In the Name of ALLAH the Merciful the Compassionate

Islamic University of Gaza
Deanery of Higher Studies
Faculty of Engineering
Computer Engineering Department

EFFICIENT DATA CLUSTERING ALGORITHMS

Mohammed B. Abubaker

Advisor

Prof. Hatem M. Hamad
(Professor of Computer Engineering)

A Thesis Submitted to the Department of Computer Engineering in partial fulfillment of the requirements for the degree of Master of Science in Computer Engineering.
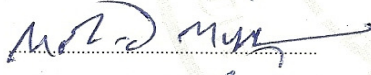
Palestine, Gaza
2011 - 1432

بِسْمِ اللهِ الرَّحْمَنِ الرَّحِيمِ

الجامعة الإسلامية – غزة
The Islamic University - Gaza

نتيجة الحكم على أطروحة ماجستير

بناءً على موافقة عمادة الدراسات العليا بالجامعة الإسلامية بغزة على تشكيل لجنة الحكم على أطروحـة الباحث/ محمـد بسـام محمـد أبـو بكـر لنيـل درجـة الماجسـتير فـي كليـة *الهندسـة* قسم هندسة الحاسوب وموضوعها:

Efficient Data Clustering Algorithms

وبعد المناقشة التي تمت اليوم الثلاثاء 26 ذو القعدة 1432هـ، الموافق 2011/10/25م الساعة الثانية عشرة ظهراً، اجتمعت لجنة الحكم على الأطروحة والمكونة من:
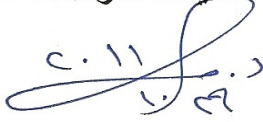
أ.د. حاتم محمود حمـاد	مشرفاً ورئيساً	
أ.د. محمد أميـن مكـي	مناقشاً داخلياً	
د. وسام محمود عاشور	مناقشاً داخلياً	

وبعد المداولة أوصت اللجنة بمنح الباحث درجة الماجستير في كلية *الهندسة*/ قسم هندسة الحاسوب.

واللجنة إذ تمنحه هذه الدرجة فإنها توصيه بتقوى الله ولزوم طاعته وأن يسخر علمه في خدمة دينه ووطنه.

والله ولي التوفيق ،،،

عميد الدراسات العليا

أ.د. فؤاد علي العاجز

TABLE OF CONTENTS

ABSTRACT

Data Clustering is one of the most important issues in data mining and machine learning. Clustering is a task of discovering homogenous groups of the studied objects. Recently, many researchers have a significant interest in developing clustering algorithms. The most problem in clustering is that we do not have prior information knowledge about the given dataset. Moreover, the choice of input parameters such as the number of clusters, number of nearest neighbors and other factors in these algorithms make the clustering more challengeable topic. Thus any incorrect choice of these parameters yields bad clustering results. Furthermore, these algorithms suffer from unsatisfactory accuracy when the dataset contains clusters with different complex shapes, densities, sizes, noise and outliers. In this thesis, we propose a new approach for unsupervised clustering task. Our approach consists of three phases of operations. In the first phase we use the most widely used clustering technique which is Kmeans algorithm for its simplicity and speed in practice. We benefit just from one run of Kmeans, despites its accuracy, to discover and analyze the given dataset by catching preliminary clusters to insure closely grouping sets. The second phase takes these initial groups for processing them in a parallel fashion using shrinking based on the convex hull of the initial groups. From the second phase we obtain a set of sub-clusters of the given dataset. Hence, the third phase considers these sub-clusters for merging process based on the Delaunay triangulation. This new algorithm is named as Kmeans-Based Convex Hull Triangulation clustering algorithm (KBCHT). We present experiments that provide the strength of our new algorithm in discovering clusters with different non-convex shapes, sizes, densities, noise and outliers even though the bad initial conditions used in its first phase. These experiments show the superiority of our proposed algorithm when comparing with most competing algorithms.

Keywords: data clustering, data mining, machine learning, homogenous groups, non-convex shapes, unsupervised clustering, Kmeans, shrinking, convex hull and triangulation.

خوارزميات فعالة للتنقيب من البيانات

للباحث: م. محمد بسام أبوبكر

تعتبر عملية جميع البيانات واحدة من أهم الموضوعات دراسة في مجال تنقيب البيانات و تعلم الآلة ، و نقصد بعملية التجميع هو اكتشاف المجموعات المتجانسة من الكائنات التى تشملها عينة الدراسة.

في الآونة الاخيرة ، عملية تطوير خوارزميات التجميع جذبت اهتمام العديد من الباحثين ، حيث إن المشكلة الأهم فيها هي أننا لا نملك معرفة مسبقة عن مجموعة بيانات الدراسة المطلوب تحليلها و ايجاد الأنماط المتجانسة فيها. و علاوة على ذلك ، فإن معظم الخوارزميات الموجودة تتطلب من المستخدم ادخال معلومات حول مجموعة بيانات الدراسة مثل عدد المجموعات الموجود ، و عدد الجيران الأقرب و غيرها من المدخلات التي جعل من هذا الموضوع أكثر الموضوعات تحدياً للباحثين. و بالتالي فإن أي اختيار لمدخل غير صحيح يعطي نتائج سيئة. و من ناحية أخرى ، تعاني هذه الخوارزميات من نتائج غير مرضية عندما تحتوي مجموعة بيانات الدراسة على أشكال معقدة مختلفة في الكثافة و الحجم و بها ضوضاء و قيم متطرفة.

في هذه الأطروحة ، عملنا على تطوير خوارزمية جديدة لتجميع البيانات لا تخضع لرحمة المدخلات ، و نهجنا على أن تتكون هذه الخوارزمية على ثلاث مراحل: في المرحلة الأولى عملنا على استخدام خوارزمية Kmeans و هي مستخدمة على نطاق واسع لبساطتها و سرعة تنفيذها ، و عمدنا على الاستفادة من تشغيلة واحدة لهذه الخوارزمية بغض النظر عن نتائجها ، و ذلك لبداية اكتشاف و تحليل مجموعة بيانات الدراسة من خلال هذه المجموعات الأولية. المرحلة الثانية تعنى بتحليل المجموعات الأولية بطريقة متوازية باستخدام عملية الانكماش استناداً على مبدأ الهيكل المحدب للمجموعات الأولية. و كنتيجة للمرحلة الثانية فإننا نحصل على مجموعة من المجموعات الأولية لبيانات الدراسة ، و بالتالي فإن المرحلة الثالثة تعتمد على مبدأ التثليث لتمجيع هذه المجموعات الصغيرة للحصول على النتيجة النهائية المرجوة . و عملنا على تسمية هذه الخوارزميـــة Kmeans-Based Convex Hull Triangulation clustering algorithm و اختصاراً بـ (KBCHT).

و قد خلصت الدراسة بناء على العديد من التجارب العملية التى أجريناها على فاعلية و قوة هذه الخوارزمية في اكتشاف المجموعات المتجانسة من بين مجموعات بيانات الدراسة المختلفة التي تحتوي على أشكال معقدة مختلفة الاحجام و الكثافة و تحتوي على ضوضاء و قيم متطرفة و ذلك بغض النظر عن الظروف الاولية السيئة المستخدمة في المرحلة الأولى . و كما و إن هذه التجارب أثبتت تفوق خوارزميتنا على أهم الخوارزميات المنافسة.

TO WHOM IT MAY CONCERN

ACKNOWLEDGMENTS

LIST OF FIGURES

LIST OF TABLES

Chapter *1*

INTRODUCTION

A lot of data can be gathered from different fields but this data is useless without proper analysis to obtain useful information. In this thesis, we focus on one of the important techniques in data mining: Clustering.

1.1   Data Clustering

Data clustering is a method of grouping similar objects together. Thus the similar objects are clustered in the same group and dissimilar objects are clustered in different ones. An illustration example of clustering is shown in Fig 1.1. Data clustering is considered as an unsupervised learning technique in which objects are grouped in unknown predefined clusters. On the contrary, classification is a supervised learning in which objects are assigned to predefined classes (clusters).
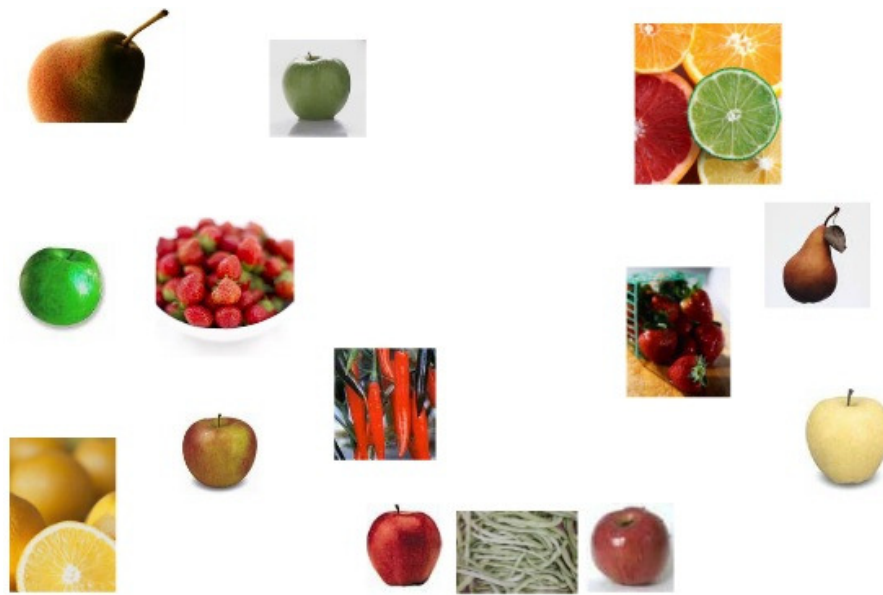
(a)
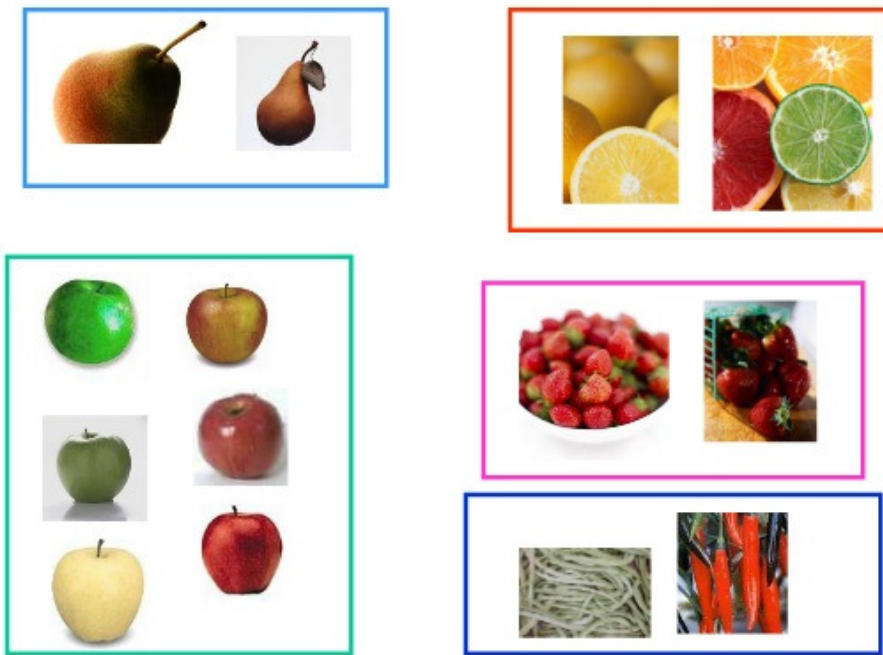
(b)

Figure 1.1: Clustering example (a) dataset contains 14 objects. (b) Objects are grouped into 5 clusters. (adapted from: *http://www.slideshare.net/pierluca.lanzi/ machine-learning-and-data-mining-08-clustering-hierarchical* last visit: July, 2011

1.1.1 Basic Concepts of Clustering

The problem of data clustering can be formulated as follows: given a dataset D that contains n objects $x_1, x_2, \ldots, x_n$ (data points, records, instances, patterns, observations, items) and each data point is in a d-dimensional space, i.e. each

data point has d dimensions (attributes, features, variables, components). This can be expressed in a matrix format as:

$$D = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1d} \\ x_{21} & x_{22} & \cdots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nd} \end{bmatrix} \qquad (1.1)$$

Data clustering is based on the similarity or dissimilarity (distance) measures between data points. Hence, these measures make the cluster analysis meaningful [28]. The high quality of clustering is to obtain high intra-cluster similarity and low inter-cluster similarity as shown in Fig. 1.2. In addition, when we use the dissimilarity (distance) concept, the latter sentence becomes: the high quality of clustering is to obtain low intra-cluster dissimilarity and high inter-cluster dissimilarity.
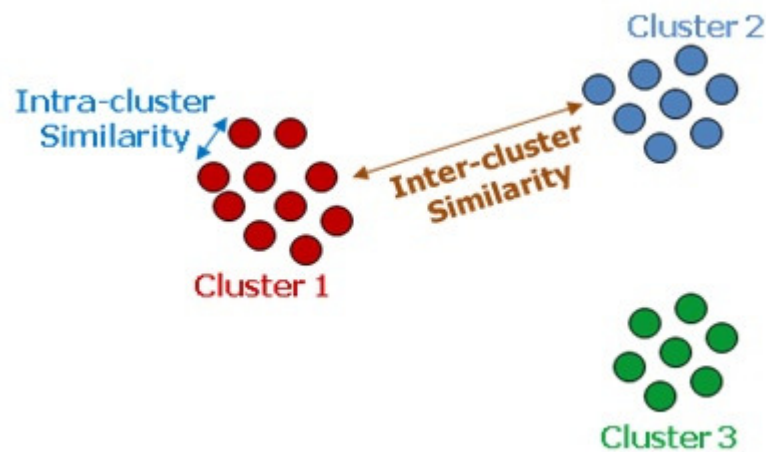
Figure 1.2: Inter-cluster and Intra-cluster similarities of clusters.

1.2   Importance of Clustering

Data clustering is one of the main tasks of data mining [1] and pattern recognition [2]. Moreover, it can be used in many applications such as:

1. Data compression [3].
2. Image analysis [5].
3. Bioinformatics [6].
4. Academics [9].
5. Search engines [79].
6. Wireless sensor networks [80].

3

7. Intrusion detection [81].

8. Business planning [82].

1.3 Motivations

The Kmeans algorithm is considered as one of the top ten algorithms in data mining [35]. A lot of researches and studies have been proposed due to its simplicity and efficiency [55]. These efforts have focused on finding possible solutions to one or more of the limitations that have been identified in page 10. Kmeans with random initialization conditions need to be rerun many times each with different conditions to find more suitable results [21]. Many algorithms have been considered to provide better seeds so the Kmeans algorithm is likely to converge to the global optimum like Minmax[43], Kmeans++ [44] and [45]. Other solutions to the initial prototypes sensitivity can be found in [46] where they defined new criterion functions for Kmeans and they proposed three algorithms: weighted Kmeans, inverse weighted Kmeans [52] and inverse exponential Kmeans [53]. Other improvements of Kmeans focus on its efficiency where the complexity of Kmeans involves the data set size, number of dimensions, number of clusters and the number of iteration to be converged. There are many works to reduce the computational load and make it faster such as [4], [47-49]. Asgharbeygi and Maleki [39] proposed a new distance metric which is the geodesic distance to ensure resistance to outliers. Several works have been introduced to extend the use of means for numerical variables, thus Kmeans can deal with categorical variables such as [50], [51].

JJ Sheu et. al. [61] proposed a new algorithm and they named it Intelligent Kmeans (IKM) for deciding the proper number of clusters, choosing a better initial prototypes and reducing the effect of outliers on the clustering result. IKM divided the range of data points for each d dimensions into M regions where M is a constant input number. One drawbacks of this method, is the choice of grid size. If it is small, it will produce a large number of clusters and vice versa.

4

Many researchers have been involved in developing solutions to the Kmeans and other clustering algorithms such as using neighborhood model [23], ant colony [24], the principle of gravity [26], genetic algorithms [25], and clustering method with constraints [27].

The problem in clustering is that we do not have prior information knowledge about the given dataset. Moreover, the choice of input parameters such as the number of clusters, number of nearest neighbors and other factors in these algorithms make the clustering more challengeable topic. Thus any incorrect choice of these parameters yields bad clustering results. Furthermore, these algorithms suffer from unsatisfactory accuracy when the dataset contains clusters with different complex shapes, densities, sizes, noise and outliers.

In this thesis we want to design a novel clustering algorithm that is able to discover clusters with arbitrary complex shapes with presence of noise and outliers without requiring a previous knowledge of the given domain. In our approach we use the concept of convex hull [62] in which it is widely used in image processing to represent the shapes. Furthermore, it has been recently used in classification methods such as [64], [65]. Moreover, in [66] they provided a method of representing on-line data streaming using a cost function based on convex hull. In which they are concerned in representing the shape of data stream as a collection of convex hulls. However, this method cannot recover clusters correctly if the values of its input parameters are not set correctly.

1.4 Thesis outlines

In Chapter 2, we introduce a general overview of data clustering categorizations and algorithms. The ways of how we can construct a graph to be used in the clustering algorithms have been mentioned too. Moreover, we also explain and summarize some of the related works.

Our proposed algorithm is presented in details in Chapter 3. In which the three phases of the proposed algorithm are explained. A simple toy example is considered to be solved by our proposed algorithm. The simulation and results

analysis based on our generated complex shaped datasets have been accomplished in the last section of Chapter 3. Chapter 4 uses a topology preserving mapping as a preprocessing to our approach and we have used 10 real datasets from UCI machine repository to show the effectiveness of our proposed algorithm. Chapter 5 gives the conclusions and future research. We also provide two appendices that illustrate how the data preparation is done and the used distance metrics in the clustering algorithms.

Chapter *2*

RELATED LITERATURE REVIEWS

2.1   Overview

The clustering problems can be categorized into two main types: fuzzy clustering and hard clustering. In fuzzy clustering, data points can belong to more than one cluster with probabilities between 0 and 1 [10], [11] which indicate the strength of the relationships between the data points and a particular cluster. One of the most popular fuzzy clustering algorithms is fuzzy c-mean algorithm [12], [13], [14]. In hard clustering, data points are divided into distinct clusters, where each data point can belong to one and only one cluster.

The hard clustering is divided into hierarchical and partitional algorithms. Hierarchical algorithms create nested relationships of clusters which can be represented as a tree structure called dendrogram [28]. Hierarchical algorithms can be divided into agglomerative and divisive hierarchical algorithms. The agglomerative hierarchical clustering starts with each data point in a single cluster. Then it repeats merging the similar pairs of clusters until all of the data points are in one cluster, such as complete linkage clustering [29] and single

linkage clustering [30]. CURE [15], ROCK [16], BIRCH [17] and Chameleon [18] are examples of this hierarchical algorithm. The divisive hierarchical algorithm reverses the operations of agglomerative clustering, it starts with all data points in one cluster and it repeats splitting large clusters into smaller ones until each data point belongs to a single cluster such as DIANA clustering algorithm [31].

