T. C.
BAHÇEŞEHİR ÜNİVERSİTESİ

NEW CLUSTER ENSEMBLE ALGORITHM WITH AUTOMATIC CLUSTER NUMBER AND NEW PRUNING TECHNIQUE FOR FAST DETECTION OF NEIGHBORS ON BINARY DATA

Master of Science Thesis

Mehmet Emin AKŞEHİRLİ

Istanbul, 2011

T. C.
BAHÇEŞEHİR ÜNİVERSİTESİ
The Graduate School of Natural and Applied Sciences
Computer Engineering

NEW CLUSTER ENSEMBLE ALGORITHM WITH AUTOMATIC CLUSTER NUMBER AND NEW PRUNING TECHNIQUE FOR FAST DETECTION OF NEIGHBORS ON BINARY DATA

Master of Science Thesis

Mehmet Emin AKŞEHİRLİ

Supervisor: Asst. Prof. Dr. Selim Necdet MİMAROĞLU

Istanbul, 2011

T. C.
BAHÇEŞEHİR ÜNİVERSİTESİ
The Graduate School of Natural and Applied Sciences
Computer Engineering

Title of the Master's Thesis	:	New Cluster Ensemble Algorithm with Automatic Cluster Number and New Pruning Technique for Fast Detection of Neighbors on Binary Data
Name/Last Name of the Student	:	Mehmet Emin AKŞEHİRLİ
Date of Thesis Defense	:	17 June 2011

The thesis has been approved by the Graduate School of Natural and Applied Sciences.

Assoc. Prof. Dr. F. Tunç BOZBURA
Acting Director

This is to certify that we have read this thesis and that we find it fully adequate in scope, quality and content, as a thesis for the degree of Master of Science.

Examining Commitee Members:

Asst. Prof. Dr. Selim Necdet MİMAROĞLU (Supervisor) :

Asst. Prof. Dr. İsmail ARI :

Asst. Prof. Dr. Devrim ÜNAY :

ACKNOWLEDGMENTS

I am very grateful to my wife Gunes and my family for their endless support, patience and trust. Without them I would not be able find the courage and strength to walk this path.

I am very thankful to my supervisor, Dr. Selim Mimaroglu for giving me the opportunity to step into an academic career and guiding me trough its challenges and benefits. He showed me what is essential for research and provided me invaluable knowledge that I will use my entire life.

I thank Dr. Ismail Ari and Dr. Devrim Unay for their precious time, effort and constructive criticism. I also thank to Dr. Taskin Kocak, Dr. Cagri Gungor and all of the BSU Computer Engineering faculty members for their guidance and mentoring.

I thank to my fellow researchers Murat Yagci and Ertunc Erdil for providing a collaborative research environment that I am very delightful to be a member of. I also thank to my fellow colleagues, Ozgur Ates, Erdem Erzurum, Jbid Arsenyan and Ceyhun Ulker for making my days in BSU worth to remember.

17 June 2011 Mehmet Emin AKŞEHİRLİ

ABSTRACT

NEW CLUSTER ENSEMBLE ALGORITHM WITH AUTOMATIC CLUSTER NUMBER AND NEW PRUNING TECHNIQUE FOR FAST DETECTION OF NEIGHBORS ON BINARY DATA

Akşehirli, Mehmet Emin

Computer Engineering
Supervisor: Asst. Prof. Dr. Selim Necdet MİMAROĞLU

June 2011, 73 Pages

Cluster analysis is to group similar, real or abstract data objects together in an unsupervised way. Cluster analysis, or clustering is a very important tool for data analysis and widely-used in almost every scientific field including data mining, machine learning, bioinformatics, and social network analysis. Unsupervised nature of clustering comes with unique opportunities and challenges. Applying the optimum clustering algorithm with correct parameters is not straight forward. Moreover, unlike classification algorithms which use the provided labels, clustering algorithms extract the information from the data itself, therefore most of the algorithms suffer from long execution times.

Combining multiple clusterings methods emerge as a promising solution that not only ease the algorithm and parameter selection for cluster analysis but also solve some unique clustering problems. In this theses we discuss the methods that combine multiple clusterings to obtain a better overall clustering of the data, including a recent method: DiCLENS. DiCLENS does not take any input arguments and finds the number of clusters automatically using objective measures. Although finding the co-associations between objects is a computationally expensive task, it is one of the strongest similarities in the field. DiCLENS utilizes a recent method to compute the similarities in an efficient way. Our experiments show that DiCLENS produces a better final clustering at almost all of the scenarios. Moreover execution time of the DiCLENS is very good compared to other methods.

We also discuss DBSCAN_BV, a novel method that improves the execution time performance of DBSCAN clustering algorithm by utilizing a pruning method on binary data and Hamming distance. DBSCAN is a well-known density-based algorithm. Even though space indexing techniques are widely used with DBSCAN, they do not perform well on categorical and binary data sets. Extensive tests show that DBSCAN_BV works up to 40 times faster than DBSCAN while keeping the same clustering accuracy. Tests also

show that the new pruning method allows the application of DBSCAN to resource limited environments.

Keywords: Clustering, Combining Multiple Clusterings, Clustering Ensemble, DBSCAN, DiCLENS

ÖZET

KÜME SAYISINI OTOMATİK BULAN BİR KÜMELENME BİRLEŞTİRME ALGORİTMASI VE İKİLİ VERİDE KOMŞULARIN HIZLI BULUNMASI İÇİN YENİ BUDAMA YÖNTEMİ

Akşehirli, Mehmet Emin

Bilgisayar Mühendisliği
Tez Danışmanı: Yrd. Doç. Dr. Selim Necdet MİMAROĞLU

Haziran 2011, 73 Sayfa

Kümeleme, birbirine benzeyen gerçek ya da soyut nesnelerin denetimsiz bir biçimde bir araya gruplanmasıdır. Küme analizi ya da kümeleme, veri analizi için çok önemli bir araçtır ve veri madenciliği, makina öğrenmesi, bioinformatik ve sosyal ağ analizi de dahil olmak üzere neredeyse bütün bilimsel alanlarda sıklıkla kullanılır. Kümelemenin denetimsiz doğası özgün fırsatlara ve sorunlara neden olur. Doğru kümeleme algoritmasını veriye uyacak parametreler ile uygulamak kolay değildir. Dahası, sağlanan etiketleri kullanan sınıflama algoritmalarının aksine kümeleme algoritmaları bilgiyi verinin kendisinden çıkarttığı için çoğu algoritmanın çalışması uzun sürer.

Çoklu kümelemeleri birleştiren metodlar yalnızca algoritma ve parametre seçimini kolaylaştıran değil aynı zamanda bazı özgün kümeleme sorunlarını da çözen, umut vadeden çözümler olarak belirmiştir. Bu tezde daha iyi bir kümeleme elde etmek için eldeki çoklu kümelemeleri birleştiren metodları ve bunlardan biri olan DiCLENS'i gösteriyoruz. DiCLENS hiç bir argüman almadan çalışır ve nesnel ölçümler kullanarak kümelerin sayısını otomatik olarak bulur. Nesneler arasında eş-atamaların bulunması fazla hesaplama gerektirse de, eş-atamalar alandaki en güçlü benzerliklerden biridir. DiCLENS benzerlikleri etkin bir biçimde hesaplamak için yeni bir metod kullanmaktadır. Deneylerimiz DiCLENS'in neredeyse bütün senaryolarda daha iyi bir sonuç kümelemesi ürettiğini göstermiştir. Dahası diğer metodlar ile karşılaştırıldığında DiCLENS'in çalışma zamanı oldukça iyidir.

Aynı zamanda, ikili veri ve Hamming uzaklığı üzerinde bir budama yöntemi kullanarak DBSCAN kümeleme algoritmasının çalışma hızı performasını artıran DBSCAN_BV'yi de gösteriyoruz. DBSCAN oldukça iyi bilinen bir yoğunluk temelli kümeleme algorit-

masıdır. Uzam dizinleme teknikleri DBSCAN ile birlikte yaygın olarak kullanılsa da, bu teknikler kategorik ve ikili veri setlerinde düşük performans gösterirler. Yoğun testler, kümeleme doğruluğu aynı kalmakla birlikte DBSCAN_BV'nin DBSCAN'den 40 kata kadar daha hızlı çalıştığını göstermiştir. Testler aynı zamanda yeni budama metodunun DBSCAN'in kaynağı sınırlı olan ortamlarda da kullanımının yolunu açtığını göstermektedir.

Anahtar Kelimeler: Kümeleme, Çoklu Kümelemelerin Birleştirilmesi, Kümeleme Topluluğu, DBSCAN, DiCLENS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Adjusted Rand Index	:	ARI
Agglomerative Nesting clustering algorithm	:	agnes
Bit Vector based DBSCAN	:	DBSCAN_BV
Byte Aligned Bitmap Coding	:	BBC
Central Processing Unit	:	CPU
Cluster-based Similarity Partitioning Algorithm	:	CSPA
Combining Multiple Clusterings using Similarity Graph	:	COMUSA
Cophenetic Correlation Coefficient	:	CPCC
Density-Based Clustering	:	DENCLUE
Density-Based Spatial Clustering of Applications with Noise	:	DBSCAN
Divisive Cluster Ensemble	:	DiCLENS
Enhanced Word Aligned Hybrid code	:	EWAH
Evidence Accumulation Algorithm	:	EAC
Expectation Maximization	:	EM
Fast DBSCAN	:	FDBSCAN
GNU is Not Unix	:	GNU
Graph-based Consensus Clustering	:	GCC
Hybrid Bi-partite Graph Formulation	:	HBGF
Hyper-Graph Partitioning Algorithm	:	HGPA
Inter Cluster Similarity	:	ECS
Intra Cluster Similarity	:	ICS
Link-based Cluster Ensembles	:	LCE
Meta-clustering Algorithm	:	MCLA
Minimal Spanning Tree	:	MST
Partitioning-based DBSCAN	:	PDBSCAN
Sampling-based DBSCAN	:	SDBSCAN
Similarity-based Minimal Spanning Tree	:	SMST
Word Aligned Hybrid code	:	WAH

LIST OF SYMBOLS

A partition, clustering of D	:	$\pi(D)$
Average distance between clusters C_i and C_j	:	$dist_{avg}(d_i, d_j)$
Big-Oh notation for computational complexity	:	O
Centroid of i^{th} cluster	:	$\mathbf{c}_i$
Class	:	$\mathbb{C}_i$
Cluster	:	C_i
Cluster Ensemble of D	:	$\Pi(D)$
Co-association matrix	:	M
Complete link distance between clusters C_i and C_j	:	$dist_{\max}(d_i, d_j)$
Data object	:	d_i, d, d'
Data set	:	D
Distance between data objects d_i and d_j	:	$dist(d_i, d_j)$
Edge set of the graph	:	E
Empty set	:	$\emptyset$
Final, output partition	:	$\pi^{\star}(D)$
Floor of a: largest integer number that is less than a	:	$\lfloor a \rfloor$
Inter Cluster Similarity	:	ECS
Inter Cluster Similarity of a partition	:	ECS_{π}
Intra Cluster Similarity	:	ICS
Intra Cluster Similarity of a partition	:	ICS_{π}
i^{th} attribute	:	$\mathbf{a_i}$
i^{th} bit value of bit vector a	:	$(a)_j$
i^{th} Meta-cluster: cluster of clusters	:	$C_i^{\star}$
k^{th} Cluster of i^{th} partition	:	C_{ik}
min-max normalized ECS_{π}	:	$\overline{ECS_{\pi}}$
min-max normalized ICS_{π}	:	$\overline{ICS_{\pi}}$
Neighborhood distance	:	ε
Number of clusters	:	k
Partition, clustering	:	π
Point in data space	:	$\mathbf{p}$,$\mathbf{p}'$
Potential neighbors of i^{th} data object	:	$\mathbf{ngh_i}$
Quality function	:	ϕ
Set of 1's in i^{th} data object	:	$Ones_i$
Set of column bit vectors of data set D	:	D_{cols}
Set of real numbers	:	$\mathbb{R}$
Set of row bit vectors of data set D	:	D_{rows}
Set Union	:	$\bigcup$
Similarity between data objects d_i and d_j	:	$sim(d_i, d_j)$
Single link distance between clusters C_i and C_j	:	$dist_{\min}(d_i, d_j)$
Size of set A	:	$\|A\|$
Sum	:	$\sum$

Vertex set of the graph	:	V
Weighted undirected graph	:	G
Weight relation of the graph	:	W

1. INTRODUCTION

1.1 CLUSTERING

Clustering, or *Cluster Analysis*, is the process of organizing data objects into previously unknown groups, i.e., clusters. Data objects in the same cluster are similar to each other and dissimilar to the data objects in the other clusters. Unlike classification, cluster analysis is an unsupervised process and it is also called *Unsupervised Learning*. Clustering methods find the relations in the data just by using the similarities of the data points. Therefore, clustering becomes the only option if a labeled training set is not avaiable, i.e., the true labels of the data is not known beforehand.

Similarity measure, which is generally defined on the attributes of a data set, has a major impact on clustering results and it must be selected according to the clustering needs. Moreover, not every similarity measure can be used with every clustering algorithm. For instance, similarity metrics that are only defined between data objects can not be used with algorithms that define pseudo points in the data space during the clustering process, such as k-means (MacQueen 1967).

Clustering is used to find the *segment*s of the data, which are very useful for data analysis. They can be labeled by an expert to define the characteristics of the data or used to define similar data objects. Also, by using representative objects for the segments (groups), data can be compressed.

There are many fields that use clustering as an essential tool for data discovery. Data Mining deal with clustering methods that work on very large data. In Machine Learning, clustering is used to detect similarities in the complex formations of data. Clustering is also used in computer vision (Fang et al. 2010, Kannan et al. 2010, Elnakib et al. 2011, Bandyopadhyay 2011), bioinformatics (Bin and Risso 2011), medical data analysis (Joshi et al. 2010, Greene et al. 2004), social network analysis (Stanoev et al. 2011), intelligent transportation systems (Yucenur and Demirel 2011, Faouzi et al. 2011), wireless sensory networks (Guo et al. 2011), and time-series data analysis (Frank et al. 2010, Lai et al. 2010).

Clustering is also used for *outlier detection*. If a data object is not a member of any group, it is called an *outlier*. Outlier detection can be used for detection of anomalies and frauds. As an example, in a medical examination, if a correlation between heat exposure and headache exists only for one patient, that patient may have serious health problems.

Clustering is a very useful but tough research field because of its unsupervised nature. It is not easy to determine the optimum clustering algorithm and tts parameters to fit to the data. Figure 1.1a, Figure 1.1b, Figure 1.1c and Figure 1.1d shows clustering results of k-means algorithm for a two dimensional data with varying parameters and configureations. In the figures every color and shape represents a cluster of the data. that the clustering result change dramatically with the parameters, even for the same clustering algorithm.

There are numerous clustering algorithms in the literature and, as stated by Jain (2010), there is still room for new algorithms. Furthermore, state of the art algorithms are still developed for obtaining better accuracy, faster execution and scalability. As explained in detail in Chapter 2, methods that combine multiple clusterings fuse the information in different clusterings to get a better clustering results.

We give a brief overview of clustering methods in Section 1.2 and methods to evaluate clustering quality in Section 1.3. We discuss a novel Clustering Ensemble Technique in Chapter 2 and a novel pruning technique for fast detection of neighbors on binary data in Chapter 3.

1.2 CLUSTERING METHODS

In this section we give a brief description of the clustering methods. Clustering methods can be grouped into following categories.

- Partitioning methods
- Hierarchical methods
- Density-based methods
- Grid-based methods
- Model-based methods

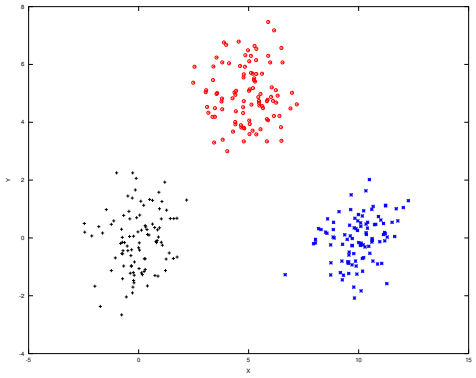

Figure 1.1a k-means with $k = 3$

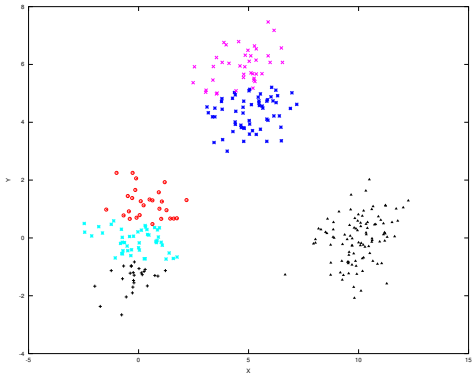

Figure 1.1b k-means with $k = 6$

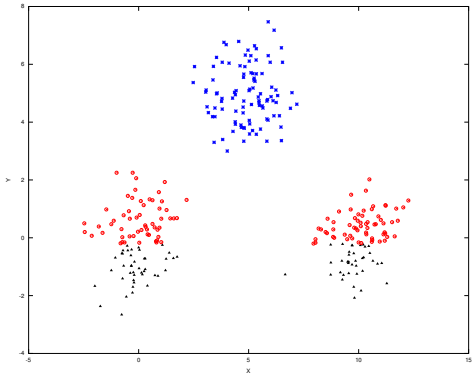

Figure 1.1c k-means on y-dimension with $k = 3$

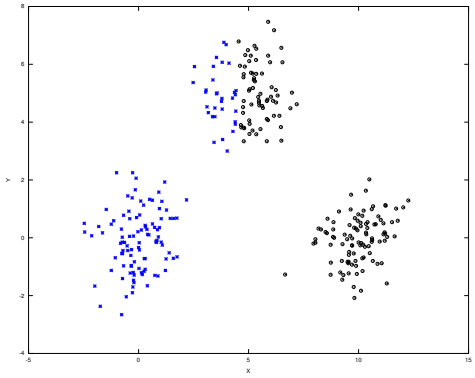

Figure 1.1d k-means on x-dimension with $k = 2$

Figure 1.1: Clustering results of k-means algorithm.

- Graph-based methods

1.2.1 Partitioning Methods

Partitioning methods divide the data set into k groups, where each group represents a cluster. Objects of the same group are expected to be similar to each other and dissimilar to objects in other groups. The most frequently used and the most well known partitioning methods are the centroid based ones, i.e. k-means and k-medoids.

k-means algorithm (MacQueen 1967) takes the number of clusters k, data set D and a distance metric as input parameters. The aim of the algorithm is to partition data set into k clusters that will minimize the sum of square errors in a clustering. k-means starts by

randomly selecting initial cluster centers. It assigns every object to its nearest cluster center. After the assignment, cluster centers are re-computed. This process is repeated until cluster centers converge.

1.2.2 Hierarchical Methods

Hierarchical methods organize the objects into a tree form where every node is a cluster consisting of its child nodes. Hierarchical clustering methods can be further divided into two sub groups: agglomerative (bottom up, merging) and divisive (top down, splitting).

AGglomerative NESting (Jardine and Sibson 1971, Sneath and Sokal 1962), which is a characteristic example of agglomerative methods, starts by considering each object as a singleton cluster. The algorithm iteratively finds the most similar clusters and merge them to form a new cluster until every object become a member of a one single cluster or another termination condition is met. Termination condition is a user specified parameter which can either be the number clusters or a data centric measure such as cluster compactness.

Divisive methods, on the other hand, start by putting all of the objects into one big cluster and then iteratively split the clusters. A good example for divisive methods is Minimal Spanning Tree (MST) (Zahn 1971) algorithm which first constructs a MST of the data and then iteratively removes the minimum weighted edge until a termination condition is met.

Hierarchical clustering methods use several ways to compute the similarity between clusters. Let $dist(d_i, d_j)$ be the distance between data objects d_i and d_j, then popular distances between two clusters C_i and C_j can be computed as follows:

- Minimum Distance, or Single Link, between C_i and C_j, $dist_{min}(C_i, C_j)$, is computed as the minimum distance between their corresponding objects.

$$dist_{min}(C_i, C_j) = \min dist(d_i, d_j), d_i \in C_i, d_j \in C_j \tag{1.1}$$

- Maximum Distance, or Complete Link, between C_i and C_j, $dist_{max}(C_i, C_j)$, is computed as the maximum distance between their corresponding objects.

$$dist_{max}(C_i, C_j) = \max dist(d_i, d_j), d_i \in C_i, d_j \in C_j \tag{1.2}$$

- Average Distance between C_i and C_j, $dist_{avg}(C_i, C_j)$, is computed as the average of pairwise distances of objects between two clusters.

$$dist_{avg}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{d_i \in C_i} \sum_{d_j \in C_j} dist(d_i, d_j) \tag{1.3}$$

1.2.3 Density-based Methods

Density-based clustering methods find the high and low density areas in the data set as clusters. Density-based methods effectively find arbitrary shaped clusters and noise when provided with correct parameters.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) (Ester et al. 1996), which is a well known density-based method, identifies the dense areas in a data set by labeling the points as *core*, *border* or *noise*. A point is labeled as core if it has a certain minimum number (*MinPts*) of neighbors in its ε neighborhood. *MinPts* and ε are user specified parameters that dramatically change the final clusters. Neighbors of core points are included into the same cluster with the core point. A *border* point is a point which is not a core point but in the ε-neighborhood of a core point. Points that are neither core nor border are labeled as *noise*.

