CAIMING ZHONG

Improvements on Graph-based Clustering Methods

Publications of the University of Eastern Finland
Dissertations in Forestry and Natural Sciences
Number 114

Academic Dissertation
To be presented by permission of the Faculty of Science and Forestry for public examination in Louhela Auditorium in Science Park at the University of Eastern Finland, Joensuu, on August 23, 2013, at 12 o'clock noon.

School of Computing

Kopijyvä Oy
Joensuu, 2013
Editors: Profs. Pertti Pasanen, Pekka Kilpeläinen, Kai Peiponen, and Matti Vornanen

Distribution:
University of Eastern Finland Library / Sales of publications
P.O. Box 107, FI-80101 Joensuu, Finland
tel. +358-50-3058396
http://www.uef.fi/kirjasto

ISBN: 978-952-61-1181-0 (printed)
ISSNL: 1798-5668
ISSN: 1798-5668
ISBN: 978-952-61-1182-7 (pdf)
ISSNL: 1798-5668
ISSN: 1798-5676

Author's address: University of Eastern Finland
School of Computing
P.O.Box 80101
80100 Joensuu
FINLAND
email: czhong@cs.uef.fi

Supervisors: Professor Pasi Fränti, Ph.D.
University of Eastern Finland
School of Computing
P.O.Box 801
80100 Joensuu
FINLAND
email: franti@cs.uef.fi

Reviewers: Dr. Samuel Rota Bulò, Ph.D
University of Venice
Dipartimento di Scienze Ambientali, Informatica e Statistica
Via Torino 155
30172 Venezia Mestre
ITALY
email: srotabul@dsi.unive.it

Professor Martti Juhola, Ph.D
University of Tampere
Department of Computer Sciences
P.O.Box 33014
Tampere
FINLAND
email: csmajuh@sis.uta.fi

Opponent: Professor Tommi Kärkkäinen, Ph.D
University of Jyväskylä
Department of Mathematical Information Technology
P.O.Box 35 (Agora)
FIN-40014 University of Jyväskylä
email: tommi.karkkainen@jyu.fi

ABSTRACT

Clustering is one of the common fundamental problems in the fields of pattern recognition, machine learning and data mining. Although many clustering algorithms exist in the literature, new ones are constantly being proposed, because a unified algorithm which can deal with all kinds of data sets does not exist. As a result, users have to select the most suitable algorithm from many candidates to achieve accurate results. However, users often have no a priori knowledge about their datasets. This is a dilemma for users, as the algorithm can only be selected with prior knowledge.

To alleviate the dilemma to some extent, clustering algorithms capable of handling diversified data sets are proposed. The proposed algorithms focus on graph-based data representation and clustering components in their own design.

For the graph-based representation, a *minimum spanning tree* (MST) and its extensions are employed, because they may imply the structure of the data. Distance measure is a crucial component of a clustering algorithm, and a suitable distance measure can reveal hidden information about the cluster structure. The mechanism is another important component that determines how the information described by the data representations or distance measures is used for cluster analysis.

This thesis has four main contributions. The first one is a new dissimilarity measure and a divisive hierarchical clustering algorithm utilizing this measure. The second one is a clustering method based on two rounds of MSTs, employed to represent the data and to disclose the cluster structures. The third one is a split-and-merge based clustering method which takes advantage of multiple rounds of MSTs to split the data set into a number of subsets, and then combines the similar ones. The last contribution is a fast and approximate MST algorithm applicable for large data sets in a Euclidean space.

Keywords: cluster analysis, similarity measure, data representation, clustering mechanism, algorithm selection

AMS Mathematics Subject Classification: 62H30, 68P05, 68R10, 68W01, 91C20
Universal Decimal Classification: 004.021, 004.421, 004.424.6, 004.62, 004.93
Library of Congress Subject Headings: Cluster analysis; Data structures (Computer science); Graph theory; Trees (Graph theory); Automatic classification; Algorithms

Yleinen suomalainen asiasanasto: tietojenkäsittely; tietorakenteet; luokitus; graafit; algoritmit

Acknowledgement

I met Prof. Pasi Fränti in January 2009. This is the most significant meeting of my research life, self-organized by our common research interest. Academically, this self-organizing process can be called clustering. Amazingly, our common research interest is this very clustering. So, clustering is really a magical and ubiquitous technique.

