.016) | 0.7363± (0.018) |
| **PAM** | 0.5012± (0.023) | 0.7114± (0.022) | 0.6107± (0.017) | 0.6930± (0.033) |
| **CC(KM,BKM)** | **0.4915±** (0.016) $t_1$=24.01 $t_2$=10.02 | **0.7042±** (0.015) $t_1$=11.46 $t_2$= 10.35 | **0.6606±** (0.020) $t_1$=14.48 $t_2$=13.64 | **0.6998±** (0.015) $t_1$=9.16 $t_2$= 6.96 |
| **Improvement (%)** | **(+12%)** | **(+11%)** | **(+13%)** | **(+5%)** |
| **CC(KM,PAM)** | **0.5935±** (0.020) $t_1$=32.60 $t_2$=13.54 | **0.6102±** (0.012) $t_1$=23.35 $t_2$= 20.19 | **0.7294±** (0.013) $t_1$=31.43 $t_2$= 32.55 | **0.5418±** (0.019) $t_1$=31.55 $t_2$= 17.75 |
| **Improvement (%)** | **(+18%)** | **(+14%)** | **(+19%)** | **(+22%)** |
| **CC(BKM,PAM)** | **0.6402±** (0.017) $t_1$=34.87 $t_2$=21.73 | **0.5188±** (0.029) $t_1$=25.31 $t_2$= 23.66 | **0.7637±** (0.012) $t_1$=40.51 $t_2$= 32.87 | **0.4943±** (0.027) $t_1$=33.35 $t_2$= 20.84 |
| **Improvement (%)** | **(+28%)** | **(+27%)** | **(+25%)** | **(+29%)** |

**Table 4. 7:** Performance Evaluation of the Cooperative and Non-cooperative Approaches [*UW*]

| (*k*=10) | ***F-measure*** | ***Entropy*** | ***Purity*** | ***SI*** |
|---|---|---|---|---|
| **KM** | 0.6988± (0.027) | 0.2579± (0.022) | 0.6879± (0.031) | 1.6921± (0.152) |
| **BKM** | 0.7520± (0.031) | 0.2281± (0.024) | 0.7334± (0.024) | 1.3772± (0.140) |
| **PAM** | 0.6463± (0.041) | 0.3592± (0.013) | 0.6490± (0.033) | 3.1932± (0.543) |
| **CC(KM,BKM)** | **0.8387±** (0.018) $t_1$=19.28 $t_2$= 10.81 | **0.1819±** (0.011) $t_1$=13.82 $t_2$= 7.82 | **0.8420±** (0.015) $t_1$=20.01 $t_2$= 17.16 | **1.0559±** (0.019) $t_1$=18.57 $t_2$=10.17 |
| **Improvement (%)** | **(+12%)** | **(+20%)** | **(+15%)** | **(+23%)** |
| **CC(KM,PAM)** | **0.8672±** (0.013) $t_1$=25.13 $t_2$=22.96 | **0.1646±** (0.028) $t_1$=11.71 $t_2$= 28.19 | **0.8734±** (0.029) $t_1$=19.54 $t_2$=22.84 | **0.9678±** (0.112) $t_1$=17.32 $t_2$=17.97 |
| **Improvement (%)** | **(+24%)** | **(+36%)** | **(+27%)** | **(+43%)** |
| **CC(BKM,PAM)** | **0.8746±** (0.024) $t_1$=13.99 $t_2$=21.49 | **0.1513±** (0.035) $t_1$=8.09 $t_2$=24.90 | **0.8819±** (0.017) $t_1$=22.58 $t_2$=28.05 | **0.8707±** (0.185) $t_1$=9.76 $t_2$=18.10 |
| **Improvement (%)** | **(+16%)** | **(+34%)** | **(+20%)** | **(+37%)** |

**Table 4. 8:** Performance Evaluation of the Cooperative and Non-cooperative Approaches [*SN*]

| (*k*=17) | ***F-measure*** | ***Entropy*** | ***Purity*** | ***SI*** |
|---|---|---|---|---|
| **KM** | 0.4927± (0.029) | 0.3787± (0.018) | 0.6449± (0.025) | 1.7441± (0.215) |
| **BKM** | 0.5281± (0.037) | 0.3585± (0.022) | 0.6867± (0.024) | 1.2876± (0.146) |
| **PAM** | 0.3412± (0.034) | 0.5831± (0.072) | 0.4787± (0.042) | 4.2001± (0.833) |
| **CC(KM,BKM)** | **0.5823±** (0.013) $t_1$=12.61 $t_2$= 6.18 | **0.3374±** (0.016) $t_1$=7.67 $t_2$= 3.45 | **0.7661±** (0.023) $t_1$=15.95 $t_2$= 10.45 | **1.0955±** (0.086) $t_1$=12.52 $t_2$= 4.96 |
| **Improvement (%)** | **(+10%)** | **(+6%)** | **(+12%)** | **(+15%)** |
| **CC(KM,PAM)** | **0.6184±** (0.015) $t_1$=17.22 $t_2$= 33.36 | **0.3244±** (0.037) $t_1$=5.90 $t_2$= 14.29 | **0.7903±** (0.043) $t_1$=13.07 $t_2$= 23.21 | **0.8531±** (0.065) $t_1$=17.74 $t_2$= 17.91 |
| **Improvement (%)** | **(+26%)** | **(+14%)** | **(+23%)** | **(+51%)** |
| **CC(BKM,PAM)** | **0.6436±** (0.022) $t_1$=11.99 $t_2$= 33.39 | **0.3153±** (0.031) $t_1$=5.08 $t_2$= 15.28 | **0.8457±** (0.029) $t_1$=18.89 $t_2$= 32.15 | **0.7517±** (0.039) $t_1$=15.85 $t_2$= 18.49 |
| **Improvement (%)** | **(+22%)** | **(+12%)** | **(+23%)** | **(+42%)** |

**Table 4. 9:** Performance Evaluation of the Cooperative and Non-cooperative Approaches [*Yahoo*]

| ($k$=20) | ***F-measure*** | ***Entropy*** | ***Purity*** | ***SI*** |
|---|---|---|---|---|
| **KM** | 0.4585± (0.011) | 0.3815± (0.025) | 0.6192± (0.024) | 2.3246± (0.134) |
| **BKM** | 0.5501± (0.031) | 0.3128± (0.029) | 0.7171± (0.017) | 1.6641± (0.089) |
| **PAM** | 0.4476± (0.016) | 0.4876± (0.034) | 0.5381± (0.053) | 2.5138± (0.206) |
| **CC(KM,BKM)** | **0.6619** ± (0.023) $t_1$=35.67 $t_2$= 12.95 | **0.2397** ± (0.014) $t_1$=22.13 $t_2$= 10.15 | **0.8078** ± (0.011) $t_1$=31.95 $t_2$= 20.03 | **1.1272** ± (0.055) $t_1$=36.97 $t_2$= 22.95 |
| **Improvement (%)** | **(+20%)** | **(+23%)** | **(+13%)** | **(+32%)** |
| **CC(KM,PAM)** | **0.4794** ± (0.010) $t_1$=6.29 $t_2$= 7.54 | **0.3674**± (0.009) $t_1$=2.373 $t_2$=15.28 | **0.6474**± (0.022) $t_1$=3.87 $t_2$= 8.51 | **2.1764**± (0.026) $t_1$=4.86 $t_2$= 7.27 |
| **Improvement (%)** | **(+5%)** | **(+4%)** | **(+5%)** | **(+6%)** |
| **CC(BKM,PAM)** | **0.6162** ± (0.018) $t_1$=8.24 $t_2$= 31.30 | **0.2687** ± (0.028) $t_1$=4.89 $t_2$=22.23 | **0.7788** ± (0.012) $t_1$=13.26 $t_2$= 19.80 | **1.3695** ± (0.033) $t_1$=13.88 $t_2$= 16.76 |
| **Improvement (%)** | **(+12%)** | **(+14%)** | **(+9%)** | **(+18%)** |

The cooperation between KM and BKM, CC(KM,BKM), achieves improvement of up to 21% in *F-measure*, up to %34 in *Entropy*, up to 19% in *Purity*, and up to 52% in *SI index* for the *Yeast* dataset. The cooperative model CC(KM,PAM) achieves improvement up to 26% in *F-measure* (*SN* dataset), up to 39% in *Entropy* (*Yeast* dataset), up to 27% in *Purity* (*UW* dataset), and up to 51% in *SI* index for the *SN* dataset. Finally, CC(BKM,PAM) achieves improvement up to 28% in *F-measure* (*Breast Cancer* dataset), up to 34% in *Entropy* (*UW* dataset), up to 25 % in *Purity* (*Breast Cancer* dataset), and up to 42% in *SI* index for the *Leukemia* dataset. It can be shown that the cooperation between the adopted clustering techniques produces clustering solutions with higher values for both *F-measure* and *Purity* and lower values for *Entropy* and *SI* index than those of the individual algorithms. The main reason for this improvement in the clustering quality is that, each of the cooperative models takes the intersection of the individual clusterings and obtains new clusterings with maximum Intra cluster homogeneity and maximum Inter-cluster separation using both the notion of similarity histograms and cooperative merging.

### 4.5.2 Scatter-F-measure Evaluation

In this sub-section, we use the *Scatter F-measure* (defined in section 3.6) as a measure of diversity between the clustering algorithms. The higher the *Scatter F-measure,* the lower improvement in the clustering quality of the cooperative models. As if there is no scattering between the clustering results of the adopted approaches (i.e. they generate the same solution), it would lead to the same set of sub-clusters as the original set of clusters. Then the final set of clusters after merging will be almost the same as that of the original non-cooperative approaches. On the other hand, when two clustering approaches generate two different clustering solutions (i.e. lower value of the *Scatter F-measure*) then a new set of sub-clusters with better homogeneity than the original clusters is generated. Thus better improvement in the clustering quality is achieved. The *Scatter F-measure*, number of sub-clusters, quality of sub-clusters (measured by the *OWSR* measure (section 3.5)) and the corresponding values of the *SI* index of the obtained *k* clusters are reported in tables 4.10 - 4.14.

**Table 4. 10:** *Scatter F-measure* and Quality of Clusters [*Yeast*]

| ***k=5*** | **CC(KM,BKM)** | **CC(KM,PAM)** | **CC(BKM,PAM)** |
|---|---|---|---|
| ***Scatter F-measure*** | 0.6956 | 0.5309 | 0.7363 |
| **# Sub-clusters** | 15 | 18 | 10 |
| **Quality of Sub-clusters (*OWSR*)↑** | 0.2871 | 0.2914 | 0.2755 |
| ***SI*↓** | 0.6162 | 0.5815 | 0.7757 |

**Table 4. 11:** *Scatter F-measure* and Quality of Clusters [*Breast Cancer*]

| ***k=4*** | **CC(KM,BKM)** | **CC(KM,PAM)** | **CC(BKM,PAM)** |
|---|---|---|---|
| ***Scattering-F-measure*** | 0.8432 | 0.6306 | 0.6175 |
| **# Sub-clusters** | 10 | 14 | 15 |
| **Quality of Sub-clusters (*OWSR*)↑** | 0.7543 | 0.8623 | 0.8955 |
| ***SI*↓** | 0.6998 | 0.5418 | 0.4943 |

**Table 4. 12:** *Scatter F-measure* and Quality of Clusters [*UW*]

| ***k*=10** | **CC(KM,BKM)** | **CC(KM,PAM)** | **CC(BKM,PAM)** |
|---|---|---|---|
| ***Scattering-F-measure*** | 0.6672 | 0.5655 | 0.4618 |
| **# Sub-clusters** | 28 | 32 | 44 |
| **Quality of Sub-clusters (*OWSR*)↑** | 0.2441 | 0.2657 | 0.2932 |
| ***SI*↓** | 1.0559 | 0.9678 | 0.8707 |

**Table 4. 13:** *Scatter F-measure* and Quality of Clusters [*SN*]

| ***k=17*** | **CC(KM,BKM)** | **CC(KM,PAM)** | **CC(BKM,PAM)** |
|---|---|---|---|
| ***Scattering-F-measure*** | 0.6109 | 0.4835 | 0.4312 |
| **# Sub-clusters** | 139 | 208 | 214 |
| **Quality of Sub-clusters (*OWSR*)↑** | 0.6473 | 0.7647 | 0.7887 |
| ***SI*↓** | 1.0955 | 0.8531 | 0.7517 |

**Table 4. 14:** *Scatter F-measure* and Quality of Clusters [*Yahoo*]

| ***k=20*** | **CC(KM,BKM)** | **CC(KM,PAM)** | **CC(BKM,PAM)** |
|---|---|---|---|
| ***Scattering-F-measure*** | 0.4075 | 0.7612 | 0.4563 |
| **# Sub-clusters** | 231 | 131 | 225 |
| **Quality of Sub-clusters (*OWSR*)↑** | 0.7788 | 0.59554 | 0.7601 |
| ***SI*↓** | 1.1272 | 2.1764 | 1.3695 |

In the *Breast Cancer* dataset, both KM and BKM are close to each in terms of their clustering solutions (measured by both internal and external quality measures) where the value of the *scatter F-measure* is 0.8432. Thus the quality of the generated clusters is almost the same as that of both of them and the percentage of improvement is only 12% for *F-measure* and 5% for *SI* Index. On the other hand, the cooperative models CC(KM,PAM) and CC(BKM,PAM) achieve an improvement in the performance of up to 28% for F-measure and 29% for SI. The main reason for this significant improvement is that either CC(BKM,PAM) or CC(KM,PAM) generates solutions that are different and that reveals different set of sub-clusters with better quality. For the *Yahoo* dataset, the least improvement in the clustering quality is provided by the CC(KM,PAM), where the cooperation results in an improvement of only 5% for *F-measure* and 6% for *SI* for large value of the *Scatter F-measure* (0.7612) while CC(KM,BKM) for example achieves improvement up to 20% for *F-measure* and 32% for *SI* at *scatter F-measure* equals 0.4075 for the same dataset. We can conclude that the scattering between the clustering results of the adopted clustering techniques enables the cooperative model to work with more homogenous set of sub-clusters that yield better final cooperative clustering results.

### 4.5.3 Performance Evaluation at *c*=3

In this section we evaluate the performance of the cooperative model by combining the clustering solutions of the three approaches, KM, BKM, and PAM together in one solution (i.e. *c*=3). We will refer to the cooperation between KM, BKM, and PAM by CC(KM,BKM,PAM). The values of both *F-measure* and *SI index* using the cooperative and non-cooperative approaches for *Leukemia*, *Yeast*, *Breast Cancer*, *UW*, *SN,* and *Yahoo* datasets are reported in Figures 4.2 and 4.3, respectively.

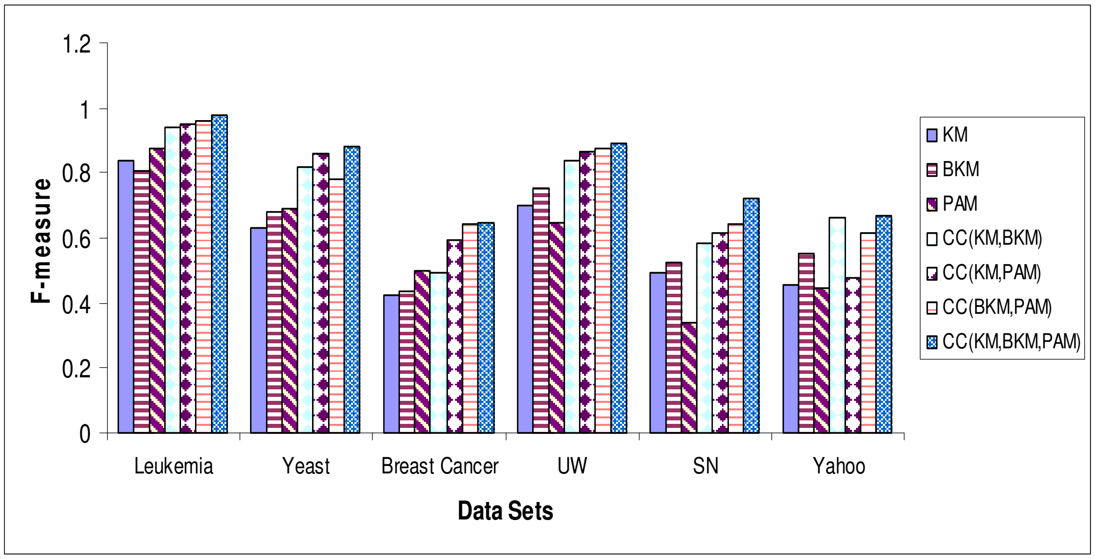


**Fig. 4. 2.** Further Improvement in *F-measure* using CC(KM,BKM,PAM)

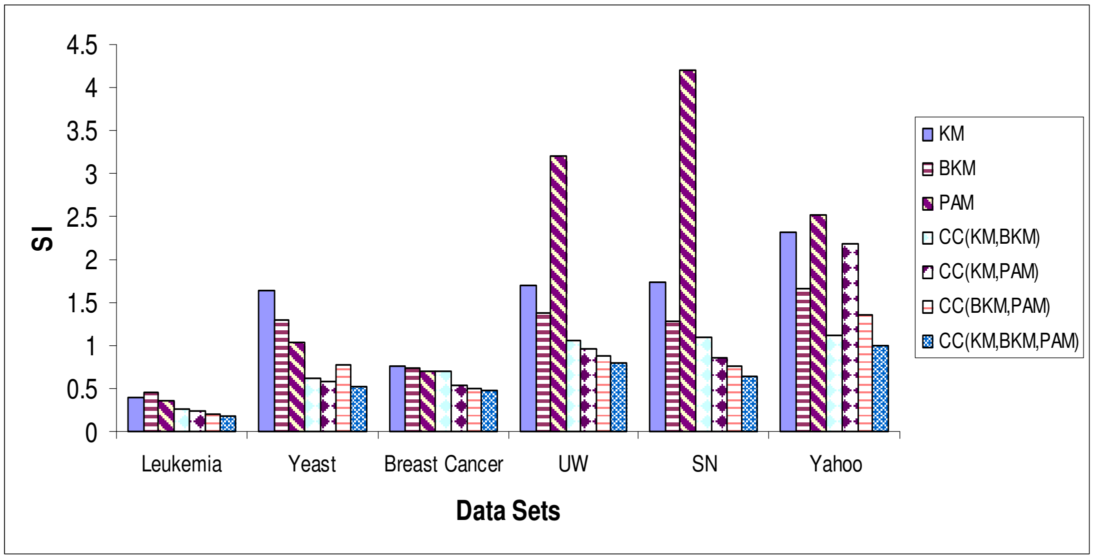


**Fig. 4. 3.** Further Improvement in *SI* using CC(KM,BKM,PAM)

We can see that the triple cooperation between the three clustering techniques achieves better clustering quality than the pair-wise cooperation measured by higher values for *F-measure* and lower values for *SI* index. This enhancement in the performance caused by the triple cooperation is mainly because a new set of sub-clusters is obtained with better homogeneity than that of the pair-wise cooperation. This set of sub-clusters acts as an agreement between the three techniques together which gives an additional confidence of the distribution of objects within clusters. This agreement directs the cooperative model in such away to group more homogenous sub-clusters than those of the pair-wise cooperation.

### 4.5.4 Performance Evaluation at *c*=4 (adding FCM)

The performance of the fuzzy c-means (FCM) [12] is added to the model as shown in Figures 4.4 and 4.5 for both the *Yeast* and *UW* datasets. We will refer to the cooperative model that combines the clustering solutions of the four techniques as CC(KM,BKM,PAM,FCM).

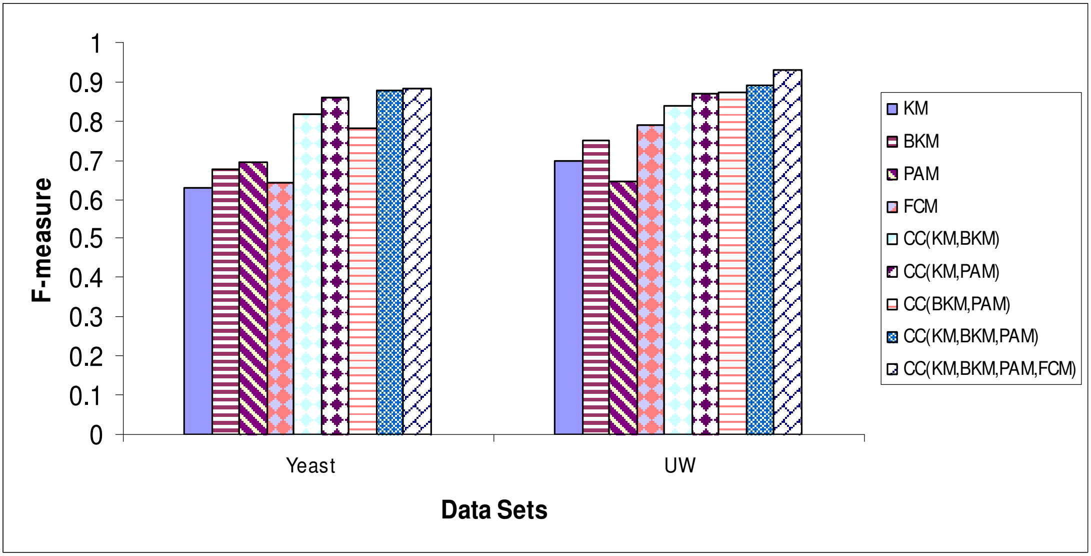


**Fig. 4. 4.** Improvement in *F-measure* by adding FCM

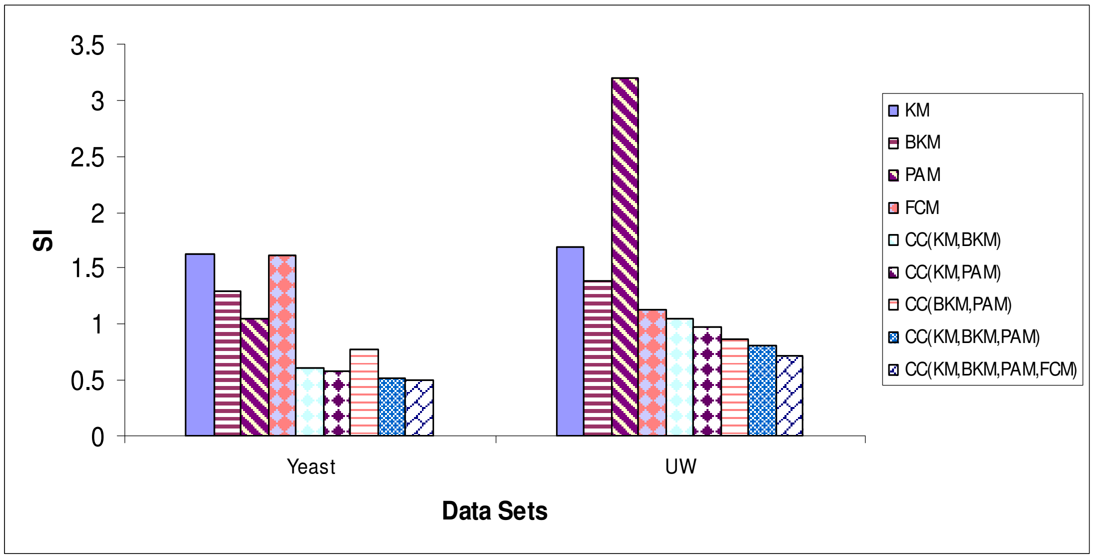


**Fig. 4. 5.** Improvement in *SI* by adding FCM

It can be shown that adding FCM to the cooperative model maintains the same clustering quality for the *Yeast* dataset, as the performance of FCM is almost the same as KM. Thus adding FCM has no additional benefit to the cooperative model. On the other hand, for the *UW* dataset, FCM has better performance than KM, this difference in the performance with better sub-clusters homogeneity provides the cooperative model CC(KM,BKM,PAM,FCM) with more homogenous set of sub-clusters that achieves a better clustering quality. Thus adding FCM provides clustering solutions with higher values of *F-measure* and lower values for the *SI* index as illustrated in figures 4.4 and 4.5, respectively.

## 4.6 Scalability of the Cooperative Clustering (CC) Model

In order to evaluate the scalability of the cooperative model in terms of number of clustering techniques, we use combinations of KM, BKM and PAM such that *c* (number of clustering techniques) ranges from 2 up to 100 algorithms as shown in Figures 4.6, 4.7, and 4.8, for *Leukemia*, *Yeast*, and *UW* datasets, respectively. In each table, we plot the ratio of number of singleton sub-clusters (sub-clusters with size 1) to the total number of sub-clusters, the quality of sub-clusters (measured by the *OWSR* measure), and the quality of the overall set of *k* clusters (measured by *F-measure*).

For the *Leukemia* dataset, it can be noticed that the cooperative model achieves better clustering results using up to 39 algorithms measured by higher values of the *OWSR* of the generated set of sub-clusters as well as the higher values of the *F-measure* for the overall set of *k* clusters. For the *Yeast* dataset, a combination of up to 37 algorithms is used to obtain better results than the original individual approaches, and finally for the UW dataset, up to 13 algorithms are invoked to obtain the best cooperative clustering results than those of the adopted non-cooperative approaches.