OPTICS (Ordering Points to Identify the Clustering Structure) (Ankerst et al. 1999) and DENCLUE (DENsity-based CLUstEring) (Hinneburg and Keim 1998) are also well known density-based clustering methods.

1.2.4 Grid-based Methods

Grid-based methods divide the object space into finite number of cells by quantizing each attribute. Defining grid size is not straight forward. Generally quantization techniques, such as equal frequency binning or equal length binning, are used to divide the attribute (Tan et al. 2005). After grid is formed each object is assigned to one of the cells.

An intuitive method is to perform a density-based clustering on grid-structure by identifying and connecting the high-density cells. Connected dense cells can be regarded as clusters. Density of a cell can be defined as the number of objects in that cell, if equal length binning is performed, or it can be the size of a cell, if equal frequency binning is performed.

Grid-based methods are very fast since they do not depend on the number of objects, however determining the correct grid size, besides other parameters, is not easy.

1.2.5 Model-based Methods

Model-based methods assume that the input data is aligned with one or more probabilistic density distributions where each distribution defines a cluster. Identifying these distributions and their parameters also identifies the clusters in the data set.

A well-known model-based method is Expectation Maximization (EM), which is a generalized and extended version of k-means. EM finds the parameters of k different probability distributions that best fit the data. EM starts with an initial parameter vector and iteratively assigns objects to the distribution that maximize the probability. After all of the objects are assigned to clusters, parameters of each distribution is updated according to the assigned objects.

1.2.6 Graph-based Methods

Graph-based clustering methods organize the data set as a graph structure. Generally, nodes of the graph are selected as data objects and the edges between them are the corresponding proximities. Graph partitioning techniques can be utilized for clustering purposes. Some key approaches of graph-based methods include; sparsifying the graph for

faster and more accurate calculation and defining a new neighborhood-based similarity metric.

Some of the most known graph-based methods are: Minimum Spanning Tree (MST) Clustering (Zahn 1971), OPOSSUM (Strehl and Ghosh 2000), Chameleon (Karypis et al. 1999) and spectral clustering (Ng et al. 2001).

1.3 EVALUATION OF CLUSTER QUALITY

Unlike classification results, validation of clustering results is not straightforward. Different clustering methods may produce different clusters on the same data set. Moreover, almost every clustering method finds a clustering even if the distribution of data is not suitable for clustering, i.e., random.

Clustering Evaluation methods can be grouped into three categories:

Unsupervised. Clustering is validated by using only the information that is available while clustering, such as the similarities between objects. Sum of squared errors is an example for unsupervised evaluation methods.

Supervised. Clustering is validated by using an information that is not used for clustering, such as real class labels. Even though supervised techniques are very accurate, external information is not available in most of the scenarios, therefore they can not be used. Entropy and Rand Index (Rand 1971) are examples for supervised validation methods.

Relative. Clustering is validated by comparing the output with another clustering of the same data. Comparison can be done by supervised or unsupervised metrics. Calculating the mutual information between two clusterings is an example for relative validity measures.

1.3.1 Unsupervised Evaluation Methods

Cohesion, or compactness, measures how similar the objects in a cluster are. If objects in a cluster are similar to each other, then the cluster is said to be compact. Mathematical

definition of cohesion is given in Equation 1.4, where $similarity(d_i, d_j)$ is the similarity between data objects d_i and d_j.

$$cohesion(C_i) = \sum_{d_i \in C_i} \sum_{d_j \in C_i} similarity(d_i, d_j) \tag{1.4}$$

Separation is the measure of dissimilarity between two clusters. In a graph based manner, separation is measured as the weighted sum of pair-wise similarities of objects between corresponding clusters. For clusters C_i and C_j mathematical definition of separation is given in Equation 1.5, where w_{ij} is the assigned weight.

$$separation(C_i, C_j) = \sum_{d_i \in C_i} \sum_{d_j \in C_j} w_{ij}\ similarity(d_i, d_j) \tag{1.5}$$

If clusters are prototype-based, cohesion and separation can be mathematically defined as in Equation 1.6 and Equation 1.7, respectively, where $\mathbf{c}_i$ represents the centroid of cluster C_i.

$$cohesion(C_i) = \sum_{d_i \in C_i} similarity(d_i, \mathbf{c}_i) \tag{1.6}$$

$$separation(C_i, C_j) = similarity(\mathbf{c}_i, \mathbf{c}_j) \tag{1.7}$$

Another definition of separation for prototype-based clusterings is given in Equation 1.8, where $\mathbf{c}$ stands for the centroid of the whole data set.

$$separation(C_i) = similarity(\mathbf{c}_i, \mathbf{c}) \tag{1.8}$$

Cohesion of a clustering is defined as the weighted sum of the cohesions of the individual clusters and similarly, separation of a clustering is defined as the wighted sum of the pairwise separations of clusters. Mathematical definition for cohesion and separation of a clustering π is given in Equation 1.9 and Equation 1.10, respectively. In the formulas w_i and w_{ij} stands for the weights that are associated with the corresponding clusters. Weights are selected according to the application needs, but in general they are selected either as a normalizing factor proportional to the number of objects in the cluster or just 1.

$$cohesion(\pi) = \sum_{C_i \in \pi} w_i \, cohesion(C_i) \tag{1.9}$$

$$separation(\pi) = \sum_{C_i \in \pi} \sum_{C_j \in \pi} w_{ij} \, seperation(C_i, C_j) \tag{1.10}$$

Cohesion and separation are widely-used validity measures for globular shaped clusters. Even if the real clusters are not globular shaped, there are similarity measures that allow a globular relation between objects. ICS (Intra Cluster Similarity) and ECS (Inter Cluster Similarity) measures, which are explained in detail in Section 2.3, are two examples for this.

Silhouette Coefficients method (Kaufman et al. 1990) uses both separation and cohesion to compute the validity of a clustering. For each object in the clustering, silhouette coefficient is calculated in 3 steps:

1. Calculate object's average similarity to other objects in its own cluster and call this value a.

2. For each cluster not having the object, calculate object's average similarity to the objects in that cluster. Find the minimum of these values and call it b.

3. Silhouette Coefficient is $s = (a - b)/max(a, b)$

s is between -1 and 1. Negative values of s indicates that the object is similar to other clusters than its own cluster, which is not desirable. Validity of a clustering, in terms of silhouette coefficients, can be calculated by finding the coefficient for all of the objects and then taking a weighted sum of them.

Cophenetic Distance between two objects is the distance between their corresponding clusters just before an agglomerative clustering algorithm puts both objects into the same cluster. When two clusters C_i and C_j are merged, all of the pair-wise cophenetic distances between their corresponding objects will be equal to the distance between C_i and C_j. Cophenetic distance between two objects is also the magnitude of the node that connects these objects in a dendogram.

CoPhenetic Correlation Coefficient, CPCC, which is the correlation between original dissimilarities and cophenetic distances, is used to evaluate how well agglomerative algorithms fits the data.

1.3.2 Supervised Evaluation Methods

If the labels of classes in the data set are known, then the clustering quality can be evaluated with respect to this external information. Clustering is performed by disregarding the real labels of the data and labels are used just to check the clustering accuracy of the method at hand. Although classification evaluation methods can be used, simple modifications are needed because unlike classification results, clustering results do not have the label information.

Clustering evaluation methods that are originated from classification evaluation methods are:

Entropy is the degree of agreement between a clustering and the real classes in terms of mutual members. It is calculated as follows: For each cluster C_i and each class $\mathbb{C}_j$; p_{ij}, the probability of the membership of the members of cluster C_i to the class $\mathbb{C}_j$

is calculated as $p_{ij} = \frac{m_{ij}}{m_i}$, where m_i is the number of objects in cluster C_i, and m_{ij} is the number of objects that is a member of both the cluster C_i and the class $\mathbb{C}_j$. Entropy of each cluster C_i is $e_i = -\sum_{j=1}^{L} p_{ij} \log_2 p_{ij}$, where L is the number of the real classes. Entropy of a clustering is the weighted sum of all entropies of all clusters, $e = \sum_{i=1}^{K} \frac{m_i}{m} e_i$, where K is the number of clusters in the clustering and m is the number of objects.

Purity measures the agreement between the clustering and the real classes at its maximum level. Purity of a clustering is calculated as $purity = \sum_{i=1}^{K} \frac{m_i}{m} \max_j p_{ij}$, where the terms are defined above.

Precision is the ratio of the number of members of both the cluster C_i and the class $\mathbb{C}_j$ to the number of members of the cluster C_i, which is calculated as $precision(C_i, \mathbb{C}_j) = p_{ij}$.

Recall is the measure of representation capability for a cluster of a class. It is calculated as the ratio of number of members of both the cluster C_i and the class $\mathbb{C}_j$ to the number of members of the class $\mathbb{C}_j$, $recall(C_i, \mathbb{C}_j) = \frac{m_{ij}}{m_j}$, where m_j is the number of members of the class $\mathbb{C}_j$.

F-measure (Larsen and Aone 1999) is the measure of agreement between a cluster and a class, that does not disregard the false positives and false negatives. It is calculated as $F(C_i, \mathbb{C}_j) = 2 \times \frac{precision(C_i,\mathbb{C}_j) \times recall(C_i,\mathbb{C}_j)}{precision(C_i,\mathbb{C}_j) + recall(C_i,\mathbb{C}_j)}$.

Methods that find the similarity between two clusterings can be used to find the similarity between the clustering and the real classes. High similarity between the clustering and he real classes implies a high quality clustering. Two of these methods, Jaccard similarity and Rand Index, use *contingency matrix* for calculation.

Contingency matrix consist of two rows and two columns, showing the pair-wise object relations between the clustering and the real classes.

Table 1.1: Contingency matrix

	Same Class	Different Class
Same Cluster	f_{11}	f_{01}
Different Cluster	f_{10}	f_{00}

Values on the cells of the Table 1.1 are defined as,

f_{11} number of object pairs that are both in the same cluster and in the same class

f_{01} number of object pairs that are in the same cluster but not in the same class

f_{10} number of object pairs that are not in the same cluster but in the same class.

f_{00} number of object pairs that are neither in the same cluster nor in the same class

Using the definitions above, Jaccard Similarity between a clustering and the real class labels is defined in Equation 1.11.

$$Jaccard = \frac{f_{11}}{f_{01} + f_{10} + f_{11}} \tag{1.11}$$

Rand Index (Rand 1971), which is another popular measure of similarity between two clusterings, is the ratio of agreements between the clusterings to the total number of cases. Formal definition of the Rand Index is given in Equation 1.12.

$$Rand = \frac{f_{00} + f_{11}}{f_{00} + f_{01} + f_{10} + f_{11}} \tag{1.12}$$

Adjusted Rand Index (ARI) (Hubert and Arabie 1985), which is a corrected for chance version of Rand Index (Rand 1971), is defined as follows: Given two clusterings $\pi^0(D) = \{C_1^0, C_2^0, \ldots, C_{|\pi^0(D)|}^0\}$ and $\pi^1(D) = \{C_1^1, C_2^1, \ldots, C_{|\pi^1(D)|}^1\}$, where $C_i^0 \cap C_j^0 = \emptyset$ for $1 \leq i, j \leq |\pi^0(D)|$, and $C_i^1 \cap C_j^1 = \emptyset$ for $1 \leq i, j \leq |\pi^1(D)|$ with variables in Table 1.2 referring to;

$$p = |\pi^\star(D)| \ , \ r = |\pi^o(D)| \ , \ n_{ij} = |C_i^d \cap C_j^\star| \tag{1.13}$$

$$n_{i.} = \sum_{j=1}^{p} n_{ij} \ , \ n_{.j} = \sum_{i=1}^{r} n_{ij} \tag{1.14}$$

Table 1.2: Abbreviations for ARI

Class \ Cluster	$C_1^\star$	$C_2^\star$	$\dots$	$C_p^\star$	**Sums**
C_1^d	n_{11}	n_{12}	$\dots$	n_{1p}	$n_{1.}$
C_2^d	n_{21}	n_{22}	$\dots$	n_{2p}	$n_{2.}$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
C_r^d	n_{r1}	n_{r2}	$\dots$	n_{rp}	$n_{r.}$
Sums	$n_{.1}$	$n_{.2}$		$n_{.p}$	$n_{..} = n$

ARI is formulated as follows:

$$\frac{\sum_{i,j} \binom{n_{ij}}{2} - \left(\sum_i \binom{n_{i.}}{2} \sum_j \binom{n_{.j}}{2}\right) / \binom{n}{2}}{\frac{1}{2}\left(\sum_i \binom{n_{i.}}{2} + \sum_j \binom{n_{.j}}{2}\right) - \left(\sum_i \binom{n_{i.}}{2} \sum_j \binom{n_{.j}}{2}\right) / \binom{n}{2}} \tag{1.15}$$

ARI takes maximum value at 1 which indicates perfect match between two clusterings $\pi^\star(D)$ and $\pi^o(D)$.

1.4 COMBINING MULTIPLE CLUSTERINGS

Combining Multiple Clusterings (Clustering Ensembles or Clustering Fusion) is combining the information from different clusterings to produce a better overall clustering. We discuss some of the motivations for combining multiple clusterings:

Choosing the Clustering Algorithm Since there are many clustering algorithms in the literature (Jain 2010), it is very hard to choose the right clustering algorithm that best fit the data set, let aside deciding the correct parameters of the clustering algorithm. Furthermore, there are data sets that can not be fit by only one clustering

algorithm. Each different clustering, produced by one of the algorithms, contain a different information about the data. Thus, instead of choosing one of the available clusterings, combining the information from the clusterings can generate a much better clustering.

Knowledge from External Sources Clustering algorithms work either with attributes of the objects or on predefined similarities between the objects. Unavailability of these crucial information may prevent the clustering algorithms to function. However, there are several situations which require these kind of restrictions, including;

- Commercial applications that treat the class names and data source as a classified information,
- Situations that the prior groupings of the data must be used,
- Distributed environments that the entire source of data is not available on each site.

Combining multiple clusterings methods are emerged as a good solution for some unique problems. We discuss some of the methods that produce a better clustering by combining the input clusterings and compare them with a novel algorithm in Chapter 2.

2. DiCLENS: DiVISIVE CLUSTERING ENSEMBLE WITH AUTOMATIC CLUSTER NUMBER

2.1 COMBINING MULTIPLE CLUSTERINGS

In this chapter we provide the formal definition of combining multiple clusterings problem and then discuss a novel combining multiple clustering algorithm that can detect the number of clusters automatically.

Definition 2.1. *A data set D is defined as a set of data objects $D = \{d_1, d_2, \ldots, d_{|D|}\}$, where d_i is a data object and $|D|$ is the number of data objects in D.*

Definition 2.2. *For a data set D, a* clustering *(partition) of D can be stated as follows:*

$$\pi_i(D) = \{C_{i1}, C_{i2}, \ldots, C_{i|\pi_i(D)|}\}, \tag{2.1}$$

where C_{ik} is a cluster of $\pi_i(D)$, $1 \leq k \leq |\pi_i(D)|$, $|\pi_i(D)|$ is the number of clusters in $\pi_i(D)$, and

$$\bigcup_{k=1}^{|\pi_i(D)|} C_{ik} \subseteq D \tag{2.2}$$

Definition 2.3. *For data set D, $\Pi(D)$ is defined as* input clusterings*, such that,*

$$\Pi(D) = \{\pi_1(D), \pi_2(D), \ldots, \pi_{|\Pi(D)|}(D)\} \tag{2.3}$$

where, $\pi_i(D)$ is a clustering of D and $|\Pi(D)|$ is the number of clusterings.

Definition 2.4. Co-association*,* co-occurrence *or* evidence accumulation*, between two objects d_i and d_j is defined as the number of clusters in input clusterings that include both d_i and d_j.*

$$sim(d_i, d_j) = |\{C_{kl} | d_i \in C_{kl}, d_j \in C_{kl}\}|, C_{kl} \in \Pi(D) \tag{2.4}$$

Definition 2.5. *Co-association matrix M is defined as a similarity matrix where $M_{ij} = sim(d_i, d_j)$.*

2.2 RELATED WORK

Cluster-based Similarity Partitioning Algorithm (CSPA) (Strehl and Ghosh 2002), shown in Algorithm 2.1, is based on co-association matrix and METIS, which is a well-known graph partitioning algorithm and software package (Karypis and Kumar 1998). Evidence Accumulation (EAC) (Fred and Jain 2005) uses hierarchical clustering algorithms to partition the co-association matrix as shown in Algorithm 2.2.

Algorithm 2.1 Cluster-Based similarity partitioning algorithm **CSPA**

Input: $\Pi(D)$: Multiple Clusterings
n: Number of Objects
k: Number of Final Clusters
Output: $\pi^*(D)$: Final Clustering

Initialize SM as an $n \times n$ matrix
// Construct similarity matrix
for all $d_i \in D$ **do**
  for all $d_j \in D$ **do**
    for all $\pi_k(D) \in \Pi(D)$ **do**
      for all $C_{kl} \in \pi_k(D)$ **do**
        if $d_i \in C_{kl} \wedge d_j \in C_{kl}$ **then**
          $SM_{ij} := SM_{ij} + 1$
$\pi^\star(D) := \mathbf{METIS}(SM, k)$
return $\pi^\star(D)$

Algorithm 2.2 Evidence accumulation **EAC**

Input: $\Pi(D)$: Multiple Clusterings, n: Number of Objects
Output: $\pi^*(D)$: Final Clustering

Initialize SM as an $n \times n$ matrix
// Construct similarity matrix
for all $d_i \in D$ **do**
  for all $d_j \in D$ **do**
    for all $\pi_k(D) \in \Pi(D)$ **do**
      for all $C_{kl} \in \pi_k(D)$ **do**
        if $d_i \in C_{kl} \wedge d_j \in C_{kl}$ **then**
          $SM_{ij} := SM_{ij} + 1$
Run Agglomerative Clustering on SM to construct $\pi^*(D)$
return $\pi^\star(D)$

COMUSA is a recent work by Mimaroglu and Erdil (2011), which uses a novel graph partitioning technique to partition the co-association matrix. It automatically finds the number of cluster when provided with correct parameters.

Meta-CLustering Algorithm (MCLA) (Strehl and Ghosh 2002), which is shown in Algorithm 2.3, creates a similarity graph of clusters. Edge weights of the graph are proportional to the Jaccard Similarity between corresponding clusters. Graph is partitioned using METIS to find meta-clusters. Objects are distributed between meta-clusters by utilizing weighted majority voting.

Algorithm 2.3 Meta-Clustering algorithm **MCLA**

Input: $\Pi(D)$: Multiple Clusterings
k: Number of Clusters In the Final Clustering
Output: $\pi^{\star}(D)$: Final Clustering
1: $G := (E, V, W)$ // Initialize Weighted Meta-Graph
2: $V := \emptyset$ // Set of Vertices
3: $E := \emptyset$ // Set of Edges
4: $W := \emptyset$, $W \subseteq E \rightarrow \mathbb{R}$ // Weight *Function* of Edges
5: **for all** $\pi_i(D) \in \Pi(D)$ **do**
6: **for all** $C_{ij} \in \pi_i(D)$ **do**
7: $V := V \cup C_{ij}$
8: **for all** $C_{ik} \in V$ **do**
9: **for all** $C_{jl} \in V, i \neq j, k \neq l$ **do**
10: $E := E \cup (C_{ik}, C_{jl})$
11: $W := W \cup (E, Jaccard(C_{ik}, C_{jl}))$
12: $\pi^{\star}(D) = \mathbf{METIS}(G, k)$
13: // Do majority voting
14: **for all** $d_i \in D$ **do**
15: Assign d_i to its most associated meta-cluster in $\pi^{\star}(D)$
16: **return** $\pi^{\star}(D)$

HyperGraph-Partitioning Algorithm (HGPA), which is also introduced in Strehl and Ghosh (2002), converts input clusterings into a hyper-graph, where vertices of the graph represent the objects and each input cluster become a hyper-edge across its member objects. Hyper-graph is partitioned by HMETIS (Karypis and Kumar 1999) algorithm to generate clusters.

Fern and Brodley (2004) proposes Hybrid Bi-partite Graph Formulation (HBGF) for cluster ensembles, which constructs a bi-partite graph, where vertices represent both objects and clusters. Unweighted edges connect the object vertices to the vertices that represent their assigned clusters. Final clustering is generated by partitioning the graph using

METIS or spectral clustering. Iam-on et al. (2010) proposes LCE: Link-based Cluster Ensembles, which improves the HBGF by augmenting the graph with edge weights.