The majority of the work presented in this thesis was undertaken from September 2009 to August 2013. In this period, during the semesters I was a lecturer in Ningbo University, China and, in the semester breaks, undertook research as a PhD student in the School of Computing in the University of Eastern Finland.

First of all, I would like to thank my supervisor Prof. Pasi Fränti. Although I was in Joensuu only during the semester breaks over the past four years, the hundreds of emails that he wrote to me witnessed his all-round guidance on my research. More important, his attitude toward work exerts a subtle influence on me. I am grateful for the constant support and help of the colleagues in SIPU, even though my visits to the lab were brief each time.

I also acknowledge the contributions of all the other co-authors: Mr. Mikko Malinen, Prof. Duoqian Miao, Dr. Ruizhi Wang and Dr. Xinmin Zhou. I would not have finished the work without your cooperation.

I would like to thank the funding parties, CIMO, NSFC (No. 61175054), their support helped me to focus on my research.

Finally, I would like to thank my wife Chunhua and daughter Wendy for their endless understanding and support when I left for Joensuu and could not take care of them.

Ningbo May 5, 2013 *Caiming Zhong*

LIST OF ORIGINAL PUBLICATIONS

This thesis is based on data presented in the following papers. Throughout the thesis, these papers will be referred to as [**P1**] – [**P5**].

[**P1**] **C. Zhong**, D. Miao, R. Wang, X. Zhou. "DIVFRP: An automatic divisive hierarchical clustering method based on the furthest reference points," *Pattern Recognition Letters*, Volume 29, Issue 16, Pages 2067–2077, December 2008.

[**P2**] **C. Zhong**, D. Miao. "A comment on 'Using locally estimated geodesic distance to optimize neighborhood graph for isometric data embedding'," *Pattern Recognition*, Volume 42, Issue 5, Pages 1012–1013, May 2009.

[**P3**] **C. Zhong,** D. Miao, R. Wang. **"**A graph-theoretical clustering method based on two rounds of minimum spanning trees," *Pattern Recognition*, Volume 43, Issue 3, Pages 752–766, March 2010.

[**P4**] **C. Zhong,** D. Miao, P. Fränti. "Minimum spanning tree based split-and-merge: A hierarchical clustering method," *Information Sciences*, Volume 181, Issue 16, Pages 3397–3410, August 2011.

[**P5**] **C. Zhong**, M. Malinen, D. Miao. P. Fränti. "A fast minimum spanning tree algorithm based on K-means," *Manuscript* (*Submitted*).

The above papers have been included at the end of the thesis with permission of their copyright holders.

AUTHOR'S CONTRIBUTION

The five papers are results of team work with joint efforts by all authors. The order of authors roughly indicates the contribution in preparing the papers, and the first author has been the principle author responsible for editing the text. In all cases, Caiming Zhong's contribution has been independent and significant.

In **P1**, Caiming Zhong contributed to the main idea of the similarity measure based on the furthest reference point, and made all implementations and experiments, and wrote the paper. Duoqian Miao improved the main idea, Ruizhi Wang contributed to writing part of the introduction, and Xinmin Zhou advised in experiments.

In **P2**, Caiming Zhong detected the defect of the paper commented. Duoqian Miao advised in writing.

In **P3**, the idea originated from Caiming Zhong, who carried out the study by making all the implementations, experiments and wrote the paper. Duoqian Miao and Ruizhi Wang advised in experiments and writing.

In **P4**, Caiming Zhong contributed to the framework of the hierarchical clustering based on split-and-merge, made all implementations and experiments, and wrote the paper. Pasi Fränti advised for improving the framework and the presentation, while Duoqian Miao advised in the experiments.

In **P5**, the idea was originally proposed and implemented by Caiming Zhong. Mikko Malinen contributed to the theoretical analysis. Pasi Fränti's contribution was improvement of the idea, experiments and writing. Duoqian Miao advised in the algorithm framework. Caiming Zhong wrote the paper.