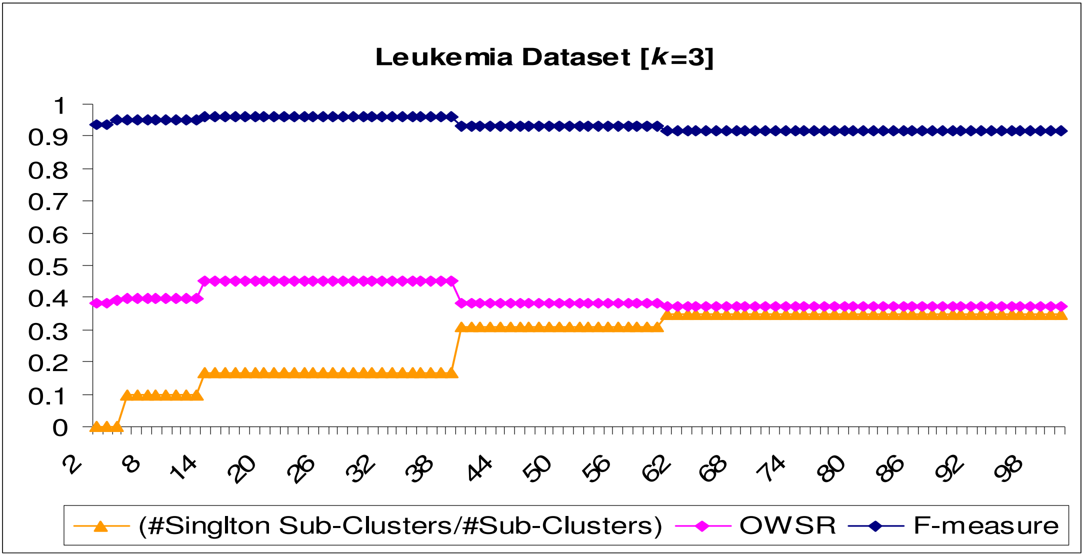


**Fig. 4. 6.** Scalability of the Cooperative Model [*Leukemia*]

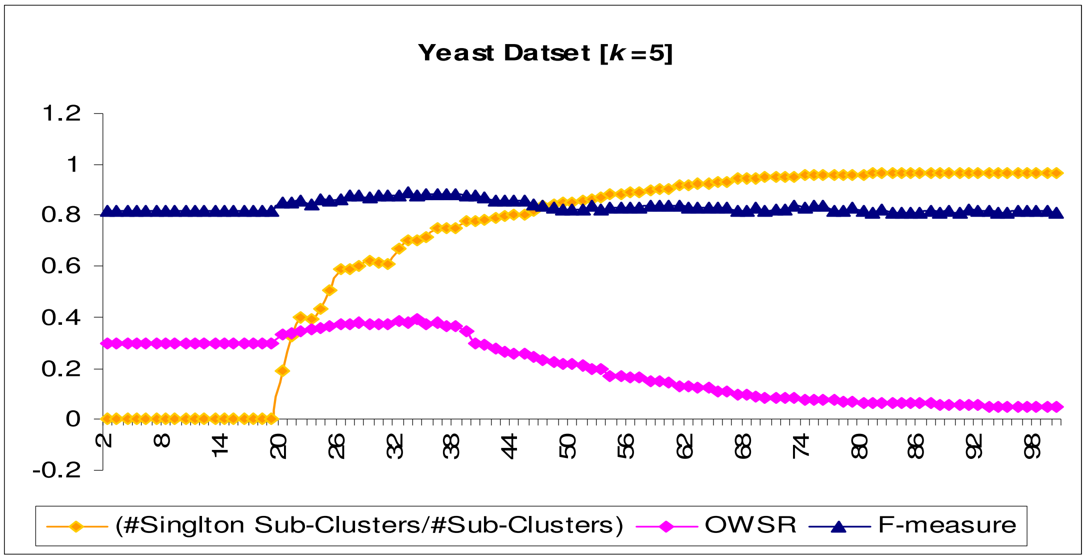


**Fig. 4. 7.** Scalability of the Cooperative Model [*Yeast*]

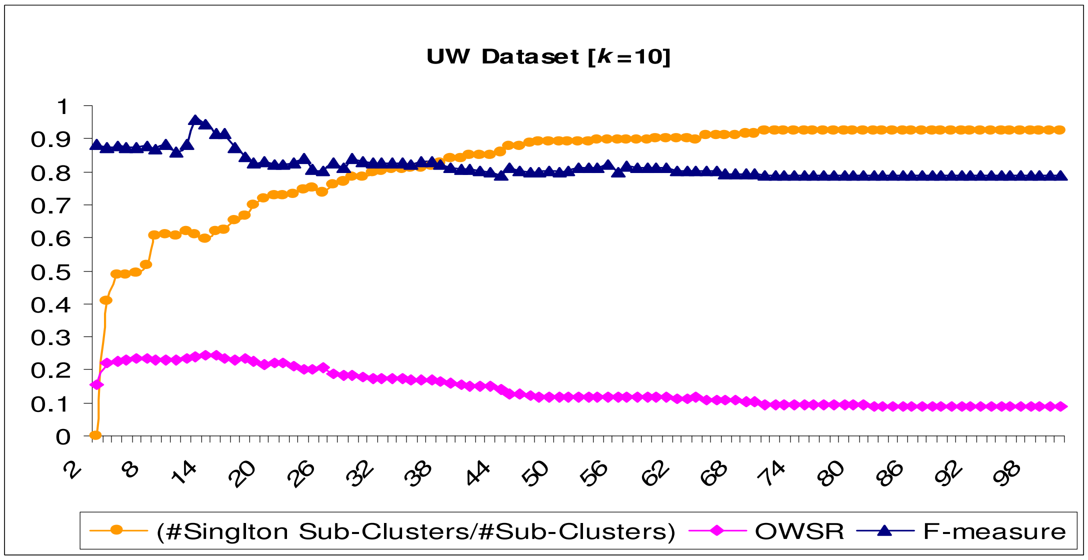


**Fig. 4. 8.** Scalability of the Cooperative Model [*UW*]

An interesting observation is that the cooperative clustering model after a specific value of $c$, $c^*$ (e.g. $c^* = 39$ in the *Leukemia* dataset), the cooperative clustering quality degrades rapidly. It is not surprising that this is the case, since at larger number of algorithms with different clustering solutions; the generated set of sub-clusters is expected to have larger number of singleton sub-clusters which drops the overall quality of the set of sub-clusters. The value of $c^*$ provides a clue of the relation between the number of sub-clusters, number of singleton sub-clusters, and the overall quality of the set of sub-clusters (measured by

*OWSR*), beyond which the number of algorithms should not be increased. An appropriate strategy for automatically detecting the value of $c^*$ is to compare the values of *OWSR* before and after adding the additional clustering techniques, if a sufficiently drop in the *OWSR* is noticed then no more algorithms can be added to the cooperative clustering model.

## 4.7 Variable Number of Clusters

As clustering is known as unsupervised classification of data, thus number of clusters is unknown as in the *Serum* dataset. In this experiment, we investigate the performance of the cooperative models as well as the individual approaches along with variable number of clusters. The proper number of clusters (i.e. natural grouping of data) is obtained based on the lowest value of the *SI* index. Fig. 4. 9 shows the performance of the cooperative models as well as the non-cooperative algorithms for the *Serum* dataset at variables number of clusters.

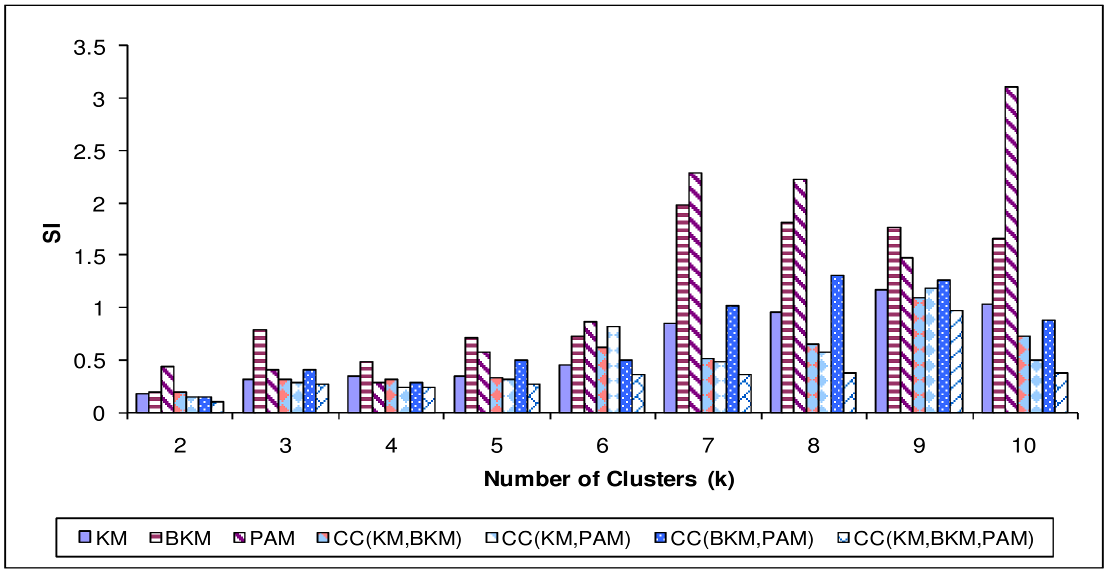


**Fig. 4. 9.** Finding Proper Number of Clusters (*k* is unknown) [*Serum*]

For the non-cooperative algorithms, it can be shown that the best performance of KM is achieved at *k*=2, BKM at *k*=2, and PAM at *k*=4. The CC(KM,BKM) has the lowest value of the *SI* index at *k*=2, the CC(KM,PAM) achieves its best performance also at *k*=2, CC(BKM,PAM) has the best results at *k*=2. Finally, for the triple cooperation model, CC(KM,BKM,PAM), the best results are obtained at *k*=2. The natural grouping of data (i.e. proper number of clusters) is determined by a majority vote between the four cooperative models as they have better clustering quality than the non-cooperative approaches. Then best performance is obtained at *k*=2.

The *UW* dataset has external information about its true class labels where *k* is known apriori as 10 clusters. In Fig. 4. 10, It can be illustrated that the four cooperative models, CC(KM,BKM), CC(KM,PAM), CC(BKM,PAM), and CC(KM,BKM,PAM) obtain the true number of clusters (*k*=10) while the non-cooperative BKM achieves its best performance at *k*=9 and PAM declares that its best results are obtained at *k*=7. KM obtains the best results at *k*=10. Thus, the cooperation between the invoked algorithms is capable of finding the proper number of clusters in data.

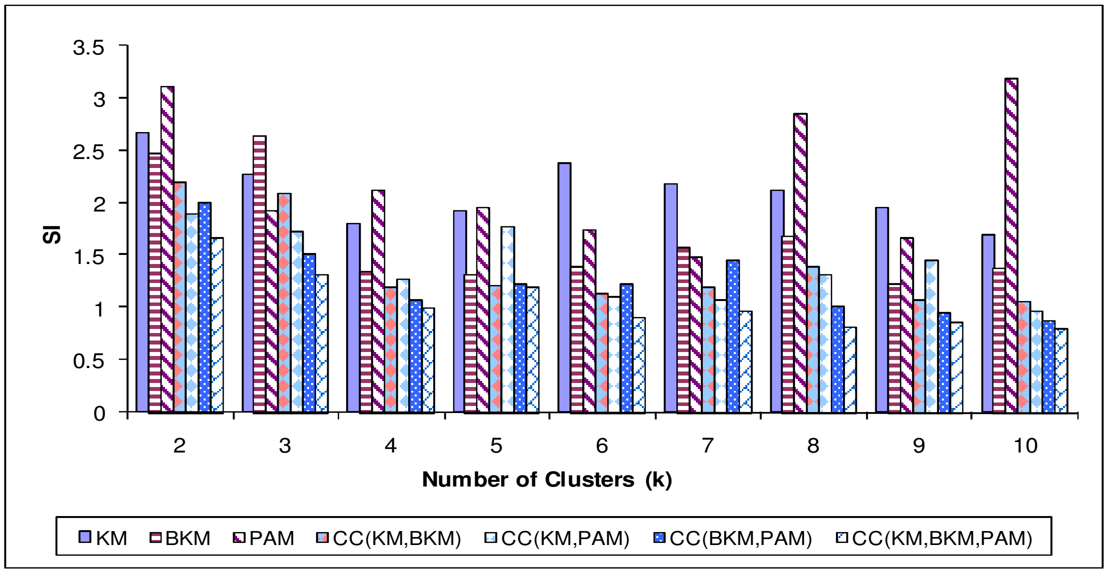


**Fig. 4. 10.** Finding Proper Number of Clusters (*k* is known) [*UW*]

For both the *Serum* and *UW* datasets, we can see that the cooperative models outperform the individual clustering techniques for variable number of clusters measured by lower values of the *SI* index.

## 4.8 Intermediate Cooperation

An additional advantage of the cooperative clustering is that, it is used to enhance the time (e.g. fast convergence) (and/or) quality performance (i.e. clustering quality) of the adopted techniques based on cooperation at the intermediate steps. Figures 4.11 and 4.12 illustrate the convergence of both KM and FCM with different initializations, respectively, with and without cooperation between them at the intermediate iterations of each algorithm for the *UW* dataset. Similar performance is achieved for the rest of the datasets. Figures 4.11 and 4.12 show that cooperation at intermediate steps enables both KM and FCM to achieve faster converge to solutions with minimum number of iterations. The main reason for this enhancement is that both KM and FCM are vulnerable to the initial seed of centroids. That means good initial centroids lead to better and faster convergence to solution. Using cooperation at the intermediate steps provides KM and FCM with a set of centroids of lower value of the objective function (Figures. 2.1 and 2.4) than the current objective function value. This new set of centroids enables KM and FCM to find the desired clustering quality with lower number of iterations. Thus KM with intermediate cooperation with FCM takes only 8 iterations to converge instead of 15 iterations. In addition, FCM needs only 6 iterations to find its local optimum instead of 11 iterations.

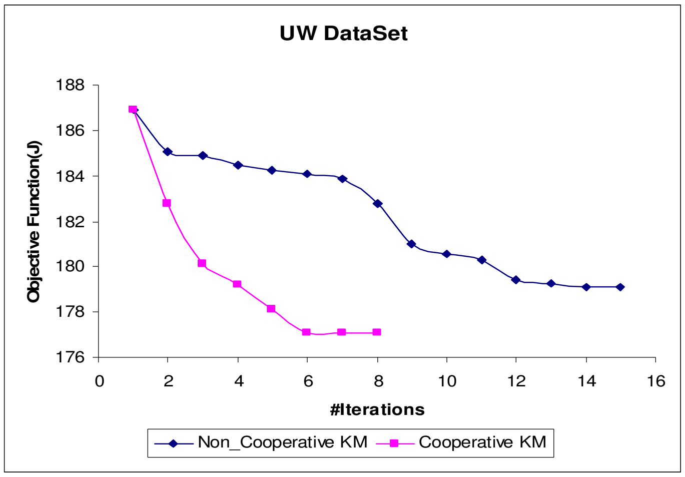


**Fig. 4. 11.** KM Convergence with and without Cooperation [*UW*]

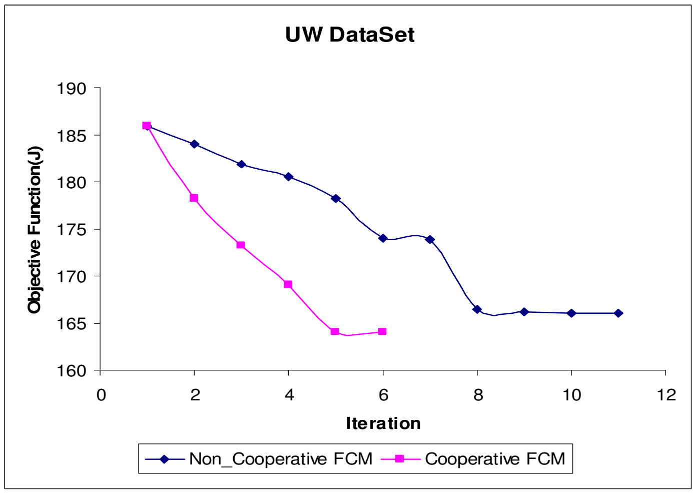


**Fig. 4. 12.** FCM Convergence with and without Cooperation [*UW*]

Fig. 4.13 shows the performance of the BKM with the intermediate feedback at each level of the hierarchical tree with cooperation with KM for the *UW* dataset. It can be shown from Fig. 4.13 that BKM takes better clustering solutions at each level with better homogeneity than the original clustering, thus in the splitting stage different partitions with less homogeneity will be selected and split further along the tree which enables the BKM to enhance its clustering quality along each level of the hierarchical tree.

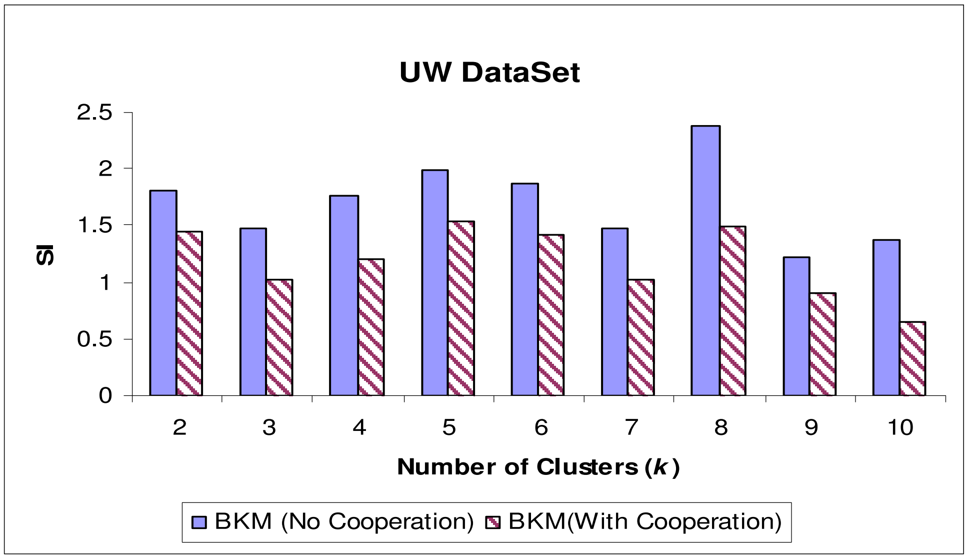


**Fig. 4. 13.** Quality of BKM using Intermediate Cooperation for Variable Number of Clusters [*UW*]

## 4.9 Discussions

In this chapter, an evaluation of the cooperative model using multiple clustering techniques with various clustering results is presented. Experiments were performed on actual gene expression datasets and text documents datasets representing different characteristics. Based on the experimental results, we can conclude that cooperative clustering achieves better clustering quality measured by both internal and external quality measures than the non-cooperative traditional clustering algorithms. Also a number of experiments were conducted to show the capability of the cooperative model to generate better clustering solutions with variable number of clusters. Also, undertaken experimental results show that the cooperative clustering model is scalable in terms of number of clustering techniques. Finally, the capability of the cooperative model to enhance the performance in terms of convergence property and clustering quality was illustrated. The conducted experiments conclude that the cooperative clustering introduced in this thesis is successful with respect to its goals. Detailed summary, conclusions, and recommendations are discussed in the last chapter.

# Chapter 5
# Outliers Detection Using Cooperative Clustering

Outlier detection refers to the problem of discovering objects that do not conform to expected behavior in a given dataset. These nonconforming objects are called *outliers* while the set of remaining objects are called *inliers*. A variety of techniques have been developed to detect outliers in several research applications including: bioinformatics and data mining [33]-[43]. Current approaches for detecting outliers using clustering techniques explore the relation of an outlier to the clusters in data. For example, in medical applications as gene expression analysis, the relation of unknown novel genes (outliers) to the gene clusters in data is important in studying the function of such novel genes. Also, in disease diagnosis analysis, the relation of an unknown pattern of symptoms (outlier) to a known cluster of symptoms can reveal important information related to the known disease. Some clustering algorithms find outliers as a side-product of the clustering process. For example, DBSCAN [51] can also handle outliers, but its main concern is clustering the dataset, not detecting outliers. The recent clustering-based outlier detection technique, *FindCBLOF* [39] is only based on the assumption that outliers form very small-sized clusters, also the detection accuracy of the *FindCBLOF* is mainly based on the clustering quality of the adopted clustering technique.

In this chapter, a novel clustering-based outlier detection method is proposed and analyzed; it is called Cooperative Clustering Outliers Detection (CCOD). Unlike the traditional clustering-based methods, e.g. *FindCBLOF*, the CCOD algorithm provides efficient outliers detection and data clustering capabilities. It uses the notion of cooperative clustering towards better discovery of outliers. The algorithm of our outlier detection method is divided into four stages. The first stage provides individual clustering. The second stage obtains the set of sub-clusters. The main objective of the third and four stages is an iterative identification of possible and candidate outliers of objects. The empirical results in chapter 6 indicate that the proposed method was successful in detecting outliers compared to the traditional clustering-based outlier's detection technique, *FindCBLOF*.

This chapter is organized as follows. Section 5.1 describes the different approaches for discovering outliers in data. Section 5.2 discusses current clustering-based detection approaches. The CCOD algorithm is presented and analyzed in section 5.3. Finally some discussions about the proposed cooperative clustering-based detection are presented in the last section.

## 5.1 Outliers Detection

Outlier's detection is a critical task in many safety critical environments as outliers indicate abnormal running conditions from which significant performance degradation may result. For example, in databases, outliers may indicate fraudulent cases or they may just denote an error by the entry clerk or misinterpretation of a missing value code, either way detection of the anomaly is vital for database consistency and integrity. Also to detect the onset of news stories, for topic detection and tracking or for traders to pinpoint equity, commodities, outperforming or underperforming commodities, this detection discovers the novelty in text [43].

***Definition***

Outlier detection can be described as follows: Given a set of *n* objects and *TopRatio*, the expected number of outliers, find the top objects *TopRatio* that are considerably dissimilar, exceptional, or inconsistent with respect to the remaining data.

Outliers may be erroneous or real in the following sense: Real outliers are observations whose actual values are very different than those observed for the rest of the data and violate plausible relationships among variables. Erroneous outliers are observations that are distorted due to misreporting or misrecording errors in the data-collection process. Outliers of either type may exert undue influence on the results of statistical analysis, so they should be identified using reliable detection methods prior to performing data analysis [102]. A more exhaustive list of applications that utilize outlier detection includes:

- Fraud detection: detecting fraudulent applications for credit cards, state benefits or detecting fraudulent usage of credit cards or mobile phones
- Intrusion detection: detecting unauthorized access in computer networks.
- Satellite image analysis: identifying novel features or misclassified features
- Medical condition monitoring such as heart-rate monitors
- Handwritten word recognition: some errors were caused by non-character images that were assigned a high character confidence value [103]
- Pharmaceutical research: identifying novel molecular structures

- Detecting unexpected entries in databases, for data mining to detect errors, frauds or valid but unexpected entries.
- Detecting mislabeled data in a training data set.

How the outliers detection systems deal with the outliers depends on the application area and the context where outliers are defined. Some of the well known outlier detections approaches are illustrated next.

### 5.1.1 Distance-based Outliers Detection

In *distance-based* outlier detection, an outlier is defined as an object that having a far distance to other objects in the data space.

***Definition***

An object $\boldsymbol{x}$ in a dataset is an *outlier* with respect to the parameters *MinPts* and *r*, if no more than *MinPts* objects in the dataset are at a distance *r* or less from $\boldsymbol{x}$.

This approach does not require any a prior knowledge of data distributions as the statistical methods do. However, this distance-based approach has certain shortcomings:

- It requires the user to specify a distance *r*, which could be difficult to determine apriori.
- It does not provide a ranking for the outliers: for instance an object with a very few neighboring objects within a distance *r* can be regarded in some sense as being a stronger outlier than an object with more neighbors within distance *r*.
- It becomes increasingly difficult to estimate parameter *r* with increasing dimensionality. Thus, if one picks radius *r* slightly small, then all objects are outliers. If one picks *r* slightly large, then no object is an outlier. So, user needs to pick *r* to a very high degree of accuracy in order to find a modest number of objects, which can be defined as outliers [104].

The definition, proposed by Ramaswamy *et al.* [105], for outliers in the high dimensional data does not require users to specify the distance parameter *r*. Instead, it is based on the distance of the *MinPts*th nearest neighbor of a point. The *Hybrid-random* algorithm [106] belongs to this family of outlier detection techniques.

### 5.1.2 Distribution-based Outliers Detection

In these techniques, the data points are modeled using a stochastic distribution and points are determined to be outliers depending upon their relationship with this model. However, with increasing dimensionality, it becomes increasingly difficult and inaccurate to estimate the multidimensional distributions of the data points.

***Definition***

An object is defined as an outlier if it is significantly different from the underlying distribution.

An on-line outlier detection algorithm called *SmartSifter* [107] takes a data sequence as input in an on-line way, learns an underlying model from the given examples and assigns a score to each object based on the underlying learned model. Thus a high score indicates a high probability that the object is an outlier. The central idea of *SmartSifter* is to learn the model with on-line learning algorithms and to calculate a score for a data. The main advantages of the method are that the computational time is inexpensive and it can deal with both categorical and continuous variables.

Also a Gaussian mixture model is used in [37] to present the normal behaviors and each datum is given a score on the basis of changes in the model. High score indicates high possibility of being an outlier. This approach has been combined with a supervised-based learning approach to obtain general patterns for outlier. The main problem with this method is that it assumes that the underlying data distribution is known a prior. However, for many applications, it is an impractical assumption and the cost for fitting data with standard distribution is significantly considerable.

### 5.1.3 Density-based Outliers Detection

*Density-based* methods have been developed for finding outliers in a spatial data. These methods can be grouped into two categories called multi-dimensional metric space-based methods and graph-based methods. In the first category, the definition of spatial neighborhood is based on Euclidean distance, while in graph-based spatial outlier detection the definition is based on graph connectivity. Density-based approaches consider both attribute values and spatial relationship in data.

***Definition***

Outliers are objects having low local density of an object’s neighborhood of objects.