In the contrary, Partitional clustering algorithm divides the dataset into a set of disjoint clusters such as Kmeans [32], [42] PAM [31] and CLARA [31]. Moreover, the partitional algorithms have been considered more appropriate for applications with large dataset, in which the construction of the dendrogram is computationally expensive [1], [37]. One of the problems in applying partitional methods is the choice of the number of clusters within the given datasets where the determination of the number of clusters is one of the most problematic issues in data clustering [7]. The partitional algorithms often use a certain objective function and produce the desired clusters by optimizing this objective function [36].

The clustering algorithms that are based on estimating the densities of data points are known as density-based methods. One of the basic density based clustering algorithm is DBSCAN [40]. It defines the density by counting the number of data points in a region specified by a predefined radius known as epsilon ε around the data point. If a data point has a number greater than or equal to predefined minimum points known as *MinPts*, then this point is treated as a core point. Non-core data points that do not have a core data point within the predefined radius are treated as noise. Then the clusters are formed around the core data points and are defined as a set of density-connected data points that is maximal with respect to density reachability. DBSCAN may behave poorly due its weak definition of data points' densities and its globally predefined parameters of ε and *MinPts*. There are many works that try to improve the well known DBSCAN such as [41], [56-60].

2.2 Similarity Graphs

Another type of clustering algorithms is based on the construction of similarity graphs in which a given set of data points is transformed into vertices and edges. The constructed graph can be used to obtain a single highly connected graph that is then partitioned by edge cutting to obtain sub graphs [72], [74], [68]. Basically, the kinds of graphs are ε-neighborhood, k-nearest neighbor and fully connected graph [2], [70], [54].

The ε-neighborhood graph connects all data points whose pairwise distances are smaller than a predefined threshold ε.

In the k-nearest neighbor graph the data point v_i (vertex) is connected with another data point in the dataset if it is in the k-nearest neighbors of v_i where k is a predefined parameter. This method lets the k-nearest neighbor produces a directed graph. The undirected graph can be obtained from the k-nearest neighbor by simply ignoring the directions of edges or by having a mutual k-nearest neighbor graph in which two vertices are connected by an edge if and only if these two vertices are among the k-nearest neighbors of each other.

The fully connected graph connects all data points that have a positive similarity measurement with each other. The similarity measure can be produced by using the Gaussian similarity function $S_{ij}=exp(-d_{ij}^2/2\sigma^2)$ where d_{ij} is the Euclidean distance between two data points x_i and x_j and the parameter σ is also a user defined one that controls the width of neighborhoods.

2.3 Kmeans Algorithm

One of the most well-known unsupervised learning algorithms for clustering datasets is Kmeans algorithm [31], [37]. The Kmeans clustering is the most widely used due to its simplicity and efficiency in various fields [33], [38]. It is also considered as the top ten algorithms in data mining [35]. The Kmeans algorithm works as follows:

1. Select a set of initial k prototypes or means throughout a dataset, where k is a user-defined parameter that represents the number of clusters in the dataset.

2. Assign each data point in a dataset to its nearest prototype **m**.

3. Update each prototype according to the average of data points assigned to it.

4. Repeat step 2 and 3 until convergence.

The Kmeans algorithm depends on minimizing the sum of squared error function which is very simple and can be easily implemented.

$$J = \frac{1}{n}\sum_{i=1}^{k}\sum_{x \in C_i}\|x - m_i\|^2 \tag{2.1}$$

Where dataset D contains n data points $\boldsymbol{x_1, x_2, \ldots, x_n}$ such that each data point is d dimensional vector in R^d, and $\boldsymbol{m_i}$ is the prototype of cluster C_i, and k is the given number of clusters.

However, it has several drawbacks: the number of clusters k in a given dataset should be known in advance, the result strongly depends on the initial prototypes, the sensitivity to noise and outliers, the problem of dead prototypes or empty clusters and the converge to local optima [34]. The Kmeans works for globular shaped, similar size and density clusters.

2.4 CURE and Chameleon Algorithms

CURE [15] uses a constant number of well scattered representative data points from all data points in the dataset to represent a cluster instead of selecting one single centroid to represent a cluster in Kmeans. These are shrunk towards the centroid of the cluster according to a user predefined shrinking factor. Then a consecutive merging of the closest pair of the cluster's representative points are occurred until the predefined number of clusters is obtained. The selection of the shrinking factor and the merging process make CURE ineffective with complex datasets and they can cause false outliers [22].

Chameleon [18] uses a graph construction based on k-nearest neighbors, and then it splits the graph into a set of small clusters using hMetis algorithm [19]. After that it merges these small clusters based on their similarity measure. It has been used to find non-convex shaped clusters, however, it cannot handle

noise and outliers and needs to set parameters correctly in order to obtain good results [22], [20].

2.5   Affinity Propagation Algorithm

Another type of clustering algorithms is called Affinity Propagation [67] that passes messages between data points to identify a set of exemplars (cluster centers) and their corresponding clusters. In contrary of selecting an initial set of cluster centers randomly and iteratively refines them such that the sum of squared error is minimized as in Kmeans; the Affinity Propagation provides a different approach that simultaneously considers all data points as candidate exemplars. Then two types of messages are exchanged between data points. The Responsibility messages are sent from data points to candidate exemplars and indicate how strongly each data point is biased to the candidate exemplar over other candidate exemplars. The Availability messages are sent from candidate exemplars to data points and reflect evidence that each candidate exemplar is available to be a cluster center of the data points. The Affinity Propagation uses the median of similarities between data points as preferences rather than the predetermined number of clusters.

2.6   Spectral Clustering Algorithm

Recently, the spectral clustering [70] has become one of the most popular clustering algorithms which outperform the traditional algorithms such as Kmeans. Furthermore, they are designed to handle non-convex shaped clusters. However, spectral clustering suffers from heavily computations. The similarity measure and graph cutting are also used in spectral clustering algorithms. The core of the spectral clustering algorithms is to use the properties of eigenvectors of Laplacian matrix for performing graph partitioning [69-76].

The Laplacian matrix is constructed by building an affinity graph matrix with a similarity measure. The common similarity measure is to use the Gaussian function S_{ij} as stated previously for its simplicity. Hence, the Laplacian matrix L is calculated as $L=D-S$ where D is the diagonal matrix

whose elements are the sum of all row elements of *S*. Then, the spectral clustering computes a column matrix of the first k eigenvectors of *L* where k is a predefined number of clusters. Thus it finds the clusters of mapped data points that corresponding to the column matrix of eigenvectors by performing Kmeans algorithm.

2.6.1 Spectral Clustering using Nystrom Method

W.-Y. Chen et. al. [76] proposed sparsification and Nystrom approaches to address the computational difficulties and to improve the results. We compare our algorithm with spectral clustering using Nystrom method because it needs less computation and does not need the prespecified number of nearest neighbors as in sparsification method. Nystrom method is a technique for finding an approximate eigendecomposition. The spectral clustering using Nystrom method uses randomly sample data points from the dataset to approximate the similarity matrix of all data points in the dataset. Then it finds the first k eigenvectors of the normalized Laplacian matrix of the Nystrom method and performs Kmeans to cluster dataset.

2.7   Topology Preserving Mapping

A topographic mapping is a transformation of high dimensional data. Furthermore, it preserves some structure in the data such as the points which are mapped close to each other share some common properties while in contrast the points which are mapped far from each other do not share a common feature or property.

The Self-organizing map (SOM) [84] and the Generative topographic mapping (GTM) [85] have been considered as very popular topology preserving mapping techniques for data visualization and dimensionality reduction. The GTM can be considered as a statistical alternative to the SOM overcoming many of its limitations such as the absence of a cost function and the lack of proof convergence [86].

2.7.1 Self-organizing Map (SOM)

The Self-organizing Map (SOM) [84] is a type of artificial neural network that is trained using unsupervised learning. SOM reduces dimensions of the given datasets by producing a map of usually one or two dimensions. Furthermore, SOM uses a neighborhood function to preserve the topological properties of the input space.

The SOM consists of components called nodes or neurons in which they are usually arranged in a hexagonal or rectangular grid. It first initializes the weights associated with each neuron by assigning them small random values. Then the SOM proceeds to three essential processes: competition, cooperation, and adaptation [28].

2.7.2 Generative Topographic Mapping (GTM)

The GTM is a statistical model for modeling the probability density of data points and finding non-linear mapping of high dimensional space onto low dimensional space.
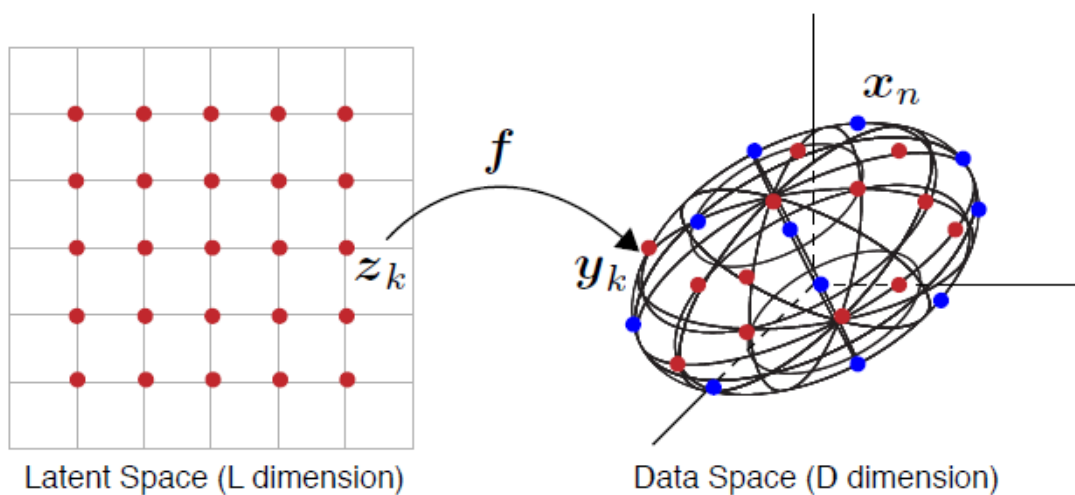
Figure 2.1: Non-linear mapping by GTM, adapted from [87].

As shown in Fig. 2.1, the basis of the GTM is to generate a grid of K latent points $z_1, z_2, ..., z_K$ in latent space. These latent points are mapped non-linearly into the data space, which contains N data points x_n ($n=1, 2, ..., N$), using a set of M fixed basis Gaussian functions, such that,

$$y_k = \Phi_k^{Tr} W \tag{2.2}$$

Where y_k denotes the mapped points in data space. The element Φ consists of M fixed basis functions. W is M x D matrix containing weight parameters, and D is the dimensionality of data space. A^{Tr} is a transpose of a matrix A.

The probability density between the mapped points y_k and data points x_n is estimated by using a Gaussian noise distribution centered on y_k with the noise inverse variance β. This probability density $p(x_n|y_k,W,\beta)$ is defined as :

$$p(x_n|y_k, W, \beta) = \left(\frac{\beta}{2\pi}\right)^{D/2} \exp\left(-\frac{\beta}{2}\|x_n - y_k\|^2\right) \tag{2.3}$$

The training step of GTM is done by optimizing its parameters using the Expectation-Maximization (EM) algorithm [2], [28] which maximizes the following log-likelihood:

$$\mathcal{L}(W, \beta) = \underset{W,\beta}{\operatorname{argmax}} \sum_{n=1}^{N} \ln\left(\frac{1}{K}\sum_{k=1}^{K} p(x_n|y_k, W, \beta)\right) \tag{2.4}$$

After convergence, we can visualize the data by projecting each data point x_i onto the latent space using one of the two ways:

- The Mode: the mode of posterior distribution:

$$y_{k\,mode} = \underset{y_k}{\operatorname{argmax}} \, p(y_k|x_i) \tag{2.5}$$

- The Mean: the mean of posterior distribution:

$$y_{k\,mean} = \sum_{k=1}^{K} y_k \, p(y_k|x_i) \tag{2.6}$$

Where $p(y_k|x)$ is the corresponding posterior distribution in the latent space for any given data point x in the data space and is defined as:

$$p(y_k|x) = \frac{p(x|y_k, W, \beta)p(y_k)}{\sum_{k'=1}^{K} p(x|y_{k'}, W, \beta)p(y_{k'})} \tag{2.7}$$

Chapter **3**

KBCHT: KMEANS-BASED CONVEX HULL TRIANGULATION CLUSTERING ALGORITHM

The problem of clustering datasets is that we have no prior knowledge information about them. Thus the majority of existing clustering algorithms try to solve it by introducing external input parameters which make these works sensitive to their inputs. In this chapter we introduce Kmeans-Based Convex Hull Triangulation clustering algorithm (KBCHT) a new clustering algorithm that studies the given dataset to find the clusters. KBCHT algorithm is able to detect clusters without pre-determination of clusters number in datasets which contain complex non-convex shapes, different sizes, densities, noise and outliers. Algorithm 3.1 provides a pseudo-code that describes the overall procedures of KBCHT algorithm.

3.1 Proposed KBCHT Algorithm

KBCHT has three phases of operations. The first phase obtains initial groups from running Kmeans algorithm just once, the second phase analyzes these initial groups to get sub-clusters and the last one merges the sub-clusters to find the final clusters in the dataset. As shown in Algorithm 3.1, KBCHT performs Kmeans algorithm on the dataset x given the number of clusters k. The use of k is just to run Kmeans as we will notice by further study of the effect of k in

Section 3.3.2. Line 2 means that the first run of Kmeans algorithm despite its bad initialization conditions has an initial set of clusters iC_i where i is from 1 to N the number of obtained clusters from Kmeans. The set iC with index i contains data points from dataset x which belong to the initial cluster i. Lines 3 to 7 describe the process of how we analyze these initial clusters to obtain a set of sub-clusters. In line 4, we construct a set of vertices which represents each initial clusters iC. This set of vertices is obtained from the convex hull of each initial clusters iC. The set iV handles these vertices which contains two indexes i and j as shown in line 4. In which the index i indicates that these vertices belong to the initial cluster i and the index j represents the vertex number of convex hull of initial cluster i in a counterclockwise order. After obtaining the vertices from the convex hull, these vertices need to be shrunk by adding new vertices from the belonged initial clusters set iC. Thus we begin with vertices drawn from a convex hull and finish with vertices of a polygon. The shrunk vertices are handled in the set sV as shown in line 5 of Algorithm 3.1. Line 6 takes the shrunk vertices sV and processes them to obtain a set of sub-clusters sC, the number of these sub-clusters S and the average distance between data points of each of the sub-clusters in sC ($sCaD$) using the delaunay triangulation [63] as will be explained later . The sub-clusters are formed by searching for closed loops vertices in the sV set. The set sC has indexed from 1 to S in which sC_i contains data points of dataset x that belong to sub-cluster i. Some of these sub-clusters could be merged together to form the final result of clusters C as shown in line 8.

Algorithm 3.1: KBCHT
1   $Begin\ initialize\ k, x, iC = \{\}, sC = \{\}, C = \{\}, iV = \{\}, sV = \{\}, sCaD = \{\}, N = 0, S = 0$
2       $iC, N \leftarrow Kmeans(x, k)$
3       $for\ i = 1\ to\ N$
4           $iV(i, j) \leftarrow Construct\ convexhull\ for\ cluster\ iC_i$
5           $sV(i, k) \leftarrow shrinkVertex(iC_i, iV(i, :))$
6           $sC, S, sCaD \leftarrow findSubClusters(iC_i, sV(i, :))$
7   end_for
8       $C \leftarrow merging(sC, S, sCaD)$
9       $Return\ C$
10 end

3.1.1 The First Phase: The use of standard Kmeans

KBCHT algorithm chooses to use the well known Kmeans algorithm as it is first step because of its simplicity and widely use as one of the top ten data mining algorithms [35].

The aim of our algorithm is to find clusters of arbitrary shapes and to detect odd patterns that exist in the datasets which Kmeans is far away from detecting them; Kmeans depends on assigning data points to their nearest mean thus the final result of it comes out as spherical shapes. Thus we can benefit from its first run with randomly thrown prototypes throughout the given dataset to catch preliminary initial clusters that insures closely grouping sets. However, Kmeans algorithm needs to be injected with k the number of clusters in the dataset. Further investigation on the effect of k has been conducted in Section 3.3.2. The k or less than k, in case of dead prototypes, resultant partitions from Kmeans could be processed and analyzed in a parallel fashion which speeds up the processing time.

In this phase we are concerned with catching initial relatively related groups and Kmeans algorithm gives relatively robust and good enough answers over a wide variety of datasets as mentioned in [46]. Hence, we have decided to work on the standard Kmeans algorithm.

Generally speaking, we can use any other method in which they offer grouping such as any of the partitional clustering algorithms that are mentioned in Chapter 1. Moreover, we can benefit from the researches that are focused on the construction of similarity graphs of the given datasets as explained in Chapter 2.

Besides that, there are many developed researches related to overcome the limitations of the standard Kmeans algorithms. Hence, the choice of one of these developed researches to be as our first phase depends on what we want and what we have. i.e, we want more accurate initial result despite the time it could take or vice versa.

Some of these researches have aimed at identifying the initial centroids locations. In [91] they avoided the initial random assignment of centroids using

sub-merger strategy and [46] focused on the sensitivity to initial centroids condition. Another research has focused on the number k of clusters such as in [92] where they proposed a measure to select the number of clusters. Others have tried to accelerate the Kmeans algorithm by avoiding many calculations of distance based on partial distance strategy like in [4].

3.1.2 The Second Phase: The Shrinking

After catching the initial groups, we want the greatest benefit from them. How we can analyze and represent them? We can go back to the topics of grid clustering in the book [28] and the research as in [62] in which they divided the group of data points into equal grids then trying to eliminate the grids that do not contain sufficient number of points. But by using this, we have stuck under the mercy of the user defined parameters. Hence, we have decided to use widely used concept of representing shapes as in image processing which is the convex hull mechanism.

As shown in lines 3 to 7 of Algorithm 3.1, this phase of KBCHT algorithm operates on the set of initial clusters that are obtained from the first phase. Each group of initial clusters is represented by its surrounding vertices on convex hull and the data points inside this convex hull. Then these vertices of each group are shrunk separately until we find the final sub-clusters. Procedure 3.1 in the next page describes the shrinking process in details.

Suppose we have one of the initial clusters obtained from the first phase of KBCHT algorithm as shown in Fig. 3.1 (page 18). The blue '*'s in Fig. 3.1 represent the data points of the given dataset which can belong to one or different final clusters of the dataset. The solid lines that are surrounding the data points in Fig. 3.1 represent the convex hull of this portion of dataset. The vertices from V_1, V_2 to V_{12} are the boundary data points drawn from the convex hull in which V_1 equals to the last vertex V_{12} and these vertices are in a counterclockwise order. As in Fig.3.1 the blue '*'s are x data points in Procedure 3.1 and the set Vs is the vertices of convex hull. In case of Fig. 3.1, Vs is from V_1 to V_{12}.