MULTI-K (Kim et al. 2009) iteratively decrements the edge values of a co-association graph. At each iteration connected components represent a clustering and final clustering is selected among these clusterings. Graph-based Consensus Clustering (GCC) (Yu et al. 2007) produces final clustering by partitioning the co-association matrix with normalized graph cut algorithm. Gionis et al. (2007) proposes to model the problem as correlation clustering and partition the co-association matrix using the methods proposed for that problem. Wang et al. (2009) partitions a probabilistic version of co-association matrix.

Topchy et al. (2003) models the combining multiple clustering problem as a median partition problem. Two recent examples for this approach can be found in Ayad and Kamel (2010) and Vega-Pons et al. (2010). Evolutinary methods for combining multiple clusterings are proposed in Cristofor and Simovici (2002) and Mohammadi et al. (2008).

2.2.1 Weaknesses of Related Work

Most of the algorithms that combine multiple clusterings, including CSPA, HGPA, MCLA, EAC, and LCE, must be provided the number of final clusters. The algorithms that work on object level does not scale well because of the size of the co-association matrix. EAC, CSPA, COMUSA and MULTI-K suffer from this drawback.

MCLA finds the similarities between clusters in terms of Jaccard. Jaccard measure finds only the one to one differences between clusters and disregards the valuable information in input clusterings.

Working solely on object level or solely on cluster level has the disadvantage of missing the information from the other level. HBGF tries to overcome this disadvantage but it still can not capture all of the relations between the objects and clusters. LCE improves HBGF by adding missing information but in exchange of additional computation.

HGPA works very fast. However because it gives every cluster equal importance, it can not produce convenient clusterings at the presence of irrelevant clusters.

Genetic formulations of the problem are not straight forward. Core concepts of genetic algorithms are not easy to define for combining multiple clustering problem. Execution times of genetic algorithms are not deterministic and in general they are very long.

Finding the median partition is an NP-hard problem, thus utilized optimization method and other parameters of the methods hugely affect the output. Moreover, inconvenient clusterings has a huge negative effect on final clustering.

2.3 DiCLENS

DiCLENS, which is shown in Algorithm 2.4, is a novel cluster ensemble method that detects the number of clusters automatically. DiCLENS finds similar clusters as the representatives of the same group of data. Similarity between two clusters is based on the co-associations of the objects of corresponding clusters.

Definition 2.6. IntEr Cluster Similarity *(ECS), or separation, between two clusters C_{ik} and C_{jl} is defined as the pair-wise similarities between the objects of the corresponding clusters,*

$$ECS(C_{ik}, C_{jl}) = \frac{1}{|C_{ik}||C_{jl}|} \sum_{d \in C_{ik}, d' \in C_{jl}} sim(d, d') \tag{2.5}$$

$ECS(C_{ik}, C_{jl})$ is pictured in Figure 2.1.

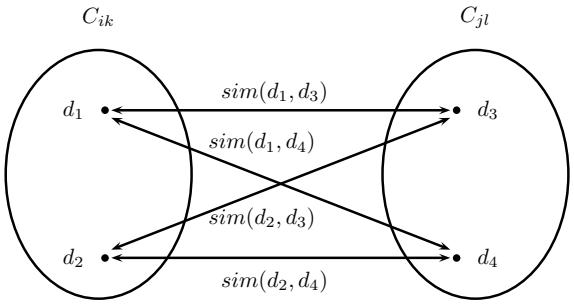

Figure 2.1: Representation of Inter Cluster Similarity (ECS)

It can be seen that the high values of $ECS(C_{ik}, C_{jl})$ resemble a high similarity between clusters C_{ik} and C_{jl}. DiCLENS finds the similar clusters by organizing the clusters as a

weighted un-directed graph G. Vertices of the graph are the clusters and weights of the edges are the ECS values between corresponding clusters, as shown in lines 5-11.

In order to obtain similarity based minimum-cost spanning tree (SMST), we run Prim's Algorithm on G as shown in line 12. But, note that cost of edges and edge weight values W have inverse relationship, since low values of similarity indicate high values of dissimilarity (cost). We explain the core of DiCLENS, `find_best_clustering` procedure (line 13), in the next section.

Algorithm 2.4 DiCLENS: **Di**visive **CL**ustering **ENS**emble with automatic cluster number

Input: $\Pi(D)$: Input Clusterings
Output: $\pi^\star(D)$: Final Clustering
1: $G := (E, V, W)$ Initialize Weighted Graph
2: $V := \emptyset$ // Set of Vertices
3: $E := \emptyset$ // Set of Edges
4: $W := \emptyset, W \subseteq E \rightarrow \mathbb{R}$ // Weight *Function* of Edges
5: **for all** $\pi_i(D) \in \Pi(D)$ **do**
6: **for all** $C_{ij} \in \pi_i$ **do**
7: $V = V \cup C_{ij}$
8: **for all** $C_{ik} \in V$ **do**
9: **for all** $C_{jl} \in V, i \neq j, k \neq l$ **do**
10: $E := E \cup (C_{ik}, C_{jl})$
11: $W := W \cup (E, ECS(C_{ik}, C_{jl}))$
 // Cost of edges and W values have inverse relationship
12: $SMST$:= `Prim's_Algorithm`(G)
13: $\pi^\star(D)$:= `find_best_clustering`($SMST$)
14: **return** $\pi^\star(D)$

2.3.1 Finding the Best Clustering Automatically

Inter Cluster Similarity concept can be applied to the whole clustering, as described in Section 1.3. Inter Cluster Similarity of a clustering, ECS_π, is defined in Equation 2.6.

$$ECS_\pi(\pi_i(D)) = \frac{1}{\binom{|\pi_i(D)|}{2}} \sum_{C_{ik}, C_{il} \in \pi_i(D), k \neq l} ECS(C_{ik}, C_{il}) \tag{2.6}$$

Clusterings that have low values of ECS_π, i.e., well-separated clusters, are preferred.

Intra Cluster Similarity (ICS), or compactness, is the measure of how similar the objects in a cluster are, as defined in Section 1.3. Intra Cluster Similarity of a cluster C_{il}, which is pictured in Figure 2.2, is calculated as shown in Equation 2.7.

$$ICS(C_{il}) = \frac{1}{\binom{|C_{il}|}{2}} \sum_{d,d' \in C_{il}} sim(d, d') \tag{2.7}$$

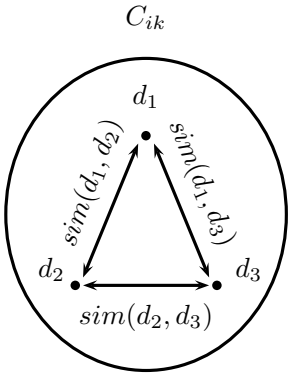

Figure 2.2: Representation of Intra Cluster Similarity (ICS)

Intra Cluster Similarity of a clustering (ICS_π) is the weighted sum of all ICS values in a clustering as shown in Equation 2.8.

$$ICS_\pi(\pi_i(D)) = \frac{1}{|\pi_i(D)|} \sum_{C_{il} \in \pi_i(D)} ICS(C_{il}) \tag{2.8}$$

We use co-occurrence based similarity, thus input clusterings are best represented with compact and well-separated clusters. Therefore, we define the quality function $\phi(\pi_i)$ as in Equation 2.9.

$$\phi(\pi_i(D)) = \overline{ICS_\pi}(\pi_i(D)) - \overline{ECS_\pi}(\pi_i(D)) \tag{2.9}$$

In Equation 2.9, $\overline{ICS_\pi}(\pi_i(D))$ and $\overline{ECS_\pi}(\pi_i(D))$ are the min-max normalized values of the $ICS_\pi(\pi_i(D))$ and $ECS_\pi(\pi_i(D))$, respectively. Normalization is utilized to scale the values between 0 and 1 so that the effects of both ICS and ECS will be equivalent to each other and small values will not be dominated by large values. Normalization procedure is shown in Algorithm 2.5.

DiCLENS finds the clustering that maximizes the quality function as the final clustering $\pi^\star(D)$, shown in 2.10.

$$\pi^\star(D) = \arg\max_{\pi_i(D)} \phi(\pi_i(D)) \tag{2.10}$$

Selection of best clustering is performed by the procedure `find_best_clustering`, which is shown in Algorithm 2.6. Removing the minimum weighted edge from the SMST increase the number of connected components, which are meta-clusters that are composed of similar clusters (line 5). Since objects can appear in more than one meta-cluster, majority voting is utilized to distribute the objects between these meta-clusters (line 7). The final clustering, generated by majority voting, is evaluated using $ICS_\pi(\pi_i(D))$ and $ECS_\pi(\pi_i(D))$ (lines 8-9).

Procedure `find_best_clustering` iteratively removes the minimum weighted edge and produces a final clustering for evaluation, until no edges are left in the SMST (lines 3-10), finally reporting the clustering with the best quality. $\overline{ICS_\pi}(\pi_i(D))$ and $\overline{ECS_\pi}(\pi_i(D))$ are calculated for each clustering (lines 11-12) and the one that maximizes Equation 2.9 is output as the final clustering $\pi^\star(D)$ (lines 15-16). There may be more than one clustering that produce the maximum value.

$ECS_\pi(\pi_i(D))$, $ICS_\pi(\pi_i(D))$ (therefore $\overline{ICS_\pi}(\pi_i(D))$ and $\overline{ECS_\pi}(\pi_i(D))$) can be computed very fast –linear with the number of total clusters in the input clusterings– using very little memory by bit vectors, and binary operations as described in Mimaroglu and Yagci (2009).

Algorithm 2.5 `normalize`(list)

Input: $list$: a list of values
Output: $normalized\_list$: min-max normalized list of values in $list$

1: $max := \max(list)$
2: $min := \min(list)$
3: $normalized\_list$:= initialize empty list
4: **for** $i := start\_of(list)$ to $end\_of(list)$ **do**
5: $\quad normalized\_list_i := \frac{list_i - min}{max - min}$
6: **return** $normalized\_list$

Algorithm 2.6 `find_best_clustering`($SMST$)

Input: $SMST$: Similarity Based Minimum-cost Spanning Tree
Output: k: Cluster Number

1: initialize empty lists: ics, $nics$, ecs, $necs$, ϕ, π
2: $k := 0$ // Store Edge Number
3: **repeat**
4: $\quad k := k + 1$
5: $\quad$ remove minimum weighted edge from $SMST$
6: $\quad \pi_{MC}(D)$ is the meta-clustering of $SMST$, each connected component is a meta-cluster
7: $\quad \pi_k(D)$:= majority_voting($\pi_{MC}(D)$)
8: $\quad ics_k := ICS_\pi(\pi_k(D))$
9: $\quad ecs_k := ECS_\pi(\pi_k(D))$
10: **until** there are some edges in $SMST$
11: $nics := \texttt{normalize}(ics)$
12: $necs := \texttt{normalize}(ecs)$
13: **for** $i := 1$ to k **do**
14: $\quad \phi_i := ics_i - ecs_i$
15: $max\_index := \max_i(\phi)$
16: **return** π_{max_index}

2.3.2 Toy Problem Demonstration

Table 2.1 shows an example clustering ensemble data set with 4 clusterings, that are represented as π_i. Clusters are represented in binary form, where value of the cell is 1 if the object is assigned to the corresponding cluster. Note that the number of clusters vary among clusterings. Moreover, π_4 is a partial clustering: d_3, d_4 and d_5 are not assigned to any of the clusters.

Table 2.1: Input clusterings on a data Set D

$\Pi(D)$	Clusters	d_1	d_2	d_3	d_4	d_5	d_6
$\pi_1(D)$	C_{11}	1	1	0	0	0	0
	C_{12}	0	0	1	1	0	0
	C_{13}	0	0	0	0	1	1
$\pi_2(D)$	C_{21}	0	0	0	0	1	1
	C_{22}	0	0	1	1	0	0
	C_{23}	1	1	0	0	0	0
$\pi_3(D)$	C_{31}	1	1	1	0	0	0
	C_{32}	0	0	0	1	1	1
$\pi_4(D)$	C_{41}	1	1	0	0	0	0
	C_{42}	0	0	0	0	0	1

SMST of the data set is shown in Figure 2.3a. DiCLENS iteratively cuts the edges of the SMST starting from the minimum weighted edge. Iteration steps with cut edge and quality value of the resulting cluster is shown in Table 2.2a. Each cut produces connected components of clusters, meta-clusters. Majority voting is utilized to distribute objects to these meta-clusters. Figure 2.3b shows the meta-clusters at the second step of Table 2.2a and Table 2.2b show the majority voting for corresponding meta-clusters. Resulting clusters are shown in Figure 2.3c.

Table 2.2: DiCLENS steps and majority voting

Table 2.2a SMST Cutting Steps

Step No.	Cut Edge	$\phi(\pi^\star(D))$
1	$C_{12} - C_{32}$	0.37
2	$C_{12} - C_{31}$	0.5
3	$C_{13} - C_{32}$	0.5
4	$C_{12} - C_{22}$	-0.27
5	$C_{11} - C_{31}$	-0.46
6	$C_{13} - C_{42}$	-0.46
7	$C_{13} - C_{21}$	-0.46
8	$C_{11} - C_{23}$	-0.46
9	$C_{11} - C_{41}$	-0.46

Table 2.2b Majority Voting

Meta-Cluster	d_1	d_2	d_3	d_4	d_5	d_6
$C_1^\star$	4	4	1	0	0	0
$C_2^\star$	0	0	2	2	0	0
$C_3^\star$	0	0	0	1	3	4

For each edge cut a clustering is formed. After all of the edges are cut quality of each clustering is calculated by the objective measure in Equation 2.9. The most qualified clustering, the clustering that has the maximum $\phi(\pi_i)$, is output as the final clustering.

For the example data set the clusterings that are generated on 2^{nd} and 3^{rd} steps have the same maximum value. Actually, both of these clusterings are exactly the same. Final output of the DiCLENS algorithm for the example data set is shown in Figure 2.3c.

2.4 EXPERIMENTAL EVALUATIONS

In this section, we present test data sets, methods for generating input clusterings, and experimental results.

We tested DiCLENS both on synthetic and real data sets. Real data sets include 34 different gene expression data sets which are shown in Table 2.3. Glass Identification (Glass), and Image Segementation (Imageseg) are obtained from the University of California Irvine Machine Learning Repository (Frank and Asuncion 2010). Synthetic data sets consist of 2-half rings (Figure A.1a), 2-curve (Figure A.1a), 4c10k (Figure A.3a), 4c20k (Figure A.4a), and 4c40k (Figure A.5a).

Synthetic data sets are not linearly separable (except 2-half rings) and they are hard to cluster using basic clustering algorithms. 4c10k, 4c20k and 4c40k data sets consist of 10000, 20000 and 40000 objects respectively.

For all of the data sets we have the original data set and true labels. We generated the clustering ensembles by utilizing various approaches including k-means algorithm with varying k-values and random sub-spacing on gene expression data sets, manually constructing clusters, randomly injecting error into the original clusters, Chameleon (Karypis et al. 1999), and hierarchical agglomerative clustering (agnes) with varying input parameter values. For k-means algorithm we randomly select k values (between 2 and $\sqrt{|D|}$) to increase diversity across clusterings. Diversity is an important aspect that increase the effectiveness of combining multiple clusterings (Hadjitodorov et al. 2006).

We mostly rely on k-means for generating our input clusterings. This situation is in harmony with the real world popularity of k-means. Moreover, some of the algorithms mentioned in Section 2.2 uses k-means as an internal part of their algorithms, i.e., MULTI-K and LCE take the original data set, execute k-means with different parameters on the data set and finally combine the resulting clusterings. Please note that our algorithm works on any type of input clustering, regardless of the generation method.

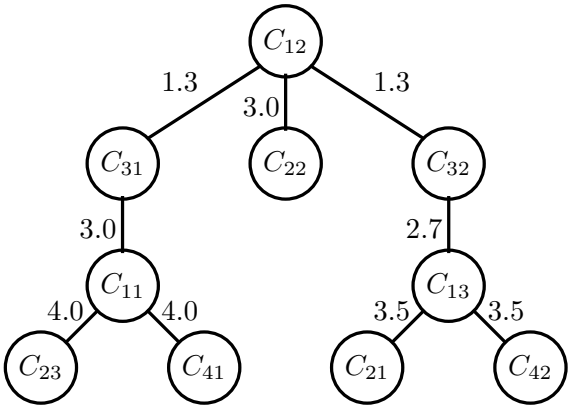

Figure 2.3a SMST of Table 2.1

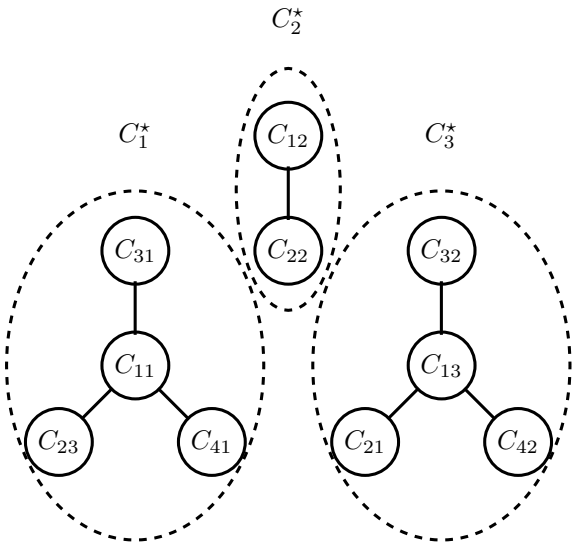

Figure 2.3b 3 Meta-Clusters

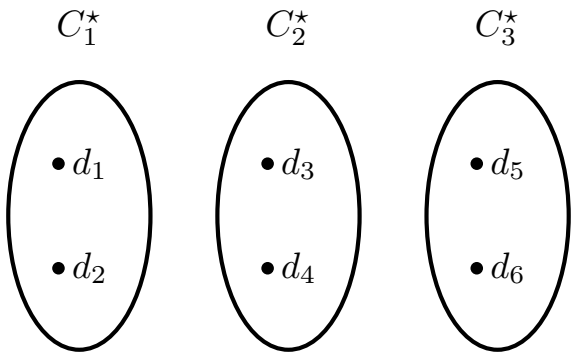

Figure 2.3c Final Clustering

Figure 2.3: Toy problem demonstration of DiCLENS

Properties of the gene expression input clusterings are shown in Table 2.4. In the same table, Method column shows the generation method of input clusterings, Features column

Table 2.3: Gene expression data sets

Data Set	*Array Type*	*Tissue*	*Total samples*	*Num of classes*	*Total Genes*	*Selected # of Genes*
Bladder carcinoma Dyrskjot et al. (2003)	Affymetrix	Bladder	40	3	7129	1203
Breast Cancer West et al. (2001)	Affymetrix	Breast	49	2	7129	1198
Breast-Colon tumors Chowdary et al. (2006)	Affymetrix	Breast, Colon	104	2	22283	182
Carcinomas Su et al. (2001)	Affymetrix	Multi-tissue	174	10	12533	1571
Central nervous system-1 Pomeroy et al. (2002)	Affymetrix	Brain	34	2	7129	857
Central nervous system-2 Pomeroy et al. (2002)	Affymetrix	Brain	42	5	7129	1379
Endometrial cancer Risinger et al. (2003)	Double Channel	Endometrium	42	4	8872	1771
Glioblastoma multiforme Liang et al. (2005)	Double Channel	Brain	37	3	24192	1411
Gliomagenesis Bredel et al. (2005)	Double Channel	Brain	50	3	41472	1739
Gliomas-1 Nutt et al. (2003)	Affymetrix	Brain	50	4	12625	1377
Gliomas-2 Nutt et al. (2003)	Affymetrix	Brain	28	2	12625	1070
Gliomas-3 Nutt et al. (2003)	Affymetrix	Brain	22	2	12625	1152
Hepatocellular carcinoma Chen et al. (2002)	Double Channel	Liver	178	2	22699	85
Leukemia-1 Yeoh et al. (2002)	Affymetrix	Bone Marrow	248	2	12625	2526
Leukemia-2 Yeoh et al. (2002)	Affymetrix	Bone Marrow	248	6	4022	1095
Leukemia-3 Armstrong et al. (2002)	Affymetrix	Blood	72	2	12582	1081
Leukemia-4 Armstrong et al. (2002)	Affymetrix	Blood	72	3	12582	2194
Leukemia-5 Golub et al. (1999)	Affymetrix	Bone Marrow	72	2	7129	1877
Leukemia-6 Golub et al. (1999)	Affymetrix	Bone Marrow	72	3	7129	1877
Lung tumor-1 Bhattacharjee et al. (2001)	Affymetrix	Lung	203	5	12600	1543
Lung tumor-2 Garber et al. (2001)	Double Channel	Lung	66	4	24192	4553
Lymphoma-1 Alizadeh et al. (2000)	Double Channel	Blood	42	2	4022	1095
Lymphoma-2 Alizadeh et al. (2000)	Double Channel	Blood	62	3	4022	2093
Lymphoma-3 Shipp et al. (2002)	Affymetrix	Blood	77	2	7129	798
Melanoma Bittner et al. (2000)	Double Channel	Skin	38	2	8067	2201
Mesothelioma Gordon et al. (2002)	Affymetrix	Lung	181	2	12533	1626
Multi-tissue Ramaswamy et al. (2001)	Affymetrix	Multi-tissue	190	14	16063	1363
Prostate cancer-1 Tomlins et al. (2007)	Double Channel	Prostate	104	5	20000	2315
Prostate cancer-2 Tomlins et al. (2007)	Double Channel	Prostate	92	4	20000	1288
Prostate cancer-3 Lapointe et al. (2004)	Double Channel	Prostate	69	3	42640	1625
Prostate cancer-4 Lapointe et al. (2004)	Double Channel	Prostate	110	4	42640	2496
Prostate cancer-5 Singh et al. (2002)	Affymetrix	Prostate	102	2	12600	339
Round blue-cell tumor Khan et al. (2001)	Double Channel	Multi-tissue	83	4	6567	1069
Serrated carcinomas Laiho et al. (2007)	Affymetrix	Colon	37	2	22883	2202

shows the ratio of features selected for random sub-spacing, $|\pi|$ column indicates the number of clusters in different clusterings, $|\Pi|$ shows the number of input clusterings and the last three columns indicates the ARI, see Section 1.3.2, values for the input clusterings. Properties of all of the other input clusterings are shown in Table 2.5. Figures A.1, A.2,

A.3, A.4 and A.5 picture the input clusterings of 2-curve, 2-half rings, 4c10k, 4c20k and 4c40k, respectively.