***Local Outlier Factor (LOF)***

*Local Outlier Factor* (LOF) [36] is the density-based method, which detects local outliers based on the local density of an object's neighborhood. LOF is intuitively a measure of difference in density between an object and its neighborhood objects. We refer to LOF as a method from multi-dimensional metric space-based category of density-based approach. In a multidimensional dataset it is more meaningful to assign for each object a degree of being an outlier. The key difference between LOF approach and existing notions of outliers is that being outlier is not a *binary property*. Local outliers are the set of objects, which relative to their local neighborhoods have low densities of the neighborhoods. Let *MinPts* specifies the minimum number of objects in the neighborhood of an object.

***Definition*** (*MinPts*-distance neighborhood of an object $\boldsymbol{x}$)

The *MinPts*-distance neighborhood of $\boldsymbol{x}$ contains every object whose distance from $\boldsymbol{x}$ is not greater than the *MinPts*-distance. These objects are called the *MinPts*-nearest neighbors of $\boldsymbol{x}$, $N_{MinPts}(\boldsymbol{x})$.

***Definition*** (*reachability distance* of an object $\boldsymbol{x}$ w.r.t. object $\boldsymbol{y}$)

The *reachability distance* of object $\boldsymbol{x}$ with respect to object $\boldsymbol{y}$ is defined as $reach\_dist_{MinPts}(\boldsymbol{x}, \boldsymbol{y}) = max\{MinPts\text{-distance}(\boldsymbol{x}), \text{distance}(\boldsymbol{x}, \boldsymbol{y})\}$.

If object $\boldsymbol{x}$ is far away from $\boldsymbol{y}$, then the reachability distance between the two is simply their actual distance. However, if they are close, the actual distance is replaced by the *MinPts-distance* of $\boldsymbol{x}$.

***Definition*** (*local reachability* density of $\boldsymbol{x}$, $lrd(\boldsymbol{x})$)

The local reachability density of an object $\boldsymbol{x}$ is the inverse of the average reachability distance from the *MinPts*-nearest neighbors of $\boldsymbol{x}$.

$$lrd_{MinPts}(\boldsymbol{x}) = 1 / \left( \frac{\sum_{y \in N_{MinPts}(x)} reach-dist_{MinPts}(\boldsymbol{x}, \boldsymbol{y})}{|N_{MinPts}(\boldsymbol{x})|} \right) \tag{5. 1}$$

The local outlier factor (LOF) is a measure of outlying-ness that is calculated for each object. LOF is the average of the ratios of the local reachability density of $\boldsymbol{x}$ and those of $\boldsymbol{x}$'s *MinPts* nearest-neighbors. The local outlier factor of an object $\boldsymbol{x}$ is defined as:

$$LOF_{MinPts}(\boldsymbol{x}) = \frac{\sum\limits_{\boldsymbol{y} \in N_{MinPts}(\boldsymbol{x})} \frac{lrd_{MinPts}(\boldsymbol{y})}{lrd_{MinPts}(\boldsymbol{x})}}{|N_{MinPts}(\boldsymbol{x})|} \quad (5.\ 2)$$

Intuitively, $\boldsymbol{x}$'s local outlier factor will be very high if its local reachability density is much lower than those of its neighbors [108],[109]. Local outliers are objects having considerable density difference from their neighboring objects, i.e. they have high LOF values.

### 5.1.4 Deviation-based Outliers Detection

*Deviation-based* outlier detection does not use statistical tests or distance-based measures to identify exceptional objects. Instead, it identifies outliers by examining the main characteristics of objects in a group. Objects that "deviate" from this description are considered outliers.

***Definition***

Outliers are discovered by inspecting the characteristics of objects and consider an object that deviates from this description as an outlier.

The sequential exception technique simulates the way in which humans can distinguish unusual objects from among a series of supposedly similar objects [38].

### 5.1.5 Clustering-based Outliers Detection

These set of techniques employ clustering approaches to discover outliers in data. Thus, the ability to detect outliers can be improved using a combined perspective from outlier detection and cluster identification.

***Definition***

*Inliers* are defined as objects that belong to large and dense clusters, while *outliers* either do not belong to any cluster or form very small clusters. Thus any object, which does not fit in any cluster, is called outlier.

The *FindCBLOF* algorithm [39] uses a clustering algorithm called *Squeezer* [40] and determines the Cluster-based Local Outliers Factor (*CBLOF*) for each object. As the focus of this chapter is on clustering-based outlier detection, the next section illustrates the state of art of discovering outliers based on clustering in more details.

## 5.2 Outliers in Clustering

The main concern of *clustering-based* outlier detection algorithms is to find *clusters* and *outliers*, which are often regarded as noise that should be removed in order to make more reliable clustering [39]. Some noisy points may be far away from the data points, whereas the others may be close. The far away noisy points would affect the result more significantly because they are more different from the data points. It is desirable to identify and remove the outliers, which are far away from all the other points in cluster [110] So, to improve the clustering such algorithms use the same process and functionality to solve both clustering and outlier discovery. Some clustering algorithms find outliers as a side-product of clustering algorithms. For example, DBSCAN [51] and ROCK [111] can also handle outliers, but their main concern is clustering the dataset, not detecting outliers. However these techniques define outliers as points, which do not lie in clusters or form very small clusters. Thus, the techniques implicitly define outliers as the background noise in which the clusters are embedded. Another class of techniques defines outliers as points, which are neither a part of a cluster nor a part of the background noise; rather they are specifically points which behave very differently from the norm [104]. The clustering-based outlier detection approach, known as *FindCBLOF* algorithm [39], assigns a cluster-based local outlier factor to each object and returns objects with the highest local factors as outliers. More details on the *FindCBLOF* algorithm are discussed next.

### 5.2.1 Find Cluster-based Local Outlier Factor (*FindCBLOF*)

To identify the physical significance of the definition of an outlier, each object is assigned an outlier factor, namely, *CBLOF*, which is a measure of both the size of the cluster the object belongs to and the distance between the object and its closest cluster (if the object lies in a *small* cluster) [39]. Here, the clustering algorithm used for partitioning the dataset into disjoint clusters can be chosen freely. The only requirement for the selected clustering algorithm is that it should have the ability to produce good clustering results. A critical problem that must be solved before defining the *cluster-based local outlier factors* is how to identify whether a cluster is *large* or *small*. Suppose $S=\{S_0,S_1,\ldots,S_{k-1}\}$ is the set of clusters in the sequence that $|S_0|\geq|S_1|\geq..\geq|S_{k-1}|$. Given two numeric parameters $\alpha$ and $\beta$, the *boundary* of *large* and *small* cluster, $u$, is defined as following (if one of the two following formulas holds):

$$(|S_0|+|S_1|+.....+|S_u|)\geq(|X|*\alpha) \tag{5. 3}$$

$$|S_u|/|S_{u+1}|\geq\beta \tag{5. 4}$$

The *CBLOF* of an object $\boldsymbol{x}$ belongs to cluster $S_i$ represented by a prototype $z_i$ is defined as:

$$CBLOF(\boldsymbol{x})|_{\alpha,\beta}=\begin{cases} |S_i|*min(\|\boldsymbol{x}-z_j\|_2, \boldsymbol{x}\in S_i, S_i \in Small\,Set \ and \ S_j \in Large\,Set) \\ |S_i|*\|\boldsymbol{x}-z_i\|_2, \boldsymbol{x}\in S_i, S_i \in Large\,Set \end{cases} \tag{5. 5}$$

The *CBLOF* of an object is determined by the size of its cluster, and the distance between the object and its closest cluster (if this object lies in *small* cluster) or the distance between the object and the cluster it belongs to (if this object belongs to *large* cluster). For the computation of distance between an object and a cluster, it is sufficient to adopt the similarity measure used in the clustering algorithm.

The *FindCBLOF* algorithm first partitions the dataset into clusters with the *Squeezer* algorithm [40]; the *Squeezer* algorithm works *only* with categorical attributes. The sets of *large* and *small* clusters are derived using the parameters $\alpha$ and $\beta$. Then, for every data point in the data set, the value of *CBLOF* is computed. Outliers are returned as objects with higher *CBLOF* values.

```
Algorithm: FindCBLOF (X, α ,β, TopRatio, A)
Input: The dataset X, two numeric parameters $\alpha$ and $\beta$, and the invoked clustering approach A
Output: The set of Outliers, OL
Begin
  Step1: Partition the dataset into a set of k clusters using the clustering Algorithm A, thus $S$=A(X,$k$,$\zeta$)
         where $S=\{S_i, i=0,..,k-1\}$ and $\zeta$ is the set of parameters of the clustering algorithm A
  Step2: Obtain the LargeSet and the SmallSet using the parameters $\alpha$, $\beta$
  Step3: For each object x in the dataset X
            If $x \in S_i$ and $S_i \in$ the SmallSet then $CBLOF(\boldsymbol{x}) = |S_i|*min(\|x-z_j\|_2, \boldsymbol{x}\in S_i, S_j \in LargeSet$
            Else $CBLOF(\boldsymbol{x}) = |S_i|*\|\boldsymbol{x}-z_i\|_2, \boldsymbol{x}\in S_i, S_i \in LargeSet$
         End
Return the set of TopRatio% objects with the highest CBLOF values as the outliers list, OL
End
```

**Fig. 5. 1.** *FindCBLOF* Algorithm

The efficiency of the clustering-based *FindCBLOF* approach for detecting outliers in data is constrained to the quality of the adopted clustering technique. In [112], it has been experimentally approved that better clustering solutions reveal better detection of outliers using the notion of *CBLOF*.

## 5.3 Outliers Detection Using Cooperative Clustering

In this section, we introduce a new clustering-based outlier's detection algorithm called Cooperative Clustering Outliers Detection (CCOD) that uses the notion of cooperative clustering towards better discovery of outliers. The purpose of our method is not only to perform data clustering but at the same time it discovers outliers. The CCOD algorithm detects outliers in a bottom-up scenario. The proposed outlier detection is divided into four stages. The first stage provides non-cooperative clustering for a set of *c* clustering algorithms, the second stage obtains the set of sub-clusters, the third stage identifies a possible set of outliers by assigning a *cooperative outlier factor* to each object in each sub-cluster, and finally the last stage returns the overall set of candidate outliers that affects on the homogeneity of the cooperative clustering process.

### 5.3.1 Cooperative Outlier Factor

Our outlier detection method employs three facts of outliers:

- Objects are considered as outliers if they either do not belong to any cluster or form very small clusters
- Outliers may exist in large clusters
- Outliers affect on the homogeneity of the clustering results of any clustering technique

Assume the similarity threshold at which the histograms in the Cooperative Clustering (CC) model (Section 3.4) were truncated for the merging process is $\delta$. Also in our detection approach we will differentiate between small and large sub-clusters in order to find the set of possible outliers at different number of partitions where a large sub-cluster means strong agreement while small sub-cluster indicates week agreement and higher possibility of being an outlier.

Let $Sb=\{Sb_0,Sb_1,\ldots,Sb_{nsb-1}\}$ be the set of $n_{sb}$ sub-clusters generated by the CC model in the sequence $|Sb_0|\geq|Sb_1|\geq..\geq|Sb_{nsb-1}|$. Given two numeric parameters $\alpha_{Sb}$ and $\beta_{Sb}$, the *boundary* of *large* and *small* sub-cluster, $v$, is defined as following (if one of the two following formulas holds):

$$(|Sb_0|+|Sb_1|+.....+|Sb_v|)\geq(|X|*\alpha_{Sb}) \tag{5. 6}$$

$$|Sb_v|/|Sb_{v+1}|\geq\beta_{Sb} \tag{5. 7}$$

Equations (5.6) and (5.7) give a quantitative measure to distinguish large and small sub-clusters, and then the sets of *large* and *small* sub-clusters are defined as:

$$\begin{cases} Large\ Subcluster\ Set\ (LSS) = \{Sb_0, Sb_1, .., Sb_v\} \\ Small\ \ Subcluster\ Set\ (SSS) = \{Sb_{v+1}, Sb_{v+2}, .., Sb_{n_{sb}-1}\} \end{cases} \quad (5.\ 8)$$

Two types of outliers are defined and proposed, the *Intra_outliers* and the *Inter_outliers*. An illustrative example of *Intra_outliers* and the *Inter_outliers* in small and large sub-clusters is illustrated in Fig. 5. 2, where we refer to inliers with the symbol x and outliers with the circle o.

***Definition*** (*Intra_outliers*)

*Intra_outliers* are objects having far distances from objects in the same sub-cluster, thus ***x*** is an intra-outlier in a sub-cluster *Sb<sub>i</sub>*, if $\forall y \in Sb_i, x \neq y, |\, Sim(x, y) < \delta \,|\ is\, maximum$, where $\delta$ is the similarity threshold and $|\, Sim(x, y) < \delta \,|$ refers to the number of pair-wise similarities that are lower than $\delta$.

***Definition*** (*Inter_outliers*)

Inter-outliers are objects of a small sub-cluster that have far distances from objects in large sub-clusters.

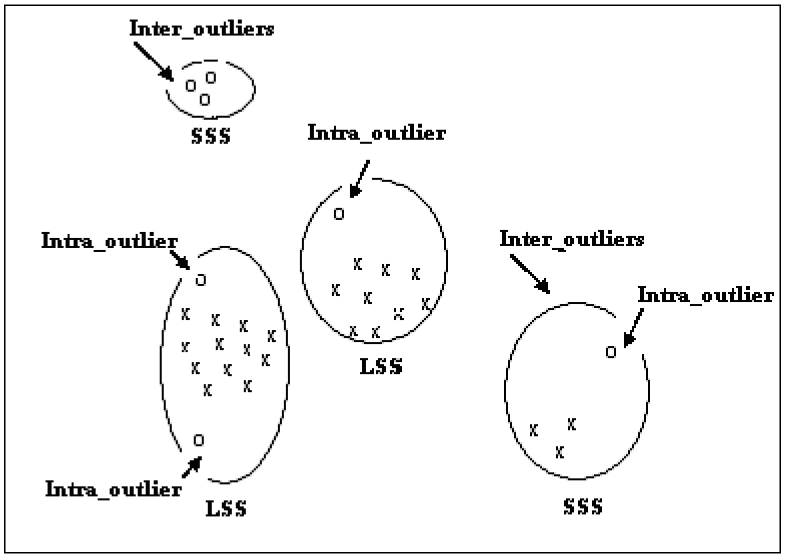


**Fig. 5. 2.** Outliers in Large and Small Sub-Clusters

Each object is assigned an outlier factor called *Cooperative Outlier Factor*, *COF*. The *COF* monitors the pair-wise similarities between objects in the same sub-cluster where objects with the highest count of similarities below the similarity threshold are identified as *Intra-Outliers*. *Intra-Outliers* can be found in both large and small sub-clusters. For objects in small sub-clusters, the *COF* combines both the weight of being *Intra-Outlier* and *Inter-Outlier*. The distances between the *Inter-outliers* and large sub-clusters are calculated as the distance to their centroids. The *COF* of an object $\boldsymbol{x}$ belongs to sub-cluster $Sb_i$ is defined as:

$$COF(\boldsymbol{x})=\begin{cases}\min\limits_{\forall Sb_j\in LSS}(1-Sim(\boldsymbol{x},c_j))+\dfrac{|Sim(\boldsymbol{x},\boldsymbol{y})<\delta|}{(|Sb_i|-1)}, \forall \boldsymbol{y}\in Sb_i, Sb_i\in SSS\\ \dfrac{|Sim(\boldsymbol{x},\boldsymbol{y})<\delta|}{(|Sb_i|-1)}, \forall \boldsymbol{y}\in Sb_i, Sb_i\in LSS\end{cases} \tag{5. 9}$$

The cosine similarity is used to compute the pair-wise similarities between objects in the same sub-clusters as well as the similarity between objects in small sub-clusters and the centroids of large sub-clusters. Objects with high values of the *COF* are considered as outliers within the set of sub-clusters.

The key difference between the CCOD and the *FindCBLOF* is that the *COF* is assigned to objects within the set of sub-clusters, which composes an additional confidence of being an outlier, where the set of sub-clusters acts as an agreement between the multiple clusterings. In addition, the *COF* takes into account that outliers may exist in both large and small sub-clusters, where it detects both Intra and Inter outliers in the set of sub-clusters which are not discovered by the traditional *CBLOF*.

### 5.3.2 Cooperative Clustering-based Outlier Detection (CCOD) Algorithm

The cooperative clustering CC model (section 3.4) generates a set of sub-clusters at different number of partitions $l$=2,..,$k$. The size of the resulting sub-clusters is smaller than the size of the original clusters. The small size of sub-clusters enables the discovering of local outliers in each sub-cluster by assigning local cooperative outlier factors (*COF*) to local objects in each sub-cluster. Local objects in each sub-cluster with the highest *COF* are selected as local outliers within sub-clusters. These discovered set of local outliers provides the possible set of outliers for the whole set of $k$ clusters. Then, in order to obtain the same number of clusters, the most similar two sub-clusters are merged. The set of possible outliers are tested against the merging process in order to identify the candidate outliers that affect on the homogeneity of the merging process which consequently affect on the overall homogeneity of the clustering procedure. The Cooperative Clustering Outlier Detection (CCOD) algorithm works in a bottom-up scenario. The bottom-up detection is an iterative approach that starts form level $l$ =2(i.e. number of partitions $l$= 2) to level $l$=$k$ (i.e. number of partitions $l$= $k$). It detects a set of outliers at level $l$, these set of outliers are considered as candidate outliers at level $l$ and possible outliers at level $l$+1.

For number of partitions $l$=2,3,..,$k$, the bottom-up cooperative clustering-based outliers detection is performed in the following four phases:

- **Phase 1** (*Non-Cooperative Clustering*): The $c$ clustering algorithms $\{A_1,A_2,..,A_c\}$ are executed concurrently, each algorithm takes its input parameters $\zeta_i$ and generates a set of $l$ clusters thus the clusterings sets $S^{Ai}(l)$ , $i$ =1,..,$c$ are generated, where $S^{Ai}(l)= A_i(X, l, \zeta_i) =\{S_j^{Ai}, 0\leq j \leq l-1\}$.
- **Phase 2** (*Sub-clusters Generation*): based on the cooperative model CC, the $c$ sets $S^{Ai}(l)$, $i$ =1,..,$c$, are employed to construct the *CCG* graph and consequently a new set of sub-clusters *Sb* is generated. Each sub-cluster $Sb_i$, $i$=0,1,…,$n_{sb}$-1 is represented with a histogram $H_i$ as a representative of the pair-wise similarities between objects in the sub-cluster.
- **Phase 3** (*Possible Outliers Detection*): for each sub-cluster $Sb_i$, a *COF* (defined in Eq.(5.9)) is assigned to each local object in the sub-cluster $Sb_i$. The *COF* is mainly based on the distribution of objects within sub-clusters and distance between objects and sub-clusters using the notion of histograms. Then, the set of *%LocalTopRatio* (a user defined parameter) outliers is selected. This set is the set of *possible* outliers $PO_l$ at level $l$.
- **Phase 4** (*Merging Process and Candidate Outliers Detection*):
  - At this stage, we consider the number of clusters as the number of sub-clusters $n_{sb}$. In order to obtain the same number of clusters as the original $l$ clusters, the two most similar sub-clusters (sub-clusters with the highest value of *mcf*) are selected for merging into a new cluster.
  - For each object ***o*** in the set $PO_l$, if removing ***o*** results in a selection of two other sub-clusters with better homogeneity (i.e. higher value of the *mcf* than the old value), then ***o*** becomes a *candidate* outlier at level $l$.
  - Finally, the selected two most similar sub-clusters are then merged.

Phase 4 is repeated until the number of clusters equals $l$. Then, the set of candidate outliers at level $l$ will be added to the set of possible outliers at level $l$+1. The four phases are repeated until the desired number of clusters $k$ is reached. The resulting set of candidate outliers are sorted according to their *COF* and the *TopRatio* outliers from the final candidate set are returned. Finally the final set of top outliers and the set of $k$ cooperative clusters are obtained. The four-phases cooperative clustering-based outlier detection algorithm is presented in Fig. 5. 3.

```
Algorithm: Cooperative Clustering-based Outlier Detection (X, SM,k, {A_i},{ζ_i},δ,LocalTopRatio,α_sb,β_sb)
Input: : Dataset X, similarity matrix SM, number of clusters k, set of clustering algorithms, $A_1,A_2,..,A_c$, a
         set of input parameters $\zeta=\{\zeta_i\}$ for each clustering technique $A_i$, similarity threshold $\delta$, number of
         local top outliers LocalTopRatio, and two numeric local parameters $\alpha_{sb},\beta_{sb}$.
Output: Set of candidate outliers $CO_k$ , and set of cooperative Clusters $S^{cooperative}(k)$
Initializations: $CO_1=\{\}$, $S^{cooperative}(k)=\{\}$
Begin
 For number of clusters $l$ =2 to $k$
     Phase 1: Generate the c clustering sets, $S^{Ai}(l)$ , $i$ =1,..,c , , $S^{Ai}(l)$ ,where $S^{Ai}(l)=A_i(X,\zeta_i, l)$
     Phase 2: Construct the CCG; a new set of disjoint sub-clusters $Sb=\{Sb_i, i=0,1,..,n_{sb}-1\}$ is
              generated.
     Phase 3: Initially, $PO_l$ ={}, assign COF factor to each object in the sub-clusters using the
              parameters $\alpha_{sb}$, $\beta_{sb}$, $\delta$, a new set of possible LocalTopRatio outliers $PO_l$ with the
              highest COF values is obtained.
     Phase 4: $S^{cooperative}(l) =Sb$, $CO_l=\{\}$,$PO_l= PO_l \cup CO_{l-1}$
         Repeat
           - Select the two most similar sub-clusters $Sb_i$, $Sb_j$
             For each object o in the list $PO_l$
                 Select the two most similar sub-clusters $Sb_x$,$Sb_y$ excluding o,
                     If($Sb_x$,$Sb_y$) has better homogeneity than ($Sb_i$,$Sb_j$) then if $o \notin CO_l$ then
                       $CO_l=CO_l+\{o\}$
             End
           - Merge the selected homogeneous sub-clusters
           - decrement number of sub-clusters, $n_{sb}$ by one
         Until (number of sub-clusters $n_{sb} = l$)
  End
Return $CO_k$ and $S^{cooperative}(k)$
End
```

**Fig. 5. 3.** The Cooperative Clustering-based Outlier Detection (CCOD)

### 5.3.3 Complexity Analysis

Assume $T^{A_1}(l)$, $T^{A_2}(l)$,..,$T^{A_c}(l)$ are the computational time complexity of the clustering techniques $A_1, A_2, .., A_c$, respectively, at a given number of clusters $l$=2,3,..,$k$. In **Phase 1**, a set of $c$ clustering approaches are employed in the CCOD algorithm, thus Phase 1 takes the computational time of the clustering approach with the maximum processing time,

$$T^{Phase\ 1} = max(T^{A_1}(l), T^{A_2}(l), .., T^{A_c}(l)) \tag{5. 10}$$

In the sub-clusters generation phase, Phase 2 takes $n$ operations to find the set of sub-clusters, where $n$ is the total number of objects, and building histograms is of order $O(|Sb_i|^2)$ thus **Phase 2** is of $O(n+|Sb_i|^2)$. In **Phase 3**, assigning *COF* for each object requires $|Sb_i|^2$ operations, Thus **Phase 3** is of order $O(|Sb_i|^2)$. Finally, in **Phase 4**, finding the most homogenous sub-clusters to be merged is of order $O(n^2_{sb})$, $n_{sb} \leq l^c$ , this merging testing is repeated for each possible outlier. Thus **Phase 4** is of order $O(LocalTopRatio* n^2_{sb})$. Updating the *CCG* is of order $O(n_{sb}^2+|S_i|*|Sb_j|)$. The total time complexity of the proposed CCOD algorithm at level $l$ is:

$$T^{CCOD}(l)=O(max(T^{A_1}(l), T^{A_2}(l), .., T^{A_c}(l)))+ O(n+|Sb_i|^2+ n_{sb}^2) \tag{5. 11}$$

## 5.4 Discussions

Using the same process and functionality to solve both clustering and outlier discovery is highly desired. Such integration will be of great benefit to discover outliers in data and consequently obtain better clustering results after eliminating the set of outliers. It is known that the capability of discovering outliers using clustering-based techniques is mainly based on the quality of the adopted clustering approach. In this chapter, we presented a novel clustering-based outliers detection algorithm (CCOD) that uses the notion of cooperative clustering towards better detection of outliers. This approach is based on assigning a cooperative outlier factor to each object and recognizing the set of candidate outliers after each merging step in the cooperative clustering model. The CCOD algorithm relies on the fact that cooperative clustering outperforms non-cooperative clustering to achieve better detection of outliers in data. Experimental results illustrate that the detection accuracy of the CCOD in terms of number of the discovered outliers is higher than that of the traditional clustering-based detections.

# Chapter 6
# Cooperative Clustering Outliers Detection: Experimental Results

In this chapter, the detection accuracy of the cooperative detection algorithms is compared to that of the *FindCBLOF* [39] approach using the non-cooperative KM [7] , BKM [13], or PAM [14] algorithms to show its important advantage of better detection of outliers. In the experiments, the CCOD($A_1$,$A_2$) refers to the cooperative detection between the clustering algorithms $A_1$ and $A_2$. Consequently, the CCOD($A_1$,$A_2$,$A_3$) refers to the cooperative detection between $A_1$, $A_2$, and $A_3$ where $A_1$, $A_2$, and $A_3 \in$ {KM, BKM, PAM}. Also we will refer to the *FindCBLOF*($A_i$) as the *FindCBLOF* detection algorithm using the individual non-cooperative clustering algorithm $A_i$. Table 6.1 illustrates the parameters setting of the adopted approaches.

**Table 6. 1:** Parameters Settings

| **Parameter** | ***Algorithm*** | ***Value*** |
|---|---|---|
| $\alpha$ | *FindCBLOF* | 90% |
| $\beta$ | *FindCBLOF* | 5 |
| $\alpha_{sb}$ | CCOD | 70% |
| $\beta_{sb}$ | CCOD | 5 |
| *MinPts* | LOF | 10 |

## 6.1 Data Sets

Experiments were performed on a number of gene expression and documents datasets with various characteristics and degree of outliers. There are two gene expression data sets and two document datasets. The gene expression datasets are *Yeast* and *Breast Cancer*, and the two document datasets are *UW* and *Yahoo*. Detailed description of the characteristics of these datasets is presented in section 4.2.