Procedure 3.1: Shrink Vertices

1	*Input*:
2	Vs: *Vertices of convex hull*
3	x: *data points inside convex hull of* Vs
4	*Begin*
5	$MAX \leftarrow$ *find maximun length of edges between* Vs
6	$AVG \leftarrow$ *find average distance among objects* x *inclosed by* Vs
7	*while* $MAX < AVG$
8	$nVs \leftarrow$ *exclude* Vs *from* x *and reconstruct convex hull*
9	$MAX \leftarrow$ *find maximun length of edges between* nVs
10	$AVG \leftarrow$ *find average distance among objects* x *inclosed by* nVs
11	$Vs \leftarrow nVs$
12	*end_while*
13	$V \leftarrow$ *Reorder* Vs *starting from vertex belonges to the longest edge* *(reserve counterclockwise order)*
14	*while* $MAX \geq AVG$ *OR Converged*
15	$P \leftarrow$ *find closest point from* x *to the line between* V_1 *and* V_2 *and its projection falls between* V_1 *and* V_2 *and resides on the left of* V_1 *and* V_2 *and the perpendicular line from* P *to the line between* V_1 *and* V_2 *does have no intersection with other edges between vertices*
16	*if such a point* P *exists*
17	*add* P *to* V *between* V_1 *and* V_2
18	*else*
19	*flag* V_1 *and* V_2 *as processed vertices*
20	*end_if*
21	$MAX \leftarrow$ *find length of longest edge and its vertices are not processed*
22	*Reorder* V *starting from vertex belonges to the longest edge and its vertices are not processed (reserve counterclockwise order)*
23	*end_while*
24	*Return* V
25	*end*

Line 5 in Procedure 3.1 computes the maximum length of edges between each two consecutive vertices and stores it in *MAX* variable. KBCHT algorithm does not use external parameter to guide the shrinking processing. It self studies the given data points to decide when and how to make the shrink. This maximum edge has the highest priority to be shrunk.

We can make the shrinking based on different criteria such as defining an external parameter to be a threshold, instead of calculating our average, like in many clustering algorithms that use threshold parameters. Thus by starting shrinking the maximum edge length until the maximum one becomes less than
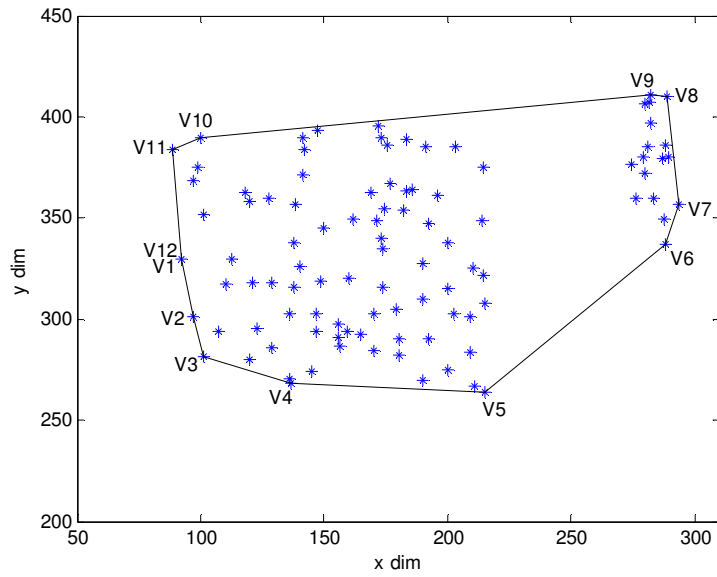
Figure 3.1: Blue '*'s represent a partition from dataset enclosed by its convex hull and V's represent Vertices on convex hull and lines between vertices represent edges.

the defined threshold. But as stated before, KBCHT algorithm does not like to be with the clemency of external factors. Hence, we decide to compute the average distance among data points that enclosed by the set of *Vs*. The vertices *Vs* are excluded from computing this average to eliminate the effect of being outliers. To compute the average distance among data points, we do not want to consider the distances between each data point and every other data point in the set; This will be computational expensive and does not reflect the actual data structure of the given set of data points. Thus we construct the Delaunay triangulation of the data points. Then the average distance *AVG* is the average length of the triangles edges.

Now we have two variables *MAX* and *AVG*. However, for starting the shrinking process the maximum edge length of convex hull *MAX* should be greater than the average distance *AVG*. If it is not, this means that the vertices of the convex hull are denser than the enclosed set of data points as shown in Fig. 3.2. In this case, we identify a new set of vertices by reconstructing the convex hull of the data points again without considering the previously obtained vertices. These are shown in lines 7 to 12 of Procedure 3.1.

20
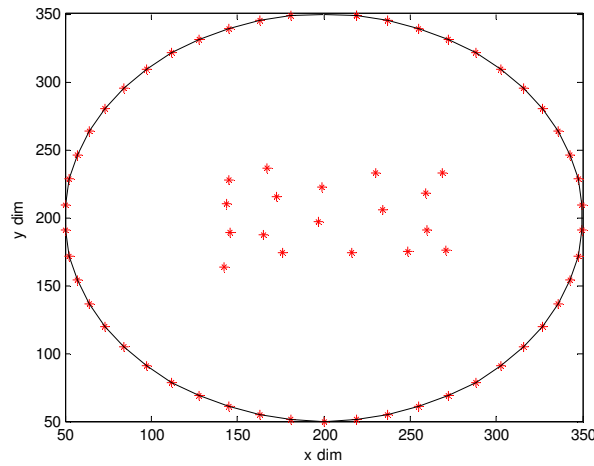
(a)
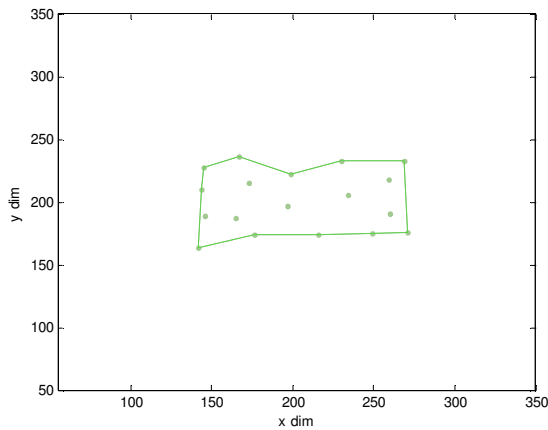
(b)
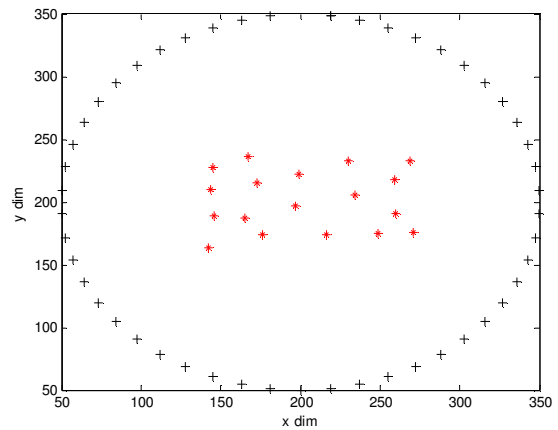
(C)

Figure 3.2: (a) Example of a set of data points, red '*'s, that vertices of the convex hull, black solid lines, are far away from a sparse cluster. (b) Shrinking result after eliminating the vertices of convex hull. (c) The final result red '*'s are in one cluster and black '+'s are in a separated cluster.

After satisfying the above condition for starting the process of shrinking, we reorder the vertices to begin from the vertex that belongs to the longest edge length (line 13 of Procedure 3.1). But we have to reserve the order of vertices in a counterclockwise order. Back to Fig. 3.1, the longest edge length is the edge between two vertices V_9 and V_{10}. So we reorder the vertices such that V_1 becomes V_9 and V_2 becomes V_{10} and so on.

How to do the shrinking: At this stage, we have a set of vertices begins from the vertex with the longest edge. We want to find another vertex from the data points to be engaged between the first two vertices. To find this new vertex, we have to find the closest data point to the line between the first two

vertices of the set *V* and its projection lies on the line between these two vertices. Consider Fig. 3.3 as an illustrated example for finding the closest point to the line. In Fig. 3.3 we want to find the closest point to the line AB. To do this we have to find the projection points (**D1** and **E1**) on the line AB from the tested points. Let **A**=(A_x,A_y), **B**=(B_x,B_y) and **D**=(D_x,D_y). Our goal is to find the point **D1**=($D1_x$,$D1_y$). Hence **D1** lies on the line AB, thus it satisfies its line equation and it can be found from: **D1**=**A**+u(**B**-**A**). To find u, the dot product of the perpendicular line from **D** to line AB and line AB is 0. Thus u can be found from the following equation [89]:

$$u = \frac{(D1_x - A_x)(B_x - A_x) + (D1_y - A_y)(B_y - A_y)}{\left((B_x - A_x)^2 + (B_y - A_y)^2\right)} \tag{3.1}$$

To guarantee that **D1** lies between **A** and **B**; the value of u should be between [0, 1]. So, the points **C** and **F** do not be considered. Then the distance from the point to the line is the distance from that point to its projection point on the line.

We reserve the order of the vertices to be counterclockwise. Thus the picked point that has to be a new vertex should also reside on the left side of the two vertices in which the new vertex has to be engaged between them. To ensure this, Back to Fig. 3.3 suppose we have two vertices A and B and we want to examine that the data point D resides on the left side of the direction from A to B. we compute the value of the following equation [90]:
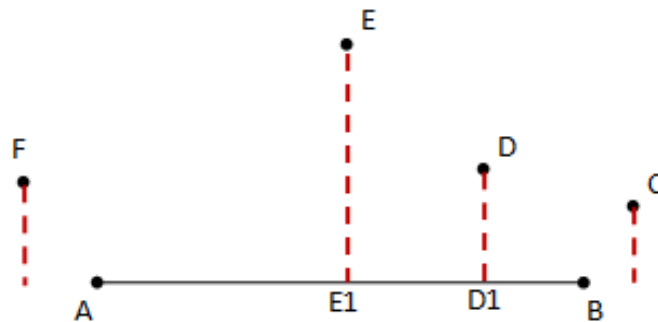
Figure 3.3: Solid line between points A and B. Dash lines are the projection of points to the line AB.

$$value = \begin{vmatrix} A_x & B_x & D_x \\ A_y & B_y & D_y \\ 1 & 1 & 1 \end{vmatrix} \qquad (3.2)$$

Where $|.|$ is the matrix determinant. If the sign of the value of equation (3.2) is positive then the examined point is on the left side. As well as it is on the right side if the sign is negative and on the straight line if it has a zero value. In fact, the equation (3.2) reflects the area of the triangle of ABD.

However, we have started the shrinking with the vertices that lay on the convex hull. Our approach in shrinking does not conserve the convexity of the shrunk vertices shape. Thus we have to provide an additional condition for shrinking which is shown in line 15 of Procedure 3.1. This condition says that the perpendicular line from the candidate vertex to the line between the two vertices in which their edge has to be shrunk should not intersect any of the lines that are between two consecutive vertices. This condition has exceptions. To be more obvious and as stated previously, the candidate vertex should be between V_1 and V_2 vertices thus if the candidate vertex is already an existing vertex, we violate the previous condition if this candidate vertex is V_3 or V_{last-1} (vertex that resides before the last one directly). Also, it should be violated if there is only one intersection and this intersection point is equal to the vertex V_1 or V_2. While the process of shrinking is going on, some vertices have to be released from the vertices set if a specific vertex has the same previous and next vertex. Hence, this vertex has to be released if its distance to next vertex is greater than the *AVG* value. The released vertices should be processed in the last phase of KBCHT algorithm. The sharp eyed readers may notice that the above condition of releasing vertices may be violated even though they should be released. This situation happens when we want to release a vertex that resides far away from a group of dense data points but its distance to next vertex is less than the overall average distance *AVG*. In this case, we add one more examine condition in which we compute the length of the two adjacent edges of this vertex. If both of them are greater than the overall average distance *AVG* and they have no close candidate vertices to be added in the set of *V*. Thus we guarantee to release this vertex from the set *V*.

23

Line 17 of procedure 3.1 says: "All of the above mentioned conditions have been satisfied". Thus we add the candidate vertex to be one of the vertices in the set V between vertex V_1 and V_2. If no such a candidate vertex in which it violates the above conditions, the two vertices V_1 and V_2 have been flagged to be processed. A new value of variable MAX should be recalculated as in line 21 of Procedure 3.1. But at this time the vertices that have a maximum edge length should not be flagged as processed. If so, find the next longest edge length until their vertices are not flagged. Then the set V is reordered again starting from the vertex that belongs to the longest edge length and their vertices are not flagged as processed. We repeat this process of shrinking again and again until the MAX value becomes less than the average distance AVG or all vertices in the set V have been flagged as processed.

What to do with the shrunk vertices: the last operation of the second phase of KBCHT algorithm is to refine the shrunk vertices and to extract useful information from the set of these vertices by finding the groups of sub-clusters entire the given dataset. Procedure 3.2 says how to find sub-clusters. After shrinking the vertices that obtained from the first phase of KBCHT algorithm in which they reside on the boundary of the convex hull of each of the resultant initial groups from the first phase. At this stage, the shrunk vertices are on the boundary of initial group of data points where this boundary forms a general polygon shape.

KBCHT algorithm calls the Procedure 3.2 in its line 6 as shown in Algorithm 3.1. Procedure 3.2 takes two inputs: V which is the resultant shrunk vertices and x which is the set of data points that enclosed by V. This x is a part of the overall given dataset. We want to distinguish these vertices in the set V thus each group of connected vertices belongs to a separate sub-cluster. We define the set L that has a number of entries equal to the number of the given vertices in V. Initially, all the vertices are assigned a label value of 0 such that $L_i=0 \forall i \in \{1,...,len\}$ where len is the number of shrunk vertices. For example if L_5 has a label value equals to 1, then this means that the V_5 of the set V is

Procedure 3.2: Find sub clusters

```
1    Input:
2         V: Shrunk Vertices of border polygon
3         x: data points enclosed by V
4    Begin
5        len ← number of vertices in V
6        for i = 1 to len
7            L_i ← 0
8        end_for
9        count ← 1
10       for i = 1 to len − 1
11           if L_i == 0
12               for j = i + 1 to len
13                   diff ← distance between V_i and V_j
14                   if diff == 0
15                       for k = i to j
16                           L_k ← count
17                       end_for
18                       count ← count + 1
19                   end_if
20               end_for
21           end_if
22       end_for
23       for i = 1 to count − 1
24           sC_i ← find objects of x inside V with label L_i
25       end_for
26       for i = 1 to count − 1
27           sCaD_i ← calculate the average distance within the sub cluster sC_i
28       end_for
29       Return sC, count − 1 and sCaD_i
30   end
```

assigned to the sub-cluster of label 1. Thus *L* handles the labels of sub-clusters that each of vertices in *V* belongs to.

In line 9 of Procedure 3.2 we define a variable *count* which it is initially set to be 1 and is incremented automatically by one when the algorithm detects a new sub-cluster. So, *count* represents the labels of sub-clusters that the vertices belong to. Procedure 3.2 enters a loop that begins from line 10 and ends up at line 22. These lines are the core of this Procedure in which we define two pointers *i* and *j* such that we fix *i* to point to a specific vertex in the set *V* and then the pointer *j* investigates each of the following vertices after that vertex which is pointed by *i*. When the pointer *j* reaches the last vertex in the set *V*, we move the pointer *i* one step forward to point to the next vertex and the pointer *j*

restarts and repeats its mission. This task will be repeated until i reaches the *len-1* vertex which is one position before the last vertex. Line 11 tests if the corresponded vertex has never assigned to sub-cluster yet. If so, the procedure enters into the loop of the pointer j. In line 13 of Procedure 3.2 we check if the vertex pointed by i (V_i) and the vertex pointed by j (V_j) are the same. If they are identical vertices, then we obtained a closed connection of vertices which also means that we catch one of the sub-clusters in the dataset. Line 16 gives the sub-cluster's label to the set L starting from its index of i through j. After that we increment the label identified by *count* by 1. This process will be repeated until we catch all of the sub-clusters in the given dataset. Before line 23 of Procedure 3.2, we have groups of vertices each belongs to different sub-clusters; these groups represent the boundary of the data points of each sub-cluster. Thus the lines from 23 to 25 find the data points themselves in which they are enclosed by each of the obtained closed connection of vertices. Hence, the set sC in Procedure 3.2 handles the data points of each sub-cluster in the dataset and it has an index starts from 1 to the number of obtained sub-clusters where each index handles the tightly data points that belong to a specific sub-cluster. Finally, for each of the obtained sub-clusters we compute the average distance between data points within each of them as explained in Section 3.1.2 and store these average distances in the group of *sCaD* as shown in lines 26 to 28.

3.1.3 The Third Phase: Merging sub-clusters

Procedure 3.3 shows the steps of the last phase of our KBCHT algorithm which is merging the obtained sub-clusters for finding the final set of clusters in the studied dataset. The set sC is defined to be a group of sub-clusters of the dataset. In addition, S is the number of these sub-clusters. Hence, $sC=\{sC_1,sC_2,…,sC_S\}$ in which each of the sub-cluster sC_i contains the data points that belong to this sub-cluster. The $sCaD=\{sCaD_1,sCaD_2,…,sCaD_S\}$ group handles the average distance within each of the sub-clusters thus $sCaD_1$

Procedure 3.3: Merging sub-clusters

1	*Input*:
2	sC: *group of sub clusters*
3	S: *Number of sub clusters*
4	$sCaD$: *group of calculated average distances within each of the sub clusters*
5	*Begin*
6	$C_1 \leftarrow sC_1$
7	*Remove entries of label 1 from sC group*
8	$aD_1 \leftarrow sCaD_1$
9	$i \leftarrow 1$
10	$j \leftarrow 2$
11	*while sC group is not empty*
12	*while* $j \leq S$
13	*if sC does not contain lable j*
14	$j \leftarrow j + 1$
15	*continue*
16	*end_if*
17	$db2c \leftarrow$ *compute the distance between two sub clusters* C_i *and* sC_j
18	*if* $db2c < aD_i$
19	*add* sC_j *to* C_i
20	*Remove entries of label j from sC group*
21	*if sC is empty*
22	*break*
23	*end_if*
24	$aD_i \leftarrow$ *recompute average distance among objects in sub cluster* C_i
25	$j \leftarrow$ *minimum value of labels in sC*
26	*else*
27	$j \leftarrow j + 1$
28	*end_if*
29	*end_while*
30	*if sC is empty*
31	*break*
32	*end_if*
33	$i \leftarrow i + 1$
34	$fl \leftarrow$ *find first label in sC*
35	*Append* sC_{fl} *to* C_i
36	*Remove entries of label fl from sC group*
37	$aD_i \leftarrow sCaD_{fl}$
38	$j \leftarrow fl + 1$
39	*end_while*
40	*Return C*
41	*end*

indicates the average distance within sub-cluster sC_1. We also have a group of clusters $C=\{C_1,C_2,...\}$ which is the output of this phase that contains a number of the resultant clusters of the given dataset in which it is self identified. Moreover, C is also the final result of KBCHT after considering released

vertices as mentioned previously. These released vertices are categorized into two groups. One of them identifies them as noise and outliers, and the other one assigns them back to their corresponding clusters.