Table 2.4: Properties of input clusterings on gene expression data sets

Data Set	Method	Features	$\|\pi\|$	$\|\Pi\|$	ARI		
					Min	Max	Average
Bladder carcinoma	agnes, k-means	25% - 50%	2 - 6	9	0.18	0.64	0.39
Breast Cancer	k-means	25% - 50%	2 - 7	10	0.08	0.42	0.25
Breast-Colon tumors	k-means	25% - 50%	2 - 10	10	0.11	0.92	0.43
Carcinomas	agnes, k-means	25% - 50%	2 - 13	11	0.10	0.63	0.42
Central nervous system-1	manual	N/A	2 - 4	6	0.21	0.61	0.44
Central nervous system-2	agnes, k-means	25% - 50%	2 - 6	10	0.23	0.54	0.39
Endometrial cancer	manual, random	N/A	4 - 5	5	0.48	0.71	0.60
Glioblastoma multiforme	k-means	75% - 85%	2 - 6	10	-0.03	0.46	0.18
Gliomagenesis	k-means	25% - 50%	2 - 7	10	0.11	0.49	0.28
Gliomas-1	manual	N/A	4 - 6	4	0.48	0.74	0.64
Gliomas-2	manual, random	N/A	2 - 5	4	0.30	0.39	0.36
Gliomas-3	manual	N/A	2 - 3	3	0.37	0.61	0.52
Hepatocellular carcinoma	k-means	75% - 85%	2 - 13	10	0.10	0.70	0.40
Leukemia-1	agnes, k-means	75% - 85%	2 - 15	11	0.10	0.87	0.24
Leukemia-2	k-means	25% - 50%	2 - 15	10	0.14	0.23	0.20
Leukemia-3	manual	N/A	2 - 5	3	0.36	0.56	0.46
Leukemia-4	k-means	75% - 85%	3 - 8	10	0.42	0.92	0.59
Leukemia-5	agnes, k-means	25% - 50%	2 - 8	11	0.15	0.89	0.45
Leukemia-6	k-means	25% - 50%	2 - 8	10	0.18	0.84	0.47
Lung tumor-1	chameleon, k-means	25% - 50%	3 - 14	11	0.10	0.28	0.19
Lung tumor-2	k-means	25% - 50%	2 - 8	10	0.08	0.32	0.19
Lymphoma-1	k-means	25% - 50%	2 - 6	10	0.02	0.43	0.17
Lymphoma-2	k-means	25% - 50%	3 - 7	10	0.20	0.52	0.33
Lymphoma-3	agnes, k-means, random	25% - 50%	2 - 8	10	-0.01	0.32	0.11
Melanoma	manual, random	N/A	2	5	0.38	0.70	0.50
Mesothelioma	k-means	25% - 50%	2 - 13	10	0.07	0.75	0.25
Multi-tissue	chameleon	100%	7 - 14	6	0.06	0.34	0.25
Prostate cancer-1	manual	N/A	5 - 7	5	0.43	0.61	0.52
Prostate cancer-2	manual	N/A	4 - 6	5	0.44	0.61	0.52
Prostate cancer-3	manual	N/A	4 - 7	4	0.24	0.64	0.42
Prostate cancer-4	manual	N/A	5 - 6	3	0.51	0.61	0.55
Prostate cancer-5	k-means	25% - 50%	2 - 10	10	0.02	0.23	0.10
Round blue-cell tumor	agnes, k-means	25% - 50%	2 - 9	9	0.10	0.90	0.49
Serrated carcinomas	manual	N/A	2 - 6	5	0.29	0.51	0.37

Table 2.5: Properties of input clusterings on other data sets

Data Set	Method	$\lvert\pi\rvert$	$\lvert\Pi\rvert$	ARI		
				Min	Max	Average
2curves	manual, random	4 - 6	3	0.33	0.78	0.532
2halfrings	k-means	2 - 5	3	0.414	0.805	0.656
4c10k	k-means	5 - 6	4	0.727	0.818	0.765
4c20k	k-means	3 - 6	9	0.57	0.928	0.739
4c40k	k-means	10	3 - 6	0.557	0.874	0.759
segmentation	k-means	7	5	0.436	0.525	0.46
glass	k-means	5 - 7	5	0.581	0.731	0.642

All of the tests are done on a computer that has 1.67GHz Intel CPU and 1.5GiB main memory, operating system of the computer is GNU with Linux 2.6 kernel. DiCLENS is implemented in Java. LCE is implemented in MatLab. Java is used for the implementation of MCLA, CSPA, and HPGA algorithms but they require the METIS software packages which are implemented in C.

Input clusterings are generated once and the same input clusterings are used for all of the methods. Quality of input and final clusterings are calculated by Adjusted Rand Index (ARI), see Section 1.3.2. Table B.1 and Table B.2 shows the ARI values for the final clusterings produced by DiCLENS, MCLA, CSPA, HPGA and LCE.

Results in Table B.1 and Table B.2 indicate that DiCLENS produces perfect clusterings, $ARI = 1.0$, on 2-halfrings, Glass and 4 of the gene expression data sets. On 28 of the gene expression data sets DiCLENS produces the best clusterings and almost perfect clusterings, $ARI \geq 0.89$, on 15 of them. Results of DiCLENS is very good, $ARI \geq 0.92$ on 2-curve, 4c10k, 4c20k and Imageseg data sets.

On 34 out of 41 data sets we used for tests, DiCLENS produces the best final clustering with the same input clusterings. Also for all of the data sets quality of clusterings that are produced by DiCLENS are better than the average quality of input clusterings. These results clearly demonstrate that DiCLENS is very useful and superior to other algorithms. Test results show that DiCLENS produces better quality clusterings than LCE which produces better results than MULTI-K, GCC, and HGBF.

Table B.3 shows the comparison of the number of clusters found by DiCLENS and the true number of classes. DiCLENS finds true number of clusters on 24 of the data sets.

Execution times of the methods are shown in Table B.4. Comparing the most accurate methods, i.e., DiCLENS and LCE, it is clear that DiCLENS is superior. DiCLENS requires more time to run because of its automatic cluster finding mechanism.

3. IMPROVING DBSCAN'S EXECUTION TIME BY USING A PRUNING TECHNIQUE ON BIT VECTORS

3.1 INTRODUCTION

In this chapter we discuss DBSCAN_BV (Mimaroglu and Aksehirli 2011) a novel pruning technique that dramatically improves the execution time of DBSCAN clustering algorithm for binary data sets and Hamming distance.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) (Ester et al. 1996) is a well known and widely used density-based clustering algorithm. DBSCAN can effectively detect arbitrary shape clusters and noise in the data set when supplied with correct parameters. Moreover, DBSCAN can automatically find the number of clusters using the density information. DBSCAN finds well separated clusters, because it merges the clusters that are not well separated.

DBSCAN algorithm, which is shown in Algorithm 3.1, identifies the dense areas in the data set as clusters and objects in the sparse areas as noise. Density and sparsity is determined by the number of objects in ε-neighborhood of the object, which are defined by two parameters: ε and $MinPts$. In Figure 3.1 there are 7 points in the ε-radius of point p.

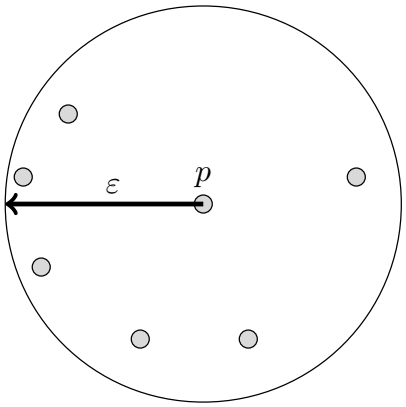

Figure 3.1: Center-based density in DBSCAN

Algorithm 3.1 DBSCAN algorithm

Input: D: data set, ε: radius, $MinPts$: minimum number of points
Output: π: Clustering

```
1:  clusterId := 0
2:  for all unvisited point p ∈ D do
3:     mark p as visited
4:     N := getNeighbors(p, ε)
5:     if sizeof(N) < MinPts then
6:        mark p as noise point
7:     else
8:        clusterId++
9:        add p to cluster clusterId
10:       for all point p' ∈ N do
11:          if p' is not visited then
12:             mark p' as visited
13:             N' := getNeighbors(p', ε)
14:             if sizeof(N') ≥ MinPts then
15:                N := N ∪ N'
16:             if p' does not belong to a cluster then
17:                add p' to cluster clusterId
18: return π
```

Each data object is labeled either as *core*, *border* or *noise*. If an object has at least $MinPts$ number of points in its ε-neighborhood then the object is labeled as a *core* point. All core points that are in the ε-neighborhood of a core point are assigned to the same cluster.

Objects that are not core points in the ε-neighborhood of core points are also assigned to the same cluster as the core point and labeled as *border* points. Objects that are not assigned to any cluster are regarded as *noise*.

In Figure 3.2, p_1 is a core point, p_2 is a border point, and p_3 is noise with respect to ε and $MinPts = 7$.

The main bottleneck of the DBSCAN algorithm is the detection of the core points, more specifically the neighborhood search. When searching for neighbors every object must be compared to every other object, which leads to $O(n^2)$ complexity, where n is the number of data points. If distance matrix is stored in the memory then the space complexity also becomes $O(n^2)$, if not then there are redundant distance computations which require additional time.

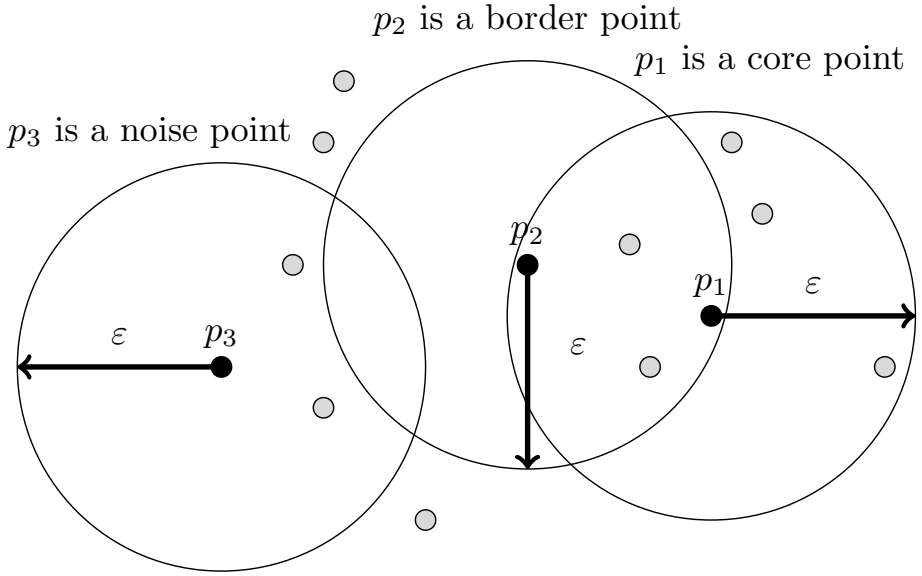

Figure 3.2: A data set labeled with respect to ε and $MinPts = 7$

We give a non-exhaustive overview of the methods that improve the execution time performance of DBSCAN in Section 3.2. Section 3.3 discusses our improvement to DBSCAN, and Section 3.4 provides experimental evaluations.

3.2 RELATED WORK

Inspecting the methods that improve the execution time performance of DBSCAN, three different techniques come forward:

1. Partition the database to reduce the search space.
2. Run DBSCAN on a subset of a data set.
3. Reduce the number of seeds.

The most time consuming process of DBSCAN is finding the neighbors of data points. One technique to speed up this process is partitioning the database to reduce the search space. In the literature, there are several methods which partition the database, execute DBSCAN only on partitions and finally merge the clusters obtained from partitions. El-Sonbaty et al. (2004) partitions the database using CLARANS (Ng and Han 2002), which is an algorithm that has good performance on large data sets. Clusters are merged using

the relative-density measure that is introduced in Karypis et al. (1999). DBSK algorithm (Rui and Chunhong 2008) executes DBSCAN with partition specific parameters on partitions that are formed by executing k-means, then merges the density-connected clusters. Partitioning-Based DBSCAN algorithm, PDBSCAN, (Zhou et al. 2000) uses statistical characteristics of the data to partition the data set over one or more data attributes. It is proposed to use different parameters for different partitions but details are not given in the paper. Clusters are merged by analysing the points near the borders of partitions. A recent method presented in Jiang and Li (2009) partitions the data set using a one-pass clustering algorithm then applies a modified version of DBSCAN to find and merge the clusters.

Zhou et al. (2000) proposes Sampling-based DBSCAN, SDBSCAN, which creates two R*-trees, one for the whole data set and the second one for the selected samples. DBSCAN algorithm is executed on the selected samples to create the clusters. Assignments of not sampled data points are performed faster by using the region queries over R*-trees. A recent method, Rough-DBSCAN (Viswanath and Babu 2009) finds some representative points using the well-known *Leaders* clustering method. Instead of all of the data set, these representative points are used as seeds for a modified version of DBSCAN. Rough-DBSCAN finds the approximate clusters using the Rough Set theory.

For dense areas in data sets, the neighbors of two close data points can be very similar. Because DBSCAN puts the density connected points into the same cluster, finding the neighbors for some of the neighbor points can be omitted. "A Fast DBSCAN algorithm", FDBSCAN, which is introduced in Zhou et al. (2000) selects the seed points as far as possible from the existing core points. A good discussion about the cost of handling of *lost points*, the points that can not be accurately clustered because of the approximation, can be found in the paper. IDBSCAN algorithm (Borah and Bhattacharyya 2004) reduces the time complexity by selecting seed points outside the neighborhoods of core points. KIDBSCAN (Tsai and Liu 2006) determines the high-density centers by running k-means beforehand and suggests these points as seeds to the IDBSCAN. KIDBSCAN increases the worst case performance of IDBSCAN.

Parallel versions of DBSCAN are explained in Zhou et al. (2000), Sakellariou et al. (2001), and Guo et al. (2002).

Intuitively, space indexing techniques such as kd-tree and R-tree are used to index the data and increase the execution time performance of DBSCAN. But these indexing techniques

work efficiently only on real valued data sets, so they become inefficient on binary and categorical data sets. As explained in detail in Section 3.3, our method works on binary data and reduces the search space when finding the neighbors of points. A comparison between our method and the mentioned indexing techniques are given in Section 3.4.

Partitioning the data set is a pre-processing step with its limitations and requirements. Beside extra time and space requirements, selection of partitioning technique and its input parameters adds another level of complexity. Limiting the number of seed points or sampling the data distorts the accuracy of the method and leads to false identification of border points and false separation of clusters.

3.3 BINARY APPROACH FOR DBSCAN

In this section we present a novel pruning method that significantly reduces the number of points returned by getNeighbors($\mathbf{p}, \varepsilon$), shown in line 4 of Algorithm 3.1, and the execution time of neighbor search. Our method works on binary data sets and Hamming distance.

A binary data set is broken down into row and column bit vectors. For example, each row bit vector ($\mathbf{d_i}$, $1 \leq i \leq 10$) in Figure 3.4a represents a data object and each column bit vector ($\mathbf{a_j}$, $1 \leq j \leq 5$) in Figure 3.4b represents an attribute of the binary data set shown in Figure 3.3, respectively. Even though keeping two copies of data set doubles the memory requirement, benefits will become apparent later.

	$\mathbf{a_1}$	$\mathbf{a_2}$	$\mathbf{a_3}$	$\mathbf{a_4}$	$\mathbf{a_5}$
$\mathbf{d_1}$	0	1	1	0	1
$\mathbf{d_2}$	0	1	0	0	1
$\mathbf{d_3}$	1	0	0	1	0
$\mathbf{d_4}$	0	1	0	1	0
$\mathbf{d_5}$	1	0	1	0	0
$\mathbf{d_6}$	0	0	0	1	1
$\mathbf{d_7}$	0	1	0	0	0
$\mathbf{d_8}$	0	0	1	1	0
$\mathbf{d_9}$	1	0	1	0	0
$\mathbf{d_{10}}$	0	0	0	1	1

Figure 3.3: A binary data set

Our pruning method is based on the following observations:

$\mathbf{d_1}$	0	1	1	0	1
$\mathbf{d_2}$	0	1	0	0	1
$\mathbf{d_3}$	1	0	0	1	0
$\mathbf{d_4}$	0	1	0	1	0
$\mathbf{d_5}$	1	0	1	0	0
$\mathbf{d_6}$	0	0	0	1	1
$\mathbf{d_7}$	0	1	0	0	0
$\mathbf{d_8}$	0	0	1	1	0
$\mathbf{d_9}$	1	0	1	0	0
$\mathbf{d_{10}}$	0	0	0	1	1

Figure 3.4a Row Bit Vectors

$\mathbf{a_1}$	$\mathbf{a_2}$	$\mathbf{a_3}$	$\mathbf{a_4}$	$\mathbf{a_5}$
0	1	1	0	1
0	1	0	0	1
1	0	0	1	0
0	1	0	1	0
1	0	1	0	0
0	0	0	1	1
0	1	0	0	0
0	0	1	1	0
1	0	1	0	0
0	0	0	1	1

Figure 3.4b Column Bit Vectors

Figure 3.4: Decomposed data set of Figure 3.3

Definition 3.1. *A binary data set D, having n objects and m attributes, can be represented either as a set of row bit vectors $D_{rows} = \{\mathbf{d_1}, \mathbf{d_2}, \ldots \mathbf{d_n}\}$, or as a set of column bit vectors $D_{cols} = \{\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_m}\}$.*

Definition 3.2. *In a binary data set D having n objects and m attributes, let $(\mathbf{d_j}) \in D_{rows}$ and $(\mathbf{a_j}) \in D_{cols}$. $(\mathbf{d_j})_i \in \{0, 1\}$ and $(\mathbf{a_j})_i \in \{0, 1\}$ denote the value of i^{th} bit of $\mathbf{d_j}$ and $\mathbf{a_j}$, respectively.*

Definition 3.3. *In a binary data set D having n objects and m attributes, let $Ones_i$ be the set of the column bit vectors that have the value 1 at i^{th} bit. That is,*

$$Ones_i = \{\mathbf{a_j} | (\mathbf{a_j})_i = 1, 1 \leq j \leq m, \mathbf{a_j} \in D_{cols}\} \tag{3.1}$$

Note that $Ones_i$ is also the set of column bit vectors that are 1 in object $\mathbf{d_i}$.

$$Ones_i = \{\mathbf{a_j} | (\mathbf{d_i})_j = 1, 1 \leq j \leq m, \mathbf{a_j} \in D_{cols}, \mathbf{d_j} \in D_{rows}\} \tag{3.2}$$

Example 3.1. *According to the data set in Figure 3.3,* $Ones_5 = \{\mathbf{a_1}, \mathbf{a_3}\}$ *and* $Ones_8 = \{\mathbf{a_3}, \mathbf{a_4}\}$.