## 6.2 Detection Accuracy

In this section, we obtained 20 runs of the *FindCBLOF*($A_i$) algorithm using KM, BKM, or PAM as well as 20 runs of each of the cooperative detection algorithms. The *t-test* is also used to assess whether the means of two groups are statistically different from each other or not. The critical *t*-value at degree of freedom equals 38 and 95% confidence interval is 2.024. More detailed discussion of the *t*-test can be found in section 4.3. For any cooperative detection algorithm CCOD($A_1$,$A_2$), $t_1$ and $t_2$, refer to the calculated $t$ values between the results of CCOD($A_1$,$A_2$) and those of $A_1$ and $A_2$, respectively. We will add $t_3$ to $t_1$ and $t_2$ such that $t_1$, $t_2$, and $t_3$ refer the $t$-test values between the combined cooperative detection algorithm CCOD($A_1$,$A_2$,$A_3$) and the three non-cooperative detection algorithms. For each of the non-cooperative detection and the cooperative detection, the *%TopRatio* outliers are returned. In the following tables, the comparison is established based on the number of matched outliers from the *TopRatio* outliers that are occurred in the top list of outliers detected by the LOF algorithm. We use the LOF just as a common base of comparison between the adopted approaches as well as the cooperative detection algorithms. The number of the selected top outliers ranges from 10% to 30% of the dataset size. We selected the *LocalTopRatio* possible outliers as $\left(\log(n_{Sb})/\log(k)\right) * TopRatio$, where $n_{sb}$ is the number of generated sub-clusters and $k$ is the number of clusters.

Tables 6.2-6.5 show the number of the discovered outliers using the cooperative detection algorithms, CCOD(KM,BKM), CCOD(KM,PAM), CCOD(BKM,PAM), and CCOD(KM,BKM,PAM) compared to that of the traditional *FindCBLOF* using KM, BKM, or PAM for the four datasets. In each table the value of the calculated $t$ is greater than the critical value of $t$ (from the $t$-distribution tables) which means that there is a statistical difference in the obtained results of the non-cooperative and cooperative approaches and thus the Null hypothesis (no significance difference) is rejected.

Also, it can be shown from tables 6.2-6.5 that the cooperative detection algorithms, CCOD(KM,BKM), CCOD(KM,PAM), CCOD(BKM,PAM), and CCOD(KM,BKM,PAM) are able to detect more outliers than the traditional *FindCBLOF* using the non-cooperative clustering approaches at different values of the *TopRatio* ranges from 10% to 30% across the four datasets. As the bottom-up cooperative detection method from level $l$ to level $l$+1 assigns a cooperative outlier factor to each object in a sub-cluster and assures that the discovered candidate outliers at level $l$ are also considered as possible outliers at level $l$+1. This set of candidate outliers are accumulated along the bottom-up path till the final set of candidate outliers for $k$ clusters is obtained. For example for the *Yeast* dataset, *FindCBLOF*(KM), *FindCBLOF*(BKM), and *FindCBLOF*(PAM) detect only 33%, 45%, and 50% of the top outliers, respectively, while the CCOD(KM,BKM), CCOD(KM,PAM), CCOD(BKM,PAM), and CCOD(KM,BKM,PAM) detect 57%, 63%, 54%, and 70%, respectively, of the top outliers at *TopRaio*=30%. Also for the *UW* dataset, the CCOD(KM,BKM,PAM) achieves up to 97% accuracy compared to only 66%, 78%, 54% accuracy of the *FindCBLOF* using the individual KM, BKM, or PAM, respectively, at *TopRaio*=30%. The same interpretation of results is achieved for the rest of datasets.

**Table 6. 2:** Number of the Detected Outliers for the *Yeast* Dataset

| *TopRatio* | *FindCBLOF* **(KM)** | *FindCBLOF* **(BKM)** | *FindCBLOF* **(PAM)** | **CCOD (KM,BKM)** | **CCOD (KM,PAM)** | **CCOD (BKM,PAM)** | **CCOD (KM,BKM,PAM)** |
|---|---|---|---|---|---|---|---|
| **10%** | 8± (4) | 13± (2) | 17±(2) | **24±(4)** $t_1$=12.64 $t_2$=11.00 | **29±(3)** $t_1$=18.78 $t_2$=14.88 | **19±(2)** $t_1$=9.48 $t_2$=3.16 | **31±(3)** $t_1$=20.57 $t_2$=22.32 $t_3$=22.32 |
| **15%** | 31± (5) | 37± (4) | 39±(3) | **46±(3)** $t_1$=11.50 $t_2$=8.05 | **53±(4)** $t_1$=15.36 $t_2$=12.52 | **43±(2)** $t_1$=4.74 $t_2$=4.96 | **57±(3)** $t_1$=19.94 $t_2$=17.88 $t_3$=18.97 |
| **20%** | 39± (6) | 50± (5) | 58±(2) | **63±(3)** $t_1$=16.00 $t_2$=9.97 | **69±(4)** $t_1$=20.46 $t_2$=11.00 | **61±(3)** $t_1$=8.43 $t_2$=3.72 | **77±(5)** $t_1$=21.75 $t_2$=17.07 $t_3$=15.77 |
| **25%** | 61± (5) | 83± (4) | 90±(3) | **105±(4)** $t_1$=26.54 $t_2$=12.64 | **113±(4)** $t_1$=36.31 $t_2$=20.57 | **99±(4)** $t_1$=17.39 $t_2$=5.36 | **120±(3)** $t_1$=45.25 $t_2$=33.09 $t_3$=31.62 |
| **30%** | 69± (7) | 96± (4) | 106±(5) | **121±(4)** $t_1$=28.84 $t_2$=19.76 | **133±(6)** $t_1$=31.27 $t_2$=15.46 | **114±(4)** $t_1$=14.23 $t_2$=5.58 | **147±(6)** $t_1$=23.47 $t_2$=31.63 $t_3$=37.83 |

**Table 6. 3:** Number of the Detected Outliers for the *Breast Cancer* Dataset

| *TopRatio* | *FindCBLOF* **(KM)** | *FindCBLOF* **(BKM)** | *FindCBLOF* **(PAM)** | **CCOD (KM,BKM)** | **CCOD (KM,PAM)** | **CCOD (BKM,PAM)** | **CCOD (KM,BKM,PAM)** |
|---|---|---|---|---|---|---|---|
| **10%** | 28±(4) | 31±(4) | 45±(3) | **55 ±(3)**<br>$t_1$=24.14<br>$t_2$=21.46 | **67±(4)**<br>$t_1$=30.83<br>$t_2$=17.39 | **79±(5)**<br>$t_1$=33.52<br>$t_2$=26.08 | **88±(6)**<br>$t_1$=37.21<br>$t_2$=35.34<br>$t_3$=28.66 |
| **15%** | 39±(2) | 48±(4) | 52±(3) | **63±(3)**<br>$t_1$=29.76<br>$t_2$=13.42 | **79±(5)**<br>$t_1$=33.21<br>$t_2$=12.27 | **83±(4)**<br>$t_1$=27.77<br>$t_2$=27.72 | **97±(7)**<br>$t_1$=35.62<br>$t_2$=27.18<br>$t_3$=26.42 |
| **20%** | 50±(3) | 62±(4) | 69±(3) | **83 ±(4)**<br>$t_1$=29.51<br>$t_2$=16.60 | **98±(6)**<br>$t_1$=32.00<br>$t_2$=19.33 | **104±(5)**<br>$t_1$=29.33<br>$t_2$=26.84 | **127±(8)**<br>$t_1$=40.30<br>$t_2$=32.50<br>$t_3$=30.35 |
| **25%** | 68±(3) | 79±(3) | 81±(4) | **97 ±(3)**<br>$t_1$=30.56<br>$t_2$=18.97 | **107±(5)**<br>$t_1$=29.91<br>$t_2$=18.16 | **118±(6)**<br>$t_1$=26.00<br>$t_2$=22.94 | **139±(8)**<br>$t_1$=37.16<br>$t_2$=31.40<br>$t_3$=21.00 |
| **30%** | 76 ±(3) | 84± (5) | 93±(4) | **116±(6)**<br>$t_1$=26.66<br>$t_2$=18.32 | **125±(7)**<br>$t_1$=28.77<br>$t_2$=17.75 | **139±(6)**<br>$t_1$=31.49<br>$t_2$=14.26 | **152±(7)**<br>$t_1$=44.62<br>$t_2$=35.35<br>$t_3$=32.72 |

**Table 6. 4:** Number of the Detected Outliers for the *UW* Dataset

| *TopRatio* | *FindCBLOF* **(KM)** | *FindCBLOF* **(BKM)** | *FindCBLOF* **(PAM)** | **CCOD (KM,BKM)** | **CCOD (KM,PAM)** | **CCOD (BKM,PAM)** | **CCOD (KM,BKM,PAM)** |
|---|---|---|---|---|---|---|---|
| **10%** | 10±(1) | 13±(2) | 6±(2) | **15±(1)** $t_1$=15.81 $t_2$=4.00 | **19±(2)** $t_1$=18.00 $t_2$=20.55 | **24±(2)** $t_1$=17.39 $t_2$=28.46 | **28±(2)** $t_1$=35.00 $t_2$=23.71 $t_3$=34.78 |
| **15%** | 23±(1) | 27±(2) | 15±(3) | **31±(3)** $t_1$=11.31 $t_2$=4.96 | **37±(2)** $t_1$=28.00 $t_2$=27.28 | **42±(2)** $t_1$=23.71 $t_2$=33.48 | **45±(3)** $t_1$=31.11 $t_2$=22.32 $t_3$=31.62 |
| **20%** | 34±(2) | 41±(3) | 26±(4) | **46±(2)** $t_1$=18.97 $t_2$=6.20 | **50±(3)** $t_1$=19.84 $t_2$=21.46 | **55±(4)** $t_1$=12.52 $t_2$=22.93 | **59±(2)** $t_1$=39.52 $t_2$=22.33 $t_3$=33.00 |
| **25%** | 53±(2) | 59±(3) | 42±(2) | **64±(3)** $t_1$=13.64 $t_2$=5.27 | **69±(4)** $t_1$=16.00 $t_2$=27.00 | **72±(3)** $t_1$=13.70 $t_2$=37.21 | **75±(4)** $t_1$=22.00 $t_2$=14.31 $t_3$=33.00 |
| **30%** | 62±(2) | 74±(3) | 51±(2) | **78±(3)** $t_1$=19.84 $t_2$=4.21 | **83±(3)** $t_1$=26.04 $t_2$=39.69 | **87±(3)** $t_1$=13.70 $t_2$=44.65 | **91±(5)** $t_1$=24.08 $t_2$=13.03 $t_3$=33.22 |

**Table 6. 5:** Number of Detected Outliers for the *Yahoo* Dataset

| *TopRatio* | *FindCBLOF* **(KM)** | *FindCBLOF* **(BKM)** | *FindCBLOF* **(PAM)** | **CCOD KM,BKM)** | **CCOD (KM,PAM)** | **CCOD (BKM,PAM)** | **CCOD (KM,BKM,PAM)** |
|---|---|---|---|---|---|---|---|
| **10%** | 35±(4) | 51±(4) | 31±(3) | **67±(3)** $t_1$=28.62 $t_2$=14.32 | **40±(3)** $t_1$=4.47 $t_2$=9.49 | **58±(3)** $t_1$=6.26 $t_2$=28.46 | **71±(4)** $t_1$=28.46 $t_2$=15.81 $t_3$=35.78 |
| **15%** | 43±(3) | 62±(4) | 40±(3) | **75±(4)** $t_1$=28.62 $t_2$=10.28 | **47±(4)** $t_1$=3.58 $t_2$=6.26 | **69±(2)** $t_1$=7.00 $t_2$=35.97 | **86±(5)** $t_1$=32.97 $t_2$=16.76 $t_3$=35.28 |
| **20%** | 64±(4) | 78±(5) | 57±(3) | **94±(5)** $t_1$=20.95 $t_2$=10.12 | **67±(2)** $t_1$=3.00 $t_2$=12.40 | **85±(3)** $t_1$=5.36 $t_2$=29.51 | **116±(7)** $t_1$=28.84 $t_2$=19.75 $t_3$=34.64 |
| **25%** | 81±(5) | 99±(4) | 72±(4) | **117±(3)** $t_1$=27.61 $t_2$=17.89 | **86±(3)** $t_1$=3.84 $t_2$=12.52 | **106±(2)** $t_1$=7.00 $t_2$=34.00 | **124±(3)** $t_1$=32.98 $t_2$=22.36 $t_3$=46.51 |
| **30%** | 94±(4) | 105±(6) | 87±(5) | **124±(7)** $t_1$=16.64 $t_2$=9.21 | **97±(3)** $t_1$=7.66 $t_2$=2.86 | **113±(3)** $t_1$=5.33 $t_2$=19.94 | **147±(6)** $t_1$=32.86 $t_2$=22.13 $t_3$=34.36 |

An interesting observation is that, the detection accuracy of each cooperative algorithm is mainly based on the original cooperative clustering model, where better clustering indicates better ability of discovering outliers. For example, for the *Yeast* dataset, we can see that the cooperation between KM and PAM, CC(KM,PAM), obtains clustering solutions of higher values of *F-measure* and *Purity* and lower values of *Entropy* and *SI* (Table 4. 5) than those of the CC(KM,BKM) and CC(BKM,PAM). Thus the corresponding CCOD(KM,PAM) detects up to 133 outliers compared to only 121 and 114 outliers by CCOD(KM,BKM) and CCOD(BKM,PAM), respectively, at *TopRatio*=30%. Also for the *UW* dataset, the CC(BKM,PAM) achieves better clustering than the other two cooperative models, thus the CCOD(BKM,PAM) discovers 87 outliers compared to 78 and 83 by the CCOD(KM,BKM) and CCOD(KM,PAM), respectively, at *TopRatio*=30%. The same interpretation of results is reported for both the *Breast Cancer* and the *Yahoo* datasets. A second interesting observations is that across the four datasets, the detection accuracy of the triple cooperation CCOD(KM,BKM,PAM) is much better than that of the other three pair-wise cooperative algorithms at different values of the *TopRatio* ranges from 10% to 30%. This better discovery of outliers is mainly based on the capability of obtaining better clustering solutions using the triple cooperative clustering than that of the pair-wise cooperative clustering models.

## 6.3 Enhancing Clustering Quality

In order to illustrate the significance of the discovered outliers using both the non-cooperative and cooperative detection methods, the clustering performance of the KM, BKM, and PAM is compared to that of the cooperative models using the *SI* index before and after removing the discovered set of outliers at variable number of top ratios for *Yeast*, *Breast Cancer*, *UW*, and *Yahoo* datasets. In figures 6.1-6.4, the *SI* at 0% means the value of the *SI* index with the existence of outliers. It can be shown that the ability of detecting more outliers in the datasets using the notion of cooperative clustering enhances the clustering quality measured by lower values of the *SI* index. For example, the CCOD(KM,BKM) achieves reduction in the *SI* index (compared to the value of *SI* at 0% *TopRatio*) of up to 56% compared to only 44% by KM and 36% by BKM (*Yeast* dataset), CCOD(KM,PAM) achieves reduction in the *SI* index of up to 61% compared to 40% by KM and 34% by PAM (*UW* dataset), CCOD(BKM,PAM) improves *SI* with a percentage of up to 62% compared to BKM (33%) and PAM (34%) (*UW* dataset) at *TopRatio*=30% after removing the discovered set of outliers. The CCOD(KM,BKM,PAM) achieves up to 62% improvement in the *SI* index for *Yeast dataset,* up to 60% improvement for *Breast Cancer* dataset, 75% improvement for *UW dataset,* and up to 60% improvement for *Yahoo* dataset at *TopRatio*=30% compared to that of the non-cooperative KM, BKM, and PAM approaches after removing the discovered set of outliers.

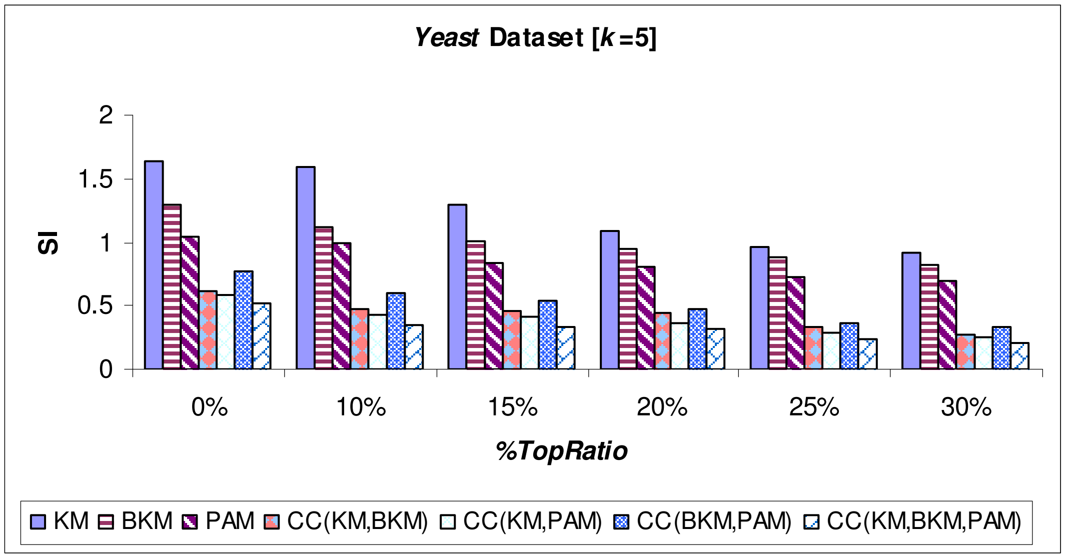


**Fig. 6. 1.** Performance Evaluation before and after Deleting Outliers [*Yeast*]

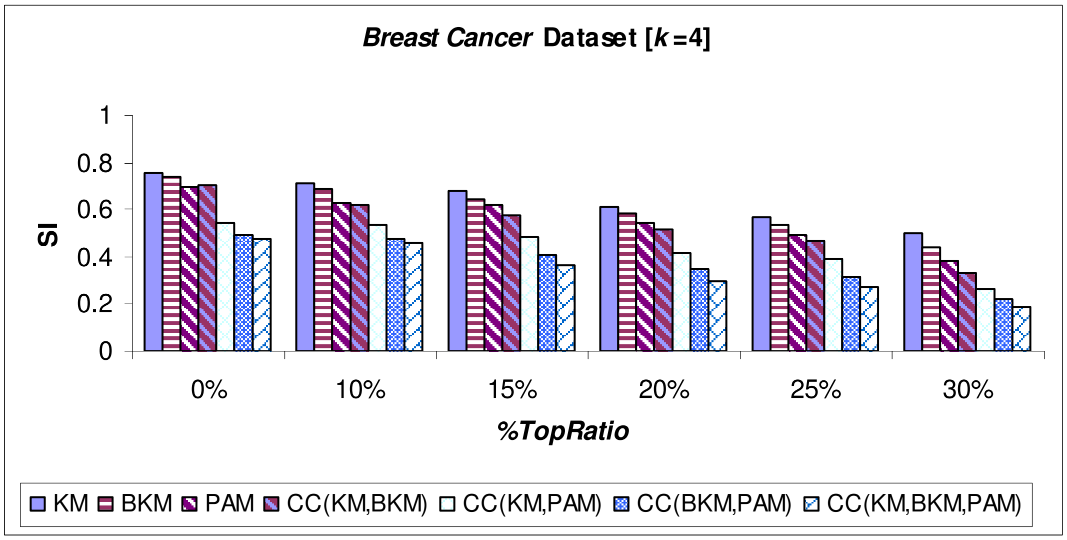


**Fig. 6. 2.** Performance Evaluation before and after Deleting Outliers [*Breast Cancer*]

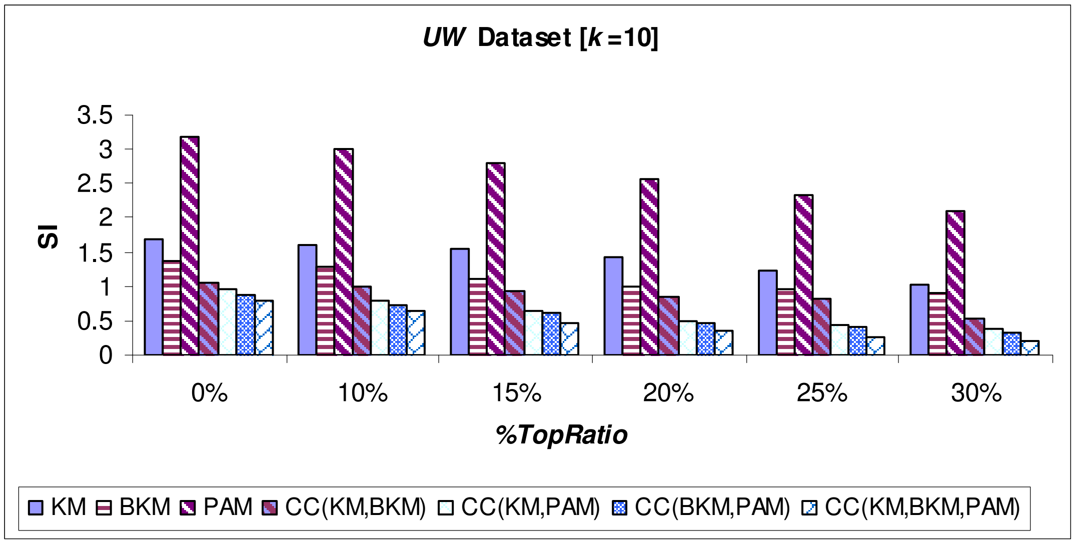


**Fig. 6. 3.** Performance Evaluation before and after Deleting Outliers [*UW*]

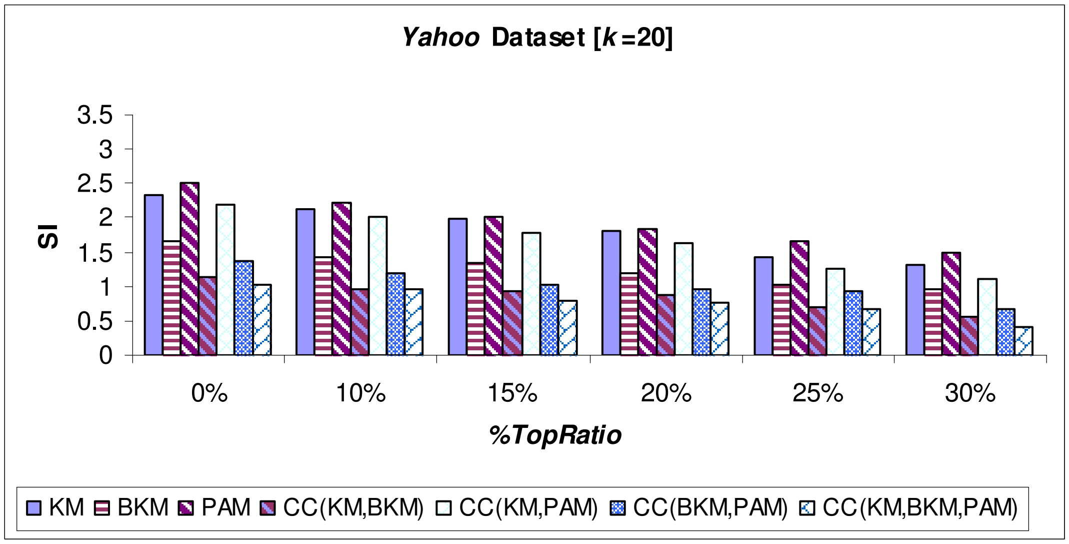


**Fig. 6. 4.** Performance Evaluation before and after Deleting Outliers [*Yahoo*]

## 6.4 Discussions

This chapter analyzes a new outlier detection method called Cooperative Clustering Outliers Detection (CCOD). It provides efficient outlier detection and data clustering capabilities in the presence of outliers. Experimentally, the CCOD is applied on both gene expression datasets and text documents datasets. Undertaken experimental results indicate that CCOD works better than the traditional clustering-based outlier's detection techniques with better improvement in the clustering quality after removing the discovered set of outliers.

# Chapter 7
# Cooperative Clustering in Distributed Super-Peer P2P Network

Traditional data clustering technologies have been fundamentally based on centralized operation; data sets were of small manageable sizes, and usually reside on one node that belongs to one organization. Today, data is of enormous sizes and is usually located on distributed nodes; examples are the World Wide Web (WWW) and distributed gene expression repositories. This has created a need for performing clustering in distributed environments. Distributed clustering solves two problems: infeasibility of collecting data at a central node, due to either technical or privacy limitations, and intractability of traditional clustering algorithms on huge data sets.

In distributed data clustering environments, adopting a flat node distribution topology can affect on the scalability of the network. To address the problem of modularity, flexibility, and scalability, a dynamic hierarchical two-tier architecture and model for cooperative clustering in distributed super-peer P2P network is presented. The proposed model is called Distributed Cooperative Clustering in super-peer P2P networks (DDCP2P). It involves a hierarchy of two layers of P2P neighborhoods. In the first layer, peers in each neighborhood are responsible for building local cooperative sub-clusters from the local data sets that they are responsible for. The main objective is to allow nodes in a network to first form independent partitioning of local data, and then they send their local clustering to a super-peer in their neighborhood to aggregate the local models from its ordinary peers (workers). Each node sends only cluster representatives to its super-peer in a form of sub-cluster's centroids extracted from the local cooperative clustering. This summarized view of local data at each node minimizes the exchange of information between nodes and their super-peers. As we move up to the next layer, sub-clusters are merged at each super-peer and at the root of the hierarchy one global clustering can be derived. The distributed cooperative clustering approach finds globally-optimized clusters. This approach achieves significant improvement in the global clustering solutions without the cost of centralized clustering.