First we add the first item of the group sC to the group C as shown in line 6 of Procedure 3.3. Then in line 7 we remove the first sub-cluster from the group sC. Thus we want to pick and remove sub-clusters from sC and add them to the group of C to one of its items or append them to be a new item or cluster in the group C. Hence, we want the group sC to be empty for having the clusters in C. Now we have to calculate the average distance between data points in sub cluster C_1 by assigning it the value of $sCaD_1$ because at this stage the group C_1 is the same as sC_1.

As stated previously, for computing the average distance of a given group of data points we construct its Delaunay triangulation to have geometry of triangles. Thus the average distance is the average length of edges of the triangles. The average distance of sub-cluster C_1 is stored in aD_1 of the group aD. The two pointers i, and j in Procedure 3.3 deal with the two groups C and sC respectively. We trace all the sub-clusters in sC thus we have to decide if one of these sub-clusters can be added to the picked item from the group C. So, we have sub-clusters that are merged together to form the final clusters. For deciding to merge or not, we compute the distance between the sub-cluster C_i and each of the remaining sub-clusters in the group sC as stated in line 13 of Procedure 3.3. We calculate the distance between two groups of data points by considering the distance between the two closest data points in which each of these two data points belongs to different group. Other methods of measuring distance between clusters can be found in [28]. If the distance between the cluster C_i and one of the sub-clusters in sC ($db2c$) is less than the average distance within the C_i, we add this sub-cluster of sC to the cluster C_i thus the cluster C_i grows. In case of finding a candidate sub-cluster from the group sC to be merged with one of the clusters in the group C, we remove this sub-cluster from the group sC and check if sC becomes empty to stop and get out from the Procedure 3.3 as shown in lines 21 to 23 and lines from 30 to 32. In

line 24 we have two merged sub-clusters. Hence, we recomputed the average distance within the grown cluster to reflect the actual average distance in it and to accept new merged sub-clusters. Line 25 reassigns a value for the pointer j to point to the previous sub-clusters in sC that does not meet the criteria of merging with old C_i. Because the characteristic of C_i before merging is differ than it after merging, thus we can have additional sub-clusters in sC in which they could be merged with the new formed cluster.

When we reach line 33 of Procedure 3.3, this means that we have already caught one of the clusters of the given dataset from merging process. As a consequence, we have to look forward to finding other clusters in the given dataset. Thus we increment the pointer i by one to point the next location in the group C in which we want to find another cluster. Since we remove each of merged sub-clusters from the group sC, its labels will not be in a regular order. Hence, in line 34 we find the first label that exists in the group sC and assign it to the variable fl. Now the first sub-cluster in the sC is appended to a new index position in the group of C. Then it has been removed from the sC. The average distance within the new inserted sub-cluster in C has assigned the value from its corresponding group of $sCaD$. We assign a new value for the pointer j which points to sub-clusters in the group sC as in line 38. We repeat the above steps until the group sC becomes empty which means that we have obtained the desired clusters from the given dataset in the group C. To make C as a final result of our algorithm, we should back to process the released vertices from the shrinking phase; this is done by assigning them to their corresponding clusters in C. This is done by telling each cluster in C to pick the similar vertices to it which occurs when the distance from the cluster and the tested vertices is less than the average distance within this cluster. The remaining vertices are considered to be odd patterns in the entire dataset and we mark them as noise and outliers. Hence, we find the final results of clusters in the given dataset in a completely unsupervised manner.

3.2 A Toy example: Moon dataset

For illustration purposes, we consider the following example in Fig. 3.4 to be solved by our KBCHT algorithm.

In this dataset we have two moons distributed as shown in Fig. 3.4(a). KBCHT performs the standard Kmeans for obtaining the initial groups of clusters to be processed. Fig. 3.4(b) shows the result from the first phase of KBCHT algorithm in which we have two groups of initial clusters, one of them is with blue 'o's data points and the second one is with red '*'s data points. Now we can process these two groups in a parallel fashion. One thread picks the blue group and the second thread picks the red one. Hence, KBCHT algorithm executes its second phase in which KBCHT constructs the convex hull of the two groups as shown in Fig. 3.4(b), the closed set of solid black lines. At this step we have identified the vertices that represent each of the groups which reside on the drawn convex hull.

Fig. 3.4(c) visualizes the process of shrinking these vertices which is the output result of the Procedure 3.1. It has been noticed that each of the two groups has two connected set of vertices and the shrunk vertices forms a general polygon shape in which KBCHT can detect convex shape as well as non-convex shape clusters. Procedure 3.2 has triggered on the result shown in Fig. 3.4(c) to find the set of sub-clusters sC for this moon dataset.

Fig. 3.4(d) shows four sub-clusters in which $sC=\{sC_1,sC_2,sC_3,sC_4\}$ such that sC_1 is the green '.'s data points, sC_2 is the black '.'s data points, sC_3 is the data points with blue '.'s and sC_4 is the data points with red '.'s. KBCHT algorithm checks these sub-clusters for merging process. Procedure 3.3 works on the result shown in Fig. 3.4(d) to find final clusters C. At the beginning, the group $C=\{sC_1\}$ that is C_1 is equal to sC_1 and sC becomes $\{sC_2,sC_3,sC_4\}$ after removing sC_1 from it. KBCHT searches for one or more of the sub-clusters in sC to be merged with C_1 according to Procedure 3.3. KBCHT finds that sC_3 should be merged with C_1 thus $C=\{sC_1+ sC_3\}$ where C_1 is the two sub-clusters sC_1 and sC_3 together. Then by the next iteration $C=\{sC_1+ sC_3, sC_2\}$ where C_2 is sC_2 and $sC=\{ sC_4\}$. According to Procedure 3.3 sC_2 should be merged with sC_4

thus $C=\{sC_1+ sC_3, sC_2+ sC_4\}$ and sC_4 is removed from sC to be an empty group. Hence, we obtain the two clusters of the moon dataset. Fig. 3.4(e) shows the final of clusters that obtained by KBCHT algorithm.

(a)

(b)

(C)

(d)

(e)
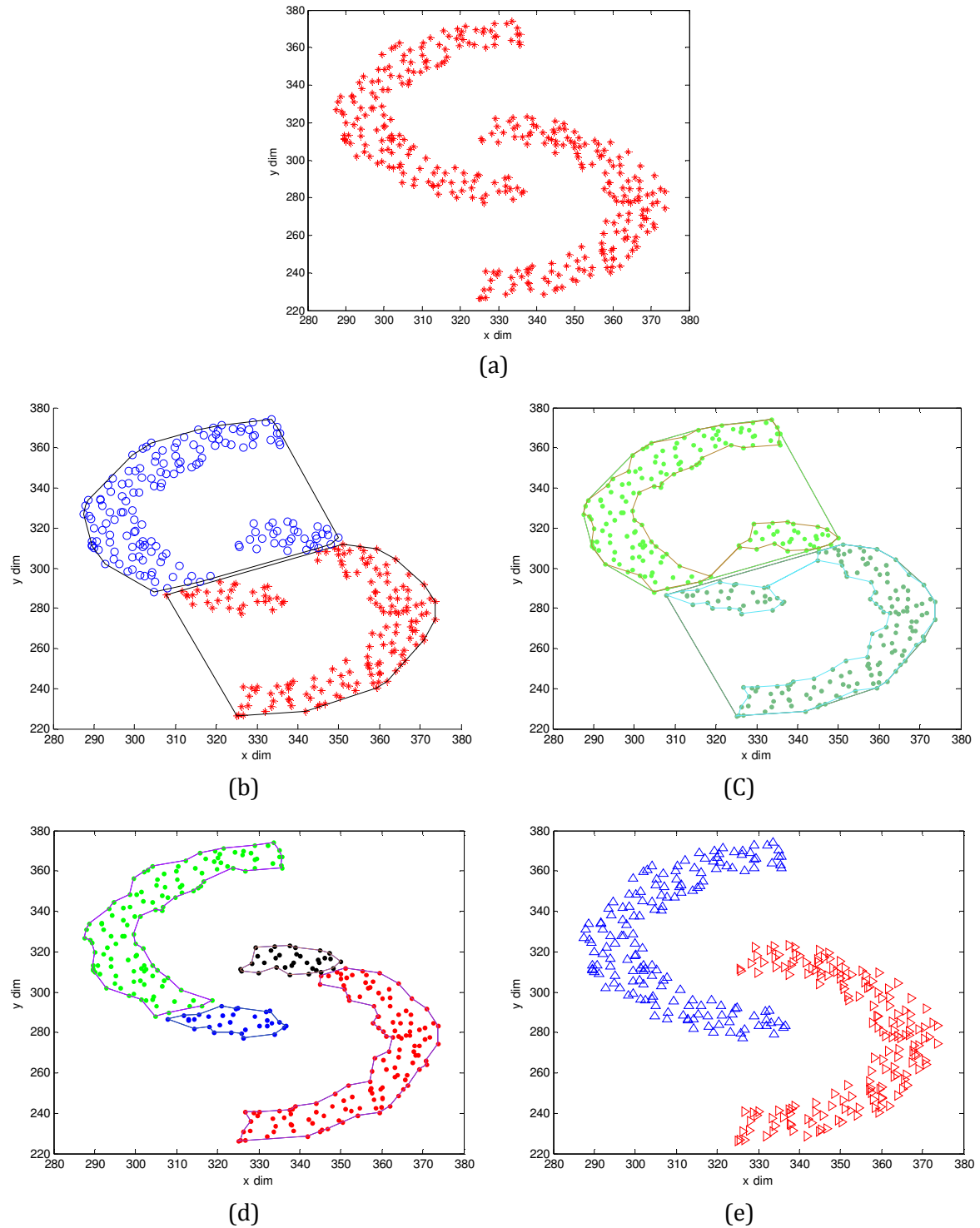
Figure 3.4: Toy example. (a) The moon dataset. (b) Result from the standard Kmeans where blue 'o's are in one cluster and red '*'s are in another cluster and each of them are enclosed by its convex hull. (c) The shrinking process. (d) The process of finding sub-clusters. (e) The final result of our KBCHT algorithm.

3.3 Simulation and Results

3.3.1 Datasets results
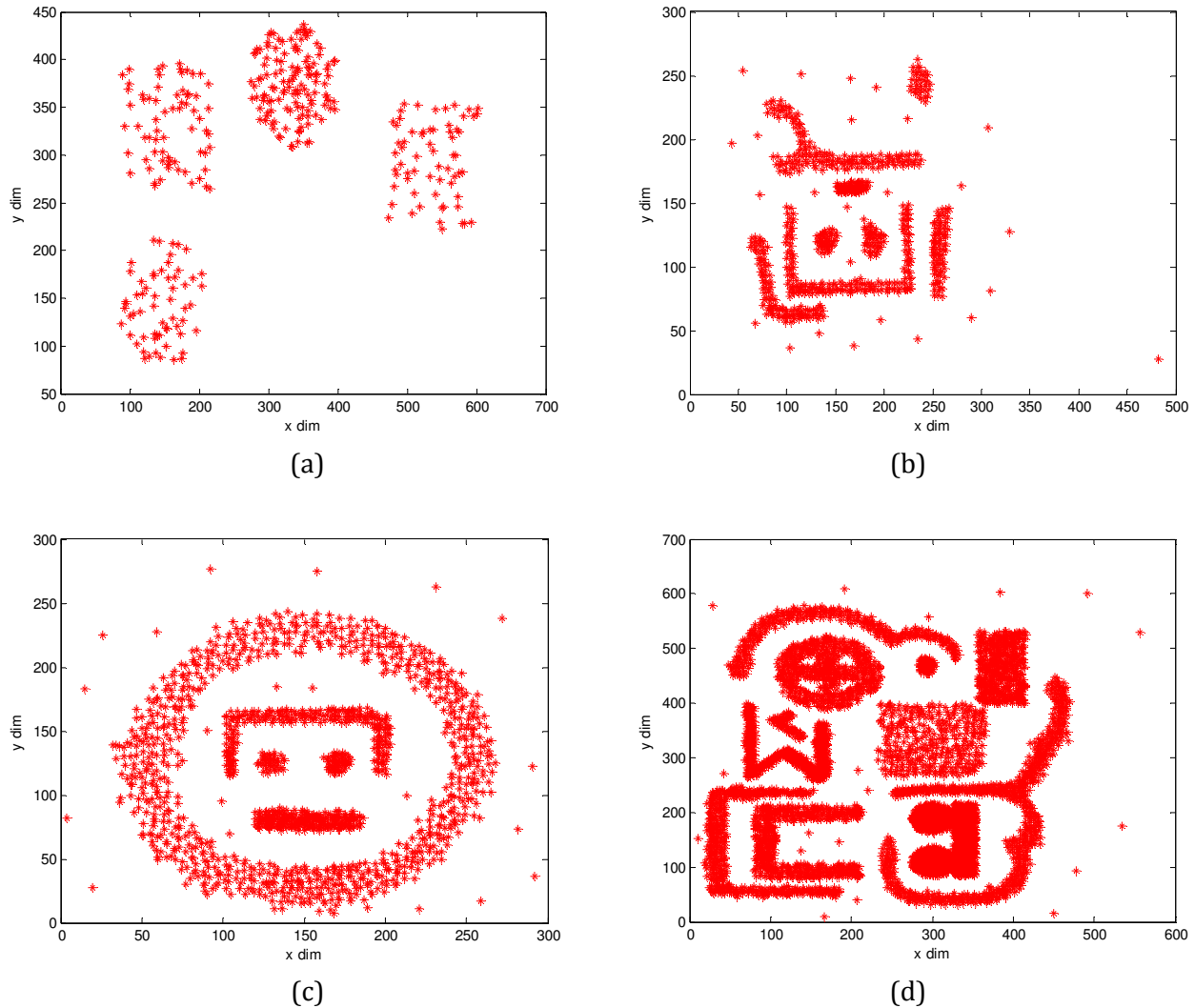
Figure 3.5: Artificial datasets. (a) DS1. (b) DS2. (c) DS3. (d) DS4.

We illustrate the strength of the KBCHT algorithm in a number of artificial two dimensional datasets, since the results can easily be verified visually. We generate four types of datasets DS1, DS2, DS3 and DS4 with 350, 1000, 2000, and 6000 data points respectively as shown in Fig. 3.5. Now, our purpose is to verify that KBCHT algorithm should be able to detect the clusters in these datasets, which can be differentiated by eyes, successfully. DS1 is considered to be a simple dataset that contains four well separated clusters, but the others

contain clusters with complex non-convex shapes, different sizes, densities, noise and outliers. We compare our algorithm with the Affinity Propagation [67] algorithm in which messages are exchanged between data points until the final clusters are found, because it does not need to be rerun many times each with different initial conditions and does not require to input the number k of clusters. Also, we compare it with one of the recently spectral clustering algorithms because they are designed to handle non-convex shaped clusters and solve the problem of expensive computations of them which is the spectral clustering using Nystrom method [76].

As stated previously, we have to do the clustering task with no prior information knowledge about the given dataset and this is the goal of our proposed KBCHT algorithm. As a consequence, we have treated the given datasets as closed black boxes. For this reason, we have used the same value of k, in case of our proposed KBCHT algorithm and Spectral clustering using Nystrom algorithm, to figure out the output with no prior information. In addition, as we have generated these four artificial datasets, we do further step in assigning the true value of k to the Spectral clustering using Nystrom algorithm

DS1: This dataset contains 350 data points distributed into 4 well separated clusters. However, it is suitable to find its clusters using the standard Kmeans. But when we perform our KBCHT its first phase does not detect the four clusters correctly. We deal with the given dataset as black boxes that mean we do not have a prior knowledge about how many clusters exist or how they are distributed. Thus we roughly choose k=5 and our KBCHT correctly identifies the four clusters as shown in Fig. 3.6(a). Fig. 3.6(b) shows the result of clustering using Affinity Propagation in which it does not use any input parameters. However, it does not correctly detect the number of clusters. It has a result of 11 clusters. Since the spectral clustering using Nystrom method has two inputs: sigma and the number of clusters which are user defined. We use a sigma to equal 20 and use the same value of k=5 for the reason stated above. Even though this is a simple dataset Fig. 3.6(c) shows the weakness of the

spectral clustering when its parameters are not identified correctly. In which it has a result of five clusters where it wrongly splits one of the clusters into two. To be more honest with the used spectral clustering, we rerun it with $k=4$ and we obtain a result that correctly identifies the four clusters as shown in Fig. 3.6(d).

DS2: This dataset contains 1000 data points distributed into 8 complex shaped clusters and also with present noise and outliers. Fig. 3.7 (a) shows the effectiveness of our KBCHT algorithm in identifying the true 8 clusters however we roughly choose $k=10$. The noise and outliers are correctly marked as the black '+'s in Fig. 3.7(a). This is due to how we measure the Delaunay triangulation average *AVG* within the initial group, where we exclude the convex hull vertices from the computation of the *AVG* which eliminates the effect of outliers. The Affinity Propagation detects 20 clusters as shown in Fig. 3.7(b). For the spectral clustering using Nystrom method with sampling equals to 400 and we set sigma=20 and also we run it one time with $k=10$ as chosen for our KBCHT and another time using correct number of clusters $k=8$. Fig. 3.7(c) and Fig. 3.7(d) improves that our proposed KBCHT outperforms it in case of cluster quality.

DS3: This dataset contains 2000 data points distributed into 5 complex shaped clusters and also with present noise and outliers. Fig. 3.8 (a) shows the strength of our KBCHT algorithm in identifying 5 clusters correctly. However, we roughly choose $k=15$. The noise and outliers are correctly marked as the black '+'s in Fig. 3.8(a). In fact, the shrinking phase of KBCHT algorithm traverses all candidate vertices that could be engaged into the representative boundary of the given initial groups, this is due to the high priority given to the longest edge between two consecutive vertices to be shrunk according to the natural of the given initial group that it is reflected by the calculated delaunay triangulation average inside this group. The Affinity Propagation detects 28 clusters as shown in Fig. 3.8(b). We proceed as above in selecting sampling of 400 and sigma=20 for the spectral clustering using Nystrom method but it triggers an error on the value of sigma thus we tuned it to be equal to 5 and also we run it one time with

k=15 as chosen for our KBCHT and another time using the correct number of clusters *k*=5. Fig. 3.8(c) shows that it does not fully detect the cluster correctly but in Fig. 3.8(d) the spectral clustering shows that it handles two clusters correctly but without identifying noise and outliers.

DS4: This dataset contains 6000 data points distributed into 11 complex shaped clusters and also with presence of noise and outliers. Fig. 3.9 (a) shows that our KBCHT algorithm finds all of the 11 clusters correctly. However, we roughly choose *k*=15. The noise and outliers are correctly marked as the black '+'s in Fig. 3.9(a). The Affinity Propagation detects 53 clusters as shown in Fig. 3.9(b). The spectral clustering using Nystrom method with sampling equals to 400 and sigma=5; the result shown in Fig. 3.9(c) with *k*=15 as chosen for our KBCHT and when using the correct number of clusters *k*=11 we obtain clusters as shown in Fig. 3.9(d).