Definition 3.4. *In a binary data set* D *having* n *objects and* m *attributes, possible neighbors of an object* $\mathbf{d_i}$ *with respect to the distance* ε *is defined by the* $\mathbf{ngh_i}$ *bit vector as*

$$\mathbf{ngh_i} = \underbrace{(\mathbf{a_{l_1}} \vee \mathbf{a_{l_2}} \vee \cdots \vee \mathbf{a_{l_k}}) \wedge \cdots \wedge (\mathbf{a_{r_1}} \vee \mathbf{a_{r_2}} \vee \cdots \vee \mathbf{a_{r_k}})}_{\binom{|Ones_i|}{k} \text{ components}}, \tag{3.3}$$

where k *is the smallest positive integer such that* $k > \varepsilon$*, each* k*-length subset of* $Ones_i$ *is logically* OR*ed in itself, and all of the* k*-length subsets are logically* AND*ed.*

Theorem 3.1. *If* $(\mathbf{ngh_i})_\mathbf{j} = 0$ *then* $dist(\mathbf{d_i}, \mathbf{d_j}) > \varepsilon$.

Proof.

$$\mathbf{ngh_i} = \underbrace{(\mathbf{a_{l_1}} \vee \mathbf{a_{l_2}} \vee \cdots \vee \mathbf{a_{l_k}}) \wedge \cdots \wedge (\mathbf{a_{r_1}} \vee \mathbf{a_{r_2}} \vee \cdots \vee \mathbf{a_{r_k}})}_{\binom{|Ones_i|}{k} \text{ components}}, \tag{3.4}$$

$(\mathbf{ngh_i})_j = 0$ implies that at least one of the components must be 0 at location j, i.e., $(\mathbf{a_{m_1}} \vee \mathbf{a_{m_2}} \vee \cdots \vee \mathbf{a_{m_k}})_j = 0$. Because of the definition of the logical OR operation, j^{th} position of every column bit vector of this component must be 0, $(\mathbf{a_{m_i}})_j = 0$ for all $1 \leq$

$\mathbf{i} \leq k$. Therefore $\mathbf{d_j}$ and $\mathbf{d_i}$ have at least k differences. Because $k > \varepsilon$ and $dist(\mathbf{d_j}, \mathbf{d_i}) \geq k$ hold, the inequality $dist(\mathbf{d_j}, \mathbf{d_i}) > \varepsilon$ must hold. Thus, $\mathbf{d_j}$ cannot be a neighbor of $\mathbf{d_i}$; as a result $\mathbf{d_j}$ can be discarded. □

Computational complexity of DBSCAN is high because it exhaustively calculates the distances between every pair of objects. Theorem 3.1 effectively discards the data objects that can not be in the ε-neighborhood of the reference object, and it only discards the distant objects, i.e., no false negatives. Instead of searching each point exhaustively, getNeighbors_BV only searchs the possible neighbors of the point $\mathbf{p}$, as shown in Algorithm 3.2. At line 2 of the algorithm $\mathbf{PsbleNghbrs}$ is initialized as a $|D|$-length bit vector filled with ones, because every data object in the data set initially is a possible neighbor. At line 3, attributes that are 1 in the $\mathbf{p}$ are extracted to $Ones_i$ set. Theorem's input parameter k, which must be the smallest positive integer that is greater than ε, is prepared at line 4. Theorem 3.1 is utilized at lines 5–8. Note that we check for the values of k and $|Ones_i|$ at line 5, since the theorem is not applicable if k is too large. If this is the case, every object in the data set is left as a possible neighbors and the standard search is performed. Lines 9–13 checks the possible neighbors to eliminate false positives.

$\binom{|Ones_i|}{k}$ can be too large for some data sets and ε values. In this case, pruning can be utilized with a small number of randomly selected k-subsets rather than every k-subset of $Ones_i$. Experimental evaluations show that even if not every k-subset is utilized, pruning provides very good results. The new procedure getNeighbors_BV replaces getNeighbors in DBSCAN resulting in a new algorithm that we call DBSCAN_BV.

It is important to note that Theorem 3.1 never produces false negatives, i.e., $PsbleNghbrs$ is always a superset of the real ε-neighbors. Also, getNeighbors_BV eliminates the false positives by checking all of the possible neighbors one by one. Therefore, accuracy of the DBSCAN_BV is always same with the original DBSCAN.

Example 3.2. *For the binary data set in Figure 3.3, corresponding row bit vectors and column bit vectors of the data set are shown in Figures 3.4Figure 3.4a and 3.4Figure 3.4b, respectively. DBSCAN_BV, with $\varepsilon = 1.7$ and $MinPts = 2$, works on object $\mathbf{d_2}$ (second row) as follows: $Ones_2 = \{\mathbf{a_2}, \mathbf{a_5}\}$ and k is selected as 2. There is only one 2-length subset of $Ones_2$: $\{\mathbf{a_2}, \mathbf{a_5}\}$. Column bit vectors $\mathbf{a_2}$ and $\mathbf{a_5}$ are ORed with each other, as shown in Figure 3.5, to obtain the $\mathbf{ngh_2}$ bit vector. The 0 values on $\mathbf{ngh_2}$ denote the objects that are located at a distance greater than $\varepsilon = 1.7$; as a result, $\{\mathbf{d_3}, \mathbf{d_5}, \mathbf{d_8}, \mathbf{d_9}\}$ can be safely pruned. Only the distances between $\mathbf{d_2}$ and $\{\mathbf{d_1}, \mathbf{d_4}, \mathbf{d_6}, \mathbf{d_7}, \mathbf{d_{10}}\}$ are calculated. The*

Algorithm 3.2 getNeighbors_BV(**p**, ε)

Input: $\mathbf{d_i}$: data object ε: radius
Output: N: Neighbors of $\mathbf{d_i}$

1: $N := \emptyset$
2: $\mathbf{PsbleNghbrs} := \mathbf{1}$ // Initially all the objects are possible neighbors
3: $Ones_i := \{\mathbf{a_j} | (\mathbf{d_i})_j = 1, 1 \leq j \leq m\}$
4: Select k as the smallest positive integer, such that $k > \varepsilon$
5: **if** $|Ones_i| \geq k$ **then**
6: // Apply Theorem 3.1
7: **for all** unique set **do**
8: $\mathbf{PsbleNghbrs} := \mathbf{PsbleNghbrs} \wedge (\mathbf{a_{l_1}} \vee \mathbf{a_{l_2}} \vee \cdots \vee \mathbf{a_{l_k}})$
9: **for all** $\mathbf{d_j} : (\mathbf{PsbleNghbrs})_j = 1$ **do**
10: // Compute Hamming distance
11: **if** $|\mathbf{d_i} \oplus \mathbf{d_j}| \leq \varepsilon$ **then**
12: // $\mathbf{d_j}$ is in the ε-neighborhood of $\mathbf{d_i}$
13: $N := N \cup \mathbf{d_j}$
14: **return** N

neighbors of $\mathbf{d_2}$ *with respect to* $\varepsilon = 1.7$ *and* $MinPts = 2$ *are the set* $N = \{\mathbf{d_1}, \mathbf{d_7}\}$. *Thus,* $\mathbf{d_2}$ *is a core object.*□

$\mathbf{a_2}$		$\mathbf{a_5}$		$\mathbf{ngh_2}$
1		1		1
1		1		1
0		0		0
1		0		1
0	$\vee$	0	=	0
0		1		1
1		0		1
0		0		0
0		0		0
0		1		1

Figure 3.5: $\mathbf{a_2} \vee \mathbf{a_5}$

3.4 EXPERIMENTAL RESULTS

Algorithms were implemented in Java language. All of the tests were done on a computer that has 2.8GHz Intel CPU and 4GiB main memory, operating system of the computer is GNU with Linux 2.6 kernel. Binarized versions of real and synthetic data sets are used for testing. Table 3.1 shows the properties of the data sets.

Table 3.1: Properties of test data sets

Data Set	Objects	Attributes	Max #1's	Min #1's	Avg #1's
Reuters	5485	14575	281	4	41
WebKB	4168	7770	2773	1	78
Letter Recognition	20000	25	5	5	5
Image Segmentation	2100	42	7	7	7
Zoo	101	21	11	3	7.6
Spect Heart Data	267	23	22	0	7.6
Syn-5K	5000	32	9	5	7
Syn-10K	10000	35	8	5	6.5
Syn-25K	25000	25	8	1	5
Syn-50K	50000	50	11	7	9

Data sets that are taken from real domain are Reuters, WebKB, Letter Recognition, Image Segmentation, Zoo, and Spect. Letter Recognition, Image Segmentation, Zoo and Spect data sets are obtained from the University of California Irvine Machine Learning Repository (Frank and Asuncion 2010). WebKB and Reuters are Text data sets. Text data sets are gone through a two step pre-processing. First step consists of standard text-specific processes such as removal of stop words and stemming while the second step is the binarization. Syn-5k, Syn-10K, Syn-25K, and Syn-50K are synthetic data sets.

$\binom{|Ones_i|}{k}$ can become too high for some data sets and ε values. DBSCAN_BV does not generate and use every k-length subset of $Ones_i$, instead it generates and uses a few randomly selected k-length subsets of $Ones_i$. Even though not using every subset results in less pruning, experimental evaluations show that 100 randomly selected subsets provide sufficient pruning even for very large data sets. If the number of attributes of the data set is not very large, all k-length subsets of $Ones_i$ can be used.

The execution time performances of DBSCAN and DBSCAN_BV on real data sets are shown in Figure A.6. DBSCAN_BV is much faster than DBSCAN on large data sets such as WebKB and Reuters. DBSCAN_BV performs faster than DBSCAN on medium size data sets, i.e., Letter Recognition and Image Segmentation. On small data sets, i.e., Zoo and Spect the results are comparable.

The execution time performance of DBSCAN and DBSCAN_BV on synthetic data sets are shown in Figure A.7. DBSCAN_BV performs much faster than DBSCAN on all of these data sets.

Experiments on both real and synthetic data sets show that the execution time gets larger while ε gets larger, see Figure A.6 and Figure A.7. To understand the relation between execution time and ε some additional tests are conducted. Figure A.8 shows the results. For large values of ε, number of neighbors is increased with the radius. Thus, pruning method becomes unavailable to detect the points outside of this radius and DBSCAN_BV falls back to the original DBSCAN algorithm.

Hamming distance is a symmetric metric, i.e. both zeros and ones in a data point have the same significance. Theorem 3.1 formulates the pruning operation for the attributes that are 1 in the reference point. But it can be extended for the attributes that are zero in the reference data point. Pruning both for *Ones* and *Zeros* decreases the number of possible neighbors. For clarity we will call this operation *two-way pruning*. Figures A.9 and A.10 show the effects of two-way pruning for some real data sets in terms of the total number of possible neighbors and execution time, respectively. Note that these results are only for finding the exact neighborhoods, not for the whole DBSCAN algorithm. For data sets that are very sparse, i.e. Image Segmentation and Letter Recognition, benefit from conducting two-way pruning is so little that the effect of it can hardly be seen. On the other hand, benefit of using zeros is much more obvious on dense data sets, i.e. Zoo and Spect. For all of the data sets, two-way pruning increases the total execution time.

DBSCAN is commonly used with space indexing techniques to reduce the time cost of neighbor search. We conduct tests to compare our pruning method against two of the most popular of these techniques, which are kd-trees (Bentley 1975) and R-trees (Guttman 1984). Because both of these methods work effectively only on real valued data sets, they are not directly applicable to our data sets. Thus, we compare them with our pruning method only on real valued data sets. To keep the comparison fair, we compare the execution time results for ε values that approximately cover the same radius. Results can be seen on Figure A.13. Please note that these tests measure just the time for finding the exact neighbors of all of the data points, not the whole DBSCAN algorithm. On Letter Recognition data set, R-tree search did not return any results in practical times, thus the corresponding results are put into Figure A.13b just for completeness.

Bit vector, or bitmap, representation of a data set can be compressed. Bitmap compression algorithms, such as Byte aligned Bitmap Coding (BBC) (Antoshenkov 1994), Word Aligned Hybrid (WAH) code (Wu et al. 2004) and Sorted Word Aligned Bitmap (Lemire et al. 2010), support bitwise set operations in compressed form. Key difference of these methods from generic compression algorithms is their focus on performance improve-

ment of bit operations on compressed form rather than sole improvement of compression ratio. It is a known fact that for some data sets, operations can be performed faster on compressed forms than their not-compressed forms. Wu et al. (2006) shows that BBC form and WAH executes faster than not-compressed form if compression ratio is below 10^{-2} and 10^{-1}, respectively. We compared the performance of using WAH compressed form against using not compressed form of our data sets. Compressed forms are 2 to 40 times slower, which is expected according to Wu et al. (2006).

Pruning technique can be very beneficial to the systems having limited resources in terms of CPU and battery capacity. Figure A.11 indicates that the discussed DBSCAN_BV algorithm completes the clustering procedure much more efficiently than the original DBSCAN algorithm. For Wireless Sensor Network deployments k-means algorithm is favored to DBSCAN algorithm because of its lower computational complexity (Hua et al. 2009). Although the complexity of DBSCAN, $O(n^2)$, is greater than the complexity of k-means, which is $O(ikn)$ where i is the iteration count, k is the number of clusters and n is the number of objects, Figure A.12 shows that DBSCAN_BV is comparable to k-means in practice. Furthermore DBSCAN can detect arbitrary shape cluster contrary to k-means which can only detect globular shape clusters. During comparison; values of ε, $MinPts$ and k are set to make DBSCAN_BV and k-means produce same number of clusters.

4. CONCLUSION

Clustering is a very important tool for data analysis and knowledge discovery. It is used in almost every field of science and engineering. Although the unsupervised nature of clustering is very effective for many tasks, it comes with its unique problems. Compared to supervised methods, such as classification, producing accurate results with unsupervised methods requires much more computation.

Choosing the clustering method and parameters of the method that fit the data is not immediate. Producing diverse range of clusterings and combining them is easier than determining the correct method and its parameters. Moreover, since different clustering algorithms can capture different aspects of the data, combining multiple clusterings can produce better clusterings.

Combining multiple clusterings problem is defined as combining the different views for the same data to produce a better grouping of the data. There are many situations where multiple views of the same data are present although the original properties of the data are missing. In these situations a new clustering must be formed just by evaluating the views.

In Chapter 2 we discussed a novel combining multiple clusterings algorithm, DiCLENS, that can find the number of clusters automatically using well-known objective measures. DiCLENS uses co-associations of objects for similarity calculations. As explained in Section 2.2 co-associations are widely used for combining multiple clusterings. Although it is very costly to find the co-associations between objects, DiCLENS uses a very efficient method to calculate these similarities. DiCLENS does not take any input arguments, which is another advantage.

Co-association based versions of objective measures such as Intra Clusters Similarity (ICS) and Extra Cluster Similarity (ECS) are used to automatically determine the number of final clusterings. Thus, DiCLENS produces compact and well-separated final clusters. Note that because DiCLENS uses co-association based similarity these clusters are the ones that best represent the input clusters. Experimental results on both artificial and real data sets show that DiCLENS performs well and works fast.

Results in Chapter 2 show that ECS-based cluster similarity graphs are superior to both co-association based object graphs and syntactical similarity-based cluster graphs. Clus-

ter graphs provide a better scalability compared to object graphs, because generally the number of clusters in a data set is much less than the number of objects. ECS is more effective as a similarity measure compared to Jaccard but computation of co-associations is very costly. However, computation cost of ECS-based graphs becomes comparable to Jaccard graphs, thanks to the fast ECS computation method.

DiCLENS also shows that using ICS and ECS for evaluating the final clusters works well. Our method can be applied to other combining multiple clustering algorithms to evaluate their final clusterings and to find the parameters that yields to best results.

As a future work, we are planing to apply different partitioning methods on the ECS-based similarity graph.

DBSCAN is a very well known and widely used clustering algorithm which can effectively detect arbitrary shape clusters and noise. However, generating a distance matrix or calculating distances between objects on-the-fly is very costly and even impractical when the number of objects is large. Some indexing techniques may speed up the neighbor search procedure but they do not perform well or not applicable to categorical or binary data.

In Chapter 3 we discussed a novel pruning method that dramatically improves the execution time of DBSCAN on Binary Data and Hamming Distance. Our pruning technique effectively eliminates the data objects that are not in the ε-neighborhood of an object, and considerably reduces the search space on some data sets.

Extensive tests show that DBSCAN_BV performs up to 40 times faster than the original DBSCAN algorithm without sacrificing clustering accuracy. Our novel pruning technique performs better than traditional space indexing techniques such as kd-tree and R-tree. Furthermore, these techniques are not practically applicable to binary data. We also show that our novel pruning method increases the efficiency of DBSCAN and makes it comparable to k-means in terms of CPU usage, and allows its usage in environments with limited resources.

We are hoping to extend our pruning method to make it applicable to other distance metrics such as Jaccard. Application of this technique to a broader range of data sets including real valued ones would be an interesting research.

REFERENCES

Books

Hinneburg, A. and Keim, D. A.: 1998, *An efficient approach to clustering in large multimedia databases with noise*, Bibliothek der Universität Konstanz.

Kaufman, L., Rousseeuw, P. J. and Corporation, E.: 1990, *Finding groups in data: an introduction to cluster analysis*, Vol. 39, Wiley Online Library.

Tan, P., Steinbach, M. and Kumar, V.: 2005, *Introduction to data mining*, Pearson Addison Wesley Boston.

Periodicals

Alizadeh, A. A., Eisen, M. B., Davis, R. E., Ma, C., Lossos, I. S., Rosenwald, A., Boldrick, J. C., Sabet, H., Tran, T., Yu, X., Powell, J. I., Yang, L., Marti, G. E., Moore, T., Hudson, J., Lu, L., Lewis, D. B., Tibshirani, R., Sherlock, G., Chan, W. C., Greiner, T. C., Weisenburger, D. D., Armitage, J. O., Warnke, R., Levy, R., Wilson, W., Grever, M. R., Byrd, J. C., Botstein, D., Brown, P. O. and Staudt, L. M.: 2000, Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling, *Nature* **403**(6769), 503–511.

Ankerst, M., Breunig, M. M., Kriegel, H. and Sander, J.: 1999, OPTICS: ordering points to identify the clustering structure, *Proceedings of the ACM SIGMOD international conference on Management of data* **28**(2), 49–60.

Antoshenkov, G.: 1994, United states patent: 5363098 - byte aligned data compression.

Armstrong, S. A., Staunton, J. E., Silverman, L. B., Pieters, R., den Boer, M. L., Minden, M. D., Sallan, S. E., Lander, E. S., Golub, T. R. and Korsmeyer, S. J.: 2002, MLL translocations specify a distinct gene expression profile that distinguishes a unique leukemia., *Nature Genetics* **30**(1), 41–47.

Ayad, H. G. and Kamel, M. S.: 2010, On voting-based consensus of cluster ensembles, *Pattern Recognition* **43**(5), 1943–1953.

Bandyopadhyay, S. K.: 2011, A survey on brain image segmentation methods, *Journal of Global Research in Computer Science* **2**(2), 4–7.

Bentley, J. L.: 1975, Multidimensional binary search trees used for associative searching, *Communications of the ACM* **18**(9), 509–517.

Bhattacharjee, A., Richards, W. G., Staunton, J., Li, C., Monti, S., Vasa, P., Ladd, C., Beheshti, J., Bueno, R., Gillette, M., Loda, M., Weber, G., Mark, E. J., Lander, E. S., Wong, W., Johnson, B. E., Golub, T. R., Sugarbaker, D. J. and Meyerson, M.: 2001, Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses., *Proceedings of National Academy of Sciences of the U.S.A.* **98**(24), 13790–13795.

Bin, R. D. and Risso, D.: 2011, A novel approach to the clustering of microarray data via nonparametric density estimation, *BMC Bioinformatics* **12**(1), 49.

Bittner, M., Meltzer, P., Chen, Y., Jiang, Y., Seftor, E., Hendrix, M., Radmacher, M., Simon, R., Yakhini, Z., Ben-Dor, A., Sampas, N., Dougherty, E., Wang, E., Marincola, F., Gooden, C., Lueders, J., Glatfelter, A., Pollock, P., Carpten, J., Gillanders, E., Leja, D., Dietrich, K., Beaudry, C., Berens, M., Alberts, D. and Sondak, V.: 2000, Molecular classification of cutaneous malignant melanoma by gene expression profiling., *Nature* **406**(6795), 536–540.

Borah, B. and Bhattacharyya, D.: 2004, An improved sampling-based DBSCAN for large spatial databases, *Intelligent Sensing and Information Processing, 2004. Proceedings of International Conference on*, pp. 92–96.

Bredel, M., Bredel, C., Juric, D., Harsh, G. R., Vogel, H., Recht, L. D. and Sikic, B. I.: 2005, Functional network analysis reveals extended gliomagenesis pathway maps and three novel MYC-interacting genes in human gliomas., *Cancer Research* **65**(19), 8679–8689.