This chapter is organized as follows. Section 7.1 gives an overview of the current approaches of distributed clustering. Section 7.2 discusses the proposed hierarchical two-tier super-peer network. The distributed cooperative clustering model is presented in section 7.3. Both the computational and communication complexity of the distributed cooperative model are discussed in section 7.4. Finally some discussions and conclusions are presented in section 7.5.

## 7.1 Overview

Huge data sets are being collected daily in different fields; e.g. retail chains, banking, biomedicine, astronomy, and many others, but it is still extremely difficult to draw conclusions or make decisions based on the collective characteristics of such disparate data. Two main approaches for performing distributed clustering can be identified:

- A common approach is to perform local clustering at each node to generate a local model. Then all local models can be transmitted to a central node that aggregates them together into one global model [76],[77],[82]. While this approach many not scale well with the number of nodes, it can be considered as a better alternative than pooling the data to one central node.
- A better strategy is that each node selects a small set of representative objects and transmits them to a central node, while combines the local representatives into one global representative of the whole data set. Then data clustering can be carried out on the global representatives [76],[79].

The two previous approaches involve one central node to facilitate the distributed clustering process. A more departing approach does not involve centralized operation, and this belongs to the peer-to-peer (P2P) class of algorithms. P2P networks can be unstructured and structured. Unstructured networks are formed arbitrarily by establishing and dropping links over time, and they usually suffer from flooding of traffic to resolve certain requests. Structured networks, on the other hand, make an assumption about the network topology and implement a certain protocol that exploits such a topology. In P2P networks, nodes (peers) communicate directly with each other to perform the clustering task [70],[71],[113]. Communication in P2P networks can be very costly if care is not taken to localize traffic, instead of relying on flooding of control or data messages.

To pave the way for discussion of the different concepts and strategies of super-peer P2P networks and for the proposed distributed cooperative clustering model and architecture, some terminologies and notations that are used throughout this chapter are summarized in Table 7.1.

**Table 7.1**: Distributed Clustering Symbols and Notations

| **Symbol** | **Definition** |
|---|---|
| $n_Q$ | Number of neighborhoods |
| $Q_i$ | The *i*th neighborhood |
| $\lvert Q_i \rvert$ | Number of peers in the neighborhood $Q_i$ |
| $SP_i$ | Super peer of a neighborhood $Q_i$ |
| $N_p$ | The *p*th peer in a network |
| $Sb\rvert_{N_p}$ | Local set of sub-clusters at node $N_p$ |
| $n_{Sb}\rvert_{N_p}$ | Number of local sub-clusters at node $N_p$ |

## 7.1.1 Pure P2P Networks

A pure P2P overlay network can be shown as an undirected graph, where the vertices correspond to nodes in the network, and the edges correspond to open connections maintained between the nodes. Two nodes maintaining an open connection between them are known as *neighbors*. Messages may be transferred in either direction along the edges. For a message to travel from one node to another node, it must travel along a path in the graph. The length of this traveled path is known as the number of *hops* taken by the message. Similarly, two nodes are said to be *h* "hops apart" if the shortest path between them has length *h*. A pure P2P topology is shown in Fig. 7. 1.

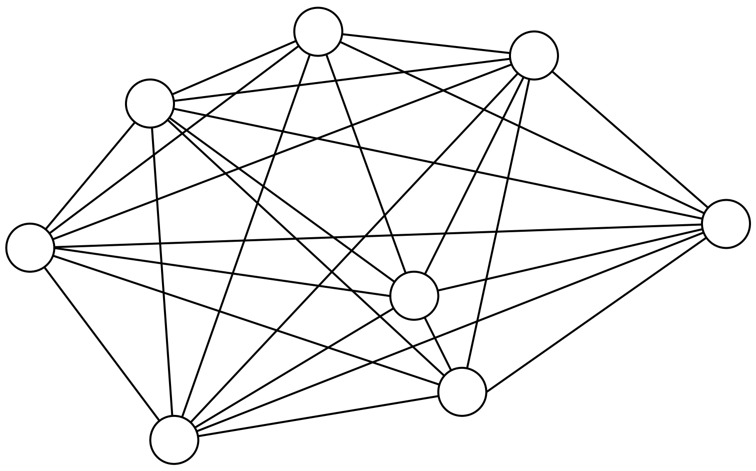


**Fig. 7. 1.** Pure P2P Topology

### 7.1.2 Super-Peers P2P Networks

At this point we will refer to facilitator node as a super-peer (*SP*) and a worker node as ordinary peer (*OP*). Many P2P systems use stronger peers (super-peers) as a representative of other ordinary nodes. A *super-peer* is a node in a peer-to-peer network that operates both as a facilitator to a set of workers (*ordinary peers*), and as an equal in a network of super-peers.

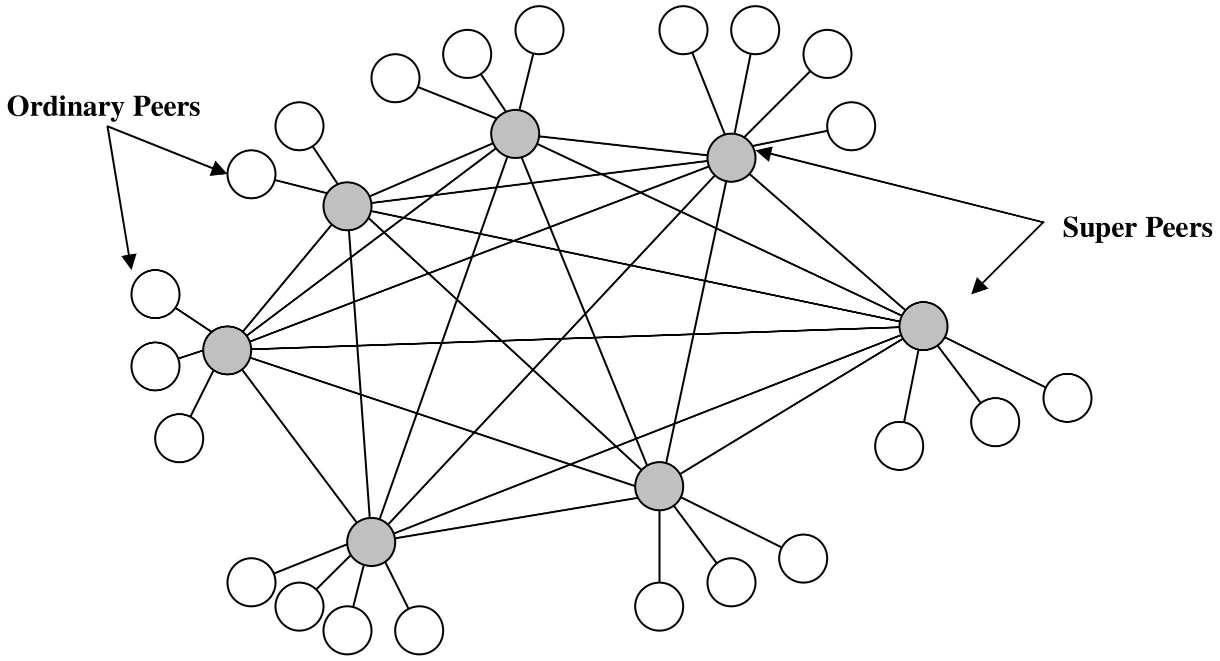


**Fig. 7. 2.** Super-Peers and Ordinary Peers

A "*super-peer network*" is simply a pure P2P network consisting of super-peers and their worker nodes. Each worker node is only connected to its super-peer. We will refer to systems that obtain super-peers as a *Super-peer based P2P system.* A pure P2P network is actually a "degenerate" super-peer network where neighborhood size is 1 (i.e. every node is a super-peer with no workers).

### 7.1.3 Neighborhoods

*A Neighborhood*, *Q*, is a group of workers (ordinary peers) forming a logical unit of isolation in an otherwise unrestricted open P2P network that are mapped to the same super-peer. Peers in a neighborhood cannot communicate with peers in other neighborhoods. Communication between neighborhoods is achieved through their respective super-peers.

$$\textit{Neighborhood}\ Q_i=(SP_i,\ OP_j\ ,j=0,1,..,|Q_i|-1) \qquad (7.\ 1)$$

Examples of *super-peer*-based networks include FastTrack [114], SODON [115], and ECSP [116]. FastTrack is a so-called second generation P2P networks. It uses super-peers to improve scalability. FastTrack was the most popular file sharing network, being mainly used for the exchange of music. SODON (Self-Organized Download Overlay Network) is a distributed multi-source content distribution system. It uses super-trackers to maintain neighborhood state information and guide peers to other neighborhood members for piece exchange. Finally, the ECSP is an *Efficient Clustered Super-Peer* architecture for P2P Networks. These architectures involve a super-peer as a representative of each neighborhood to communicate with other super-peers to obtain the global model at the root peer.

## 7.2 Two-Tier Hierarchical Overlay Super-peer P2P Network

The proposed distributed architecture deviates from the standard definition of P2P networks, which typically involve loose structure (or no structure at all), based on peer connections that are created and dropped frequently. The DCCP2P model on the other hand, is based on a two-tier hierarchy structure that is designed up front, upon which the peer network is formed. Although we focus on a two tier-hierarchy of super-peer neighborhoods, the architecture can be extended to a general tier hierarchy in a straightforward manner. In the two-tier architecture, the lower layer represents neighborhoods of peers and the higher layer connects representatives from each neighborhood (i.e. super-peers) to ensure good global connectivity. The two-tier hierarchical super-peer P2P architecture is illustrated in Fig. 7.3.

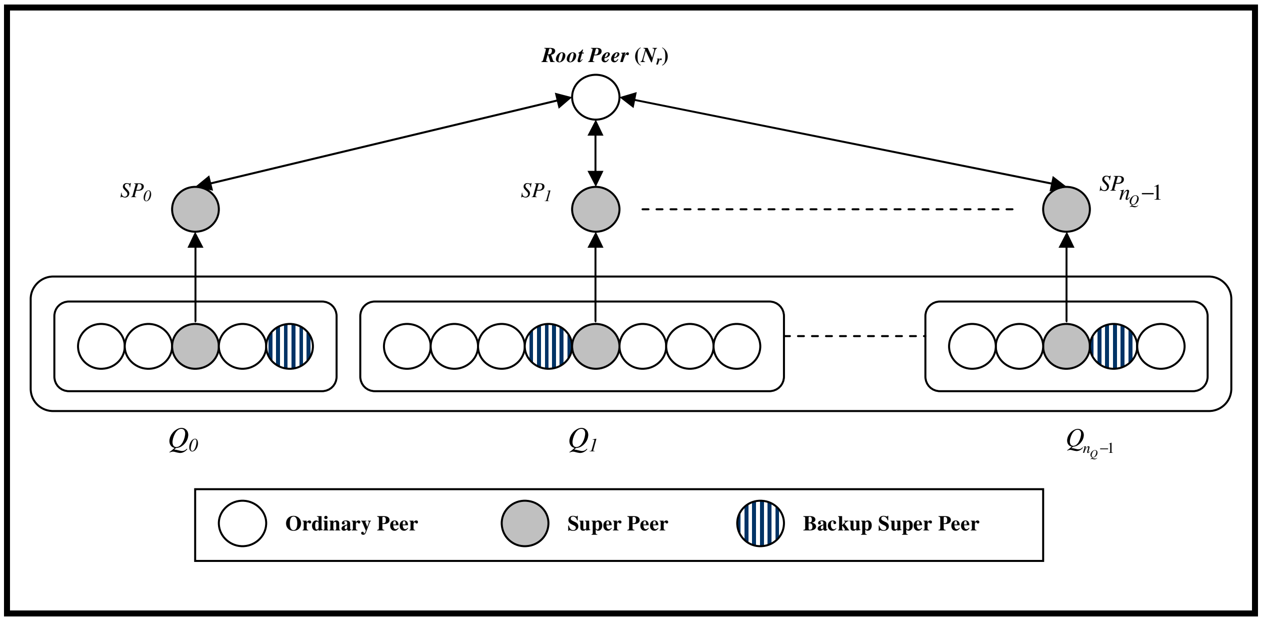


**Fig. 7. 3**. Two-tier Hierarchical Super-peer P2P Network

The following neighborhood properties are enforced in the proposed architecture:

- Each node receives information (in forms of messages) only from its super-peer
- Each node is connected only to either super-peer or peers in the same neighborhood
- A set of neighborhoods, $\boldsymbol{Q}=\{Q_i\}, i=0,1,..,n_Q-1$ covers the first overlay network
- Neighborhoods do not overlap: $\forall i, j \neq i : Q_i \cap Q_j = \varnothing$
- A node must belong to some neighborhood: $\forall N_p, p = 0,1,.., P-1 : N_p \in \text{some}\, Q_i$

The number of neighborhoods in the networks $n_Q$ depends on the number of partitions generated by the proposed *peer-clustering* algorithm described next. At one extreme of the resulting clustering solution when $n_Q$=1 there is only one neighborhood that contains all peers. On the other hand, when $n_Q$=P, where P is the total number of nodes in the network, there are P neighborhoods, each containing one peer only. In between the value of $n_Q$ is set to a value that determines the number of neighborhoods as a result of the peer-clustering algorithm and consequently the size of each neighborhood. In addition, *super-peers* are selected from ordinary peers in a neighborhood to act as cluster leaders and service providers using the *super-peer selection* algorithm as illustrated in sub-section 7.2.2.

### 7.2.1 Peer-Clustering Algorithm

The first problem in flat peer-to-peer networks is that random connections among peers in the network make it possible for geographically far-distance peers to connect and consequently an increase in the connection time is obtained. The second problem is that a large number of peers can cause bottleneck in the network bandwidth. It is possible to obviate these problems by using a hierarchical architecture to construct large-scale P2P overlay network in a form of a structured architecture. In such an approach, the network is composed of several (or tens of) clusters of peers. Each cluster behaves as a neighborhood of ordinary peers and it selects a super-peer as a representative peer for the cluster. When peers in P2P topology form clusters (representing, for example, thematic partitioning), then the whole topology is arranged in hierarchical structure of neighborhoods and super-peers. Current approaches for clustering P2P network use a graph-based partitioning techniques which is of order $O(P^2)$ [116] which is extremely expensive for large number of nodes in the network.

For simplicity, we propose a peer-clustering algorithm that uses a variant of the *k*-medoids clustering algorithm [14] to cluster the pure P2P network into a set of $n_Q$ neighborhoods (clusters) such that the total communication cost is minimized as shown in Fig. 7. 4. We assume that the P2P network is formed as a graph where its nodes are the set of peers and its edges are the set of connections between peers. The weight of each edge is the geographical proximity distance between nodes; this is simulated by measuring the similarity of the local objects. Such that each node $N_p$, $p$=0,..,P-1 is represented by the centroid of its local dataset $X_p$. We adopted the *cosine* similarity (Eq. (2.5)) to calculate the similarity between nodes, where similarity close to 1 means close peers in the network.

```
Algorithm: Peer-Clustering (P, {$N_p$},$n_Q$)
Input: Number of peers in the network P, set of peers {$N_p$}, and number of neighborhoods (clusters) $n_Q$.
Output: a set of $n_Q$ neighborhoods.
Begin
    Step1: Partition the set of P nodes into $n_Q$ initial clusters; we determine $n_Q$ initial medoids (seed points)
           first by randomly choosing $n_Q$ nodes (represented by their centroids) locating to act as the $n_Q$
           cluster's representatives.
    Step2: Proceed through the list of nodes in the network, assigning a node to the specific cluster whose
           medoid is the shortest in terms of proximity distance. Re-computation of the medoid is done for
           the cluster having gained a new node as in the traditional PAM algorithm (Fig. 2. 3)
    Step3: Repeat Step 2 until no more assignments take place.
Return $n_Q$ neighborhoods with the corresponding representatives (medoids)
End
```

**Fig. 7. 4.** Peer-Clustering Algorithm

The method attempts to minimize the sum of the within cluster variances where peers are organized into groups such that peers in the same group are topologically close to each other.

### 7.2.2 Selection of Super Peers (SP)

The *SP* selection problem is highly challenging because, in the pure P2P network, a large number of *SP* must be selected from a huge and dynamically changing network in which neither the peer's characteristics nor the network topology are known a prior [117]. Often simple strategy such as random selection is used, where each *OP* chooses a random *SP* from the whole network. The selected super-peer is of maximally distant from other super-peers. Although this technique is simple, it does not deal well with the heterogeneity of the participating peers both in terms of dynamic capabilities, a content similarity, and geographical allocations. In our two-tier overlay super-peer network, if we can group peers according to their proximity, we can further reduce average cost for message delivery between peers and increase the network bandwidth usage by using the proposed *super-peer selection* algorithm. Our criteria for selecting a peer form a neighborhood of ordinary nodes to become a candidate super-peer are based on the following factors:

- A super peer should not have limited capabilities that can cause bottlenecks capacities.
- A super peer is centrally distant from all existing ordinary peers in a neighborhood.

The selected super-peers should be evenly distributed in a neighborhood such that each cluster has a super-peer at its center. However, this architecture is more sensitive to the failure of super peers and faces similar problems of central servers. Thus, in order to enhance the reliability and scalability of the system, the super-peer selection algorithm allows each neighborhood to select a backup peer, which copies the entire neighborhood (cluster) state information periodically from the super-peer.

```
Algorithm: Peer-Selection ($Q_i$)
Input: a neighborhood of peers, $Q_i$
Output: Super-peer SP and a back-up super-peer bSP
Begin
  Step1: Select peer SP from $Q_i$ such that SP has a central distance from all peers in the neighborhood,
         this super peer is selected as the cluster medoid that is returned by the peer-clustering algorithm
         (F.g. 7.4).
  Step2: Select the second peer with the closest distance to the central super-peer as a back-up super-peer,
         bSP, and copy the entire neighborhood state information into it.
Return SP and bSP
End
```

**Fig. 7. 5.** Super-Peer Selection Algorithm

In general, the entire P2P network is structured into two-tier hierarchical structure which clearly separates the super-peers neighborhoods from the ordinary-peers neighborhood such that the overall communication cost is minimized. In addition, this hierarchical structure maintains data privacy between peers in the same neighborhood. Where ordinary peers only establish a connection with its super peer and the local datasets at each ordinary peer is capsulated by its peer. The only form of information that is transferred to the super-peer is a cluster summary (i.e. sub-clusters centroids). First layer is a clustered P2P network of P nodes, which is partitioned into a set of $n_Q$ neighborhood using the peer-clustering algorithm. A connection is only established between peers in a neighborhood and its super-peer. The second layer is a network of super-peers nodes connected to the root peer. The root peer is responsible of aggregating the local models allocated at each super-peer and building the global model out of the whole data set as if it was transferred into a central node. This hierarchical architecture reduces flooding problems usually encountered in large P2P networks. The construction of the proposed two-tier architecture is illustrated in Fig. 7. **6**.

```
Algorithm: Two-Tier Super-peer Network-Construction({$X_p$},{$N_p$}, P, $n_Q$)
Input: Local datasets {$X_p$}, set of nodes {$N_p$}, number of nodes P, and number of neighborhoods $n_Q$.
Output: Two-tier Network (T2N)
Initialization: T2N={ }, allocate $N_r$ as a root peer
Begin
        Step1: Let $c_p$ be the centroid of the local dataset $X_p$. Represent each node $N_p$ with the
                corresponding centroid $c_p$.
        Step2: Neighborhoods {$Q_i$}= Peer-Clustering(P,{$N_p$},$n_Q$)
                Add each neighborhood $Q_i$ to the first layer of T2N
        Step3: For each neighborhood $Q_i$, $i$=0,1..,$n_Q$-1
                {$SP_i$, $bSP_i$}=Peer-Selection($Q_i$)
                - Designate $SP_i$, $bSP_i$ as the super-peer and backup super-peer for the neighborhood $Q_i$
                - Add each super-peer $SP_i$ ,$i$=0,1,..,$n_Q$-1 to the second layer of T2N
                - Connect peers in the neighborhood $Q_i$ to their super peer $SP_i$
               End
        Step4: Connect the selected super-peers to the root peer $N_r$ at the top-level of the hierarchy.
        Return (T2N)
End
```

**Fig. 7. 6.** Two-Tier Super-peer Network Construction

## 7.3 Distributed Cooperative Clustering in a Hierarchical Super-Peer P2P Network (DCCP2P)

In the centralized cooperative clustering CC approach, the dataset is centralized and the cooperative clustering process is performed on the data. The DCCP2P is a distributed globally-optimized clustering technique. It is a centroid-based clustering algorithm, where a set of centroids is generated to describe the clustering solution both on the local level and at the global level. The distributed cooperative algorithm is a distributed variant of the cooperative clustering model (chapter 3) in a distributed super-peer P2P network that involves a series of steps including:

- Generation of local sub-clusters at each peer
- Each ordinary peer sends representatives of its local sub-clusters to its super-peer in the local neighborhood
- Each super peer builds one cooperative solution from the local representatives it receives and then it transfers the cooperative solution to the root peer, and finally,
- At the root peer, building the global clustering solution is taken place

### 7.3.1 Building Local Models

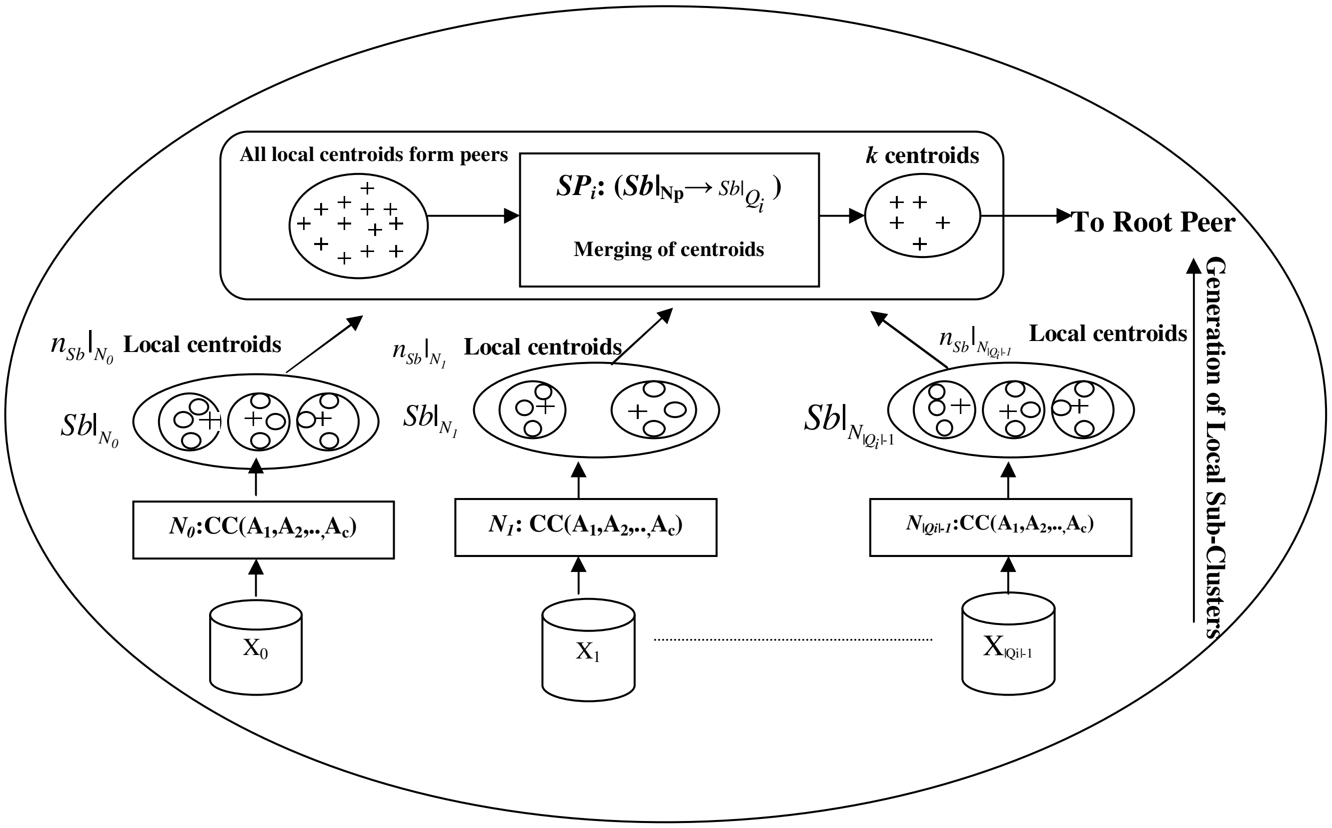
In the DCCP2P, each node $N_p$, $p$=0,..,P-1, obtains a set of local sub-clusters $Sb|_{Np}$ from the local data set $X_p$ it owns using the notion of cooperative sub-clusters memberships (Eq. (3.1)). This set acts as the agreement between the $c$ clustering algorithms on clustering the local data set $X_p$ into a set of $k$ clusters. Let $n_{sb}|_{Np}$ be the number of local sub-clusters at node $N_p$. One strives to characterize the distributed data distribution *via* clustering using high-level information that provides a trade-off between privacy and quality. Thus each node in a neighborhood $Q_i$ sends representatives of its sub-clusters to its super-peer in the same neighborhood. If each peer sends its local histograms (as in centralized cooperative clustering ) to its super-peer, the size of the transmitted information will be of order ($n_p^2$) where $n_p$ is the number of local objects at peer $N_p$, which is extremely large for huge datasets. In addition the super-peer will perform an additional overhead of accumulating the pair-wise similarities between objects from different peers. Thus, in order to minimize the communication cost taken by sending messages between peers and their super-peer, we assume that each sub-cluster $Sb_j$ is represented by a representative, $c_j$ a sub-cluster centroid, so local models at each node are represented by a set of centroids instead of transmitting the whole histograms. Thus the DCCP2P is considered as an approximate distributed clustering algorithm. The communication between peers and super-peer is facilitated through the *send* and *receive* operations (messages). The size of the messages is computed by the size of the representatives sent from each peer.