In the next Section, we do further analysis of the given results in which we measure the obtained clustering quality rather than subjectively evaluating the results.
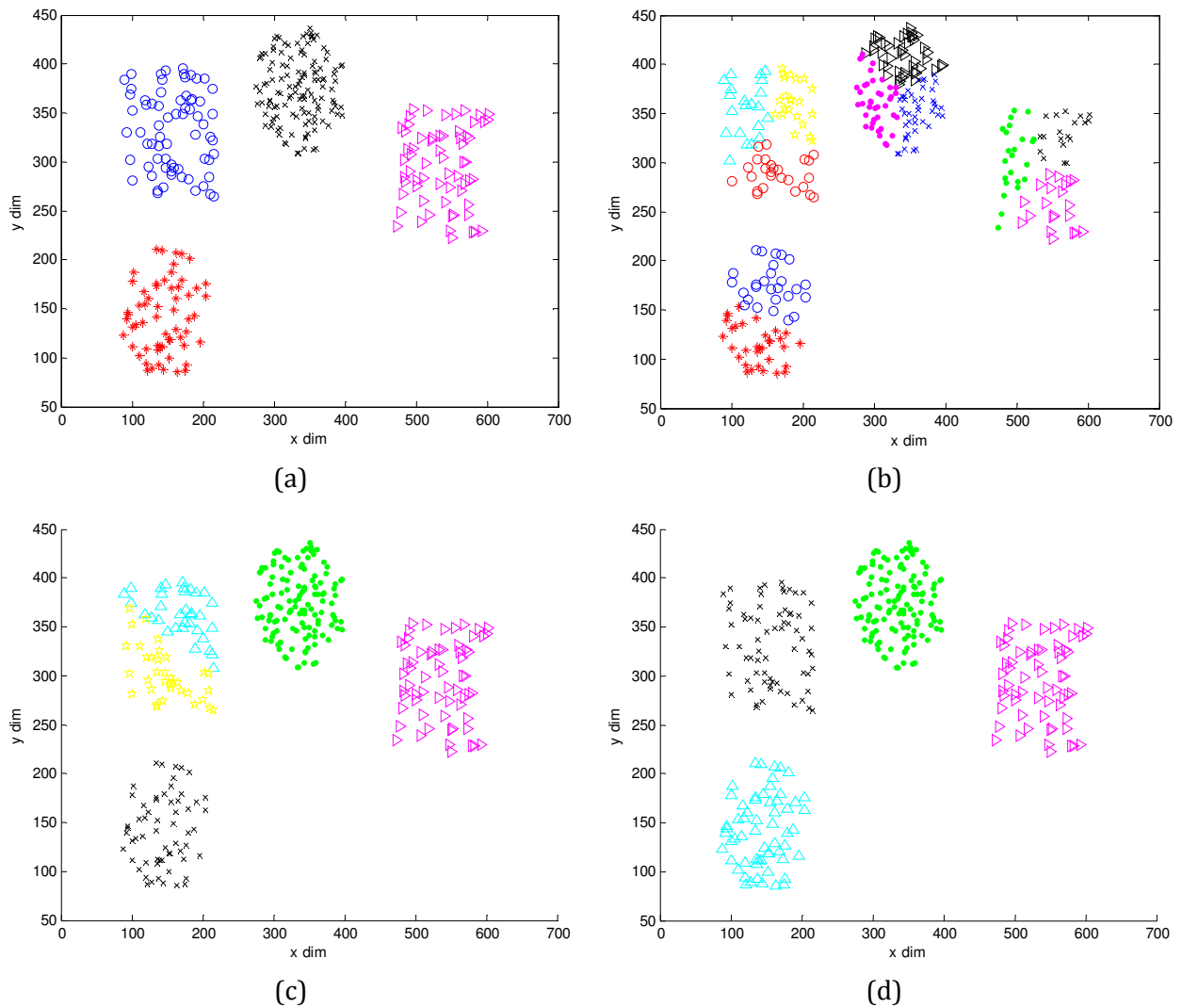
Figure 3.6: Clustering Results of DS1: (a) our proposed KBCHT (using k=5). (b) Affinity Propagation. (c) Spectral clustering using Nystrom method (samples=100, sigma=20 and k=5). (d) Spectral clustering using Nystrom method (samples=100, sigma=20 and k=4).

(a)

(b)

(c)

(d)
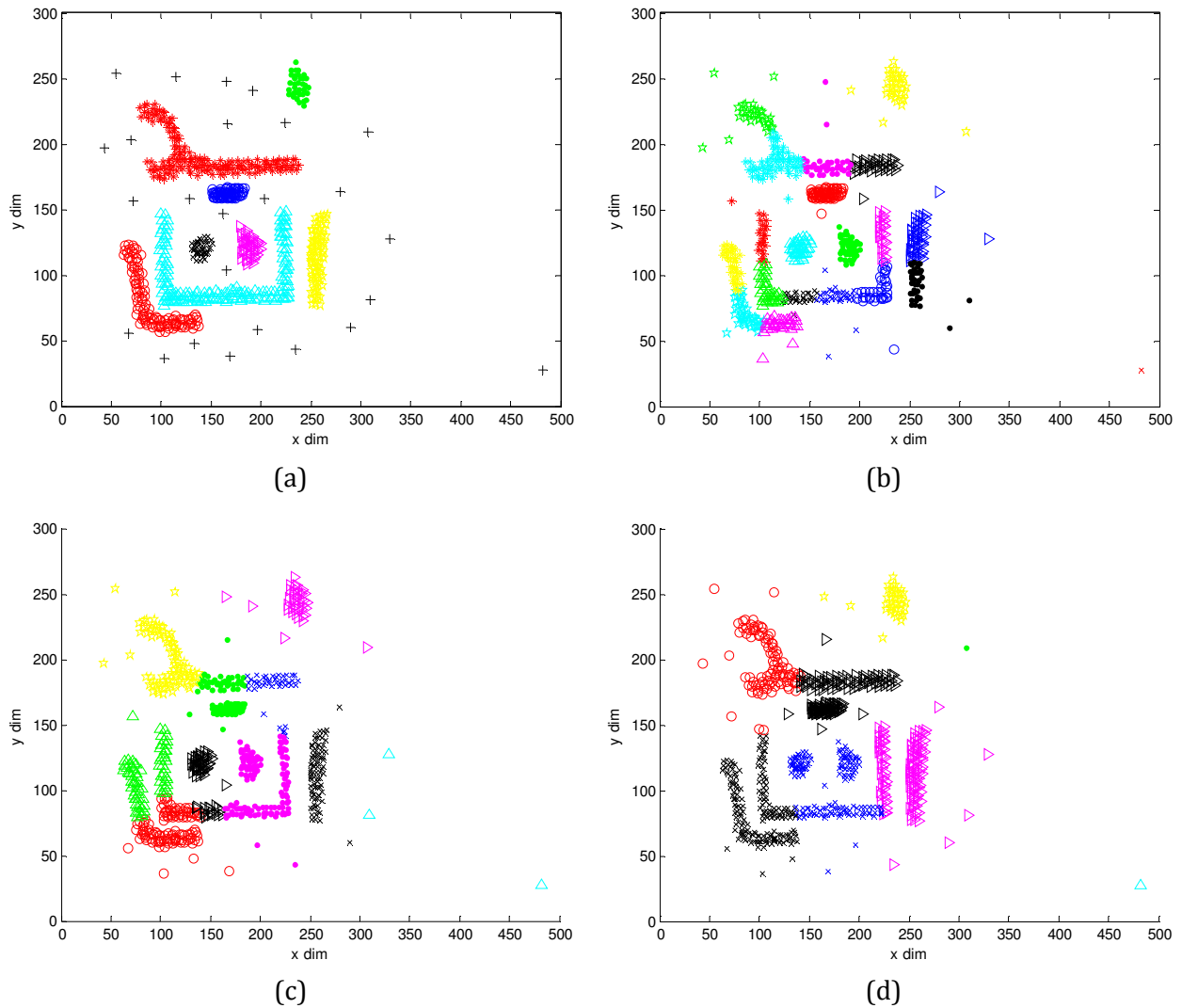
Figure 3.7: Clustering Results of DS2: (a) our proposed KBCHT (using k=10). (b) Affinity Propagation. (c) Spectral clustering using Nystrom method (samples=400, sigma=20 and k=10). (d) Spectral clustering using Nystrom method (samples=400, sigma=20 and k=8).
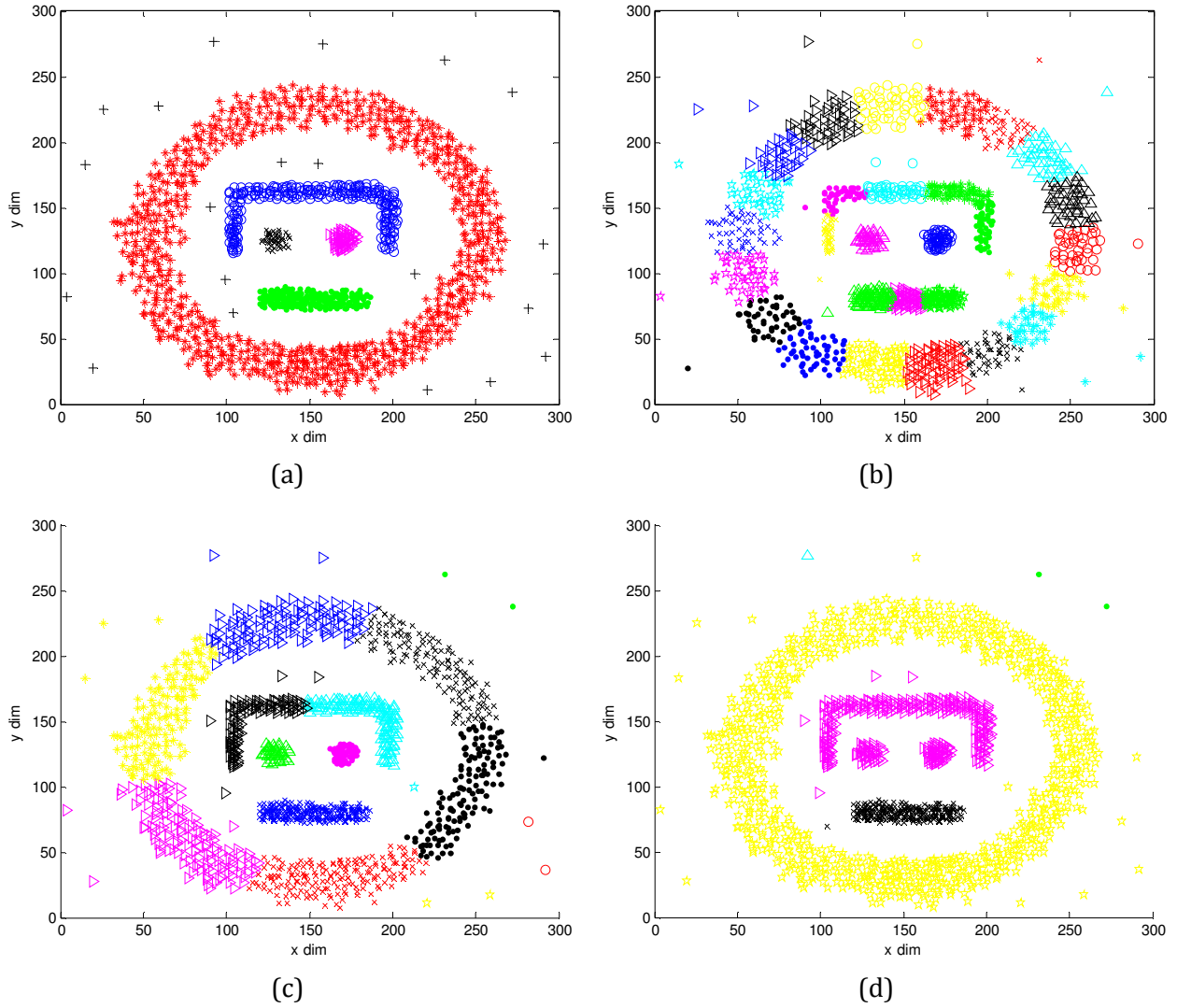
Figure 3.8: Clustering Results of DS3: (a) our proposed KBCHT (using k=15). (b) Affinity Propagation. (c) Spectral clustering using Nystrom method (samples=400, sigma=5 and k=15). (d) Spectral clustering using Nystrom method (samples=400, sigma=5 and k=5).
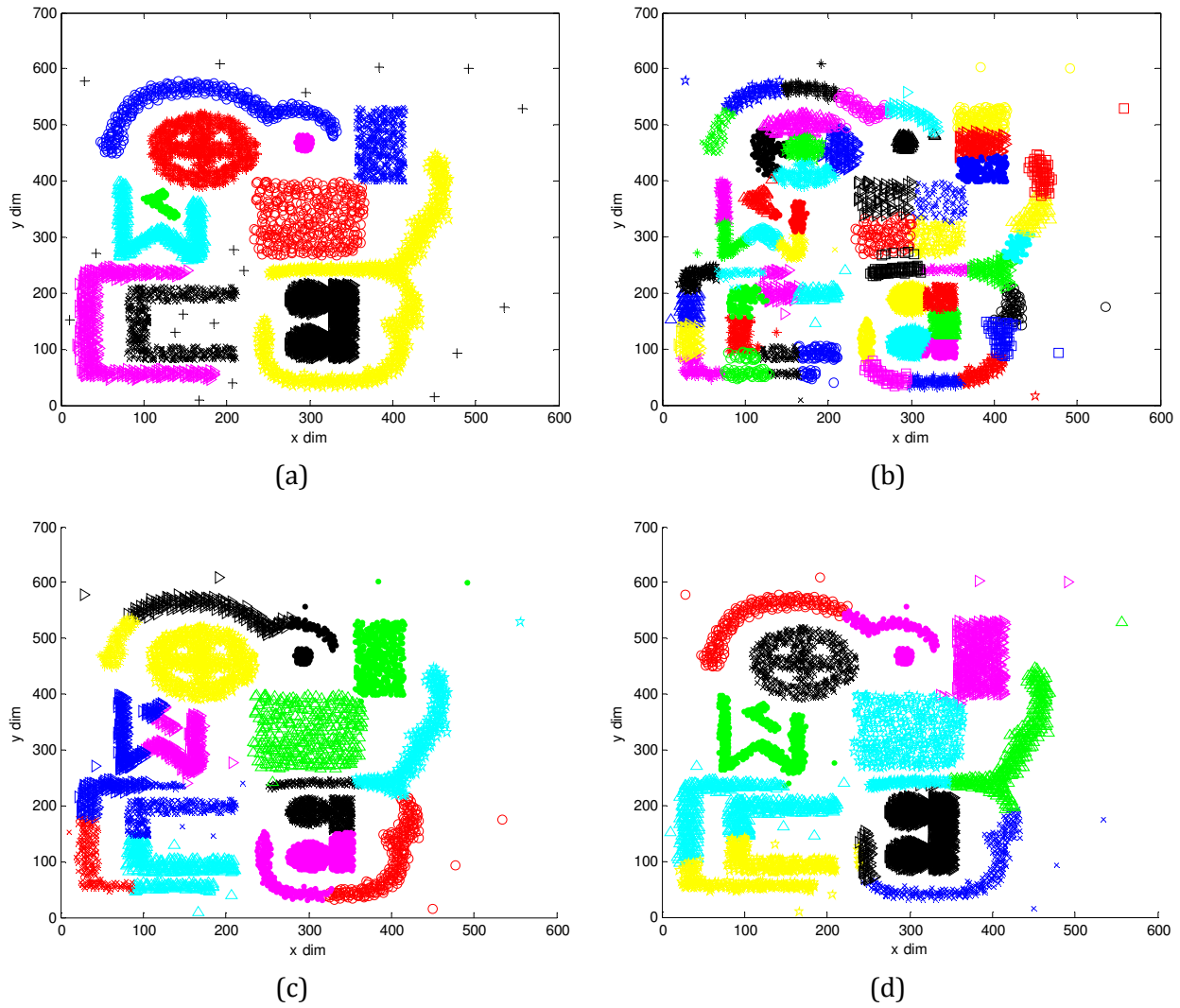
Figure 3.9: Clustering Results of DS4: (a) our proposed KBCHT (using k=15). (b) Affinity Propagation. (c) Spectral clustering using Nystrom method (samples=400, sigma=20 and k=15). (d) Spectral clustering using Nystrom method (samples=400, sigma=20 and k=11).

3.3.2 Performance Analysis

In order to prove the effectiveness and the strength of our proposed KBCHT algorithm, we construct experiments for measuring the performance in case of time cost and clustering accuracy according to the visual results shown in Section 3.3.1. Table 3.1 shows the comparison of the performance of our proposed KBCHT algorithm and the used clustering algorithms in which we have been used in Section 3.3.1 for each of the results that obtained visually in previous section. We measure the time that it takes for completing its mission of clustering in seconds. Furthermore, it is important to measure how they are accurate in identifying the final results which is more important. For measuring the clustering accuracy, we follow the technique used in [77] and [76] to evaluate the quality of the resultant clusters and we use the following equation:

$$Accuracy = \frac{\sum_{i=1}^{n} \delta(y_i, map(c_i))}{n} \tag{3.3}$$

Where n is the number of data points in the given dataset, y_i and c_i are the true cluster label and the obtained cluster label respectively. The delta function $\delta(y, c)$ equals 1 if $y=c$ and equals 0 otherwise. To measure the accuracy, we need a permutation function that maps each obtained cluster label to a true cluster label. Moreover, for achieving the optimal matching the Hungarian algorithm [78] is used.

As shown in Table 3.1, our proposed KBCHT algorithm has a clustering accuracy as high as 100% for all of the datasets that have been used. For DS1, we have also an accuracy of 100% for the spectral clustering using Nystrom method but when we use the exact number of clusters that included in DS1 even in this case our proposed KBCHT outperforms it in the execution time. But when using the same value of k, our algorithm has the superiority in both the time cost and the clustering accuracy. To find the final results of clusters in DS1, our KBCHT takes 0.12 sec, 0.03 sec and 0.12 sec for the first, the second and the third of its phase respectively.

For the DS2, our proposed KBCHT algorithm executes the first phase in 0.3 sec, phase two in 1.03 sec and the last phase in 0.87 sec. Hence the overall time

cost is 2.2 sec as shown in Table 1. Our KBCHT takes 1.27 sec, 0.92 sec and 5.11 sec for its three phases respectively for the DS3. And for the DS4 it spends the most of its time in executing the third phase in which it takes 32.55 sec, the first and the second phases are executed in 5.1 sec and 10.84 sec respectively. However, for the DS4 our algorithm takes much time than the spectral clustering algorithm, the overall performance of KBCHT improves the clustering results in case of the quality of the results compared with this small gap reported in the execution time.

Table 3.1: Comparison of the performance of our proposed KBCHT algorithm and existing algorithms with time cost and clustering accuracy.