Chen, X., Cheung, S. T., So, S., Fan, S. T., Barry, C., Higgins, J., Lai, K., Ji, J., Dudoit, S., Ng, I. O. L., van de Rijn, M., Botstein, D. and Brown, P. O.: 2002, Gene expression patterns in human liver cancers, *Molecular Biology of the Cell* **13**(6), 1929–1939.

Chowdary, D., Lathrop, J., Skelton, J., Curtin, K., Briggs, T., Zhang, Y., Yu, J., Wang, Y. and Mazumder, A.: 2006, Prognostic gene expression signatures can be measured in tissues collected in RNAlater preservative., *Journal of Molecular Diagnostics* **8**(1), 31–39.

Cristofor, D. and Simovici, D. A.: 2002, Finding median partitions using information-theoretical-based genetic algorithms, *Journal of Universal Computer Science* **8**(2), 153–172.

Dyrskjot, L., Thykjaer, T., Kruh0ffer, M., Jensen, J. L., Marcussen, N., Hamilton-Dutoit, S., Wolf, H. and Orntoft, T. F.: 2003, Identifying distinct classes of bladder carcinoma using microarrays., *Nature Genetics* **33**(1), 90–96.

El-Sonbaty, Y., Ismail, M. and Farouk, M.: 2004, An efficient density based clustering algorithm for large databases, *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pp. 673–677.

Elnakib, A., Gimel'farb, G., Suri, J. S. and El-Baz, A.: 2011, Medical image segmentation: A brief survey, *in* A. S. El-Baz, R. A. U, A. F. Laine and J. S. Suri (eds), *Multi Modality State-of-the-Art Medical Image Segmentation and Registration Methodologies*, Springer New York, New York, NY, pp. 1–39.

Ester, M., Kriegel, H., Sander, J. and Xu, X.: 1996, A density-based algorithm for discovering clusters in large spatial databases with noise, *Proceedings of the 2nd International Conference on Knowledge Discovery and Data mining*, Vol. 1996, Portland: AAAI Press, pp. 226–231.

Fang, Y., Zhen, Z., Huang, Z. and Zhang, C.: 2010, Multi-objective fuzzy clustering method for image segmentation based on Variable-Length intelligent optimization algorithm, *in* Z. Cai, C. Hu, Z. Kang and Y. Liu (eds), *Advances in Computation and Intelligence*, Vol. 6382, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 329–337.

Faouzi, N. E., Leung, H. and Kurian, A.: 2011, Data fusion in intelligent transportation systems: Progress and challenges - a survey, *Information Fusion* **12**(1), 4–10.

Fern, X. Z. and Brodley, C. E.: 2004, Solving cluster ensemble problems by bipartite graph partitioning, *Proceedings of the twenty-first international conference on Machine learning*, ACM, Banff, Alberta, Canada, p. 36.

Frank, J., Mannor, S. and Precup, D.: 2010, A novel similarity measure for time series data with applications to gait and activity recognition, *Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing* p. 407–408. ACM ID: 1864460.

Fred, A. and Jain, A.: 2005, Combining multiple clusterings using evidence accumulation, *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **27**(6), 835–850.

Garber, M. E., Troyanskaya, O. G., Schluens, K., Petersen, S., Thaesler, Z., Pacyna-Gengelbach, M., van de Rijn, M., Rosen, G. D., Perou, C. M., Whyte, R. I., Altman, R. B., Brown, P. O., Botstein, D. and Petersen, I.: 2001, Diversity of gene expression in adenocarcinoma of the lung., *Proceedings of National Academy of Sciences of the U.S.A.* **98**(24), 13784–13789.

Gionis, A., Mannila, H. and Tsaparas, P.: 2007, Clustering aggregation, *Knowledge Discovery from Data, ACM Transactions on* **1**(1), 4.

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D. and Lander, E. S.: 1999, Molecular classification of cancer: class discovery and class prediction by gene expression monitoring., *Science* **286**(5439), 531–537.

Gordon, G. J., Jensen, R. V., Hsiao, L., Gullans, S. R., Blumenstock, J. E., Ramaswamy, S., Richards, W. G., Sugarbaker, D. J. and Bueno, R.: 2002, Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma., *Cancer Research* **62**(17), 4963–4967.

Greene, D., Tsymbal, A., Bolshakova, N. and Cunningham, P.: 2004, Ensemble clustering in medical diagnostics, *Computer-Based Medical Systems, IEEE Symposium on*, Vol. 0, IEEE Computer Society, Los Alamitos, CA, USA, p. 576.

Guo, L., Ren, M., Li, J., Liu, Y. and Ai, C.: 2011, H-cluster: A novel efficient algorithm for data clustering in sensor networks, *Journal of Communications* **6**(2), 168–178.

Guo, Y., Grossman, R., Xu, X., Jäger, J. and Kriegel, H.: 2002, A fast parallel clustering algorithm for large spatial databases, *High Performance Data Mining*, Springer US, pp. 263–290.

Guttman, A.: 1984, R-trees: a dynamic index structure for spatial searching, *Proceedings of the ACM SIGMOD international conference on Management of data* **14**(2), 47–57.

Hadjitodorov, S. T., Kuncheva, L. I. and Todorova, L. P.: 2006, Moderate diversity for better cluster ensembles, *Information Fusion* **7**(3), 264–275.

Hua, M., Lau, M., Pei, J. and Wu, K.: 2009, Continuous K-Means Monitoring with Low Reporting Cost in Sensor Networks, *Knowledge and Data Engineering, IEEE Transactions on* **21**(12), 1679–1691.

Hubert, L. and Arabie, P.: 1985, Comparing partitions, *Journal of Classification* **2**(1), 193–218.

Iam-on, N., Boongoen, T. and Garrett, S.: 2010, LCE: a link-based cluster ensemble method for improved gene expression data analysis, *Bioinformatics* **26**(12), 1513–1519.

Jain, A.: 2010, Data clustering: 50 years beyond K-means, *Pattern Recognition Letters* **31**(8), 651–666.

Jardine, N. and Sibson, R.: 1971, Mathematical taxonomy, *London etc.: John Wiley* .

Jiang, S. and Li, X.: 2009, A hybrid clustering algorithm, *Fuzzy Systems and Knowledge Discovery, Fourth International Conference on*, Vol. 1, IEEE Computer Society, Los Alamitos, CA, USA, pp. 366–370.

Joshi, A., Gangopadhyay, A., Banerjee, M., Baffoe-Bonnie, G., Mohanlal, V. and Wali, R.: 2010, A clustering method to study the loss of kidney function following kidney transplantation, *International Journal of Biomedical Engineering and Technology* **3**(1), 64–82.

Kannan, S., Sathya, A., Ramathilagam, S. and Devi, R.: 2010, Novel segmentation algorithm in segmenting medical images, *Journal of Systems and Software* **83**(12), 2487–2495.

Karypis, G., Han, E. and Kumar, V.: 1999, Chameleon: hierarchical clustering using dynamic modeling, *Computer* **32**(8), 68–75.

Karypis, G. and Kumar, V.: 1998, Multilevel algorithms for Multi-Constraint graph partitioning, *Supercomputing, 1998. SC98. IEEE/ACM Conference on*, p. 28.

Karypis, G. and Kumar, V.: 1999, Multilevel k-way hypergraph partitioning, *Design Automation Conference, 1999. Proceedings. 36th*, pp. 343–348.

Khan, J., Wei, J. S., Ringner, M., Saal, L. H., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Antonescu, C. R., Peterson, C. and Meltzer, P. S.: 2001, Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks., *Nature Medical* **7**(6), 673–679.

Kim, E., Kim, S., Ashlock, D. and Nam, D.: 2009, MULTI-K: accurate classification of microarray subtypes using ensemble k-means clustering, *BMC Bioinformatics* **10**(1), 260.

Lai, C., Chung, P. and Tseng, V. S.: 2010, A novel two-level clustering method for time series data analysis, *Expert Systems with Applications* **37**(9), 6319–6326.

Laiho, P., Kokko, A., Vanharanta, S., Salovaara, R., Sammalkorpi, H., Jarvinen, H., Mecklin, J., Karttunen, T. J., Tuppurainen, K., Davalos, V., Schwartz, S., Arango, D., Makinen, M. J. and Aaltonen, L. A.: 2007, Serrated carcinomas form a subclass of colorectal cancer with distinct molecular basis., *Oncogene* **26**(2), 312–320.

Lapointe, J., Li, C., Higgins, J. P., van de Rijn, M., Bair, E., Montgomery, K., Ferrari, M., Egevad, L., Rayford, W., Bergerheim, U., Ekman, P., DeMarzo, A. M., Tibshirani, R., Botstein, D., Brown, P. O., Brooks, J. D. and Pollack, J. R.: 2004, Gene expression profiling identifies clinically relevant subtypes of prostate cancer., *Proceedings of National Academy of Sciences of the U.S.A.* **101**(3), 811–816.

Larsen, B. and Aone, C.: 1999, Fast and effective text mining using linear-time document clustering, *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, p. 16–22.

Lemire, D., Kaser, O. and Aouiche, K.: 2010, Sorting improves word-aligned bitmap indexes, *Data & Knowledge Engineering* **69**(1), 3–28.

Liang, Y., Diehn, M., Watson, N., Bollen, A. W., Aldape, K. D., Nicholas, M. K., Lamborn, K. R., Berger, M. S., Botstein, D., Brown, P. O. and Israel, M. A.: 2005, Gene expression profiling reveals molecularly and clinically distinct subtypes of glioblastoma multiforme., *Proceedings of National Academy of Sciences of the U.S.A.* **102**(16), 5814–5819.

MacQueen, J.: 1967, Some methods for classification and analysis of multivariate observations, *Proceedings of Fifth Berkeley Symposium on Mathematics, Statistics and Probability (Berkeley, Calif., 1965/66)*, Univ. California Press, Berkeley, Calif., pp. Vol. I: Statistics, pp. 281–297.

Mimaroglu, S. and Aksehirli, E.: 2011, Improving DBSCAN's execution time by using a pruning technique on bit vectors, *Pattern Recognition Letters* **32**(13), 1572 – 1580.

Mimaroglu, S. and Erdil, E.: 2011, Combining multiple clusterings using similarity graph, *Pattern Recognition* **44**(3), 694–703.

Mimaroglu, S. and Yagci, A. M.: 2009, A binary method for fast computation of inter and intra cluster similarities for combining multiple clusterings, *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, ACM, Seoul, Korea, pp. 452–456.

Mohammadi, M., Nikanjam, A. and Rahmani, A.: 2008, An evolutionary approach to clustering ensemble, *Natural Computation, 2008. ICNC '08. Fourth International Conference on*, Vol. 3, pp. 77–82.

Ng, A., Jordan, M. and Weiss, Y.: 2001, On spectral clustering: Analysis and an algorithm, *Advances in Neural Information Processing Systems 14: Proceeding of the 2001 Conference*, pp. 849–856.

Ng, R. and Han, J.: 2002, CLARANS: a method for clustering objects for spatial data mining, *Knowledge and Data Engineering, IEEE Transactions on* **14**(5), 1003–1016.

Nutt, C. L., Mani, D. R., Betensky, R. A., Tamayo, P., Cairncross, J. G., Ladd, C., Pohl, U., Hartmann, C., McLaughlin, M. E., Batchelor, T. T., Black, P. M., von Deimling, A., Pomeroy, S. L., Golub, T. R. and Louis, D. N.: 2003, Gene expression-based classification of malignant gliomas correlates better with survival than histological classification., *Cancer Research* **63**(7), 1602–1607.

Pomeroy, S. L., Tamayo, P., Gaasenbeek, M., Sturla, L. M., Angelo, M., McLaughlin, M. E., Kim, J. Y. H., Goumnerova, L. C., Black, P. M., Lau, C., Allen, J. C., Zagzag, D., Olson, J. M., Curran, T., Wetmore, C., Biegel, J. A., Poggio, T., Mukherjee, S., Rifkin, R., Califano, A., Stolovitzky, G., Louis, D. N., Mesirov, J. P., Lander, E. S. and Golub, T. R.: 2002, Prediction of central nervous system embryonal tumour outcome based on gene expression., *Nature* **415**(6870), 436–442.

Ramaswamy, S., Tamayo, P., Rifkin, R., Mukherjee, S., Yeang, C. H., Angelo, M., Ladd, C., Reich, M., Latulippe, E., Mesirov, J. P., Poggio, T., Gerald, W., Loda, M., Lander, E. S. and Golub, T. R.: 2001, Multiclass cancer diagnosis using tumor gene expression signatures., *Proceedings of National Academy of Sciences of the U.S.A.* **98**(26), 15149–15154.

Rand, W. M.: 1971, Objective criteria for the evaluation of clustering methods, *Journal of the American Statistical Association* **66**(336), 846–850.

Risinger, J. I., Maxwell, G. L., Chandramouli, G. V. R., Jazaeri, A., Aprelikova, O., Patterson, T., Berchuck, A. and Barrett, J. C.: 2003, Microarray analysis reveals distinct gene expression profiles among different histologic types of endometrial cancer., *Cancer Research* **63**(1), 6–11.

Rui, X. and Chunhong, D.: 2008, An improved clustering algorithm, *Computational Intelligence and Design, 2008. ISCID '08. International Symposium on*, Vol. 1, pp. 394–397.

Sakellariou, R., Gurd, J., Freeman, L., Keane, J., Arlia, D. and Coppola, M.: 2001, Experiments in parallel clustering with DBSCAN, *Euro-Par 2001 Parallel Processing*, Vol. 2150 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 326–331.

Shipp, M. A., Ross, K. N., Tamayo, P., Weng, A. P., Kutok, J. L., Aguiar, R. C. T., Gaasenbeek, M., Angelo, M., Reich, M., Pinkus, G. S., Ray, T. S., Koval, M. A., Last, K. W., Norton, A., Lister, T. A., Mesirov, J., Neuberg, D. S., Lander, E. S., Aster, J. C. and Golub, T. R.: 2002, Diffuse large b-cell lymphoma outcome prediction

by gene-expression profiling and supervised machine learning., *Nature Medicine* **8**(1), 68–74.

Singh, D., Febbo, P. G., Ross, K., Jackson, D. G., Manola, J., Ladd, C., Tamayo, P., Renshaw, A. A., D'Amico, A. V., Richie, J. P., Lander, E. S., Loda, M., Kantoff, P. W., Golub, T. R. and Sellers, W. R.: 2002, Gene expression correlates of clinical prostate cancer behavior., *Cancer Cell* **1**(2), 203–209.

Sneath, P. H. and Sokal, R. R.: 1962, Numerical taxonomy, *Nature* **193**, 855–860.

Stanoev, A., Trpevski, I. and Kocarev, L.: 2011, An agglomerative clustering technique based on a global similarity metric, *in* M. Gusev and P. Mitrevski (eds), *ICT Innovations 2010*, Vol. 83, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 266–275.

Strehl, A. and Ghosh, J.: 2000, A scalable approach to balanced, high-dimensional clustering of market-baskets, *High Performance Computing—HiPC 2000* pp. 525–536.

Strehl, A. and Ghosh, J.: 2002, Cluster ensembles - a knowledge reuse framework for combining multiple partitions, *Journal on Machine Learning Research (JMLR)* **3**, 583–617.

Su, A. I., Welsh, J. B., Sapinoso, L. M., Kern, S. G., Dimitrov, P., Lapp, H., Schultz, P. G., Powell, S. M., Moskaluk, C. A., Frierson, H. F. and Hampton, G. M.: 2001, Molecular classification of human carcinomas by use of gene expression signatures., *Cancer Research* **61**(20), 7388–7393.

Tomlins, S. A., Mehra, R., Rhodes, D. R., Cao, X., Wang, L., Dhanasekaran, S. M., Kalyana-Sundaram, S., Wei, J. T., Rubin, M. A., Pienta, K. J., Shah, R. B. and Chinnaiyan, A. M.: 2007, Integrative molecular concept modeling of prostate cancer progression., *Nature Genetics* **39**(1), 41–51.

Topchy, A., Jain, A. K. and Punch, W.: 2003, Combining multiple weak clusterings, *Proceedings of the Third IEEE International Conference on Data Mining*, p. 331.

Tsai, C. and Liu, C.: 2006, KIDBSCAN: a new efficient data clustering algorithm, *Artificial Intelligence and Soft Computing – ICAISC 2006*, Vol. 4029 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 702–711.

Vega-Pons, S., Correa-Morris, J. and Ruiz-Shulcloper, J.: 2010, Weighted partition consensus via kernels, *Pattern Recognition* **43**(8), 2712–2724.

Viswanath, P. and Babu, V. S.: 2009, Rough-DBSCAN: a fast hybrid density based clustering method for large data sets, *Pattern Recognition Letters* **30**(16), 1477–1488.

Wang, X., Yang, C. and Zhou, J.: 2009, Clustering aggregation by probability accumulation, *Pattern Recognition* **42**(5), 668–675.

West, M., Blanchette, C., Dressman, H., Huang, E., Ishida, S., Spang, R., Zuzan, H., Olson, J. A., Marks, J. R. and Nevins, J. R.: 2001, Predicting the clinical status of human breast cancer by using gene expression profiles., *Proceedings of National Academy of Sciences of the U.S.A.* **98**(20), 11462–11467.

Wu, K., Otoo, E. J. and Shoshani, A.: 2006, Optimizing bitmap indices with efficient compression, *Database Systems, ACM Transaction on* **31**(1), 1–38.

Wu, K., Shoshani, A. and Otoo, E.: 2004, United states patent: 6831575 - word aligned bitmap compression method, data structure, and apparatus.

Yeoh, E., Ross, M. E., Shurtleff, S. A., Williams, W. K., Patel, D., Mahfouz, R., Behm, F. G., Raimondi, S. C., Relling, M. V., Patel, A., Cheng, C., Campana, D., Wilkins, D., Zhou, X., Li, J., Liu, H., Pui, C., Evans, W. E., Naeve, C., Wong, L. and Downing, J. R.: 2002, Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling., *Cancer Cell* **1**(2), 133–143.

Yu, Z., Wong, H. and Wang, H.: 2007, Graph-based consensus clustering for class discovery from gene expression data, *Bioinformatics* **23**(21), 2888–2896.

Yucenur, G. N. and Demirel, N. C.: 2011, A new geometric shape-based genetic clustering algorithm for the multi-depot vehicle routing problem, *Expert Systems with Applications* **38**(9), 11859–11865.

Zahn, C. T.: 1971, Graph-theoretical methods for detecting and describing gestalt clusters, *Computers, IEEE Transactions on* **100**(1), 68–86.

Zhou, A., Zhou, S., Cao, J., Fan, Y. and Hu, Y.: 2000, Approaches for scaling DBSCAN algorithm to large spatial databases, *Journal of Computer Science and Technology* **15**(6), 509–526.