### 7.3.2 Global Model

The super-peer of a neighborhood $Q_i$ is responsible of building an aggregated cooperative clustering solution of the local models it receives from the ordinary peers in the same neighborhood. The underlying aggregation process comprises of the following steps:

- Each peer sends a set of sub-clusters centroids $\{c_l\}|_{Np}$ to its super-peer, $SP_i$, such that the super peer $SP_i$ receives order of $O(|Q_i|*\{c_l\}|_{Np})$ centroids where $|Q_i|$ is the size of the neighborhood $Q_i$.
- The goal is to obtain a number of clusters $k$ at the super-peer from the set of received centroids, so the $SP_i$ merges centroids such that the overall homogeneity of clusters in the neighborhood $Q_i$ is maximized, where the two local sub-clusters $Sb_l$ and $Sb_j$ with the closest centroids $c_l$ and $c_j$ are merged. The new generated centroid is calculated as $(c_l*|Sb_l|+c_j*|Sbj|)/(|Sb_l|+|Sb_j|)$.
- The merging step is repeated at the super-peer until the number of centroids equals $k$.

The generation of cooperative centroids at super-peer $SP_i$ within a neighborhood $Q_i$ is shown in Fig. 7.7.



All local centroids form peers
$SP_i$: $(Sb|_{Np} \rightarrow Sb|_{Q_i})$
Merging of centroids
$k$ centroids
To Root Peer
Generation of Local Sub-Clusters
$n_{Sb}|_{N_0}$ Local centroids
$n_{Sb}|_{N_1}$ Local centroids
$n_{Sb}|_{N_{|Q_i|-1}}$ Local centroids
$Sb|_{N_0}$
$Sb|_{N_1}$
$Sb|_{N_{|Q_i|-1}}$
$N_0$:CC($A_1,A_2,..,A_c$)
$N_1$: CC($A_1,A_2,..,A_c$)
$N_{|Qi|-1}$:CC($A_1,A_2,..,A_c$)
$X_0$
$X_1$
$X_{|Qi|-1}$

**Fig. 7. 7.** Cooperative Centroids Generation within a Neighborhood $Q_i$ at Super-peer $SP_i$

Once a neighborhood $Q_i$, $i$=0,1,..,$n_Q$-1, converges to a set of cluster centroids at its super-peer $SP_i$, those centroids are acquired by the root-peer in order to build the global model from all neighborhoods. The root peer receives a set of $k$ centroids from each super-peer such that $k*n_Q$ centroids are available for merging. The root peer follows the same merging procedure as super-peers; it merges two clusters with the closest centroids until the number of global centroids equals $k$. These final $k$ centroids represent the global model of the distributed cooperative clustering model. The Distributed cooperative clustering in the two-tier hierarchical super-peer network is illustrated in Fig. 7. 8.

```
Algorithm: DCCP2P({$A_c$},$k$,{$X_p$},{$N_p$}, P, $n_Q$)
Input: $c$ clustering techniques {$A_i$}, number of clusters $k$, local datasets {$X_p$}, set of nodes {$N_p$}, number
        of nodes P, and number of neighborhoods $n_Q$.
Output: global $k$ centroids
Begin
T2N=Two-Tier Super-peer Network-Construction ({$X_p$},{$N_p$}, P, $n_Q$), $N_r$ is the root peer
        For each neighborhood $Q_i \in$ T2N, $i$ =0,1,..,$n_Q$-1
            For each peer $N_p$, $p$=0,1,..,|$Q_i$|-1 in the neighborhood $Q_i$
                - Perform the $c$ clustering algorithms, {$A_c$} synchronously on the local dataset $X_p$.
                - A set $Sb|_{Np}$ is obtained using Eq.(3.1), then each local sub-cluster is represented by its
                  centroid such that $n_{sb}|_{Np}$ centroids are available at node $N_p$.
                - Send the local centroids to the super-peer $SP_i$.
            End
            - The super-peer $SP_i$ recursively merges the received centroids into $k$ centroids, such that the
              two sub-clusters |$Sb_l$|, |$Sb_j$| with the closest centroids $c_l$ and $c_j$ are merged. The new
              generated centroid is calculated as ($c_l$*|$Sb_l$|+$c_j$*|$Sbj$|)/(|$Sb_l$|+|$Sb_j$|).
            - Send the merged $k$ centroids to the root peer $N_r$
        End
Recursively merge the received $k$*$n_Q$ centroids into $k$ centroids in the same scenario as super-peers
Return the $k$ global centroids.
End
```

**Fig. 7. 8.** DCCP2P Clustering

Due to the dynamic environment of peer-to-peer networks, we also allow peers to leave and join the two-tier super-peer network using two novel algorithms, the peer joining and peer leaving algorithms.

### 7.3.3 Peer-leaving

The P2P network is known of its dynamic nature, thus the overlays of super-peers or ordinary peers may be broken by peers' ungraceful departures. This departure of peers causes incompleteness of the two-tier structure. When peer leaves a network, two possible alternatives are taken into consideration:

- **A super-peer *$SP_i$* fails or simply leaves**: all its ordinary peers become temporarily disconnected until they can find a new candidate super-peer to connect to. The new super-peer is a selected backup peer that takes over the idle super-peer, which is assigned by the super-peer selection algorithm. Thus once a super-peer fails, automatically the backup peer takes place.
- **An ordinary peer *$N_p$* leaves or turns offline**: let the set of local data objects of the leaving peer is identified as $\{X_p^-\}$.
    - The corresponding super-peer frees the resources corresponding to its connection to the leaving peer after observing a missing signal from the departing peer
    - The super-peer de-allocates the datasets assigned to the failed peer
    - Then it updates the received set of local models by removing the received set of centroids from the idle node. Then it remerges the new set of centroids
    - The super-peer sends a new copy of the updated cooperative centroids to the root peer to update its global model in the same manner

### 7.3.4 Peer-Joining

Peers are allowed to join the neighborhood in which it is closest to its super-peer among all super-peers. As a result, all peers in a neighborhood are proximate in the underlying network topology. If there is more than one neighborhood, the new peer $N_p$ receives a list of all super-peers. From these super-peers, the new arriving peer selects the nearest super-peer and joins its neighborhood. Assume the incremental datasets associated with the joining peer as $\{X_p^+\}$. The new peer performs the cooperative membership generation procedure and obtains a new set of sub-clusters. Each sub-cluster is represented with the corresponding centroid. Then $N_p$ sends the generated centroids to the corresponding selected super-peer. The super peer follows the following steps:

- It starts remerging the set of sub-clusters based on the new added set of centroids
- It sends an update message with the newly generated centroids to the root peer to update its global model

## 7.4 Complexity Analysis

The complexity of the DCCP2P model is divided into two parts, the computational complexity ($T^{comp}$) and the communication complexity ($T^{comm}$).

### 7.4.1 Computation Complexity

Assume the whole data set size for all nodes is X. Data is equally divided among nodes, so each node $N_p$ holds $n_p$=|X$_p$|=|X|/P local data objects. At the first layer, we have $n_Q$ neighborhoods each with variables sizes. In each neighborhood $Q_i$, each node performs the cooperative clustering that involves $c$ clustering techniques each of computational complexity time equals $T^{Ac}$, and then it needs additional $n_p$ operations to find the set of sub-clusters. Thus node $N_p$ in neighborhood $Q_i$ needs computational complexity time $T^{Np}= max(T^{Ac})+n_p$. Assume the super-peer $SP_i$ of a neighborhood $Q_i$ receives $n_{sb}$ centroids from all nodes, it performs $O(d*n^2_{sb})$ operations for merging the set of sub-clusters, where $d$ is the dimension of each centroid. The total computational complexity within neighborhood $Q_i$ is defined as:

$$T^{comp}(Q_i) = O(\max_{p=0,..,|Q_i|-1}(T^{Np})) + O(d * n^2{}_{sb}) \tag{7. 2}$$

Since each neighborhood's computations are done in parallel with other neighborhoods then the total computational complexity at the first layer of the network is:

$$T^{Comp}(Layer1) = \max_{i=0,..,n_Q-1}(T^{comp}(Q_i)) \tag{7. 3}$$

At the root peer, a total of $n_Q*k$ centroids are received, and it needs a total of $d*n_Q*k*(n_Q*k$ -1)/2 operations for obtaining only a set of $k$ global centroids. The total computational complexity of the second layer is defined as:

$$T^{Comp}(Layer2) = O(d * \ (n_Q * k)^2) \tag{7. 4}$$

The total computational cost is calculated as:

$$T^{Comp} = T^{Comp}(Layer1) + T^{Comp}(Layer2) \tag{7. 5}$$

### 7.4.2 Communication Cost

The communication cost can be divided into two parts, *Intra-Neighborhood Messaging Cost* and *Inter-Supper-peers Messaging Cost*. The *Intra-Neighborhood Messaging Cost* is calculated within a neighborhood $Q_i$ of $|Q_i|$ peers at the fist layer of the network between peers and their super-peer, and the *Inter-Super-peers Messaging Cost* is computed based on the communication cost between a super-peer and the root peer at the second layer of the network.

At the first layer, for number of clusters $k$, every peer $N_p$ in neighborhood $Q_i$ sends $n_{sb}|_{Np}$ messages to its super-peer $SP_i$, and each message of size $d$. Then the communication complexity for a neighborhood $Q_i$ is:

$$Intra-Neighborhood\ Messaging\ Cost\ (T^{comm}(Q_i)) = \sum_{p=0}^{|Qi|-1} (n_{sb}\,|_{N_p}) * d \qquad (7.\ 6)$$

We can see that the *Intra-Neighborhood Messaging cost* is greatly affected by the size of neighborhood and consequently the number of generated sub-clusters at each node.

The total communication requirement for the first layer is:

$$T^{comm}(Layer1) = \sum_{i=0}^{n_Q-1} (T^{comm}(Q_i)) \qquad (7.\ 7)$$

At the root peer, each super-peer sends *k* messages each of size *d*. Then the communication cost at the second layer is calculated as:

$$Inter\text{-}Supper\text{-}peers\ Messaging\ Cost\ (T^{comm}(Layer2)) = k * n_Q * d \qquad (7.\ 8)$$

The total communication cost is calculated as:

$$T^{Comm} = T^{Comm}(Layer1) + T^{Comm}(Layer2) \qquad (7.\ 9)$$

## 7.5 Discussions

In this chapter, the Distributed Cooperative Clustering (DCCP2P) in super-peer P2P networks architecture and model were presented. DCCP2P aims to computing a single set of clusters across all nodes in the network, and addresses scalability and modularity through the concept of hierarchical two-tier super-peer network. Using the DCCP2P model, we can partition the problem into a modular way, solve each part individually, and then successively combine solutions to find a global solution. By developing this approach, we avoid two main problems in the current state of art of distributed data clustering (1) we avoid high communication cost usually associated with a structured fully connected network, and (2) we avoid uncertainty in the network topology usually introduced by unstructured P2P networks. Experiments performed on the distributed cooperative clustering model in the next chapter show that we can achieve comparable results to centralized cooperative clustering with high gain in speedup with large number of nodes, showing the scalability of the distributed DCCP2P model.

# Chapter 8
# Cooperative Clustering in Super P2P Networks: Experimental Analysis

In this chapter, three of the well known clustering techniques namely, KM, BKM, and PAM are invoked. Detailed discussions of these algorithms are presented in section 2.1.4. We will refer to the distributed cooperation between KM and BKM as DCCP2P(KM,BKM), the distributed cooperation between KM and PAM as DCCP2P(KM,PAM), distributed cooperation between BKM and PAM as DCCP2P(BKM,PAM), and finally the distributed cooperation between the three adopted algorithms together as DCCP2P(KM,BKM,PAM). Experimental results on the distributed cooperative clustering models in super-peer P2P networks show that we can achieve good speedup with large number of nodes, showing the scalability of the distributed cooperative models in terms of number of nodes. The rest of the chapter gives detailed description of the experiments and provides interpretation of results and discussions on their implications. Both internal and external quality measures (Section 2.1.3) are called for to test the performance of the generated global model.

## 8.1 Experimental Setup

A simulation environment was used to evaluate the cooperative clustering model on distributed super-peer P2P network. During simulation, data was partitioned randomly over all nodes of the network. The number of clusters *k* was specified to the *c* algorithms such that it corresponds to the actual number of classes in each dataset.

## 8.2 Data Sets

Experiments on large number of nodes and different number of neighborhoods require a large number of data to keep the size of each local datasets reasonable. For this reason, the datasets *Breast Cancer* of 7129 objects and *Yahoo* of 2340 objects are used to avoid fine-grained partitioning of data across such a large number of nodes. Detailed discussions on these datasets are presented in section 4.2.

## 8.3 Evaluation Measures

Two aspects of the distributed cooperative models were evaluated: clustering quality and speedup. For evaluating clustering quality, we relied on both external (*F-measure*) and internal (*SI index*) evaluation measures. Detailed discussion of the clustering quality measures is given in section 2.1.3. *Speedup* is a measure of the relative increase in the speed of the distributed algorithm on P nodes over the centralized approach. For evaluating the distributed models, it is calculated as the ratio of the time taken by the centralized cooperative clustering, CC($A_1$,$A_2$) (or CC($A_1$,$A_2$,$A_3$)) model to the time taken by the distributed model DCCP2P($A_1$,$A_2$) (or DCCP2P($A_1$,$A_2$,$A_3$)), including communication cost, where $A_1$ , $A_2$ , and $A_3 \in$ {KM, BKM, PAM}.

## 8.4 Distributed Clustering Performance Evaluation

Experiments are performed on a network of 50 nodes with different size of neighborhoods, $n_Q$= 3, 5, and 7 neighborhoods. The effect of neighborhood's size on the clustering quality (*F-measure* and *SI*) and speedup over the centralized cooperative clustering were measured.

Tables 8.1- 8.4 summarize those results for the *Yahoo* dataset for the three network configurations, 3-Neighborhoods, 5-Neighborhoods, and 7-Neighborhoods. For each configuration the results of 20 runs are obtained, and the average value bounded by its standard deviation is reported. We can see that changing number of neighborhoods $n_Q$ affects on the quality of the distributed cooperative models. The main reason is that the peer-clustering algorithm (section 7.2) partitions the network of peers into number of neighborhoods with the minimum communication cost such that similar peers are grouped into the same neighborhood. It can be shown from Tables 8.1-8.4 that the four cooperative models obtain the best clustering solutions measured by high values for *F-measure* and lower values for the *SI* index than those of the centralized cooperative approaches at number of neighborhoods $n_Q$ equals 5. In a network of 50 nodes and number of neighborhoods equal 5, the DCCP2P(KM,BKM) achieves a speedup up to 37 and the DCCP2P(KM,PAM) achieves a speedup up to 42, the DCCP2P(BKM,PAM) has a speedup of up to 39, and finally the triple cooperation distributed model, DCCP2P(KM,BKM,PAM) achieves a speedup up to 33.

The same results for the *Breast Cancer* dataset are illustrated in Table 8.5 to Table 8.8, respectively. Tables 8.5 -8.8 show that the best performance of the cooperative models compared to the centralized approaches is achieved at number of neighborhoods $n_Q$ equals 7. In a network of 50 nodes and for a 7-neighborhoods configuration, the DCCP2P(KM,BKM) model achieves a speedup up to 39 and the DCCP2P(KM,PAM) model achieves a speedup up to 36, the DCCP2P(BKM,PAM) model obtains a speedup of up to 35, and finally the triple distributed cooperative model, DCCP2P(KM,BKM,PAM) achieves a speedup up to 34.

**Table 8.1:** Distributed DCCP2P(KM,BKM ) *vs.* Centralized CC(KM,BKM) [*Yahoo*]

| | Centralized | | 3-Neighorhoods | | | 5-Neighborhoods | | | 7-Neighborhoods | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | F-measure | SI | F-measure | SI | Speedup | F-measure | SI | Speedup | F-measure | SI | Speedup |
| **Average** | 0.6619 | 1.1272 | 0.6712 | 1.0453 | 39.7652 | 0.7182 | 0.8108 | 37.072 | 0.6677 | 1.1138 | 33.369 |
| **Std-Dev** | ±0.023 | ±0.055 | ±0.019 | ±0.026 | ±2.312 | ±0.022 | ±0.044 | ±2.776 | ±0.031 | ±0.035 | ±1.594 |

**Table 8.2:** Distributed DCCP2P(KM,PAM ) *vs.* Centralized CC(KM,PAM) [*Yahoo*]

| | Centralized | | 3-Neighorhoods | | | 5-Neighborhoods | | | 7-Neighborhoods | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | F-measure | SI | F-measure | SI | Speedup | F-measure | SI | Speedup | F-measure | SI | Speedup |
| **Average** | 0.4794 | 2.1764 | 0.4861 | 2.0037 | 43.587 | 0.5642 | 1.4041 | 42.332 | 0.4817 | 2.1182 | 37.839 |
| **Std-Dev** | ±0.010 | ±0.026 | ±0.017 | ±0.034 | ±2.018 | ±0.032 | ±0.023 | ±2.345 | ±0.027 | ±0.041 | ±1.712 |

**Table 8.3:** Distributed DCCP2P(BKM,PAM ) *vs.* Centralized CC(BKM,PAM) [*Yahoo*]

| | Centralized | | 3-Neighorhoods | | | 5-Neighborhoods | | | 7-Neighborhoods | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | F-measure | SI | F-measure | SI | Speedup | F-measure | SI | Speedup | F-measure | SI | Speedup |
| **Average** | 0.6162 | 1.3695 | 0.6234 | 1.2486 | 40.775 | 0.6983 | 0.9642 | 39.231 | 0.6203 | 1.2685 | 37.981 |
| **Std-Dev** | ±0.018 | ±0.033 | ±0.023 | ±0.042 | ±1.784 | ±0.028 | ±0.021 | ±1.954 | ±0.032 | ±0.023 | ±2.394 |

**Table 8.4:** Distributed DCCP2P(KM,BKM,PAM ) *vs.* Centralized CC(KM,BKM,PAM) [*Yahoo*]

| | Centralized | | 3-Neighorhoods | | | 5-Neighborhoods | | | 7-Neighborhoods | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | F-measure | SI | F-measure | SI | Speedup | F-measure | SI | Speedup | F-measure | SI | Speedup |
| **Average** | 0.6698 | 1.0073 | 0.6811 | 0.8905 | 33.457 | 0.7434 | 0.6523 | 32.993 | 0.6702 | 0.9234 | 31.021 |
| **Std-Dev** | ±0.022 | ±0.064 | ±0.036 | ±0.025 | ±2.341 | ±0.031 | ±0.037 | ±1.934 | ±0.029 | ±0.026 | ±2.002 |

**Table 8.5:** Distributed DCCP2P(KM,BKM ) *vs.* Centralized CC(KM,BKM) [*Breast Cancer*]

| | Centralized | | 3-Neighorhoods | | | 5-Neighborhoods | | | 7-Neighborhoods | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | F-measure | SI | F-measure | SI | Speedup | F-measure | SI | Speedup | F-measure | SI | Speedup |
| **Average** | 0.4915 | 0.6998 | 0.5011 | 0.6836 | 41.235 | 0.5297 | 0.6771 | 40.485 | 0.5911 | 0.5374 | 39.212 |
| **Std-Dev** | ±0.016 | ±0.015 | ±0.039 | ±0.025 | ±3.073 | ±0.025 | ±0.023 | ±2.176 | ±0.022 | ±0.017 | ±2.342 |

**Table 8.6:** Distributed DCCP2P(KM,PAM ) *vs.* Centralized CC(KM,PAM) [*Breast Cancer*]

| | Centralized | | 3-Neighorhoods | | | 5-Neighborhoods | | | 7-Neighborhoods | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | F-measure | SI | F-measure | SI | Speedup | F-measure | SI | Speedup | F-measure | SI | Speedup |
| **Average** | 0.5935 | 0.5418 | 0.6112 | 0.5231 | 41.449 | 0.6328 | 0.4847 | 39.384 | 0.7033 | 0.3717 | 36.518 |
| **Std-Dev** | ±0.020 | ±0.019 | ±0.021 | ±0.026 | ±2.029 | ±0.033 | ±0.042 | ±3.112 | ±0.036 | ±0.011 | ±3.020 |

**Table 8.7:** Distributed DCCP2P(BKM,PAM) *vs.* Centralized CC(BKM,PAM) [*Breast Cancer*]

| | Centralized | | 3-Neighorhoods | | | 5-Neighborhoods | | | 7-Neighborhoods | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | F-measure | SI | F-measure | SI | Speedup | F-measure | SI | Speedup | F-measure | SI | Speedup |
| **Average** | 0.6402 | 0.4943 | 0.6472 | 0.4727 | 38.024 | 0.6512 | 0.4632 | 36.985 | 0.7422 | 0.3172 | 35.401 |
| **Std-Dev** | ±0.017 | ±0.027 | ±0.034 | ±0.013 | ±2.992 | ±0.037 | ±0.010 | ±2.923 | ±0.036 | ±0.029 | ±2.134 |

**Table 8.8:** Distributed DCCP2P(KM,BKM,PAM) *vs.* Centralized CC(KM,BKM,PAM) [*Breast Cancer*]

| | Centralized | | 3-Neighorhoods | | | 5-Neighborhoods | | | 7-Neighborhoods | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | F-measure | SI | F-measure | SI | Speedup | F-measure | SI | Speedup | F-measure | SI | Speedup |
| **Average** | 0.6487 | 0.4767 | 0.6538 | 0.4728 | 37.222 | 0.6639 | 0.4525 | 36.394 | 0.7611 | 0.2783 | 34.246 |
| **Std-Dev** | ±0.023 | ±0.031 | ±0.024 | ±0.017 | ±2.345 | ±0.024 | ±0.053 | ±2.212 | ±0.041 | ±0.029 | ±3.274 |

## 8.5 Scalability of the Network

In this sub-section, a set of experiments are conducted to test the effect of increasing number of nodes on the clustering quality as well as the speedup of the distributed cooperative models in the network with a fixed number of neighborhoods $n_Q$=5 for the *Yahoo* dataset and $n_Q$=7 for the *Breast Cancer* dataset.

Figures 8.1-8.3 report the outcome of those experiments for networks sizes range from 5-100 nodes for the *Yahoo* dataset. The first observation that we can notice in figures 8.1 and 8.2 is that increasing number of nodes affects on the quality of the cooperative model. We can see that the cooperative models obtain better clustering solutions with increasing number of nodes where the distribution of data is changing within each node. This change in the data distribution provides diversity in the clustering solutions of the adopted KM, BKM and PAM algorithms. The resulting diversity in the solutions generates sub-clusters with better homogeneity that reveals better overall global clustering quality.

The second observation is that, for networks with larger number of nodes above a specific number of nodes (P > $P^{critical}$), the clustering quality is dropped. Thus after some value of P equals $P^{critical}$, the data is finely partitioned and consequently, the quality of the distributed cooperative models degrades rapidly. The value of $P^{critical}$ provides a clue of the relation between the data set size and the number of nodes, beyond which the number of nodes should not be increased without increasing the dataset size. An appropriate strategy for automatically detecting the value of $P^{critical}$ is to compare the values of *F-measure* and *SI* before and after adding nodes, if a sufficiently drop in the *F-measure* and increase in the *SI* is noticed then the network growth should be suspended until more data is available. We can see from figures 8.1 and 8.2 that the value of $P^{critical}$ for the *Yahoo* dataset equals 50 nodes. Fig. 8.3 shows the speedup of each of the cooperative models along with increasing number of nodes. It can be shown that at P= $P^{critical}$= 50 nodes, DCCP2P(KM,BKM) achieves a speedup up to 37, DCCP2P (KM,PAM) achieves a speedup up to 42, and DCCP2P(KM,BKM,PAM) achieve a speedup up to 39, while the DCCP2P(KM,BKM,PAM) achieves a speedup up to 33.