Dataset	Algorithm	Time Cost (sec)	Clustering Accuracy (%)
DS1	Proposed KBCHT (k=5)	0.27	100
	Affinity Propagation	7.58	39.62
	Spectral clustering using Nystrom (samples=100,sigma=20,k=5)	0.31	89.94
	Spectral clustering using Nystrom (samples=100,sigma=20,k=4)	0.29	100
DS2	Proposed KBCHT (k=10)	2.20	100
	Affinity Propagation	21.38	53.06
	Spectral clustering using Nystrom (samples=400,sigma=20,k=10)	3.19	52.02
	Spectral clustering using Nystrom (samples=400,sigma=20,k=8)	2.97	60.76
DS3	Proposed KBCHT (k=15)	7.30	100
	Affinity Propagation	41.78	23.69
	Spectral clustering using Nystrom (samples=400,sigma=5,k=15)	10.39	44.95
	Spectral clustering using Nystrom (samples=400,sigma=5,k=5)	10.27	91.84
DS4	Proposed KBCHT (k=15)	48.49	100
	Affinity Propagation	453.98	25.62
	Spectral clustering using Nystrom (samples=400,sigma=20,k=15)	43.23	65.12
	Spectral clustering using Nystrom (samples=400,sigma=20,k=11)	43.71	66.21

Does the choice of *k* in our proposed algorithm affect the performance: As we have mentioned in the previous section, the proposed KBCHT does not require prior knowledge information for finding the final results of clusters in a given dataset. It makes self analysis to discover the clusters. But how it performs when changing the value of *k*? For this purpose we construct experiments on the DS3 and we choose *k* to be from 2 to 70. For selecting *k*=2, the DS3 has been partitioned into two initial groups from the first phase as shown in Fig. 3.10(a). In this case all the overhead will be concentrated on the second phase of KBCHT in which the shrinking process investigates more complex shaped group and more vertices to find a set of shrunk vertices. KBCHT does not accept to have only one partition from the first phase. The two partitions of first phase are illustrated in Fig. 3.10 (a) that takes 0.34 sec. Fig. 3.10(b) and 3.10(c) show the second phase operations of shrinking vertices and finding sub-clusters (8 sub-clusters) respectively. In which the effect of noise and outliers are eliminated as shown in Fig. 3.10(c). The second phase takes 18.13 sec to be accomplished. Fig. 3.10(d) shows the final result of KBCHT after merging phase and detecting noise and outliers which it takes 2.75 sec.

Fig. 3.11 shows how each of the three phases of our KBCHT algorithm performs with different values of *k*. It has been shown that the most consuming phase is the third one with the increasing values of *k*. Since the number of comparisons increased between pairs of sub-clusters. As increasing the value of *k*, the second phase execution time drops dramatically. Because as the group under shrinking process becomes smaller, the complexity of the group decreases thus it can be handled faster. The most challenging case on shrinking of DS3 is when *k*=2 as shown in Fig. 3.10(b) in which each group has long embedded boundaries to be investigated by the shrinking process. KBCHT was proud to face this challenge to prove its efficiency and strength in handling this type of dataset. In Fig. 3.12 we show the overall time cost of the KBCHT algorithm when varying the number of *k*. As the value of *k* increased the overall time cost has increased too. The best value of *k* is when it equals 10. At this
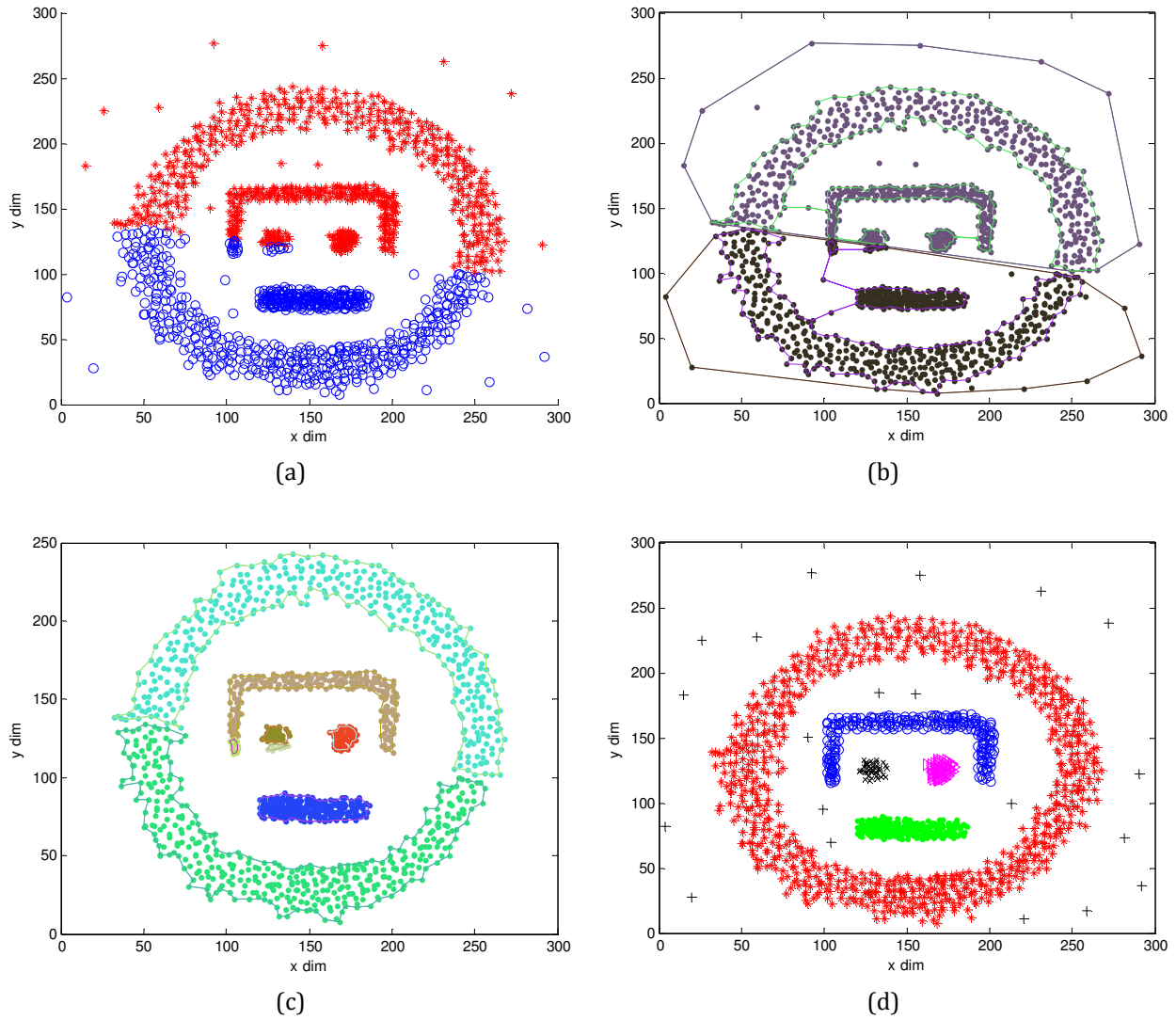
Figure 3.10: Analysis of KBCHT when *k*=2 on DS3. (a) Result from the first phase. (b) The Shrinking process. (c) the sub-clusters which is found by KBCHT (d) The final result of KBCHT after merging sub-clusters and identifying noise and outliers.

value of *k*, the execution time spends less time among other values and equals to 6.49 sec. However, we choose the value of *k* to obtain initial groups from the first phase of KBCHT. The resultant number of partitions is not always the same as *k*. This is because KBCHT uses the standard Kmeans in its first phase and it may suffer from bad dropped initial prototypes thus a problem of dead prototypes may be occurred. Fig. 3.13 shows the exact number of obtained partitions from the first phase as the value of *k* increases. In case of *k*=15 in this experiment, KBCHT obtains only 7 partitions from the first phase which it takes 0.89 sec to have them. According to this 11 sub-clusters have been

considered in 2.72 sec then the merging process to find the final clusters takes 3.93 sec thus the overall time cost is 7.54 sec. However, in Fig. 3.8(a) and Table 3.1 when we have used the value of k equals 15, we caught 14 initial partitions from the first phase of KBCHT and 16 sub-clusters to be merged. We evaluate the clustering accuracy for each used value of k. As illustrated in Fig. 3.14, KBCHT maintains a steady state of high accuracy rate as 100% for the majority of the used k. For k=50, 55, 60 and 70, the clustering accuracy is 99.21%, 98.30%, 99.94% and 99.65% respectively.
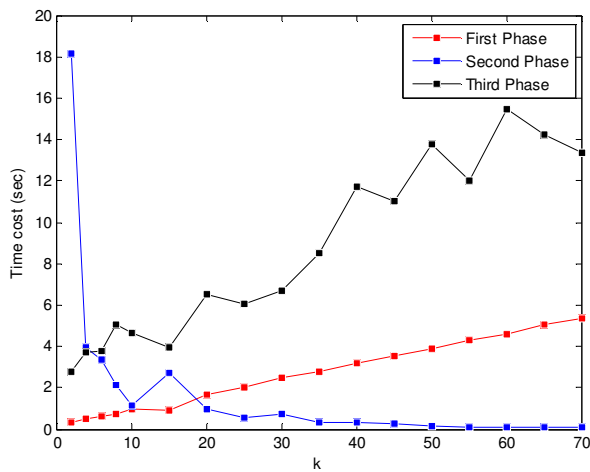
Figure 3.11: Time cost (sec) of KBCHT three phases vs. varying number of k. red: first phase, blue: second phase and black: third phase. (based on DS3)
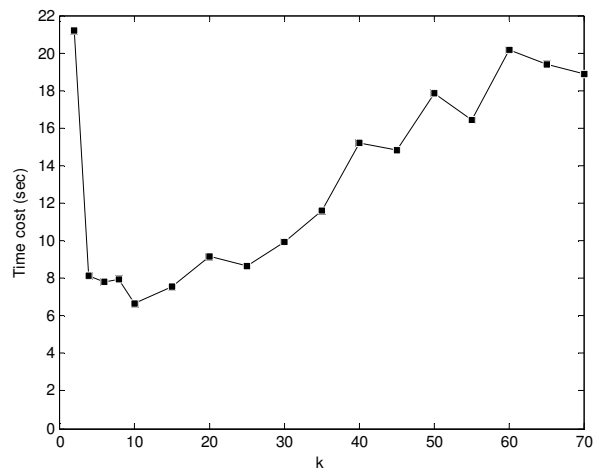
Figure 3.12: Time cost (sec) of KBCHT vs. varying number of k. (based on DS3)
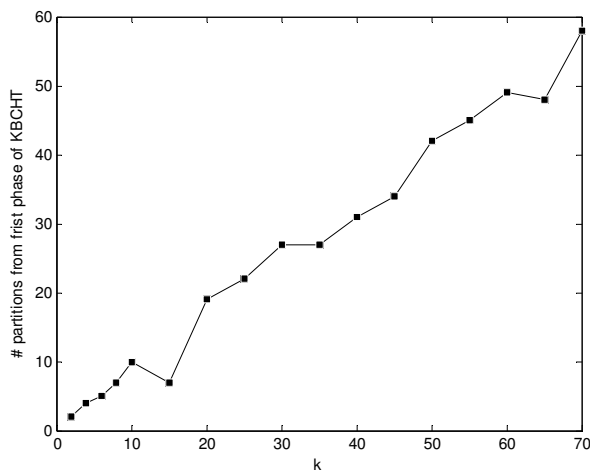
Figure 3.13: Obtained number of initial partitions from the first phase vs. varying number of k. (based on DS3)
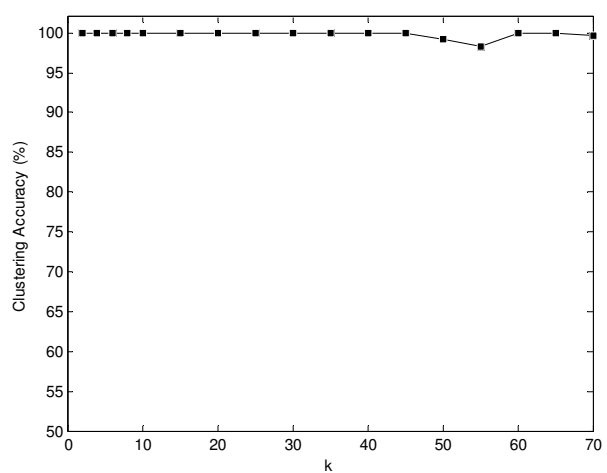
Figure 3.14: Measuring the clustering accuracy (%) of KBCHT vs. varying number of k. (based on DS3)

We analyze the result of how *k* affects our KBCHT algorithm based on one of our complex generated dataset. To do further investigation on *k*, thus we can relatively make some specific judgment. We have generated relatively simple dataset contains 800 data points distributed into 5 clusters which is named DS5 as shown in Fig. 3.15.

We have repeated the same experiment situation as we did for DS3 in which we run KBCHT algorithm on the dataset DS5 with varying *k* from 2 to 70.

Fig. 3.16 shows time cost in seconds for each phase of KBCHT algorithm. It is obvious that the third phase is the most consuming one especially when *k* increased. The obtained result in Fig. 3.16 is somewhat close in its concept to the result obtained in Fig. 3.11. Fig. 3.17 provides the overall time cost in seconds for KBCHT algorithm in which it starts high when *k*=2 then it is dropped then as *k* increased the execution time increased. Fig. 3.18 gives an overview of the actual number of partitions that obtained from the first phase since it suffers from bas initial condition that caused dead prototypes. In addition, Fig. 3.19 shows the clustering accuracy as increasing the value of *k* which maintains the highest steadily state.
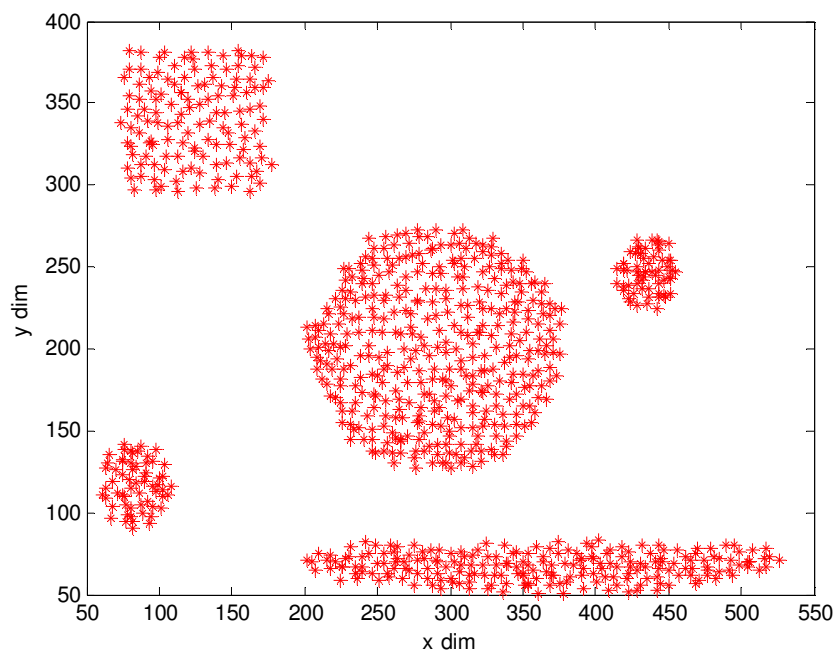
Figure 3.15: Artificial dataset DS5.
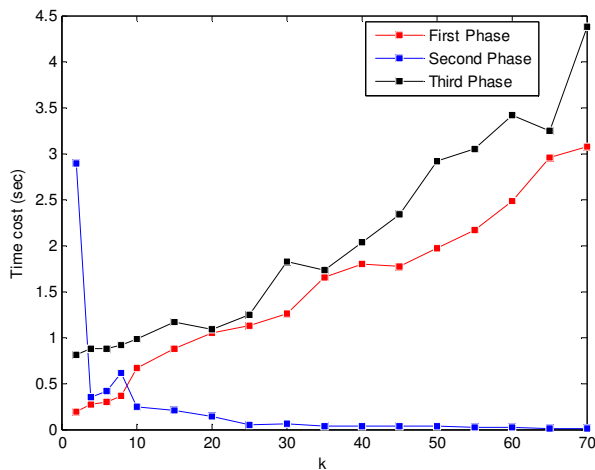
Figure 3.16: Time cost (sec) of KBCHT three phases vs. varying number of *k*. red: first phase, blue: second phase and black: third phase. (based on DS5)
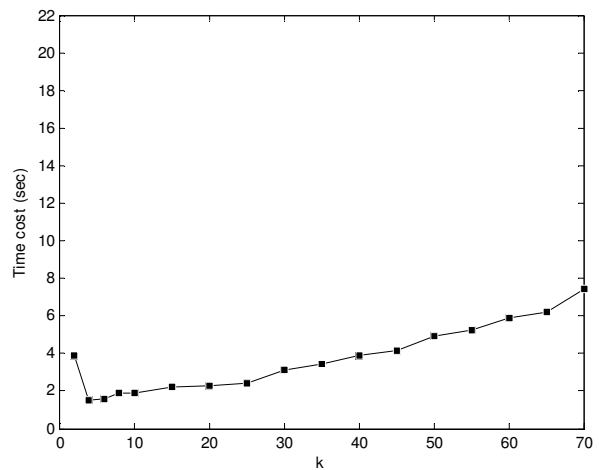
Figure 3.17: Time cost (sec) of KBCHT vs. varying number of *k*. (based on DS5)
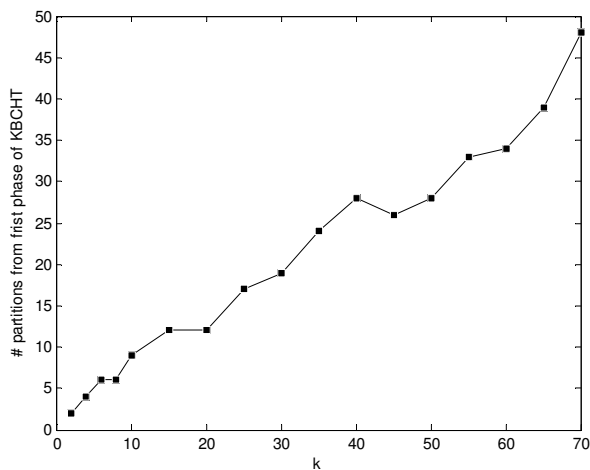

Figure 3.18: Obtained number of initial partitions from the first phase vs. varying number of *k*. (based on DS5)
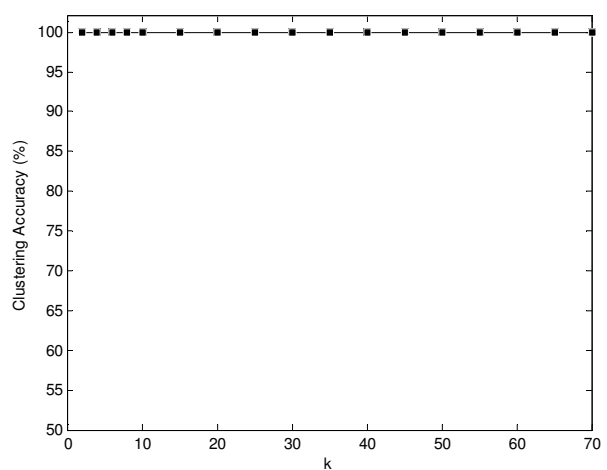
Figure 3.19: Measuring the clustering accuracy (%) of KBCHT vs. varying number of *k*. (based on DS5)

Experimenting on DS3 and DS5, we can notice that the overhead is high with respect to the execution time when *k*=2 and drops dramatically when *k*=4. In case of clustering accuracy, we can conclude that when we have applied some of the large values of *k*, the accuracy may suffer slightly in case of complex shaped dataset.