Other References

Frank, A. and Asuncion, A.: 2010, *UCI Machine Learning Repository*, University of California, Irvine, School of Information and Computer Sciences.
URL: *http://archive.ics.uci.edu/ml*

APPENDICES

APPENDIX A. FIGURES

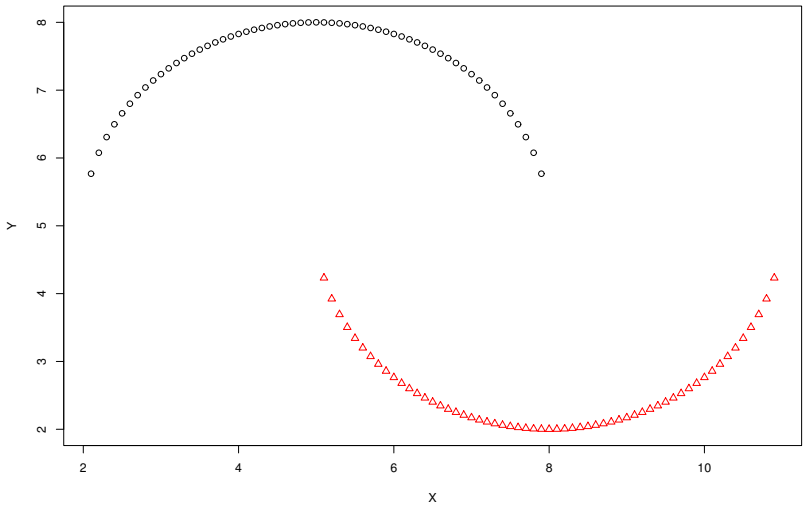

Figure A.1a DiCLENS Final Clustering, ARI:1.0

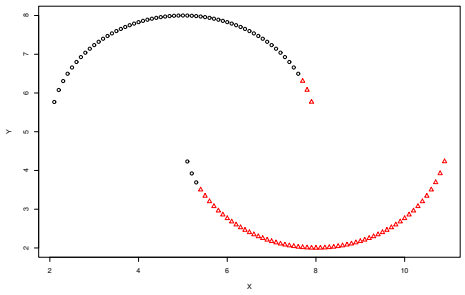

Figure A.1b Input 1, ARI:0.80

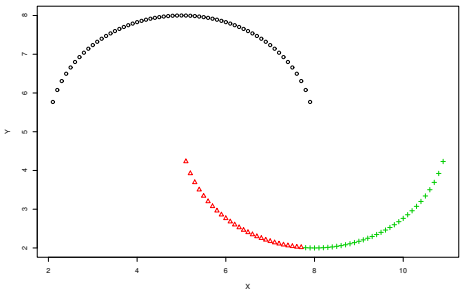

Figure A.1c Input 2, ARI:0.75

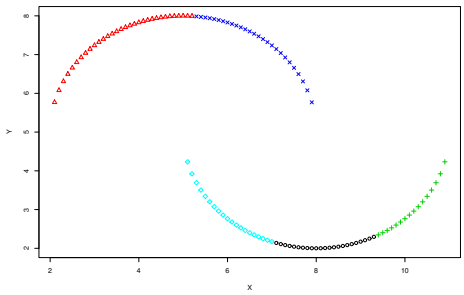

Figure A.1d Input 3, ARI:0.41

Figure A.1: DiCLENS on 2-half rings data set, and 3 input clusterings

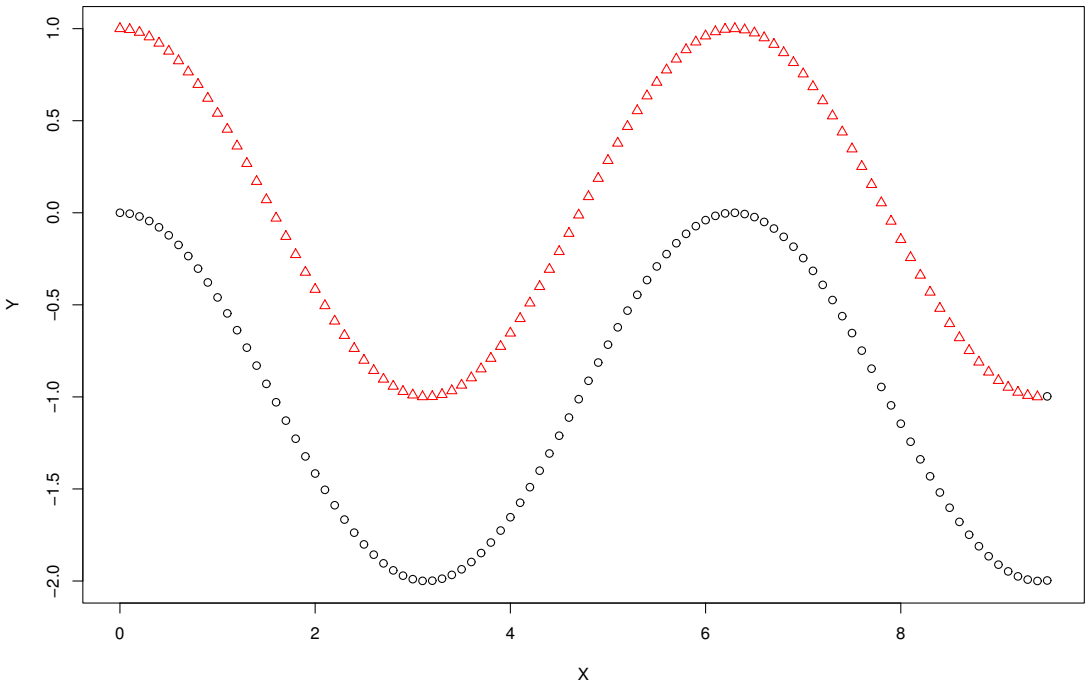

Figure A.2a DiCLENS Final Clustering, ARI:0.98

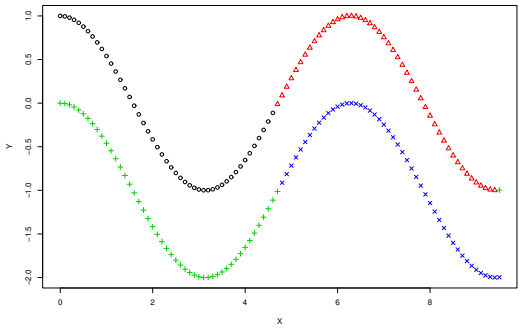

Figure A.2b Input 1, ARI:0.49

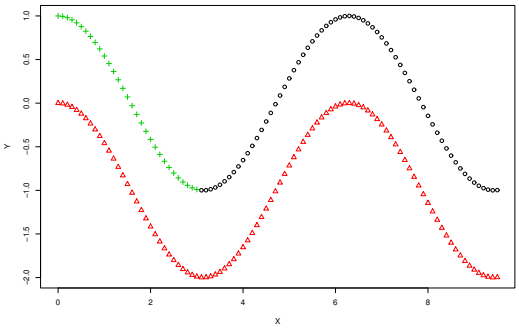

Figure A.2c Input 2, ARI:0.78

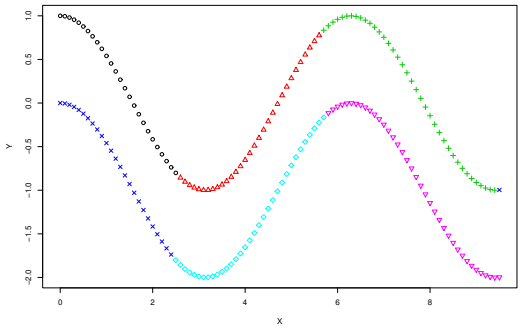

Figure A.2d Input 3, ARI:0.33

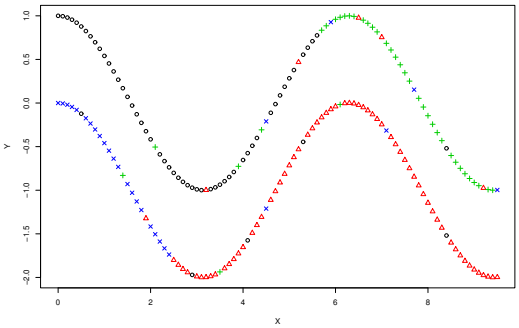

Figure A.2e Input 4, ARI: 0.38

Figure A.2: DiCLENS on 2-curve data set, and 4 input clusterings

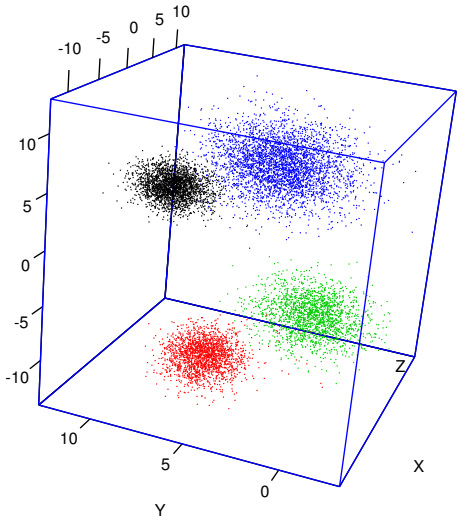

Figure A.3a DiCLENS Final Clustering, ARI:0.99

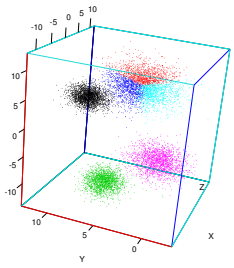

Figure A.3b Input 1, ARI:0.76

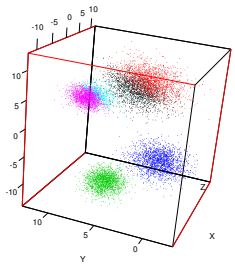

Figure A.3c Input 2, ARI:0.73

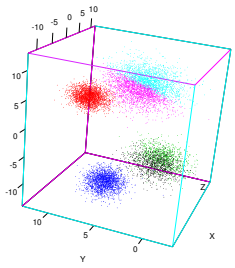

Figure A.3d Input 3, ARI:0.76

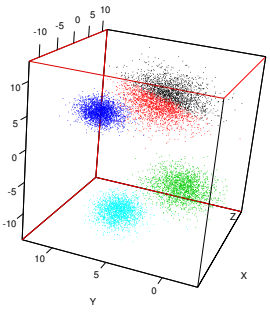

Figure A.3e Input 4, ARI:0.82

Figure A.3: DiCLENS on 4c10k data set, and 4 input clusterings

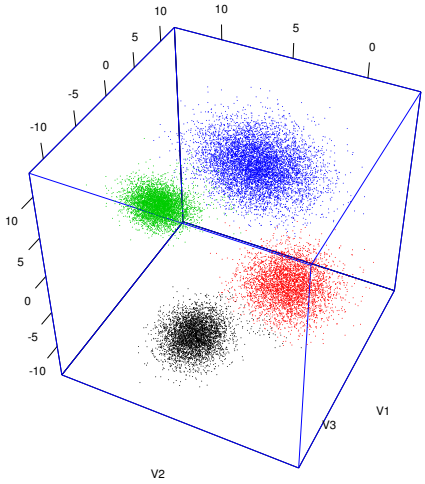

Figure A.4a DiCLENS Final Clustering, ARI:0.98

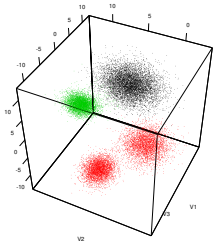

Figure A.4b Input 1, ARI:0.80

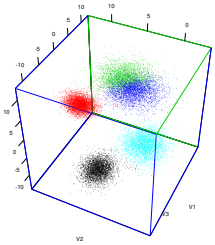

Figure A.4c Input 2, ARI:0.81

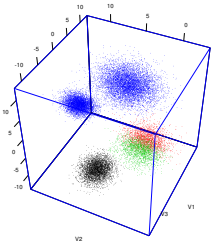

Figure A.4d Input 3, ARI:0.57

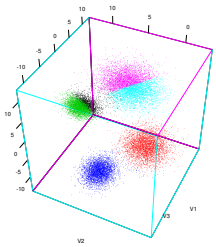

Figure A.4e Input 4, ARI:0.72

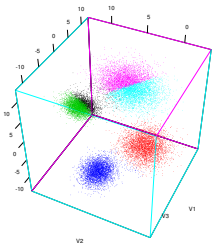

Figure A.4f Input 5, ARI:0.64

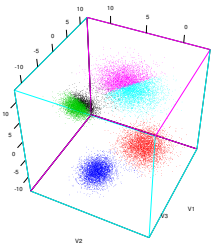

Figure A.4g Input 6, ARI:0.81

Figure A.4: DiCLENS on 4c20k data set and the first 6 of the input clusterings

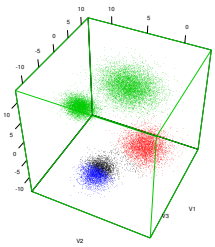

Figure A.4h Input 7, ARI:0.57

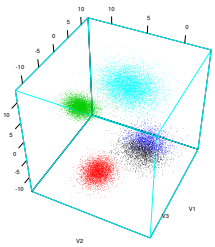

Figure A.4i Input 8, ARI:0.93

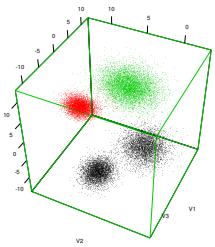

Figure A.4j Input 9, ARI:0.80

Figure A.4: The last 3 of the input clusterings of 4c20k data set

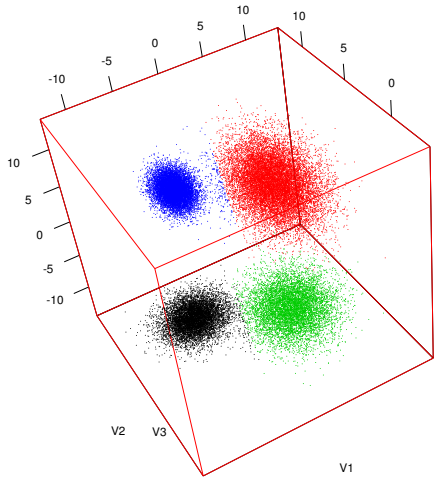

Figure A.5a DiCLENS Final Clustering, ARI:0.98

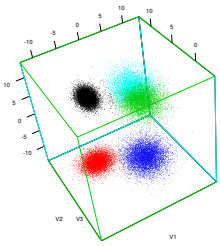

Figure A.5b Input 1, ARI:0.81

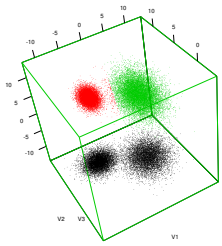

Figure A.5c Input 2, ARI:0.80

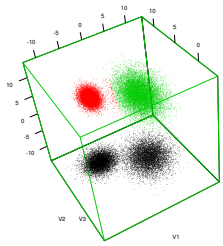

Figure A.5d Input 3, ARI:0.80

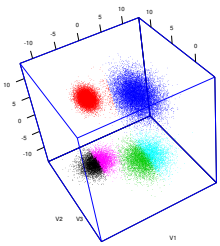

Figure A.5e Input 4, ARI:0.87

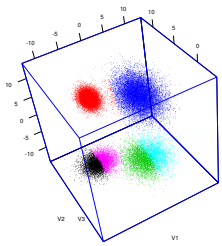

Figure A.5f Input 5, ARI:0.58

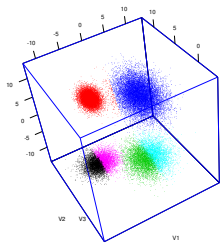

Figure A.5g Input 6, ARI:0.76

Figure A.5: DiCLENS on 4c40k data set, and the first 6 of the input clusterings

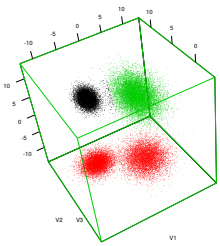

Figure A.5h Input 7, ARI:0.80

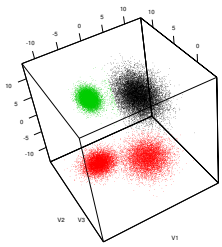

Figure A.5i Input 8, ARI:0.80

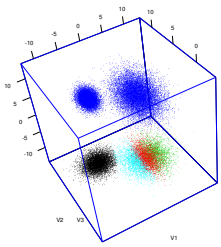

Figure A.5j Input 9, ARI:0.56

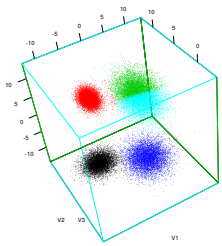

Figure A.5k Input 10, ARI:0.81

Figure A.5: The last 4 of the input clusterings of 4c40k data set

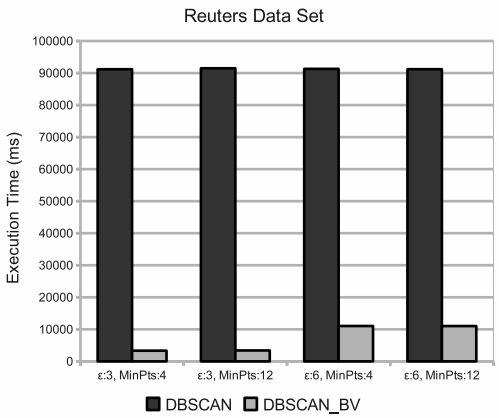

Figure A.6a Reuters

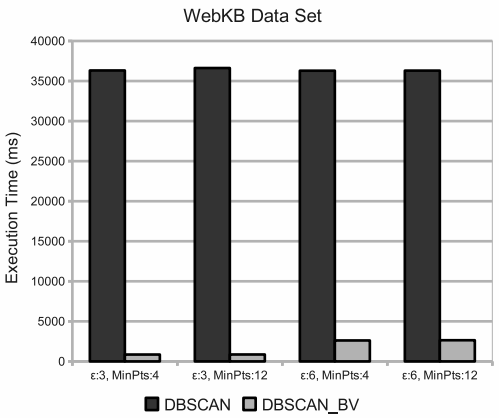

Figure A.6b WebKB

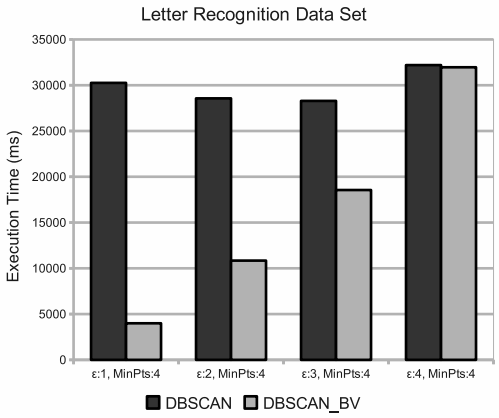

Figure A.6c Letter Recognition

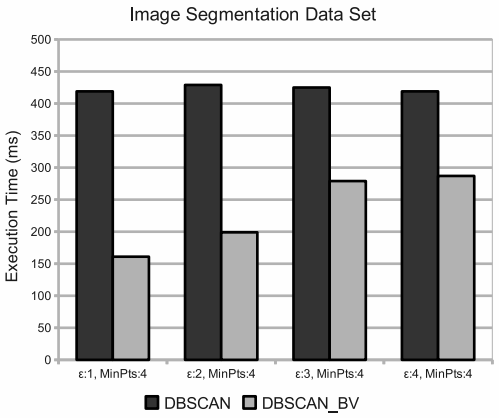

Figure A.6d Image Segmentation

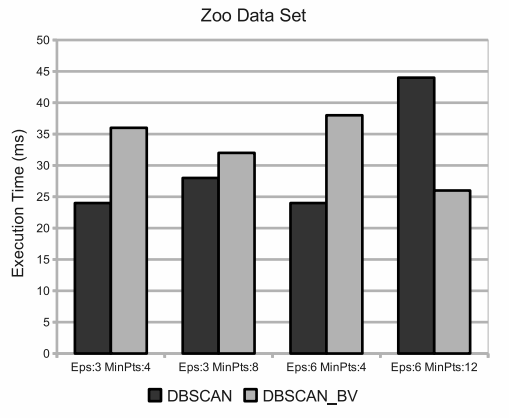

Figure A.6e Zoo

Figure A.6: Performance Comparison on real data sets

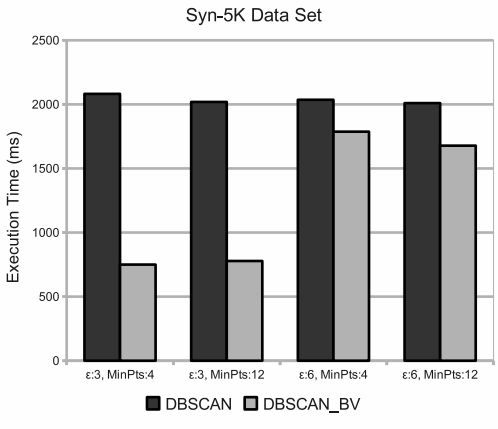

Figure A.7a Syn-5K

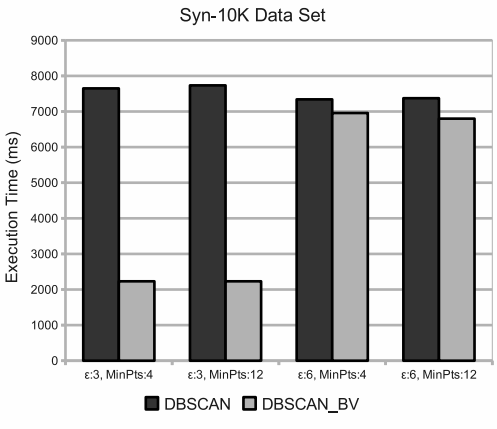

Figure A.7b Syn-10K

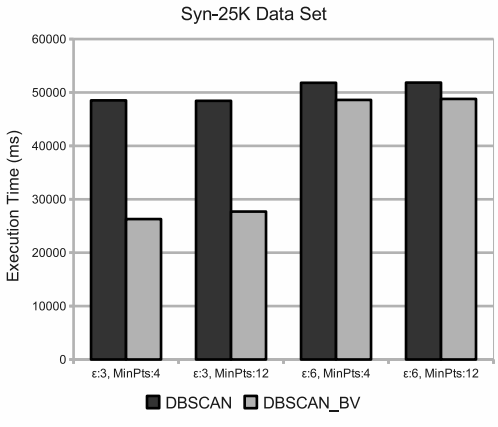

Figure A.7c Syn-25K

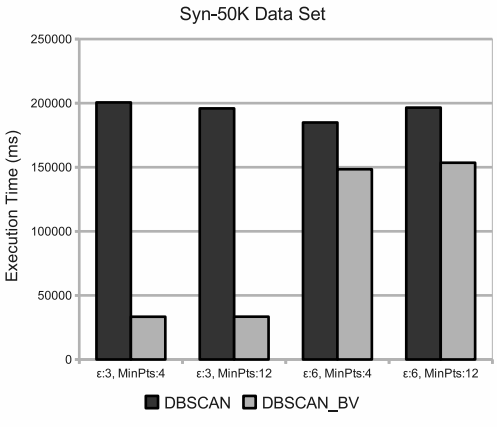

Figure A.7d Syn-50K

Figure A.7: Performance Comparison on synthetic data sets

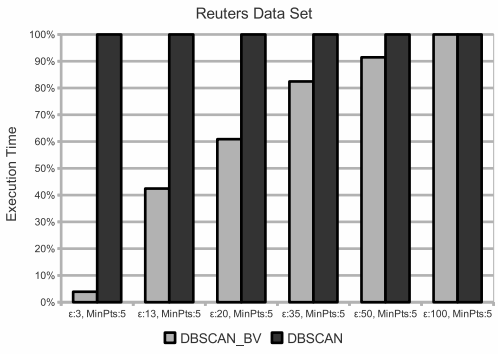

Figure A.8a Reuters

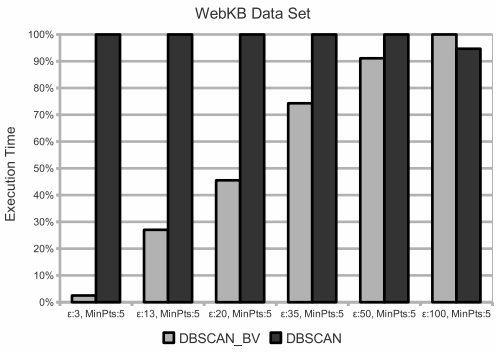

Figure A.8b WebKB

Figure A.8: Text data sets with a large range of ε values

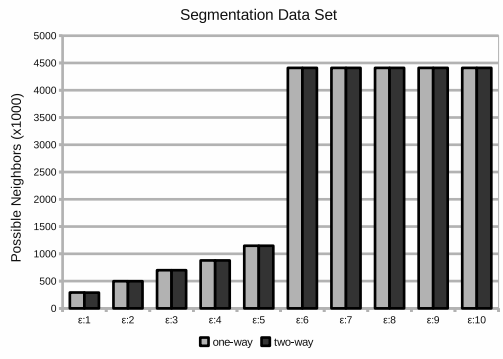

Figure A.9a Image Segmentation

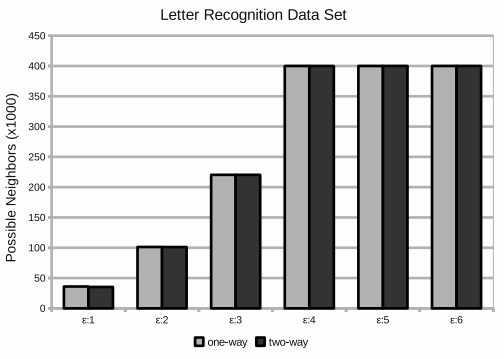

Figure A.9b Letter Recognition

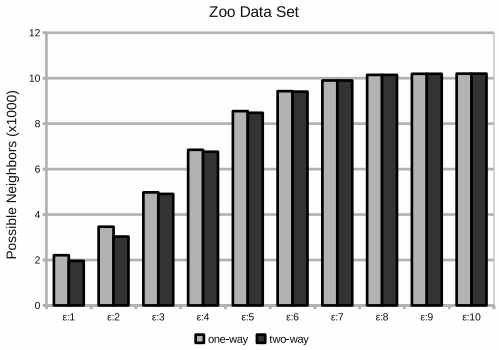

Figure A.9c Zoo

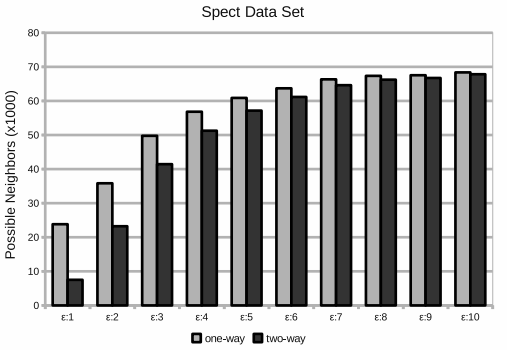

Figure A.9d Spect

Figure A.9: Possible neighbors detected with two-way and one-way pruning

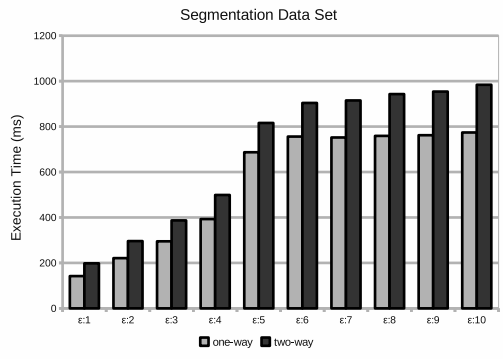

Figure A.10a Image Segmentation

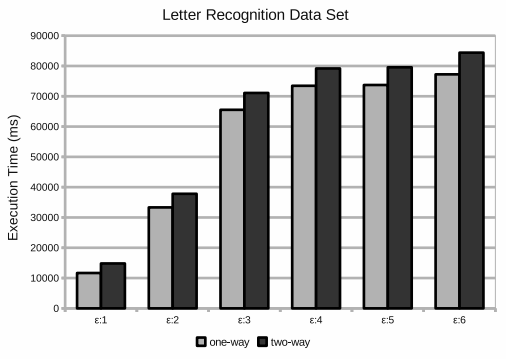

Figure A.10b Letter Recognition

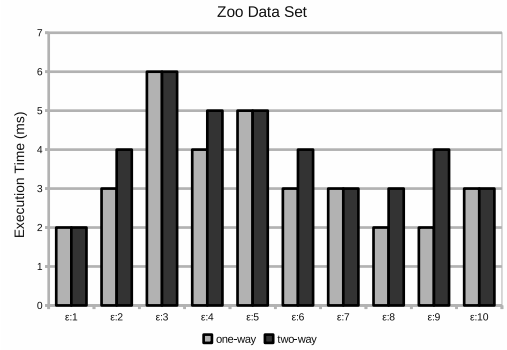

Figure A.10c Zoo

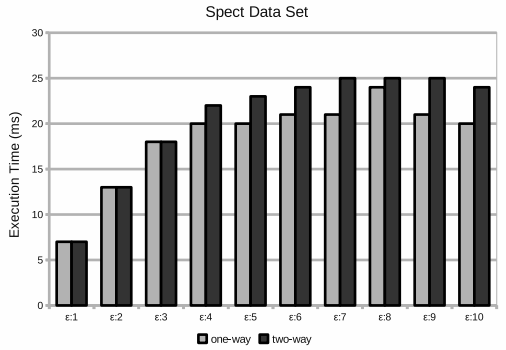

Figure A.10d Spect

Figure A.10: Execution time of two-way and one-way pruning

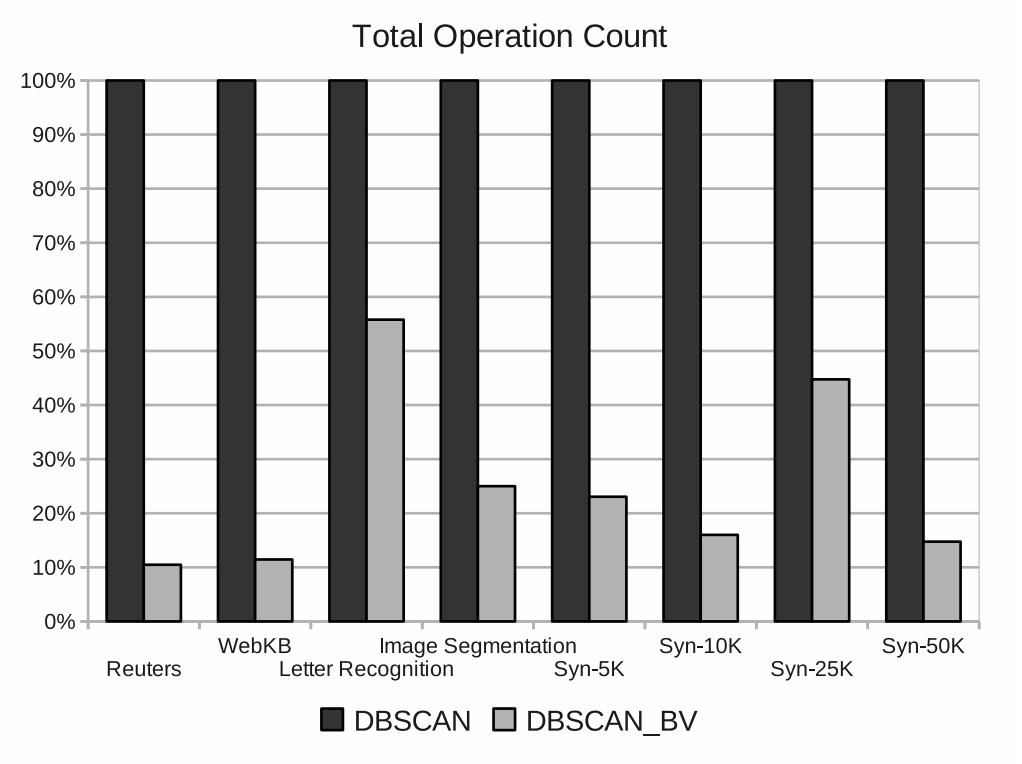

Figure A.11: Total number of operations

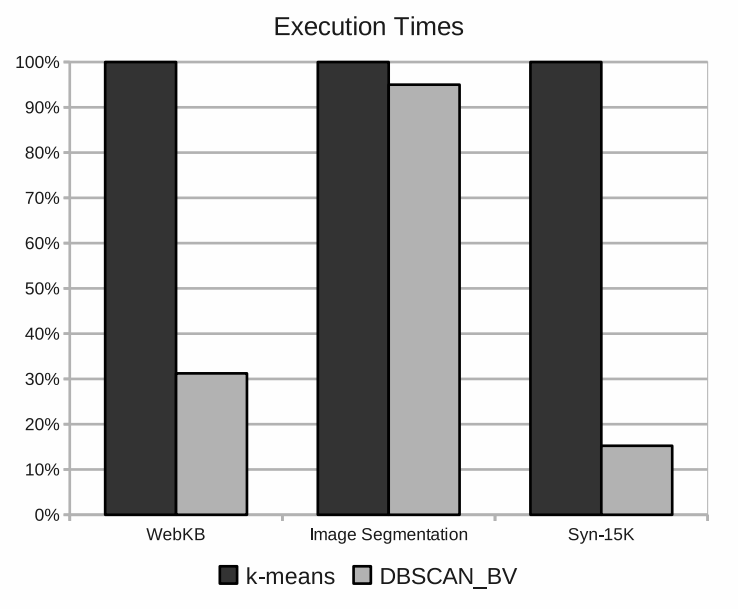

Figure A.12: k-means vs. DBSCAN_BV

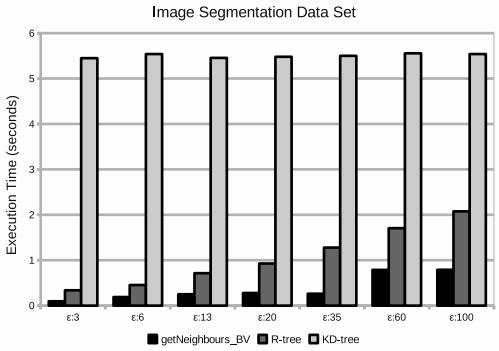

Figure A.13a Image Segmentation

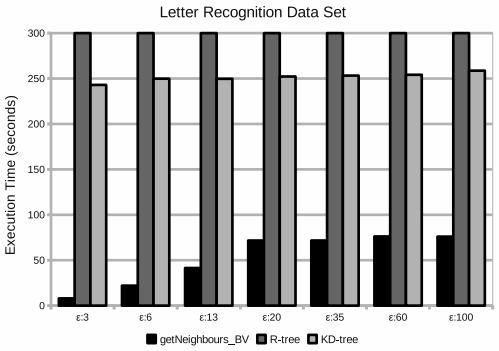

Figure A.13b Letter Recognition

Figure A.13: getNeighbors_BV **versus R-tree and** kd**-tree**

APPENDIX B. TABLES

Table B.1: Quality results of final clusterings on non-biological data sets

Data Set	*DiCLENS*	*MCLA*	*CSPA*	*HGPA*	*LCE*	*COMUSA*
2-curve	**0.98**	**0.98**	**0.98**	0.29	**0.98**	0.28
2-half ring	**1.00**	**1.00**	**1.00**	0.58	**1.00**	0.09