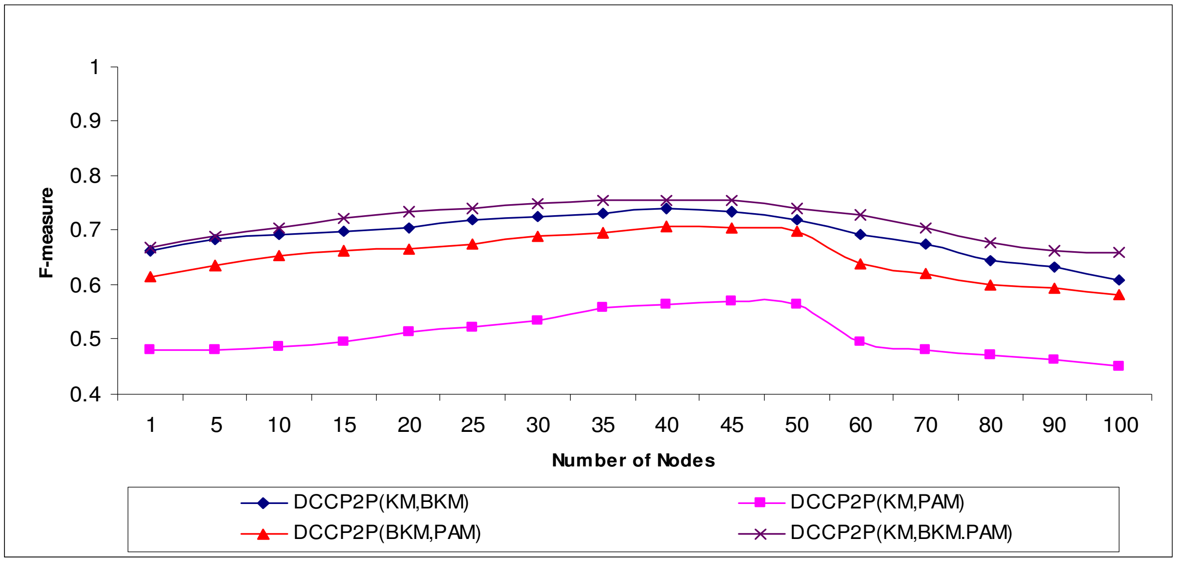


**Fig. 8. 1.** Quality of the Distributed Cooperative Clustering Models measured by *F-measure* [*Yahoo*]

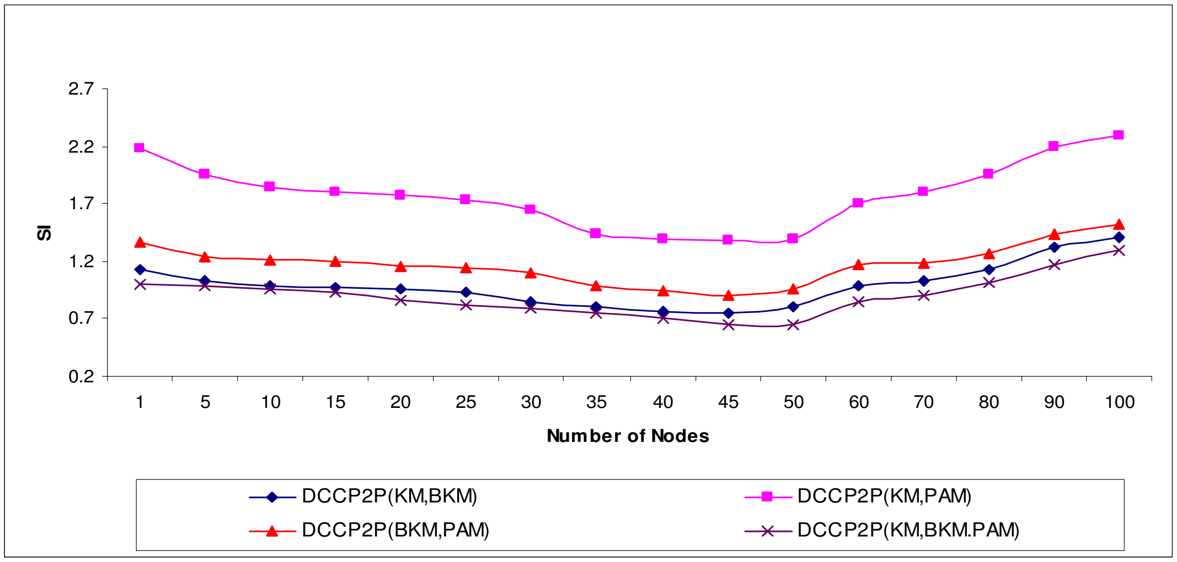
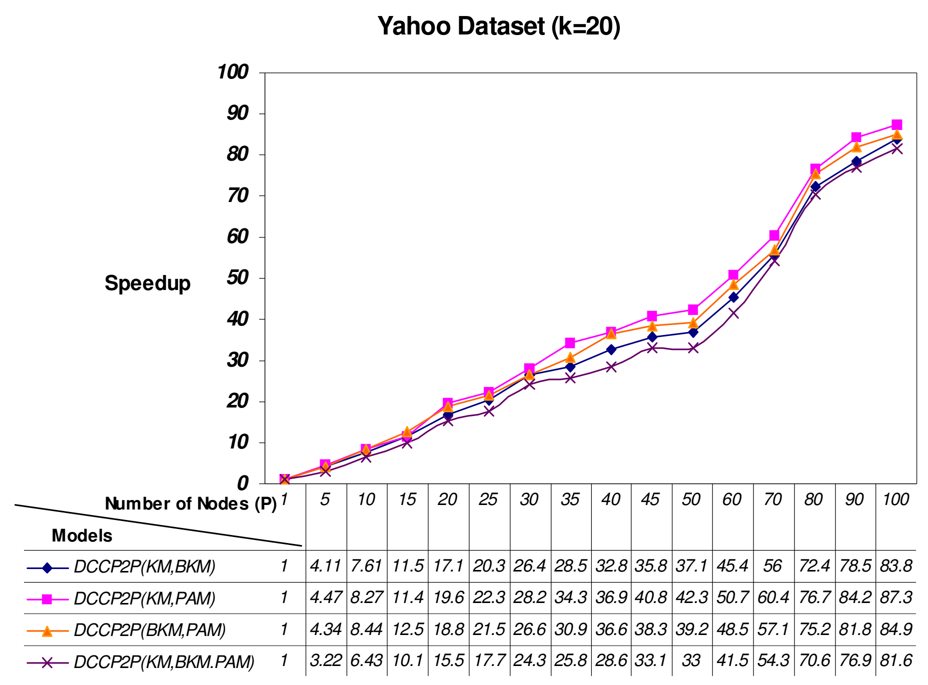


**Fig. 8. 2.** Quality of the Distributed Cooperative Clustering Models measured by *SI* [*Yahoo*]



| Models \ Number of Nodes (P) | 1 | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 | 60 | 70 | 80 | 90 | 100 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ◆ DCCP2P(KM,BKM) | 1 | 4.11 | 7.61 | 11.5 | 17.1 | 20.3 | 26.4 | 28.5 | 32.8 | 35.8 | 37.1 | 45.4 | 56 | 72.4 | 78.5 | 83.8 |
| ■ DCCP2P(KM,PAM) | 1 | 4.47 | 8.27 | 11.4 | 19.6 | 22.3 | 28.2 | 34.3 | 36.9 | 40.8 | 42.3 | 50.7 | 60.4 | 76.7 | 84.2 | 87.3 |
| ▲ DCCP2P(BKM,PAM) | 1 | 4.34 | 8.44 | 12.5 | 18.8 | 21.5 | 26.6 | 30.9 | 36.6 | 38.3 | 39.2 | 48.5 | 57.1 | 75.2 | 81.8 | 84.9 |
| × DCCP2P(KM,BKM.PAM) | 1 | 3.22 | 6.43 | 10.1 | 15.5 | 17.7 | 24.3 | 25.8 | 28.6 | 33.1 | 33 | 41.5 | 54.3 | 70.6 | 76.9 | 81.6 |

**Fig. 8.3.** Scalability of the Distributed Cooperative Clustering Models [*Yahoo*]

Results are also reported in Figures 8.4-8.6 showing the quality and speedup of the cooperative models compared to that of the centralized cooperative approaches with increasing number of nodes for the *Breast Cancer* dataset. Figures 8.4 and 8.5 show that the value of $P^{critical}$ after which the performance starts to degrade for the *Breast Cancer* dataset is 60 nodes. Fig. 8.6 shows the speedup of each cooperative model along with increasing number of nodes. We can see that at P= $P^{critical}$= 60, the DCCP2P(KM,BKM) achieves a speedup up to 49, DCCP2P (BKM,PAM) achieves a speedup up to 46, DCCP2P(KM,PAM) achieves a speedup up to 45, and finally the DCCP2P(KM,BKM,PAM) achieves a speedup up to 43.

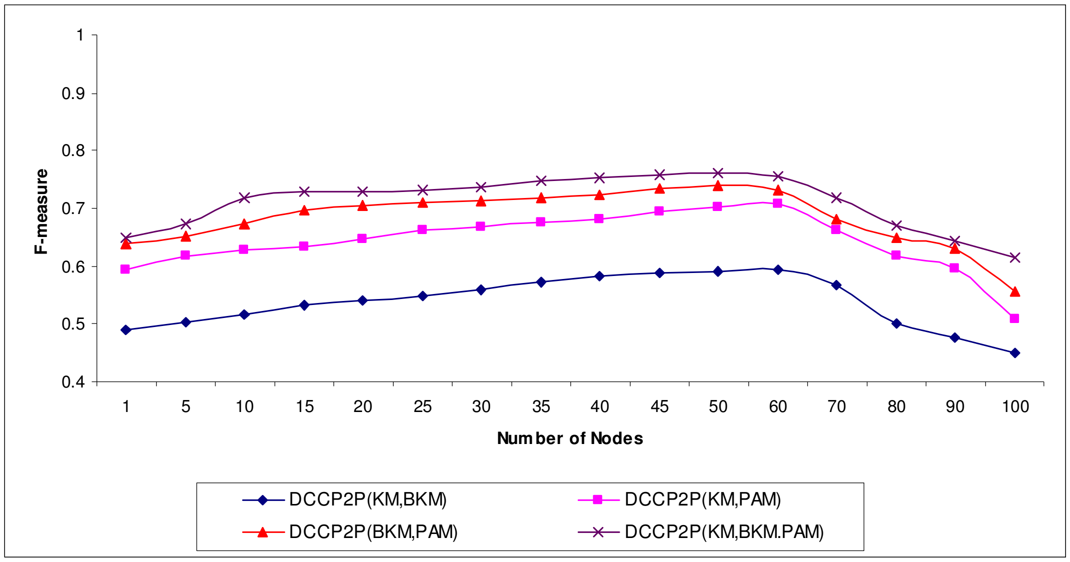


**Fig. 8. 4.** Quality of the Distributed Cooperative Clustering Models (*F-measure)* [*Breast Cancer*]

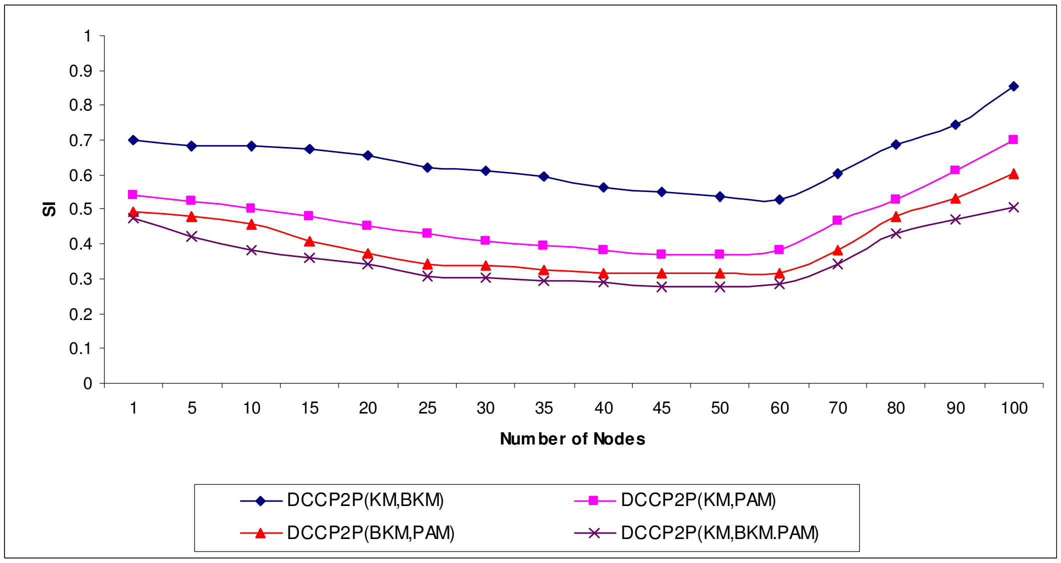
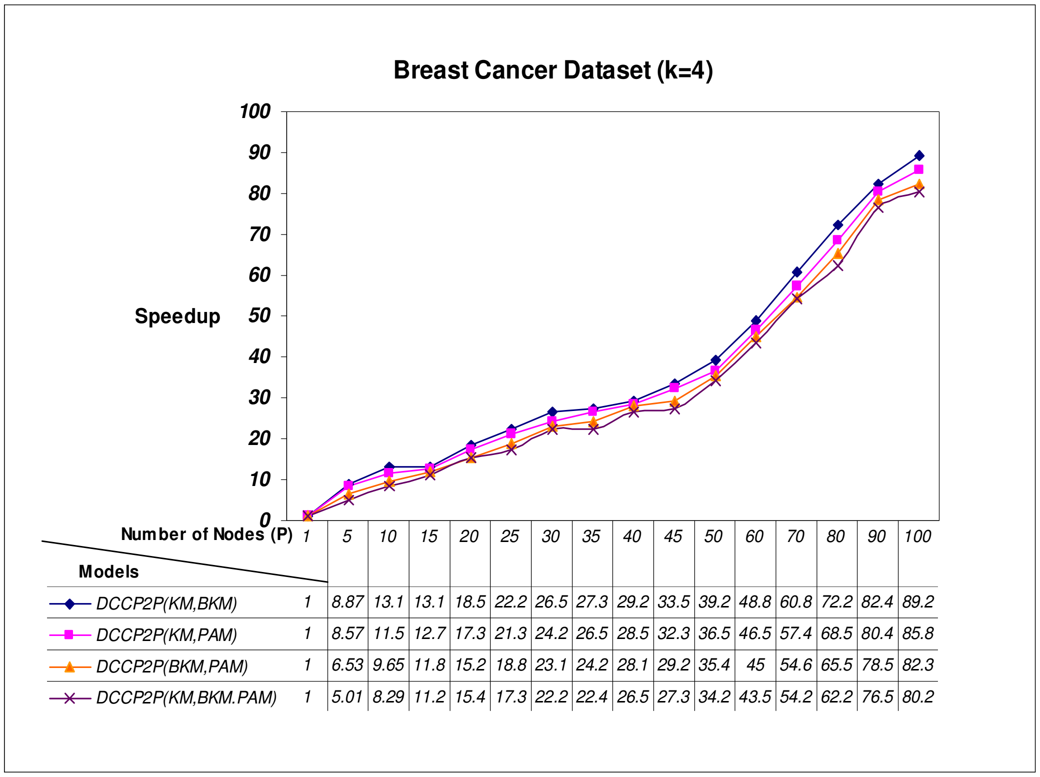


**Fig. 8. 5.** Quality of the Distributed Cooperative Clustering Models (*SI)* [*Breast Cancer*]



| Models \ Number of Nodes (P) | 1 | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 | 60 | 70 | 80 | 90 | 100 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ◆ DCCP2P(KM,BKM) | 1 | 8.87 | 13.1 | 13.1 | 18.5 | 22.2 | 26.5 | 27.3 | 29.2 | 33.5 | 39.2 | 48.8 | 60.8 | 72.2 | 82.4 | 89.2 |
| ■ DCCP2P(KM,PAM) | 1 | 8.57 | 11.5 | 12.7 | 17.3 | 21.3 | 24.2 | 26.5 | 28.5 | 32.3 | 36.5 | 46.5 | 57.4 | 68.5 | 80.4 | 85.8 |
| ▲ DCCP2P(BKM,PAM) | 1 | 6.53 | 9.65 | 11.8 | 15.2 | 18.8 | 23.1 | 24.2 | 28.1 | 29.2 | 35.4 | 45 | 54.6 | 65.5 | 78.5 | 82.3 |
| × DCCP2P(KM,BKM.PAM) | 1 | 5.01 | 8.29 | 11.2 | 15.4 | 17.3 | 22.2 | 22.4 | 26.5 | 27.3 | 34.2 | 43.5 | 54.2 | 62.2 | 76.5 | 80.2 |

**Fig. 8.6.** Scalability of the Distributed Cooperative Clustering Models [*Breast Cancer*]

## 8.6 Discussions

In this chapter an evaluation of the various methods and algorithms presented in this chapter was presented. Experiments were performed on actual document and gene expression datasets representing different characteristics. Evaluation of the distributed cooperative clustering models with networks of different sizes, neighborhoods and configurations was presented and discussed. Based on the experimental results, we can conclude that the distributed cooperative clustering methods in this thesis are successful with respect to their goals, with certain limitations that mostly can be accommodated. In addition to improving quality and gaining speedup, providing accurate interpretations for cooperative clustering results through contingency and merging phases makes the results available for interpretations. Detailed summary, conclusions and recommendations are discussed in the next chapter.

# Chapter 9
# Conclusions and Future Research

## 9.1 Conclusions and Thesis Contributions

In this thesis new models and algorithms have been proposed to advance the performance of data clustering, outlier's detection, and distributed data clustering. In this section, we conclude and summarize the main contributions of this thesis as follows:

### 9.1.1 Cooperative Clustering

In the problem of data clustering, we proposed a new cooperative clustering (CC) model. The cooperative model finds the agreement between multiple clusterings in forms of sub-clusters. Each sub-cluster is represented with a statistical concise representation of data within the sub-cluster; this form of representation is identified as a Similarity-Histogram (SH). A cooperative Contingency Graph (*CCG*) is built, in which nodes are the set of sub-clusters and edges are weighted by cohesiveness merging factors for merging sub-clusters. The merged sub-clusters attain new clusters with better homogeneity which direct the clustering process into a more coherent grouping of data. The cooperative model specifically (1) obtains better clustering quality than the traditional non-cooperative clustering techniques, (2) enhances the clustering quality of the adopted clustering techniques, and (3) handles datasets with different configurations. Experimental results show that the cooperative model achieves its goals with comparable results to the adopted clustering techniques.

### 9.1.2 Cooperative Clustering Outliers Detection

In the area of clustering-based outlier detection, we introduced a new cooperative clustering outlier's detection (CCOD) algorithm for better detection of outliers. The proposed CCOD algorithm comprises of four stages, the first stage includes a synchronous non-cooperative clustering where *c* clusterings solutions are obtained. The second stage obtains the set of sub-clusters as an agreement between the invoked clustering algorithms. The CCOD algorithm then assigns a cooperative outlier factor (*COF*) to each object based on the size of sub-clusters and the distributions of similarities within sub-clusters before a certain threshold. This outlier factor identifies the set of possible outliers within the set of sub-clusters. Finally, in order to find the global set of outliers within the whole set of clusters, a bottom-up detection scenario is performed.

The CCOD outperforms the traditional clustering-based outlier's detection, e.g. *FindCBLOF*, as it distinguishes objects with low or high similarities within small and large sub-clusters, in addition candidate outliers are found based on their effect on the merging process of the cooperative clustering model.

### 9.1.3 Cooperative Clustering in Super-Peer P2P Networks

Distributed environments provide both opportunities and challenges for distributed data clustering. In the context of distributed data clustering, we proposed a dynamic two-tier super-peer P2P cooperative clustering architecture and model for obtaining globally-optimized clusters. The distributed model is called Distributed Cooperative Clustering in Peer-to-Peer networks (DCCP2P). The proposed distributed model addresses the modularity and scalability of the network in terms of number of nodes and consequently the scalability of the distributed approach. It involves a hierarchy of two layers of P2P neighborhoods. In the first layer, peers in each neighborhood are responsible for building local models from the local data sets that they are responsible for. This summarized view of local data at each node minimizes the exchange of information between nodes and their super-peers. As we move up to the next layer, clusters are merged at each super-peer and at the root of the hierarchy one global clustering can be derived. This approach achieves significant improvement in the global clustering solutions without the cost of centralized clustering. Experimental results showed that good speedup can be achieved using this model with comparable results to centralized cooperative clustering.

## 9.2 Challenges and Future work

A number of challenges arose during this research. Some of those challenges and how they were addressed with extended future work are listed here.

### 9.2.1 Challenges

**Data**

Both gene expression and text documents datasets with different characteristics are used in this thesis. We used the *Vector Space Model* (VSM), which is the most common document representation model used in text mining.  Other representations success such as (1) the N-gram [118] model, the N-gram model assumes that the document is a sequence of characters, (2) the Suffix tree model [119] which finds common phrases suffixes between documents and builds a suffix tree where each node represents part of a phrase and associated with it are the documents containing this phrase-suffix, and (3) Finally, a phrase-based approach proposed in [82] to facilitate matching phrases effectively between documents.

**Large Datasets & High Dimensionality**

Due to the new trends on text mining and bioinformatics, the size of datasets in addition to its dimensionality have become extremely large (tens of thousands in both data and features levels). To be able to properly handle such high size and dimensionality, sampling and feature selection methods are employed to bring the size of data and the number of dimensions to a manageable level. Both sampling and feature selection are two wide research areas. We used a simple document feature selection method to reduce the number of features in large datasets. As for the text document datasets *UW* and *Yahoo*, features of weight equals 1 were removed. This reduced features set was used during cooperative clustering, outlier detection and distributed cooperative clustering.

**Similarity Measures**

Unsupervised cluster analysis is considered highly dependent on and sensitive to similarity measures. A bad similarity measure can have drastic effect on the clustering quality. In the work presented in this thesis we counted on the *cosine similarity* as a widely similarity measure in text mining and bioinformatics. Finding better similarity measures that are applicable on different types of datasets is of interest.

**Clustering Algorithms**

There is a wide range of clustering algorithms in the literature including: Partitional clustering, hierarchical clustering, graph-based clustering, model-based clustering, and many others. In the experimental section we employed four of the well known clustering algorithm from the two main family of clustering, Partitional and Hierarchical. Of course investigating the capability of the cooperative model with all clustering techniques is infeasible, but adding some other clustering algorithms to the model will be of interest.

**Clustering Quality Measures**

Two evaluation methodologies can be chosen for evaluating the clustering quality of a clustering algorithm, which can be complimentary, evaluation against a reference classification (ground truth) and evaluation against internal structure of the clusters. We used both external (*F-measure*, *Entropy*, and *Purity*) and internal quality measures (*SI* index and *OWSR*) to assess the clustering quality of the proposed models as well as the contemporary approaches. Investigating the quality performance of the cooperative clustering model using additional quality measures (e.g. the figure of merit validity measure) is an interesting issue.

**Distributed Architecture**

Simulation of peer-to-peer networks is not a trivial process. Experiments that involve hundreds on nodes were not possible due to limited number of computers available. Due to these limitations of resources, and in order to maintain simplicity and feasibility, we opted to do simulations on a single computer, rather than on a real network.

### 9.2.2 Future Work

The future work planned to be addressed can be categorized into three areas:

**Cooperative Data Clustering**

In the area of cooperative clustering, future directions include:

- Finding the optimum value of the similarity threshold as well as the optimum bin size in each histogram.
- Applying the same cooperative methodology if the number of the generated clusters is different from one partitioning to another and employing a new membership function to find the intersection between the $c$ clustering solutions.
- Developing a scatter F-measure that finds the diversity in the clustering solutions between two or more approaches.
- Comparing the time and quality performances of the cooperative clustering model to those of the current state of art in combining multiple clusterings such as ensemble clustering and hybrid clustering.
- Calculating the value of $c^*$ (the maximum number of cluster techniques in the cooperative model to maintain better clustering quality) is done experimentally, proving the value of $c^*$ theoretically will be of interest as a future work.

**Clustering-Based Outliers Detection**

For outlier's detection, the following issues are of interest as a future work to the current work in this thesis:

- Comparing the detection accuracy of the CCOD with that if the cooperative outlier factor is assigned to each object after the cooperative clustering is performed.
- Determining the proper value of the ratio between the *LocalTopRatio* and the *TopRatio* parameters in order to achieve the desired detection capability of the proposed algorithm.
- Testing the scalability of the detection approach using more than three clustering techniques
- Developing a distributed outlier's detection using cooperative clustering can be investigated to examine the performance of the cooperative clustering in detecting outliers in distributed networks and also finding the global set of outliers across the whole network needs more investigation.

**Distributed Data Clustering**

For distributed clustering, the following issues can be investigated as a future work of this thesis:

- So far data partitioning has been done equally across nodes. Different partitioning strategies can be investigated to see the effect of unbalanced distribution of data in a network.
- Also, in the current architecture we use two-tier P2P architecture, an extension to multiple overlay hierarchical networks will be of interest.
- The ability of designing informative distributed internal and external quality measures that are suitable for distributed hierarchical super-peer networks is considered as interesting issue, more research work is needed in this track.
- A formal evaluation of the value of the critical number of nodes in the network after which the performance of the distributed cooperative model decays will be of interest as a future work.
- Finally, the effect of the set of parameters used in the distributed cooperative model needs to be formally analyzed to provide error estimates that can guide the proposed model towards its goals.

## 9.3 List of Publications

The work in this thesis has resulted in a number of publications (Journal, conference proceedings, as well as posters) which are listed below.

### Journal Articles: Submitted

- **R. Kashef, M. S. Kamel**, "*Enhanced Bisecting K-means Clustering Using Intermediate Cooperation*", Pattern Recognition, submitted and under second revision.
- **R. Kashef, M. S. Kamel,** "*Cooperative Clustering Model*", Pattern Recognition, Submitted.

### Journal Articles: In Preparation

- **R. Kashef, M. S. Kamel,** "*Cooperative Clustering Outliers Detection*", written and will be submitted.
- **R. Kashef, M. S. Kamel**, *"Cooperative Clustering in Distributed Super-Peer P2P networks",* in progress.

### Conference Proceedings

- **R. Kashef, M. S. Kamel**, "*Towards Better Outliers Detection for Gene Expression Datasets*", 2008 IEEE International Conference on Biocomputation, Bioinformatics, and Biomedical Technologies (BIOTECHNO08), pp: 149-154, 2008.
- **R. Kashef, M. S. Kamel, "**Distributed Cooperative Partitional Divisive Clustering for Gene Expression Datasets*", the 2008 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational (CIBCB08), 2008.
- **R. Kashef, M. S. Kamel**, "*Efficient Bisecting k-medoids and Its Application in Gene Expression Analysis*", International Conference on Image Analysis and Recognition (ICIAR08), pp: 423-343, 2008.
- **R. Kashef, M. S. Kamel**, "*Hard-Fuzzy Clustering: A Cooperative-based Approach*", 2007 IEEE International Conference on Systems, Man and Cybernetics. pp: 425-430, 2007.
- **R. Kashef, M. S. Kamel**, "*Cooperative Partitional-Divisive Clustering and Its Application in Gene Expression Analysis*", IEEE 7th International Conference on BioInformatics and BioEngineering (BIBE07), pp: 116-122, 2007.

- **R. Kashef, M. S. Kamel**, "*Distributed Cooperative Hard-fuzzy Document Clustering*", 3rd Annual Scientific Conference of the LORNET Research Network (I2LOR 2006), Montreal, Nov 8-10, 2006.