For DS3, the best result can be achieved when *k*=10. Moreover, the result may be acceptable when *k* is ranging between 5 and 15. On the other side of

DS5, we have obtained the best result when k=4. Furthermore, when k is between 4 and 10 the result could be acceptable. Hence, we can choose the lower bound of k to be 5. Of course, this does not prevent the interested researchers from using any approaches that offers auto determination of the value of k for Kmeans algorithm. However, we recommend that the value of k is not less than 5.

Chapter *4*

KBCHT: A Topology Preserving Mapping As A Preprocessing

4.1 Overview

In the former Chapter, we have proved the efficiency of our proposed KBCHT algorithm on our generated 2-D artificial datasets. In this Chapter, we demonstrate the performance of our KBCHT algorithm based on high dimensional real datasets. In order to handle real datasets adequately, there high dimensionality needs to be reduced for coping the cures of dimensionality and other undesired features of high dimensional space [83]. The dimensionality reduction can be used as a visualization tool of high dimensional data [83].

There are many researches that are concerned in reducing the high dimensionality for data analysis. Furthermore, to enhance the performance of classification or clustering on these data, their dimensionality should be reduced before applying classification or clustering techniques [83, 93-96]. As a consequence, we can benefit from any of the available techniques in reducing the dimensionality of the given datasets to be engaged with our proposed KBCHT algorithm.

In this thesis, we have used a topographic mapping technique in conjugate with our KBCHT algorithm. A topographic mapping is a transformation of high

dimensional data which preserves some structure in the data such as the points which are mapped close to each other share some common properties while in contrast the points which are mapped far from each other do not share a common feature or property.

In this Chapter, we use the Generative Topographic Mapping (GTM) as a basis preprocessing step for our proposed KBCHT algorithm.

4.2   Generative Topographic Mapping (GTM)

The GTM is a statistical model for modeling the probability density of data points and finding non-linear mapping of high dimensional space onto low dimensional space.

The basis of the GTM is to generate a grid of K latent points $z_1, z_2, …, z_K$ in latent space. These latent points are mapped non-linearly into the data space using a set of M basis Gaussian functions with respect to a set of weights W, where W is M x D, and D is the dimensionality of data space. The GTM is optimized using the Expectation-Maximization (EM) algorithm [2], [28].

4.3   Simulation Experiments

After validating the performance of the proposed algorithm on complex artificial datasets, we have accomplished many experiments to evaluate the efficiency of our proposed algorithm on high dimensional datasets over the same competing algorithms as in Chapter 3. All the experiments in this thesis have been performed using an Intel dual at 1.87 GHz with 2 GB of RAM. A brief description of the used real datasets and simulation results will be presented in the following sub sections.

4.3.1 Real Datasets

In our simulation experiments we have used 10 real datasets. All of these datasets are available from the UCI machine learning repository [88]. General information about these datasets is given in Table 4.1.

Some of these real datasets have missing values and different types of variables with different scales. Hence, they would be preprocessing before engaging them into the clustering process (Appendix B explains how to prepare the data before clustering).

Table 4.1: The Descriptions of the used UCI datasets

No.	Datasets	# of Objects	# of Dimensions	# of clusters
1	Iris	150	4	3
2	Libras movements	360	90	15
3	Wine	178	3	3
4	Glass	214	10	6
5	Pendigits	10992	16	10
6	Image Segmentation	2310	19	7
7	Spambase	4601	57	2
8	Yeast	1484	8	10
9	Arrhythmia	452	279	16
10	Dermatology	366	34	6

4.3.2 Parameters Setting

As stated previously, we have to do the clustering task with no prior information knowledge about the given dataset and this is the goal of our proposed KBCHT algorithm. As a consequence, we have treated the given datasets as closed black boxes. As a result in our experiments, the value of k is set to 5 for all of the used real datasets except for the largest used dataset (Pendigits) we use a value of 15. The same value of k is used for both our proposed KBCHT algorithm and the spectral clustering using Nystrom algorithm.

In this Chapter and as mentioned previously, we have used the GTM as a basis preprocessing step for our proposed algorithm. Hence, the grid of latent points and the grid of base functions have been chosen to be 10x10 and 3x3 respectively for all of the 10 used UCI datasets.

Moreover, for the spectral clustering using Nystrom algorithm, the value of sigma is set to 50 and the sample size has been chosen to be 100 for all of the

used datasets. But we have faced a problem that the used default sample size of 100 is not valid for some datasets. Thus, we have chosen a sample size of :

150 for Wine, Glass, Yeast, Arrhythmia and Dermatology.

300 for Image segmentation.

1500 for Pendigits.

4601 for Spambase.

In which the Spambase dataset does not accept to have any sample size except the value of its whole number of objects.

For the Affinity Propagation algorithm, we have used its default setting as in [67].

4.3.3 Results Analysis

For illustration of the effectiveness of our algorithm, we have conducted the same performance measurements that we already used in Chapter 3.

Table 4.2 and Fig. 4.3 summarize the obtained simulation results on real datasets in which our proposed algorithm outperforms the other algorithms.

For Iris dataset, as an example, our proposed KBCHT using GTM has the highest performance comparing with other algorithms. It has an accuracy of 88%. In contrast, the spectral clustering using Nystrom has an accuracy of 57.33% and the Affinity Propagation has 33.33% clustering accuracy. Moreover, our proposed KBCHT using GTM obtained more accurate result faster than the other algorithms.

Fig. 4.1 shows the result of GTM on the Iris dataset, where the Iris dataset has been projected on 2 dimensional space. It is obvious that the Iris dataset has one linearly separable cluster from the other two clusters; in which the other two clusters are overlapped. The GTM takes 0.26 sec, the first phase of KBCHT takes 0.06 sec, the second phase takes 0.14 sec and the last phase executed in 0.11 sec. Hence, the overall time cost for the proposed KBCHT using GTM is 0.57 sec.
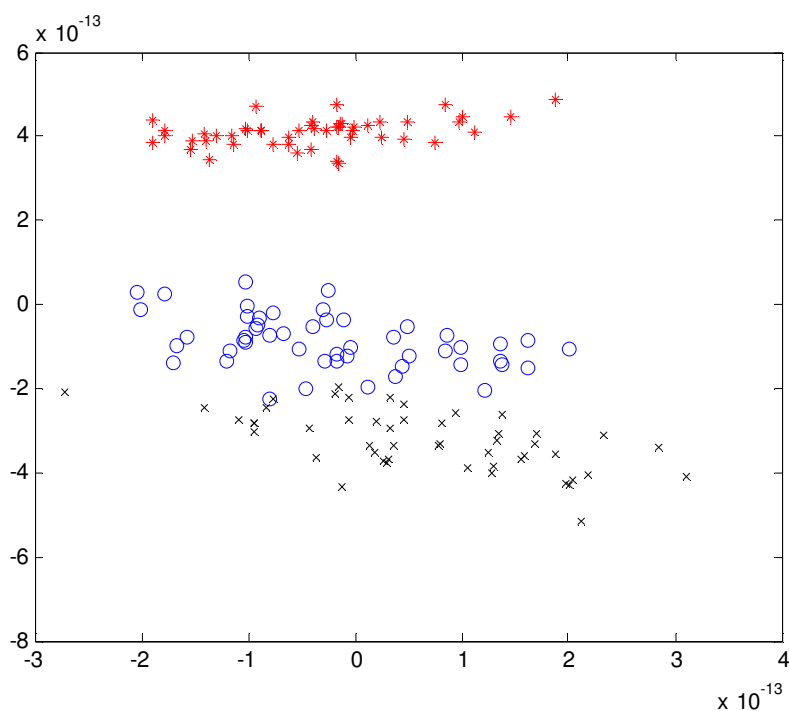
Figure 4.1: The result of using GTM on Iris dataset (the three clusters are distinguished using different color and symbol for each cluster).

The results of experiments on the rest of the used UCI datasets are summarized in Table 4.2. In which the performance of our proposed KBCHT using GTM and exiting algorithms has been compared with time cost in seconds and the clustering accuracy which is calculated as mentioned in Chapter 3.

Table 4.2: Comparisons of the performance of our proposed KBCHT algorithm using GTM and existing algorithms with time cost and clustering accuracy on real datasets.

Datasets	Algorithms	Time Cost (sec)	Clustering Accuracy (%)
Iris	Proposed KBCHT using GTM	0.57	88.00
	Affinity Propagation	6.03	33.33
	Spectral clustering using Nystrom	0.61	57.33
Libras Movements	Proposed KBCHT using GTM	0.99	30.56
	Affinity Propagation	14.64	6.67
	Spectral clustering using Nystrom	0.94	24.72
Wine	Proposed KBCHT using GTM	0.63	93.82
	Affinity Propagation	5.77	39.88
	Spectral clustering using Nystrom	0.79	65.16
Glass	Proposed KBCHT using GTM	0.70	65.02
	Affinity Propagation	6.81	48.59
	Spectral clustering using Nystrom	1.29	63.55
Pendigits	Proposed KBCHT using GTM	103.13	38.27
	Affinity Propagation	-	-
	Spectral clustering using Nystrom	212.11	29.10
Image Segmentation	Proposed KBCHT using GTM	3.88	31.52
	Affinity Propagation	214.32	14.29
	Spectral clustering using Nystrom	3.81	14.58
Spambase	Proposed KBCHT using GTM	93.03	78.91
	Affinity Propagation	1172.53	60.59
	Spectral clustering using Nystrom	6447.31	62.53
Yeast	Proposed KBCHT using GTM	6.80	39.80
	Affinity Propagation	106.53	31.19
	Spectral clustering using Nystrom	2.80	20.21
Arrhythmia	Proposed KBCHT using GTM	2.13	56.94
	Affinity Propagation	13.78	13.71
	Spectral clustering using Nystrom	2.38	55.75
Dermatology	Proposed KBCHT using GTM	0.97	86.07
	Affinity Propagation	10.59	22.95
	Spectral clustering using Nystrom	1.45	58.47

For illustration purpose and easy to compare, we provide the clustering accuracy that obtained from all the used datasets by our proposed algorithm and the existing algorithm in Fig. 4.2. On these datasets our proposed algorithm achieves the highest accuracy compared with other algorithms, the results indicates the effectiveness and the strength of our proposed algorithm. From Table 4.2 and Fig 4.2, we can notice that the Affinity Propagation algorithm failed to provide result on the Pendigits dataset due to memory limitation which gives an indication that the Affinity Propagation algorithm suffers from memory difficulties.
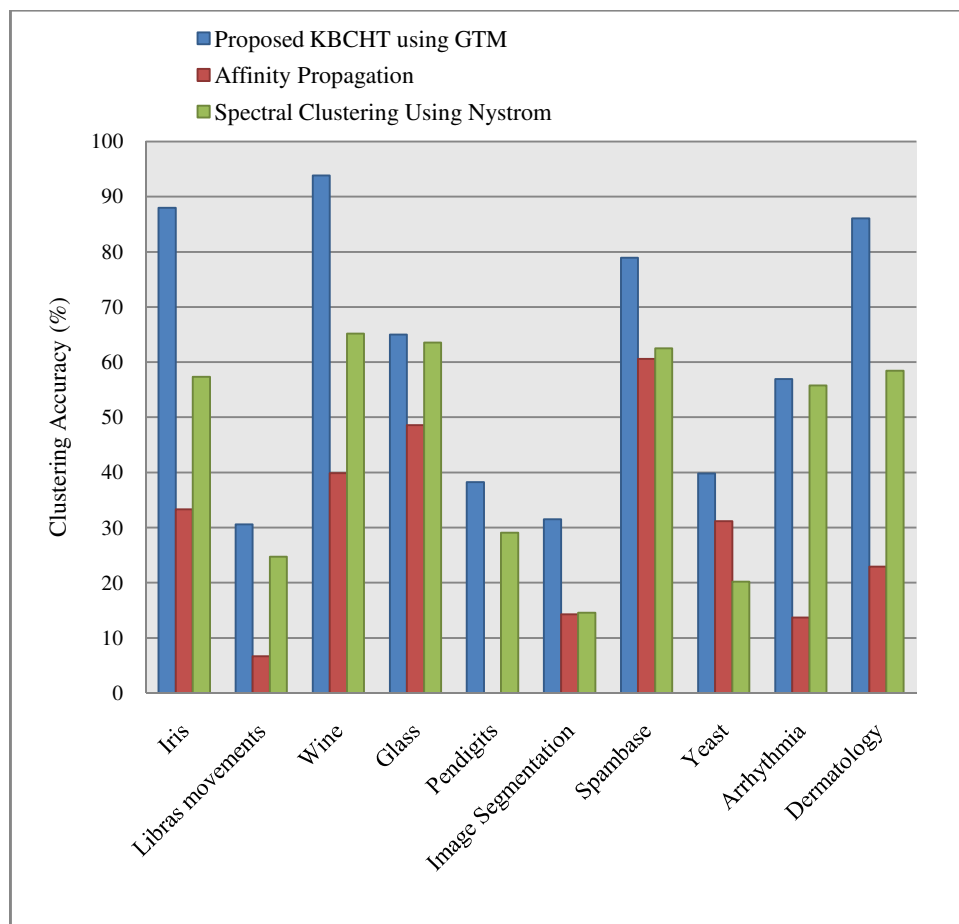
Figure 4.2: Clustering accuracy results on real datasets

Chapter **5**

CONCLUSIONS AND FUTURE RESEARCH

5.1 Conclusions and Future Research

In this thesis, we have introduced a new clustering algorithm named KBCHT (**K**means-**B**ased **C**onvex **H**ull **T**riangulation clustering algorithm) that is able to detect clusters with complex non-convex shapes, different sizes, densities, noise and outliers.

The idea behind the KBCHT is to divide the clustering task into three phases. The first phase benefit from the first run of the widely used simple Kmeans algorithm in which the experiments show that KBCHT gives good results despites it is badly chosen initial conditions. Then in the second phase KBCHT takes the initial groups from the first phase and constructs a convex hull for each group. The vertices obtained by convex hull are shrunk until we catch a set of sub-clusters in which the shrinking mechanism does not maintain the convexity of processed groups. The last phase applies merging of sub-clusters based on the Delaunay triangulation. We also have used one of the famous topology preserving mapping methods which is the Generative Topographic Mapping (GTM). The GTM have been used as a preprocessing step to our proposed KBCHT algorithm. The simulation results show the

superiority of our proposed algorithm over recently introduced ones in case of the clustering accuracy and time cost.

In future works and according to obtained results in Fig. 3.11, the third phase consumes the most time in the KBCHT since we consider all data points in each sub-clusters, thus we will do farther analysis to benefit from the sampling techniques to make it faster. While we have assumed the Euclidean distance throughout this thesis, we can use general metrics as shown in appendix C to show the effect of the used metric on our proposed algorithm. The computed value of *AVG* average from Delaunay triangulation can be recomputed each time we have a shrunk vertex according to our proposed mechanism, thus we may have more robust value. The proposed KBCHT algorithm can be implemented in a distributed environment. Also as have shown in Chapter 4, the GTM uses a fixed grid of latent points and a fixed grid of Gaussian basis function. Hence the initialization of GTM is the same despite the used data set. This may affect the training results. Thus it needs further investigations to adaptively choose the initial parameters based on the given dataset. Furthermore, in future work we can conduct more experiments that directly apply the proposed algorithm to high dimensional datasets without preprocessing the data.

Appendices

Appendix *A*

SIMULATION ENVIRONMENT

We have implemented our proposed KBCHT algorithm using MATLAB (version 7.9.0.529 (R2009b)). Furthermore, all the experiments in this thesis have been performed using 64-bit windows environment of Intel dual at 1.87 GHz with 2 GB of RAM.

In fact, MATLAB (MATrix LABoratory) is a high-level language and interactive environment that enables you to perform computationally intensive tasks faster than with traditional programming languages such as C, C++, and Fortran.

In addition, we have used the MATLAB implementation of both Affinity Propagation algorithm and Spectral clustering using Nystrom algorithm. These implementations are available from the websites (last visit on Oct. 2011):

- http://www.psi.toronto.edu/index.php?q=affinity%20propagation
- http://alumni.cs.ucsb.edu/~wychen/sc.html

Appendix *B*

DATA PREPARATION

Dealing with missing values

The collected data can be represented in a matrix notation (two modes) as in equation 1.1 of chapter 1. Hence, the missing values could be in records or columns. In record: if the most of data are missing then we delete this record. In column: if the most of data are missing then we delete this column. But if the missing values are not too much, we have to replace these missing values with:

1. The average value of all column data in which the missing occurred and then is done before running clustering algorithms.

2. By making a pre clustering algorithm:

    2.1 Make clusters based on all variables, then take average value of objects that belong to the same cluster to replace the missing values.

    2.2 To know which one variable (column) is more similar to the one with missing values. Then replace the missing values with the average value of the most similar variable.

Normalization

The attributes of the entire dataset may be collected from different scales. Hence, the large scale ones may cause the bias. So, data should be normalized

thus all attributes of the dataset would be in the same scale. Moreover, the objects of the dataset can be described by different types of attributes such as nominal, categorical, ordinal, interval or ratio. There are different approaches to cluster these objects. One of them is convert attributes of different types into attributes of the same type. A more detail of scale conversion could be found in chapter 3 of [28].

Sampling

Sampling can be used to pick a portion of large dataset because processing the entire dataset is too expensive and time consuming. The sampling may be obtained randomly which is the simplest way. But to be more effective, it should reflect an approximation of the same characteristic as the original dataset.

Dimensionality reduction

High dimensional dataset requires a large amount of time and memory to be analyzed. There are two major types of dimensionality reduction methods: feature transformation and feature selection.

In feature transformation, the original high dimensional space is projected linearly or non-linearly into a lower dimensional space.

In feature selection, this method selects a subset of meaningful dimensions from the original high dimensions of the entire dataset.

Appendix *C*

DISTANCE METRICS

In this section, we present some of the common used metrics. The Minkowski (Manhatten, Euclidean, and Maximum distances), the Mahalanobis distance and the Cosine distance.