Contents

1 Introduction

Clustering (or cluster analysis) is one of the most common problems in pattern recognition and machine learning. The goal, without any a priori knowledge, is to partition a data set into different groups so that the data points within a group are similar while those among different groups are dissimilar. In the more than 50 years since one of the classical clustering algorithms K-means was proposed [1], researchers have been being interested in the study of clustering, and new algorithms have been being presented continuously in the literature.

An essential problem of pattern recognition is to classify the objects to be analyzed. If the analysis is based on a priori knowledge about the objects, it is called *classification*, otherwise, *clustering*. For example, for a given set of images of human facial expressions, each has been labeled as happy, angry, or sad. For a new unlabeled image of the same type, one can determine its label according to the labeled image set. This process is classification. If the set of images have no labels available, the process of classifying these images into subsets is clustering.

The purpose of machine learning is to design algorithms that enable computers to learn how to make decisions. When the algorithms work with empirical data, the learning process is called *supervised learning*, while when no empirical data is available, the learning is called *unsupervised learning*.

1.1 CONCEPT OF CLUSTERING

So far, there has been no standard definition of a cluster. Although Jain and Dubes have given a functional definition in [2], a cluster is a group of similar objects, they point out that even for an identical data set, different users probably have different motivations, and therefore, it is difficult to give an operational definition.

Everitt [3] describes a cluster from the following three viewpoints:

1. A cluster is a set of entities that are alike, and entities from different clusters are not alike.
2. A cluster is an aggregation of points in the test space such that the distance between any two points in the cluster is less than the distance between any point in the cluster and any point outside of the cluster.
3. Clusters may be described as connected regions of a multi-dimensional space containing a relatively high density of points, separated from other such regions by a region containing a relatively low density of points.

The last two definitions are more operational than the first one. Since one cannot universally and operationally define a cluster, there exist a diversity of clustering algorithms in the literature.

The formulization of the clustering problem can be described as follows. For a given data set, $\mathbf{X}=\{\mathbf{x}_1,\ldots,\mathbf{x}_i,\ldots,\mathbf{x}_N\}$, where x_{ij} is the j^{th} attribute of data point $\mathbf{x}_i$, clustering is used to find a partition of $\mathbf{X}$, $C=\{C_1,\ldots,C_i,\ldots,C_K\}$, where $K\leq N$, such that:

1. $Ci \neq \varnothing, i = 1,\cdots,K$
2. $\bigcup_{i=1}^{K} C_i = \mathbf{X}$
3. $C_i \cap C_j = \varnothing$, $i, j = 1,\cdots,K$ and $i \neq j$.

A partition that satisfies the above three conditions is called a hard partition, and each data point belongs to one and only one cluster. A hard partition is usually generated by a traditional clustering approach, while a fuzzy clustering approach extends the notion of a hard partition by introducing a membership matrix **U,** of which an element u_{ij} indicates that to what extent $\mathbf{x}_i$ belonging to C_K [4]. In this dissertation we only focus our study on hard partitions.

In general, the design of a clustering algorithm consists of the components shown in Figure 1.1.

The feature extraction component determines the features of the data from which the distance or similarity between a pair of data points is computed. The data representation component transforms the data into a form that can discover the hidden information of the cluster structure. The distance measure component defines the similarity of any pair of data points, while the clustering mechanism component partitions the data set into subsets with the measure. The cluster validity index is employed to evaluate the partition and the result interpretation unit

translates the clustering into knowledge. This dissertation mainly focuses on data representation, distance measure and clustering mechanism.