Glass	**1.00**	0.99	0.51	0.21	0.73	0.08
4c10k	**0.99**	0.79	0.68	0	0.98	0.24
4c20k	**0.98**	**0.98**	N/A	0	**0.98**	N/A
4c40k	0.80	**0.98**	N/A	0	**0.98**	N/A
Imageseg	**0.92**	**0.92**	0.91	0.62	0.89	0

Table B.2: Quality results of final clusterings on gene expression data sets

Data Set	*DiCLENS*	*MCLA*	*CSPA*	*HGPA*	*LCE*	*COMUSA*
Bladder carcinoma	**0.59**	0.56	0.32	0.36	0.39	0.05
Breast Cancer	**0.63**	0.56	0.44	0.50	0.56	0.23
Breast-Colon tumors	**0.92**	0.85	0.62	0.75	**0.92**	0.39
Carcinomas	0.45	0.47	0.41	0.45	**0.57**	0.15
Central nervous system-1	**1.00**	0.88	0.33	0.33	**1.00**	0.55
Central nervous system-2	0.51	0.51	0.36	0.44	**0.61**	0.27
Endometrial cancer	**1.00**	0.92	0.55	0.42	0.92	0.37
Glioblastoma multiforme	**0.46**	0.16	0.13	0.13	0.16	0.06
Gliomagenesis	**0.55**	0.38	0.25	0.34	0.37	0.13
Gliomas-1	**0.92**	**0.92**	0.73	0.88	**0.92**	0.74
Gliomas-2	0.72	0.60	0.39	0.35	**1.00**	0.32
Gliomas-3	**1.00**	**1.00**	0.51	0.27	**1.00**	0.74
Hepatocellular carcinoma	**0.72**	0.62	0.65	0.02	0.64	0.05
Leukemia-1	**0.96**	0.48	0.10	0.11	**0.96**	0.33
Leukemia-2	**0.38**	0.31	0.25	0.26	0.37	0.01
Leukemia-3	**0.94**	**0.94**	0.44	0.24	0.56	0.15
Leukemia-4	**0.92**	**0.92**	0.81	**0.92**	**0.92**	0.65
Leukemia-5	**0.94**	0.84	0.52	0.33	0.79	0.42
Leukemia-6	**0.84**	0.74	0.54	0.48	0.79	0.05
Lung tumor-1	**0.55**	0.29	0.12	0.11	0.30	0.47
Lung tumor-2	**0.28**	0.25	0.09	0.09	0.15	0.03
Lymphoma-1	**0.50**	0.26	0.37	0.12	0.37	0.02
Lymphoma-2	**0.83**	0.37	0.39	0.35	0.36	0.20
Lymphoma-3	**0.31**	0.20	0.25	0.17	0.25	0.02
Melanoma	**0.89**	**0.89**	**0.89**	0.20	0.70	0.25
Mesothelioma	**0.89**	0.78	0.12	0.14	0.78	0.49
Multi-tissue	0.38	0.32	0.32	0.32	**0.41**	0.39
Prostate cancer-1	**0.89**	**0.89**	0.58	0.52	0.66	0.26
Prostate cancer-2	0.90	**0.92**	0.60	0.47	0.79	0.25
Prostate cancer-3	**0.65**	0.50	0.46	0.27	0.47	0.02
Prostate cancer-4	0.78	0.71	0.44	0.32	**0.86**	0.02
Prostate cancer-5	**0.13**	0.07	0.07	0.05	0.02	0.11
Round blue-cell tumor	**0.94**	0.66	0.49	0.70	0.89	0.27
Serrated carcinomas	**1.00**	0.88	0.20	0.15	**1.00**	0.11

Table B.3: Number of clusters

Data Set	True Cluster #	DiCLENS
Bladder carcinoma	**3**	**2**
Breast Cancer	**2**	**2**
Breast-Colon tumors	**2**	**2**
Carcinomas	10	6
Central nervous system-1	**2**	**2**
Central nervous system-2	**5**	**4**
Endometrial cancer	**4**	**4**
Glioblastoma multiforme	**3**	**2**
Gliomagenesis	**3**	**2**
Gliomas-1	**4**	**4**
Gliomas-2	**2**	**2**
Gliomas-3	**2**	**2**
Hepatocellular carcinoma	**2**	**3**
Leukemia-1	**2**	**2**
Leukemia-2	6	3
Leukemia-3	**2**	**2**
Leukemia-4	**3**	**3**
Leukemia-5	**2**	**2**
Leukemia-6	**3**	**3**
Lung tumor-1	5	3
Lung tumor-2	4	2
Lymphoma-1	**2**	**2**
Lymphoma-2	**3**	**2**
Lymphoma-3	**2**	**3**
Melanoma	**2**	**2**
Mesothelioma	**2**	**2**
Multi-tissue	14	5
Prostate cancer-1	**5**	**5**
Prostate cancer-2	**4**	**5**
Prostate cancer-3	**3**	**2**
Prostate cancer-4	**4**	**5**
Prostate cancer-5	2	6
Round blue-cell tumor	**4**	**5**
Serrated carcinomas	**2**	**2**
2-curve	**2**	**2**
2-half rings	**2**	**2**
4c10k	**4**	**4**
4c20k	**4**	**4**
4c40k	**4**	**4**
Glass	**6**	**6**
Imageseg	**7**	**7**

Table B.4: Execution time results of clustering ensemble methods (msec)

Data Set	*DiCLENS*	*MCLA*	*CSPA*	*HGPA*	*LCE*	*COMUSA*
Bladder carcinoma	59	6	5	59	205	54
Breast Cancer	87	7	7	55	182	103
Breast-Colon tumors	295	7	13	69	369	97
Carcinomas	1139	12	37	249	1533	63
Central nervous system-1	10	5	11	19	67	76
Central nervous system-2	68	7	11	89	340	45
Endometrial cancer	14	7	6	46	103	6
Glioblastoma multiforme	71	6	7	53	169	4
Gliomagenesis	58	10	7	62	297	7
Gliomas-1	28	6	7	61	132	17
Gliomas-2	17	6	12	30	91	3
Gliomas-3	11	7	10	27	50	6
Hepatocellular carcinoma	124	7	22	46	420	57
Leukemia-1	448	8	44	72	476	173
Leukemia-2	2431	16	81	202	2195	132
Leukemia-3	5	7	8	25	78	27
Leukemia-4	207	9	9	124	321	38
Leukemia-5	67	6	8	54	211	13
Leukemia-6	103	7	7	102	268	7
Lung tumor-1	1491	13	31	240	1162	134
Lung tumor-2	69	7	8	82	361	15
Lymphoma-1	124	15	7	36	176	8
Lymphoma-2	134	10	9	81	352	12
Lymphoma-3	162	7	12	55	289	16
Melanoma	2	5	6	11	41	4
Mesothelioma	622	9	25	72	511	101
Multi-tissue	677	10	41	216	3793	97
Prostate cancer-1	38	6	12	79	438	26
Prostate cancer-2	29	6	13	58	187	50
Prostate cancer-3	17	7	8	43	147	53
Prostate cancer-4	29	6	14	48	157	33
Prostate cancer-5	365	10	12	77	463	21
Round blue-cell tumor	112	10	11	93	302	34
Serrated carcinomas	51	6	9	25	72	4
2-curve	58	12	29	20	125	91
2-half ring	5	6	19	13	80	71
Glass	84	8	31	55	543	100
4c10k	281	12	36134	551	8585	862288
4c20k	2068	152	N/A	1654	29319	N/A
4c40k	6018	118	N/A	5119	113894	N/A
Imageseg	943	49	5375	200	8876	11977

CV

Name Surname	:	Mehmet Emin AKŞEHİRLİ
Address	:	Fulya mah. Ayşecik sok. Doğan Ap. No:16/6 Şişli-İstanbul - TURKEY
Date and Place of Birth	:	14.01.1984 Ankara
Languages	:	Turkish (native), English (fluent)
B.S.	:	Istanbul Technical University - Computer Engineering
M.S.	:	Bahçeşehir University
Institute	:	The Graduate School of Natural and Applied Sciences
Program	:	Computer Engineering
Publications	:	S. Mimaroglu and E. Aksehirli, 2011, Improving DBSCAN's Performance by Using a Pruning Technique on Bit Vectors, *Pattern Recognition Letters*, 32(13), 1572–1580, Elsevier S. Mimaroglu and E. Aksehirli, 2010, DICLENS: Divisive Clustering Ensemble with Automatic Cluster Number, *Journal*, under revision
Work Experience	:	Bahcesehir University Computer Engineering Department *Research and Teaching Assistant* (Istanbul, 2010 - today) Industry *Software Engineer, System Engineer* (Turkey, 2006 - 2009)