Minkowski distance

A metric or distance is used to measure the similarity or dissimilarity between objects. While we have assumed the Euclidean distance throughout this thesis, a more general metric between two objects **X** and **Y** could be addressed. This general metric is named Minkowski metric and is defined as (*p*-norm):

$$d_p(\boldsymbol{X}, \boldsymbol{Y}) = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{1/p} , p \geq 1 \qquad (C.1)$$

Where *d* is the dimension of objects **X** ($x_1, x_2, \ldots x_d$) and **Y** ($y_1, y_2, \ldots y_d$). Thus the Euclidean distance is when $p = 2$ (2-norm). And when $p = 1$ (1-norm) d_p is called Manhattan or city block distance which is the shortest path between **X** and **Y** where each segment of the path is parallel to a coordinate axis. And when $p = \infty$ (∞-norm) is called maximum distance which is the distance between **X** and **Y** corresponds to the maximum distance between the projections of **X** and **Y** onto each of the *d* coordinate axes.

Mahalanobis distance

Mahalanobis distance removes the distance distortion caused by linear combinations of attributes. It is defined as:

$$d_{mah}(X, Y) = \sqrt{(X - Y)\Sigma^{-1}(X - Y)^T} \qquad (C.2)$$

Where Σ is the covariance matrix of the dataset. If $\Sigma = Identity$, this is the Euclidean distance.

An important property of the Mahalanobis distance is that it is invariant under nonsingular transformation. It suffers from high computation needed to compute the covariance matrix based on all objects in the dataset.

Cosine distance

Cosine distance calculates the cosine of the angle between two vectors. It can be defined as:

$$d_{cos}(X, Y) = \frac{X.Y}{|X||Y|} \qquad (C.3)$$

Where $X.Y$ is the dot product and $|.|$ is the (2-norm).

References

[1] A. Jain, M. Murty, and P. Flynn, "Data Clustering: A review," ACM Computing Surveys (CSUR), Vol. 31, issue (3), pp. 264-323, Sept. 1999.

[2] R. Duda, P. Hart, and D. Stork, "Pattern Classification," John Wiley & Sons, second edition, 2001.

[3] A. Gersho and R. Gray, "Vector Quantization and Signal Compression," Kulwer Academic, Boston, 1992.

[4] M. Al- Zoubi, A. Hudaib, A. Huneiti and B. Hammo, "New Efficient Strategy to Accelerate k-Means Clustering Algorithm," American Journal of Applied Science, Vol. 5, No. 9, pp. 1247-1250, 2008.

[5] M. Celebi, "Effective Initialization of K-means for Color Quantization," Proceeding of the IEEE International Conference on Image Processing, pp. 1649-1652, 2009.

[6] M. Borodovsky and J. McIninch, "Recognition of genes in DNA sequence with ambiguities, " Biosystems, Vol. 30, issues 1-3, pp. 161-171, 1993

[7] X. Wang, W. Qiu and R. H. Zamar "CLUES: A Non-parametric Clustering Method Based on Local Shrinking," Journal Computational Statistics & Data Analysis, Vol. 52, Issue 1, pp. 286-298, 2007.

[8] M. Khalilian, N. Mustapha, N. Suliman, and A. Mamat, "A Novel Kmeans Based Clustering Algorithm for High Dimensional Datasets," Proceedings of the International Multi Conference on Engineers and Computer Scientists (*IMECS 2010*), Vol. I, Hong Kong, March 2010.

[9] O. Oyelade, O. Oladipupo and I. Obagbuwa, "Application of Kmeans Clustering Algorithm for Prediction of Students' Academic Performance," International Journal of Computer Science and Information Security, vol. 7, no. 1, pp. 292-295, 2010.

[10] J. Bezdek and N. Pal, "Fuzzy Models for Pattern Recognition," IEEE press, New York, NY, USA, 1992.

[11] D. Karaboga and C. Ozturk, "Fuzzy Clustering with Artificial Bee Colony Algorithm," Scientific Research and Essays, Vol. 5(14), pp. 1899-1902, 2010.

[12] J. Bezdek, "Pattern Recognition with Fuzzy Objective Function Algorithms," Plenum Press, New York, NY, USA, 1981.

[13] J. Bezdek, J. Keller, R. Krishnapuram, and N. Pal, "Fuzzy Models and Algorithms for Pattern and Image Processing," Kluwer, Dordrecht, Netherland, 1999.

[14] F. Hoppner, F. Klawonn, R. Kruse, and T. Runkler, "Fuzzy Cluster Analysis," J. Wiley & Sons, Chichester, England, 1999.

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

[16] S. Guha, R. Rastogi, and K. Shim, "ROCK: A Robust Clustering Algorithm for Categorical Attributes," Information Systems, Vol. 25, No. 5, pp.345-366, 2000.

[17] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An Efficient Clustering Method for Very Large Databases," Proceedings of ACM SIGMOD Workshop Research Issues on Data Mining and Knowledge Discovery, pp. 103-114, 1996.

[18] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical Clustering Using Dynamic Modeling," IEEE Computer, Vol. 32, No. 8, pp. 68-75, 1999.

[19] G. Karypis and V. Kumar, "Multilevel Algorithms for Multi-Constraint Graph Partitioning," Technical Report, University of Minnesota, May 1998.

[20] CAO Chang-hu, and LI Ya-fei, "Improved Chameleon Algorithm for Cluster Analysis," Journal of Science Technology and Engineering, issues 33, 2010.

[21] P. Bhattacharya and M. L. Gavrilova, "CRYSTAL: A New Density Based Fast and Efficient Clustering Algorithm," Proceeding of the 3rd International Symposium on Voronoi Diagrams in Science and Engineering, pp. 102-111, 2006.

[22] A. Foss and O. Zaiane, "A Parameterless Method for Efficiently Discovering Clusters of Arbitrary Shape in Large Datasets," In IEEE International Conference on Data Mining, pp. 179-186, 2002.

[23] F. Cao, J. Liang, and G. Jiang, "An Initialization Method for Kmeans Algorithm Using Neighborhood Model," Journal Computers & Mathematics with Applications, Vol. 58, Issue 3, pp. 474-483, Aug. 2009.

[24] X. Liu and H. Fu, "An Effective Clustering Algorithm with Ant Colony," Journal of Computers, Vol. 5, No. 4, pp. 598-605, 2010.

[25] H.-J. Lin, F.-W. Yang, and Y.-T. Kao, "An Efficient GA-based Clustering Technique," Tamkang Journal of Science and Engineering, Vol. 8, No 2, pp. 113-122, 2005.

[26] T. Zhang, and H. Qu, "An Improved Clustering Algorithm," Proceedings of the Third International Symposium on Computer Science and Computational Technology (ISCSCT '10) Jiaozuo, China, pp. 112-115, Aug. 2010.

[27] V.-V. Vu, N. Labroche, and B. Bouchon-Meunier, "An Efficient Active Constraint Selection Algorithm for Clustering," 20th International Conference on Pattern Recognition, pp.2969-2972, 2010

[28] G. Gan, Ch. Ma, and J. Wu, "Data Clustering: Theory, Algorithms, and Applications," ASA-SIAM series on Statistics and Applied Probability, SIAM, 2007.

[29] D. Defays, "An Efficient Algorithm for A Complete Link Method," The Computer Journal, Vol. 20, pp. 364-366, 1977.

[30] R. Sibson, "SLINK: an Optimally Efficient Algorithm for the Single Link Cluster Method," The Computer Journal, Vol. 16, No. 1, pp. 30-34, 1973.

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

[32] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," 5th Berkeley Symp. Math. Statist. Prob., Vol. 1, pp. 281-297, 1967.

[33] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy," Advances in Knowledge Discovery and Data Mining," AAAI/MIT press, 1996.

[34]R. Xu and D. Wunsch II, "Computational Intelligence in Clustering Algorithms, With Applications," In A. Iske and J. Levesley, Eds., Algorithms for Approximation, Proceedings of 5th International Conference, Chester, England, UK, Springer, Heidelberg, pp. 31-50, 2007.

[35] XindongWu and et. Al., "Top 10 Algorithms in Data Mining," Journal of Knowledge and Information Systems, Vol. 14, Issues 1-37, 2008.

[36] P. Hansen and B. Jaumard, "Cluster analysis and mathematical programming," Mathematical Programming, pp. 191-215, 1997.

[37]R. Xu, and D. Wunsch, "Survey of clustering algorithms," IEEE Transactions on Neural Networks, pp. 645-678, 2005.

[38] B.Bahmani Firouzi, T. Niknam, and M. Nayeripour, "A New Evolutionary Algorithm for Cluster Analysis," Proceeding of world Academy of Science, Engineering and Technology, Vol. 36, Dec. 2008.

[39]N. Asgharbeygi, and A. Maleki, "Geodesic Kmeans Clustering," 19th International Conference on Pattern Recognition, 2008.

[40] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, " A Density-based Algorithm for Discovering Clusters in Large Spatial Data sets with Noise," 2nd International Conference on Knowledge Discovery and Data Mining, pp. 226-231, 1996.

[41]M. Ankerst, M. Breunig, H. P. Kriegel, and J. Sander, "OPTICS: Ordering Objects to Identify the Clustering Structure," Proceedings of ACM SIGMOD in International Conference on Management of Data, pp. 49–60, 1999.

[42]E. Forgy, "Cluster Analysis of Multivariate Data: Efficiency vs. Interpretability of Classification," Biometrics, Vol. 21, 1965.

[43] D. Hochbaum, and D. Shmoys, "A Best Possible Heuristic for the K-Center Problem," Math. Oper. Res., Vol. 10, No. 2, pp. 180-184, 1985.

[44] D. Arthur, and S. Vassilvitskii, "Kmeans++: The Advantages of Careful Seeding," Proceeding of SODA'07, pp. 1027-1035, 2007.

[45] M. Al-Daoud, "A New Algorithm for Clustering Initialization," Proceeding World Academy of Science, Engineering, and Technology, Vol. 4, 2005.

[46] W. Barbakh, and C. Fyfe, "Local vs. Global Interactions in Clustering Algorithms: Advances over Kmeans," International Journal of knowledge-based and Intelligent Engineering Systems, Vol. 12, 2008.

[47] Jim Z.C. Lai, and T. J. Huang, "Fast Global Kmeans Clustering Using Cluster Membership and Inequality," Pattern Recognition, Vol. 43, pp. 1954-1963, 2010.

[48] G. Frahling, and Ch. Sohler, "A Fast Kmeans Implementation Using Coresets," International Journal of Computational Geometry and Applications, Vol. 18, Issue 6, pp. 605-625, 2008.

[49] L. Taoying, and Y. Chen, "An Improved Kmeans for Clustering Using Entropy Weighting measures," 7th World Congress on Intelligent Control and Automation, 2008.

[50] S. Gupara, K. Rao, and V. Bhatnagar, "Kmeans Clustering Algorithm for Categorical Attributes," Proceeding 1st International Conf. on Data Warehousing and Knowledge Discovery, pp. 203-208, Italy, 1999.

[51] Z. Huang, "Extensions to The Kmeans Algorithms for Clustering Large Data Sets with Categorical Values," Data Mining & Knowledge Discovery, Vol. 2, pp. 283-304, 1998.

[52] W. Barbakh and C. Fyfe. "Inverse Weighted Clustering Algorithm," Computing and Information Systems, 11(2), pp. 10-18, May 2007. ISSN 1352-9404.

[53] W. Barbakh. "The Family of Inverse Exponential Kmeans Algorithms," Computing and Information Systems, 11(1), pp. 1-10, February 2007. ISSN 1352-9404.

[54] W. Barbakh, and C. Fyfe, "Clustering and Visualization with Alternative Similarity Functions." The 7th WSEAS international conference on artificial intelligence, knowledge, engineering and data bases, AIKED'08, pp. 238–244. University of Cambridge, UK. 2008.

[55] A. K. Jain, "Data clustering: 50 years beyond K-means." Pattern Recognition Letters, Vol. 31, No. 8, pp. 651-666, 2010.

[56] B. Borah, and D.K. Bhattacharyya, "DDSC: A Density Differentiated Spatial Clustering Technique." Journal of Computers, Vol. 3, No. 2, pp. 72-79, 2008.

[57] A. Ram, S. Jalal, A. S. Jalal, and M. Kumar, "A Density Based Algorithm for Discovering Density Varied Clusters in Large Spatial Databases." International Journal of Computer Applications, Vol. 3, No. 6, June 2010.

[58] K. Mumtaz and K. Duraiswamy "A Novel Density Based Improved Kmeans Clustering algorithm – Dbkmeans," International Journal on Computer Science and Engineering (IJCSE), Vol. 2, No. 2, pp. 213-218, 2010.

[59] A. Fahim, A. Salem, F. Torkey, and M. Ramadan, "Density Clustering Based on Radius of Data (DCBRD)," International Journal of Mathematical and Computer Sciences, Vol. 3 No. 2, pp. 80-86, 2007.

[60] S. Kisilevich, F. Mansmann, and D. Keim, "P-DBSCAN: A Density Based Clustering Algorithm for Exploration and Analysis of Attractive Areas of Geo-tagged Photos," Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research & Application, Washington, D.C., 2010

[61] J.J Sheu, W.M Chen, W.B Tsai and K.T Chu, "An Intelligent Initialization Method for the Kmeans Clustering Algorithm." International Journal of Innovative Computing, Information and Control, Vol. 6, No. 6, pp. 2551-2566, June 2010.

[62] Convex Hull, "http://en.wikipedia.org/wiki/Convex_hull," last visit: June, 2011.

[63] Delaunay triangulation, "http://en.wikipedia.org/wiki/Delaunay_triangulation," last visit: June, 2011.

[64] G. Nalbantov, P. Groenen, J. Bioch, "Nearest Convex Hull Classification," Erasmus University Rotterdam, Econometric Institute in its series Econometric Institute Report, Dec, 2006. (http://repub.eur.nl/publications/index/728919268/NCH10.pdf)

[65] X. Zhou, and Y. Shi, "Nearest Neighbor Convex Hull Classification Method for Face Recognition," Proceedings of the 9th International Conference on Computational Science, pp. 570-577, 2009.

[66] J. Hershberger, N. Shrivastava, and S. Suri, "Summarizing Spatial Data Streams Using ClusterHulls," Journal of Experimental Algorithmics (JEA), Vol. 13, Feb. 2009.

[67] B. J. Frey, and D. Dueck, "Clustering by Passing Messages Between Data Points," Science, Vol. 315, pp. 972-949, 2007.

[68] E. Hartuv and R. Shamir, "A Clustering Algorithm Based on Graph Connectivity," Information Processing Letters, Vol. 76, Nos. 4-6, pp. 175-181, 2000.

[69] M. Fielder, "A Property of Eigenvectors of Nonnegative Symmetric Matrices and Its Application to Graph Theory," Czechoslovak Mathematical Journal, Vol. 25, No. 100, pp.619-633, 1975.

[70] U. Luxburg, "A Tutorial on Spectral Clustering," Statistics and Computing, Vol. 17, No. 4, pp. 395-416, 2007.

[71] X. Zhang, J. Li, and H. Yu, "Local Density Adaptive Similarity Measurement for Spectral Clustering," Pattern Recognition Letters, 32, pp. 352-358, 2011.

[72] J. Santos, J. Marques de Sa, and L. Alexandre, "LEGClust: A Clustering Algorithm Based on Layered Entropic Subgraphs," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 30, No. 1, pp. 62-75, 2008.

[73] A. Ny, M. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an Algorithm," Advances in Neural Information Processing Systems, Vol. 14, 2001.

[74] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 8, pp. 888-905, 2000.

[75] D. Verma and Meila, "A Comparison of Spectral Clustering Algorithms," Technical Report UW-CSE-03-05-01, Washington University, 2003.

[76] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, E. Chang, "Parallel Spectral Clustering in Distributed Systems," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 33, No. 3, pp. 568–586, 2011.

[77] M. Wu and B. Scholkopf, "A Local Learning Approach for Clustering," Proceedings of NIPS, pp. 1529–1536, 2007.

[78] C. Papadimitriou and K. Steiglitz, "Combinatorial Optimization: Algorithms and Complexity," Dover, New York, 1998.

[79] T. Liu, C. Rosenberg and H. Rowley, "Clustering Billions of Images with Large Scale Nearest Neighbor Search," Proceedings of the Eighth IEEE Workshop on Applications of Computer Vision (WACV), 2007.

[80] K. Akkaya, F. Senel and B. McLaughlan, "Clustering of Wireless Sensor and Actor Networks Based on Sensor Distribution and Connectivity," Journal of Parallel and Distributed Computing, Vol. 69, No. 6, pp. 573-587, 2009.

[81] L. Portnoy, E. Eskin and S. Stolfo, "Intrusion Detection with Unlabeled Data using Clustering," In Proceedings of ACM CSS Workshop on Data Mining Applied to

Security (DMSA), pp. 5-8, 2001.

[82] Cluster Analysis, "http://en.wikipedia.org/wiki/Cluster_analysis," last visit: July, 2011.

[83] L. van der Maaten, E. Postma, and H. van den Herik, "Dimensionality reduction: A Comparative Review," Technical Report, MICC, Maastricht University, The Netherlands, 2007.

[84] T. Khonen, "Self-Organizing Maps," Springer, 1995.

[85] C. Bishop, M. Svensen, and C. Williams, "GTM: The Generative Topographic Mapping," Neural Computation, Vol. 10, No. 1, pp. 215–234, 1998.

[86] C. Bishop, M. Svensen, and C. Williams, "Developments of the Generative Topographic Mapping," Neurocomputing, vol. 21, No. 1, pp. 203-224, 1998.

[87] J.Y. Choi, J. Qiu, M.E. Pierce, and G. Fox, "Generative Topographic Mapping by Deterministic Annealing," In Proceedings of International Conference on Membrane Computing, pp.47-56, 2010.

[88] UCI Machine Learning Repository, Available from (http://archive.ics.uci.edu/ml). Last visit: Sep., 2011.

[89] Minimum Distance between a Point and a Line, Available from (http://paulbourke.net/geometry/pointline/). Last visit: Sep., 2011.

[90] Convex Hull Geometric, Available from (http://www.cse.unsw.edu.au/ ~lambert/java/3d/triangle.html). Last visit: Sep., 2011.

[91] C. Zhang and S. Xia, "K-means Clustering Algorithm with Improved Initial Center," Second International Workshop on Knowledge Discovery and Data Mining, pp. 790-792, 2009.

[92] D. Pham, S. Dimov, and C. Nguyen, "Selection of K in K-means Clustering," Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, Vol. 219, No. 1, pp. 103-119, 2005.

[93] G. Yu, H. Peng, J. Wei, and Q. Ma, "Enhanced Locality Preserving Projections using Robust Path Based Similarity," Neurocomputing, Vol. 74, No. 4, pp. 598-605, 2011.

[94] R. Harrison, and K. Pasupa, "Sparse Multinomial Kernel Discriminant Analysis (sMKDA)," Pattern Recognition, Vol. 42, no. 9, pp. 1795-1802, 2009.

[95] L. Zhang, L. Qiao, and S. Chen, "Graph-optimized Locality Preserving Projections," Pattern Recognition, Vol. 43, No. 6, pp. 1993-2002, 2010.

[96] H. Liu, J. Sun, L. Liu, and H. Zhang, "Feature Selection with Dynamic Mutual Information," Pattern Recognition, Vol. 42, No. 7, pp. 1330-1339, 2009.