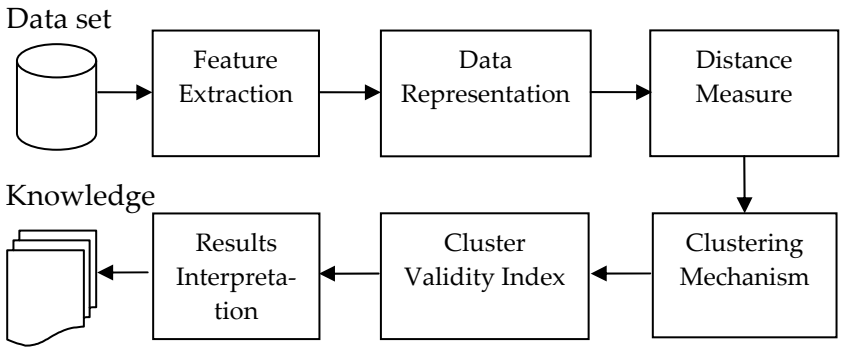


Figure 1.1 *Components of cluster analysis*

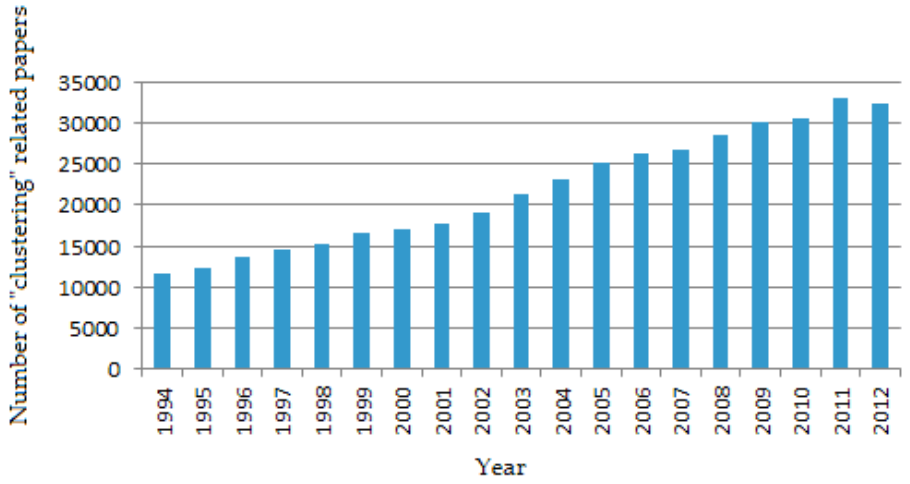


Figure 1.2. *Number of papers indexed by Web of Science with keyword clustering from 1994 to 2012. The three searched databases were Science Citation Index Expanded, Social Sciences Citation Index and Arts & Humanities Citation Index.*

1.2 CATEGORIES OF CLUSTERING ALGORITHMS

Clustering algorithms have been increasing continuously in the literature during the last two decades. In Figure 1.2, the annual numbers of published papers indexed by Web of Science are displayed.

These indicate that cluster analysis is an important research topic. These algorithms can be categorized roughly into two groups: *hierarchical clustering* and *partitional clustering* [4].

Hierarchical clustering algorithms can be further divided into *divisive* [4–8] and *agglomerative* [9–12]. A divisive algorithm begins by taking the whole data set as a cluster, repeatedly selecting a cluster and bisecting it until the number of clusters is as defined. An agglomerative algorithm performs the clustering in the opposite way: It considers one data point as a cluster, and iteratively combines two clusters until the specified number of clusters is achieved.

Partitional clustering can be based on *sum-of-squared error*, a *graph* or a *model*. A typical sum-of-squared error based clustering is K-means. It randomly selects K data points as the initial cluster centers, repeatedly assigns each data point to its nearest center to form K clusters and re-computes the cluster centers until the centers are fixed. Graph-based algorithms represent the data set as a graph, where a vertex denotes a data point and the weight of an edge denotes the similarity between the two points connected by the edge. Then a graph cut, such as *normalized cut*, is applied to cut the graph into sub-graphs, and one sub-graph is a cluster. Well-known algorithms of this kind are [13–15]. Model-based clustering describes the data set with some mathematical models [16–18], for example, Gaussian distributions.

1.3 APPLICATIONS OF CLUSTERING

In computer science, some data analysis tasks, such as image segmentation [19, 20], speaker diarization [21, 22] and text analysis [23, 24], involve clustering. Image segmentation is a fundamental problem in computer vision where the task is to partition an image into several homogeneous regions. If a region is viewed as a cluster, then the segmentation is a clustering. An image segmentation example is shown in Figure 1.3. In speaker diarization, the task is to cluster speech segments coming from the same speaker. Text clustering is used to categorize documents into different groups according to their content. For example, the results of a search engine on a given keyword can be

clustered so that users are given the overview of the results. Figure 1.4 shows the visualization of the search results [25].