**Posters**

- **R. Kashef, M. S. Kamel,** "*Cooperative Clustering and its Application in Gene Expression Analysis*", Poster Presentation, Graduate Research Conference, 2008.
- **R. Kashef, M. S. Kamel,** "*Hard-Fuzzy Clustering: A Cooperative-based Approach*", Poster Presentation, Graduate Research Conference, 2007.
- **R. Kashef, M. S. Kamel,** "*Cooperative Partitional-Divisive Clustering and Its Application in Text Mining*", L2lOR2007, Montreal, Canada.
- **R. Kashef, M. S. Kamel**, "*Distributed Cooperative Hard-fuzzy Document Clustering*", Poster Presentation, Knowledge and Data Mining Workshop (KDM06), University of Waterloo, Waterloo, Canada, 2006.

# Appendix A
# Message Passing Interface

The Message Passing Interface (MPI) [86] communication tool consists of a group of processes that have only local memory but are able to communicate with other processes by sending and receiving messages. It is a defining feature of the message-passing model, such that data transfers from the local memory of one process to the local memory of another process require operation to be performed by both processes. MPI is a standardized, portable, and widely available message passing system. A typical parallel program can be written in any high level language, which is then compiled and linked with the MPI library. The resultant object code is distributed to each process for parallel execution. Each process is assigned a unique identifier between 0 and *P*-1 where P is the number of parallel processes. Depending on its process identifier, each process may follow a distinct execution path through the same code. These processes may communicate with each other by calling appropriate routines in the MPI library, which encapsulates the details of communications between various processes. Table 1 lists some of the various MPI routines, which are used in most of the Single Program Multiple Data (SPMD) clustering algorithms.

**Table 1**: Functionality of MPI routines

| **MPI routine** | **Functionality** |
|---|---|
| MPI_init() | Initialize the MPI execution environment |
| MPI_Comm_size() | Returns the number of processes. |
| MPI_Comm_rank() | Returns the process identifier for the calling process. |
| MPI_Bcast(message,root) | Broadcasts “message” from a process with identifier “root” to all the processes. |
| MPI_Allreduce(A,B,MPI_SUM) | Sums all the local copies of “A” in all the processes (reduction operation) and replace the result in “B” on all the processes (broadcast operation) |
| MPI_Wtime() | Returns the number of seconds since some fixed, arbitrary point of time in the past. |
| MPI_Finalize() | Terminate the MPI execution environment |

# Bibliography

[1] A. Jain, M. Murty, and P. Flynn. "*Data Clustering: A Review*". ACM computing surveys, Vol. 31, pp: 264-323, 1999.

[2] R. Xu, "*Survey of Clustering Algorithms*", IEEE Transactions on Neural Networks, Vol.16, Issue 3, pp: 645- 678, 2005.

[3] R. Duda, D. Stork and P. Hart, Pattern Classification, John Wiley, 2001.

[4] J. Hartigan, Clustering Algorithms, John Wiley & Sons, New York, NY, 1975.

[5] F. Karray and C. Desilva, Soft Computing and Intelligent Systems Design Theory, Tools and Applications, Addison Wesley Publishing, 2004.

[6] M. Steinbach, G. Karypis, and V. Kumar, "*A Comparison of Document Clustering Techniques*", Proceeding of the KDD Workshop on Text Mining, pp: 109-110, 2000.

[7] J. Hartigan and M. Wong. "*A k-means Clustering Algorithm*", Applied Statistics, Vol. 28, pp: 100-108, 1979.

[8] S. Guha, R. Rastogi, and K. Shim, "*CURE: An Efficient Clustering Algorithm for Large Databases*". In Proceedings ACM SIGMOD International Conference on Management of Data, pp: 73-84. ACM Press, 1998.

[9] T. Zhang, R. Ramakrishnan, and M. Livny. "*BIRCH: An Efficient Data Clustering Method for very Large Databases*". In *Proc. of ACM-SIGMOD Int. Conf. Management of Data (SIG-MOD'96)*, pp: 103-114, 1996.

[10] M. Vrahatis, B. Boutsinas, P. Alevizos, and G. Pavlides, "*The New k-windows Algorithm for Improving the k-means Clustering Algorithm*", Journal of Complexity Vol.18, pp: 375-391, 2002.

[11] D. Boley. "*Principal Direction Divisive Partitioning*". *Data Mining and Knowledge Discovery*, 2 (4), pp: 325-344, 1998.

[12] J. Bezdek, R. Ehrlich, and W. Full, "*The Fuzzy C-Means Clustering Algorithm*", Computers and Geosciences, Vol.10, pp: 191-203, 1984.

[13] S. Savaresi and D. Boley. "*On the Performance of Bisecting K-means and PDDP*". In Proc. of the 1st SIAM Int. Conf. on Data Mining, pp: 1-14, 2001.

[14] L. Kaufmann and P. Rousseeuw, Finding groups in data, Wiley, 1990.

[15] R. Ng, J. Han, *"Efficient and Effective Clustering Methods for Spatial Data Mining*". In VLDB, pp: 144–155, 1994.

[16] K. Hammouda and M. Kamel, "*Incremental Document Clustering Using Cluster Similarity Histograms*", The IEEE/WIC International Conference on Web Intelligence (WI 2003), pp: 597-601, 2003.

[17] K. Hammouda and M. Kamel, "*Efficient Phrase-based Document Indexing for Web Document Clustering*", IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 10, pp: 1279-1296, 2004.

[18] A. Ben-Dor, R. Shamir, and Z. Yakhini, "*Clustering Gene Expression Patterns*", Journal of Computational Biology, 6(3/4), pp: 281–297, 1999.

[19] B. Borah, D. Bhattacharyya, "*An Improved Sampling-based DBSCAN for Large Spatial Databases*". In Proceedings of the International Conference on Intelligent Sensing and Information, pp: 92-96, 2004.

[20] R. Shamir, and R. Sharan, "*Algorithmic Approaches to Clustering Gene Expression Data*", in Current Topics in Computational Biology, MIT Press, pp: 269-299, 2002.

[21] E. Hartuv, and R. Shamir, "*A Clustering Algorithm Based on Graph Connectivity*". Information Processing Letters, 76(200) pp: 175-181, 2000.

[22] N. Yousri, M. Ismail and M.S. Kamel, "*Discovering Connected Patterns in Gene Expression Arrays*", IEEE Symposium on Computational intelligence and Bioinformatics and Computational Biology (CIBCB), pp: 113-120, 2007.

[23] Y. Lu, S. Lu, F. Fotouhi, Y. Deng, and S. Brown, "*Incremental Genetic K-means Algorithm and its Application in Gene Expression Data Analysis*", BMC Bioinformatics, 5(172), 2004.

[24] K. Cios, W. Pedrycs, and R. Swiniarski, Data Mining Methods for Knowledge Discovery. Kluwer Academic Publishers, 1998.

[25] R. Krishnapuram, A. Joshi, and L. Yi, "*A fuzzy Relative of the k-medoids Algorithm with Application to Web Document and Snippet Clustering*", In IEEE International Conference on Fuzzy Systems, pp: 1281-1286,1999.

[26] B. Frey and D. Dueck, "*Clustering by Passing Messages Between Data Points*", *Science* 16, Vol. 315. no. 5814, pp: 972–976, 2007.

[27] F. Bach, M. Jordan, "*Learning Spectral Clustering*", In: Thrun, S., Saul, L., Schölkopf, B. (eds.). Advances in Neural Information Processing Systems (NIPS) 16, pp: 305-312. MIT Press, 2004.

[28] Y. Qian and C. Suen. "*Clustering Combination Method*". In International Conference on Pattern Recognition. ICPR 2000, volume 2, pp: 732–735, 2000.

[29] A. Strehl and J. Ghosh. "*Cluster Ensembles – Knowledge Reuse Framework for Combining Partitionings*". In conference on Artificial Intelligence (AAAI 2002), pp: 93–98, AAAI/MIT Press, 2002.

[30] H. Ayad, M. Kamel, "*Cumulative Voting Consensus Method for Partitions with Variable Number of Clusters*," IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Computer Society Digital Library. IEEE Computer Society, Volume 30, Issue 1, pp: 160-173, 2008.

[31] Y. Eng, C. Kwoh, and Z. Zhou, "*On the Two-Level Hybrid Clustering Algorithm*" AISAT04 - International Conference on Artificial Intelligence in Science and Technology, pp: 138-142. 2004.

[32] S. Xu and J. Zhang, "*A Hybrid Parallel Web Document Clustering Algorithm and its Performance Study*", Journal of Supercomputing, Vol. 30, Issue 2, pp: 117-131, 2004.

[33] V. Barnett, T. Lewis, Outliers in Statistic Data, John Wiley's Publisher, NY, 1994.

[34] M. Knorr, and R. Ng, "*Algorithms for Mining Distance-Based Outliers in Large Datasets*", Very Large Data Bases Conference Proceedings, pp: 24-27,1998.

[35] F. Angiulli, C. Pizzuti, "*Fast Outlier Detection in High Dimensional Spaces*". Proc of PKDD'02, pp: 15-26, 2002.

[36] M. Breunig, H. Kriegel, R. Ng, J. Sander, "*LOF: Identifying Density based Local Outliers*", Proc. ACM SIGMOD 2000. Int. Conf. On Management of Data, pp: 93-104, 2000.

[37] K. Yamanishi, J. Takeuchi, G. Williams and P. Milne, "*On-Line Unsupervised Outlier Detection Using Finite Mixtures with Discounting Learning Algorithms*". In Proceedings of the

International Conference on Knowledge Discovery and Data Mining, Volume 8, Issue 3, pp: 275 - 300, 2004.

[38] A. Arning, R. Agrawal, and P. Raghavan. "*A Linear Method for Deviation Detection in Large Databases*". In: Proc of KDD'96, pp: 164-169, 1996.

[39] Z. He, X. Xu and S. Deng, **"***Discovering Cluster-based Local Outliers*", Pattern Recognition Letters Volume 24, Issues 9-10, , pp: 1641-1650, 2003.

[40] Z. He, X. Xu, and S. Deng,"*Squeezer: An Efficient Algorithm for Clustering Categorical Data*". J. Comput. Sci. Technol. 17 5, pp: 611–624. 2002.

[41] S. Ramaswamy, R. Rastogi, and K. Shim, "*Efficient Algorithms for Mining Outliers from Large Datasets*", Proceedings of the ACM SIGMOD international conference on Management of data, pp:427-438, 2000.

[42] S. Papadimitriou, H. Kitagawa, P.Gibbons, and C. Faloutsos, "*LOCI: Fast Outlier Detection Using the Local Correlation Integral*", 19th International Conference on Data Engineering (ICDE03), pp: 315-326, 2003.

[43] D. Hawkins, Identification of Outliers, Chapman and Hall, London 1980.

[44] S. Datta, C. Giannella, and H. Kargupta. "*K-means Clustering Over a Large, Dynamic Network*". In SIAM International Conference on Data Mining (SDM06), pp: 153–164, 2006.

[45] K. Hammouda and M. Kamel. "*Distributed Collaborative Web Document Clustering Using Cluster Keyphrase Summaries*". Information Fusion, Special Issue on Web Information Fusion, pp: 465-480, 2008.

[46] U. Maulik, S. Bandyopadhyay, "*Performance Evaluation of Some Clustering Algorithms and Validity Indices*", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.24, No.12, pp: 1650-1654, 2002.

[47] X. Xie and G. Beni, "*A Validity Measure for Fuzzy Clustering*", IEEE Transactions on Pattern Analysis and machine Intelligence, Vol.13, No.4, pp: 841-846, 1991.

[48] J. Bezdek and N. Pal, "*Some New Indexes of Cluster Validity*". IEEE Transactions on Systems and Cybernetics – Part B: Cybernetics, Vol.28, No.3, pp: 301-315, 1998.

[49] R. Kashef and M. Kamel, "*Efficient Bisecting K-medoids and its Application for Gene Expression Datasets*", International Conference on Image Analysis and Recognition, (ICIAR08), pp: 423-343, 2008.

[50] J. Jang, C. Sun and E. Mizutani, Neuro-Fuzzy and Soft Computing. Prentice Hall, 1997.

[51] M. Ester, H-P. Kriegel, J. Sander, and X. Xu, "A *Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*". In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, pp: 226–231, 1996.

[52] P. Alevizos, "*An algorithm for Orthogonal Range Search in d≥3 dimensions*", proceeding of the 14th European workshop on Computational Geometry, 1998.

[53] P. Alevizos, B. Boutsinas, D. Tasoulis, and M. Vrahatis, "*Improving the Orthogonal Range Search k-windows Clustering Algorithm*", Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence, pp: 239-245, 2002.

[54] J. Jackson. A User's Guide to Principal Components. John Wiley, 1991.

[55] C. Olson, "*Parallel Algorithms for Hierarchical Clustering*", Parallel Computing, Vol. 21, pp: 1313-1325, 1995.

[56] D. Judd, P. Mckinely, and A. Jain, "*Large-Scale Parallel Data Clustering*", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.20, No.8, pp: 871-876, 1998.

[57] X. Xu and J. Jger, and H. Kriegel, "*A Fast Parallel Clustering Algorithm for Large Spatial Databases*". Data Mining and Knowledge Discovery, Vol. 3, pp: 263-290, 1999.

[58] P. Bradley, U. Fayyad, and C. Reina, "*Scaling Clustering Algorithms to Large Databases*". In Proceeding of the Fourth International Conference on Knowledge Discovery and Data Mining, pp: 9-15, AAAI Press, 1998.

[59] H. Nagesh, S. Goil, and A. Choudhary, "*A Scalable Parallel Subspace Clustering Algorithm for Massive Data Sets*", Proceedings of International Conference on Parallel Processing, pp: 477-484, 2000.

[60] V. Kumar, A. Grama, A. Gupta, and G. Karpis, Introduction to Parallel Computing, Design and Analysis of Algorithms, Benjamin / Cummings Pub. Co., Don Mills, 1994.

[61] Petrosino and M. Verde, "*P-AFLC: a Parallel Scalable Fuzzy Clustering Algorithm*", ICPR2004, Proceedings of the 17th International Conference on. Pattern Recognition, pp: 809-812, 2004.

[62] K. Stoffel and A. Belkoniene, "*Parallel K-Means Clustering for Large Data Sets*", Proceedings Euro-Par '99, LNCS 1685, pp: 1451-1454, 1999.

[63] T. Kwok, K. Smith, S. Lozano, and D. Taniar. "*Parallel Fuzzy c-Means Clustering for Large Data Sets*", Proceedings EUROPAR02, vol. 2400 of *LNCS*, pp: 365-374, 2002.

[64] P. Alevizos and D. Tasoulis, and M. Vrahatis. "*Parallelizing the Unsupervised k-windows Clustering Algorithm*". Lecture Notes in Computer Science, Vol. 3019, pp: 225-232, 2004.

[65] D. Jiménez and V. Vidal, "*Parallel Implementation of Information Retrieval Clustering Models*", Lecture Notes in Computer Science, Vol. 3402, pp: 129-141, 2005.

[66] C. Pizzuti and D. Talia, "*P-AutoClass: Scalable Parallel Clustering for Mining Large Data Sets*", IEEE Transactions on Knowledge and Data Engineering, Vol.15, Issue 3, pp: 629-641, 2003.

[67] R. Hanisch," *Distributed Data Systems and Services for Astronomy and the Space Sciences*", in ASP Conf. Ser., Vol. 21 6, Astronomical Data Analysis Software and Systems IX, pp: 201-210, 2000.

[68] D. Tasoulis and M. Vrahatis. "*Unsupervised Distributed Clustering*", *IASTED* International Conference on Parallel and Distributed Computing and Networks, pp: 347-351, 2004.

[69] S. Krishnaswamy, S. Loke, and A. Zaslavsky. "*Cost Models for Heterogeneous Distributed Data Mining*". In Proceedings of the 12th International Conference on Software Engineering and Knowledge Engineering (SEKE), pp: 31–38, 2000.

[70] S. Datta, K. Bhaduri, C. Giannella, R. Wol, and H. Kargupta. "*Distributed Data Mining in Peer-to-Peer Networks*". IEEE Internet Computing, 10(4), pp: 18–26, 2006.

[71] S. Datta, C. Giannella, and H. Kargupta. "*K-means Clustering over Peer-to-Peer Networks*". In Workshop on High Performance and Distributed Mining (HPDM05). SIAM International Conference on Data Mining (SDM05), 2005.

[72] E. Johnson and H. Kargupta, "*Collective, Hierarchical Clustering from Distributed, Heterogeneous Data*", Large-Scale Parallel KDD Systems, Data Mining, Lecture Notes in Computer Science, Vol. 1759, pp: 221-244, Springer Verlag, 1999.

[73] H. Kargupta, W. Huang, K. Sivakumar, E. Johnson, "*Distributed Clustering Using Collective Principal Component Analysis*", Knowledge and Information Systems, Journal. Vol. 3, pp: 422-448, 2001.

[74] H. Kargupta and P. Chan, Advances in Distributed and Parallel knowledge Discovery. AAAI/MIT Press, 2000.

[75] H. Kargupta, B. Park, D. Hershberger, and E. Johnson, "*Collective Data Mining: A new Perspective toward Distributed Data Mining*". Advances in Distributed and Parallel Knowledge Discovery, pp: 131-178, AAAI/MIT Press, 1999.

[76] M. Klusch, S. Lodi, and G. Moro. "*Agent-based Distributed Data Mining: The KDEC scheme*". In AgentLink, pp: 104-122, 2003.

[77] N. Samatova, G. Ostrouchov, A. Geist, and A. Melechko. "*RACHET: An efficient cover-based Merging of Clustering Hierarchies from Distributed Datasets*". Distributed and Parallel Databases, 11(2), pp: 157–180, 2002.

[78] S. Merugu and J. Ghosh. "*Privacy-preserving Distributed Clustering using Generative Models*". In IEEE International Conference on Data Mining (ICDM), pp: 211-218, 2003.

[79] E. Januzaj, H. Kriegel, and M. Pfeifle. "*DBDC: Density based Distributed Clustering*". In EDBT, pp: 88–105, 2004

[80] K. Hammouda and M. Kamel. "*Collaborative Document Clustering*". In Proceedings of the Sixth SIAM International Conference on Data Mining (SDM06), pp: 453-463, Bethesda, MD, 2006.

[81] J. Li and R. Morris, "*Document Clustering for Distributed Full Text Search*". In 2nd MIT Student Oxygen Workshop, 2002.

[82] J. da Silva, C. Giannella, R. Bhargava, H. Kargupta, and M. Klusch. "*Distributed Data Mining and Agents*". Engineering Applications of Artificial Intelligence, 18(7), pp: 791–807, 2005.

[83] K. Hammouda, "*Distributed Document Clustering and Cluster Summarization in Peer-to-Peer Environments*", PhD Thesis, Dept. of System Design Engineering, University of Waterloo, 2007.

[84] Dhillon and D. Modha. "*A Data Clustering Algorithm on Distributed Memory Multiprocessors*", Large Scale Parallel Data Mining, LNCS Vol. 1759, pp: 245-260, Springer, 2000.

[85] Z. Hung. "*k-means-type Algorithms on Distributed Memory Computer*", International Journal of High Speed Computing, Vol. 11, pp: 75-91, 2000.

[86] W. Gropp E. Lusk and A. Skjellum, "*Using MPI, Portable Parallel Programming with Message Passing Interface*". The MIT Press, Cambridge, MA, 1996. [http://www-unix.mcs.anl.gov/mpi/mpich].

[87] E. Januzaj, H. Kriegel, M. Pfeifle. "*Towards Effective and Efficient Distributed Clustering*", Proceedings International Workshop on Clustering Large Data Sets (ICDM), pp: 49-58, 2003.

[88] S. Monti, P. Tamayo, J. Mesirov, and T. Golub, "*Consensus Clustering: A resampling-based Method for Class Discovery and Visualization of Gene expression Microarray data*", Kluwer Academic Publishers, pp: 91-118, 2003.

[89] S. Tavazoie, J. Hughes, M. Campbell, R. Cho and G. Church. "*Systematic Determination of Genetic Network Architecture. Nature Genetics*" V.22: pp: 281-285, 1999.

[90] M. West, C. Blanchette, H. Dressman, E. Huang, S. Ishida, R. Spang, H. Zuzan, J. Olson, J.Marks, and J. Nevins. "*Predicting the Clinical Status of Human Breast Cancer by using Gene Expression Profiles*" Proc Natl Acad Sci, pp: 11462 –11467, 2001.

[91] www.sciencemag.org/feature/data/984559.sh.

[92] T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. Mesirov, H. Coller, M. Loh, J. Downing, M.A. Caligiuri, C. Bloomfield, and E. Lander, "*Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring*", in Science Vol. 286. no. 5439, pp: 531-537, 1999.

[93] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. S. Lander, and T. Golub. "*Interpreting Patterns of Gene Expression with Self-organizing Maps: Methods and Application to Hematopoietic Differentiation*". PNAS, 96, pp: 2907-2912, 1999.

[94] M. Eisen, P. Spellman, P. Brown, and D. Botstein, "*Cluster Analysis and Display of Genome-wide Expression Patterns*", Proceedings of the National Academy of Sciences of the United States of America, Volume 95, Issue 25, pp: 14863-14868, 1998.

[95] D. Boley, M. Gini, R. Gross, S. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. "*Partitioning-Based Clustering for Web Document Categorization*". Decision Support Systems, Vol.27, pp: 329-341, 1999.

[96] D. Boley, M. Gini, R. Gross, S. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. "*Document Categorization and Query Generation on the World Wide Web using WebACE*". AI Review, 13(5-6), pp: 365–391, 1999.

[97] K. Aas and L. Eikvil. "*Text Categorization: A Survey*". Technical Report 941, Norwegian Computing Center, 1999.

[98] G. Salton, A. Wong, and C. Yang. "*A Vector Space Model for Automatic Indexing*". Communications of the ACM, 18(11), pp: 613-620, 1975.

[99] M. F. Porter. "*An algorithm for suffix stripping*". Program, 14(3), pp: 130-137, 1980.

[100] G. Snedecor and W. Cochran, Statistical Methods, Eighth Edition, Iowa State University Press, 1989.

[101] http://www.itl.nist.gov/div898/handbook/eda/section3/eda353.htm.

[102] B. Ghosh-Dastidar and J.L. Schafer, "*Outlier Detection and Editing Procedures for Continuous Multivariate Data*". Journal of Official Statistics, Vol.22, No.3, pp: 487-506, 2006.

[103] J. Liu and P. Gader, "*Neural Networks with Enhanced Outlier Rejection Ability for Off-line Handwritten Word Recognition*". The journal of the Pattern Recognition society, Volume 35, Issue 10, pp: 2061- 2071, 2002.

[104] C. Aggarwal and P. Yu, "*Outlier Detection for High Dimensional Data*". In Proceedings of the ACM SIGMOD International Conference on Management of *Data*, Volume 30, Issue 2, pp: 37 - 46, 2001.

[105] S. Ramaswamy, R. Rastogi and K. Shim, "*Efficient Algorithms for Mining Outliers from Large Data Sets*". In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Volume 29, Issue 2, pp: 427-438, 2000.

[106] E. Knorr and R. Ng, "*A Unified Notion of Outliers: Properties and Computation*". In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, pp: 219 -222, 1997.

[107] K. Yamanishi and J. Takeuchi, "*Discovering Outlier Filtering Rules from Unlabeled Data: Combining a Supervised Learner with an Unsupervised Learner*". In Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp: 389 -394, 2001.

[108] W. Jin, A. Tung and J. Han, "*Mining Top-n Local Outliers in Large Databases*". In Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp: 293-298, 2001.

[109] G. Williams, R. Baxter, H. He, S. Hawkins, and L. Gu, "*A Comparative Study for RNN for Outlier Detection in Data Mining*". In Proceedings of the 2nd IEEE International Conference on Data Mining, pp: 709-712, 2002.

[110] M. Jaing, S. Tseng, and C. Su, "*Two-phase Clustering Process for Outlier Detection*". *Pattern Recognition Letters*, Volume 22, Issue 6–7, pp: 691–700, 2001.

[111] S. Guha, R. Rastogi, and K. Shim, "*ROCK: A Robust Clustering Algorithm for Categorical Attributes*". In Proceedings of the 15th International Conference on Data Engineering (ICDE), pp: 512-521, 1999.

[112] R. Kashef and M. Kamel, "*Towards Better Detection of Outliers*", IEEE International Conference on BioInfomratics and BioEngineering BIOTECHNO, pp: 149-154, 2008.

[113] M. Eisenhardt, W. Muller, and A. Henrich. "*Classifying Documents by Distributed p2p Clustering*". In Informatik 2003: Innovative Information Technology Uses, pp: 286-291, 2003.

[114] FastTrack website, http://www.fasttrack.nu/

[115] P. Zheng and C. Wang, "*SODON: A High Availability Multi-Source Content Distribution Overlay*," in Proceedings of ICCCN, pp: 87-92. 2004.

[116] J. Li, and S. Vuong, "An *Efficient Clustered Architecture for P2P Networks*," in Proceedings of the 18th International Conference on Advanced Information Networking *and Applications*, pp: 278-283. 2004.

[117] V. Lo, D. Zhou, Y. Liu, C. Dickey, and J. Li, "*Scalable Super-node Selection in Peer-to-Peer Overlay Networks*", Proceedings of the 2005 Second International Workshop on Hot Topics in Peer-to- Peer Systems (HOT-P2P'05), pp: 18-27, 2005.

[118] C. Y. Suen. "*N-gram Statistics for Natural Language Understanding and Text Processing*", IEEE Transactions on Pattern Analysis and Machine Intelligence", 1(2), pp: 164–172, 1979.

[119] O. Zamir and O. Etzioni. "*Web Document Clustering: A Feasibility Demonstration"*. In Proceedings of the 21st Annual International ACM SIGIR Conference, pp: 46–54, 1998.