Clustering is frequently used in biology and medicine; for example, gene expression data analysis [26–28], protein structure analysis [29, 30], gene function prediction [31, 32], protein function prediction [33], and diagnosis and treatment of diseases [34, 35]. The clustering of gene expression data can be categorized into three groups:

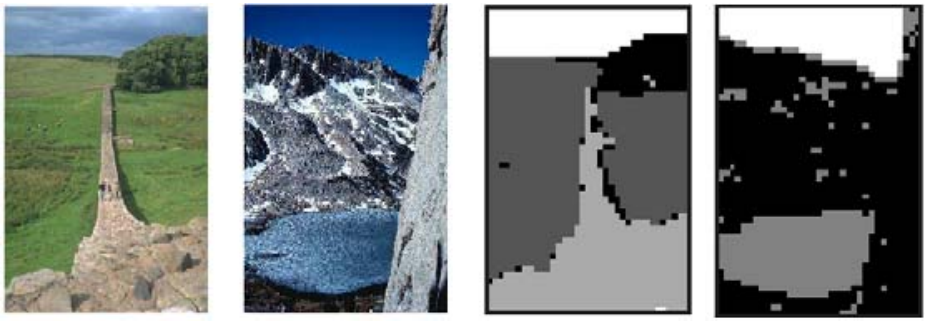


Figure 1.3. *Application of clustering on image segmentation. The left two are the original images, and the right two are the corresponding segmentations.*

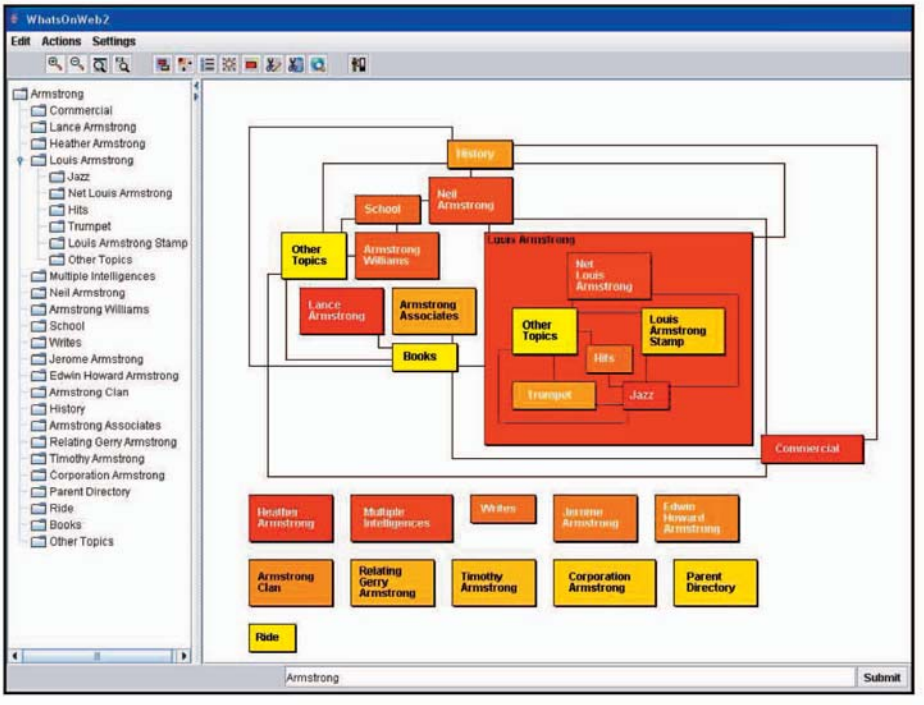


Figure 1.4. *Visualization of clustering a search result given the keyword Armstrong.*

1. If genes are taken as the data points and samples (conditions for measuring the genes) are viewed as the features of the data points, then clustering of the genes can discover those with a similar expression pattern;

2. If the samples are taken as the data points and the genes are viewed as the features, then clustering of the samples can discover the phenotype;

3. Simultaneous clustering of the genes and the samples can discover the genes involved in regulation and the association conditions.

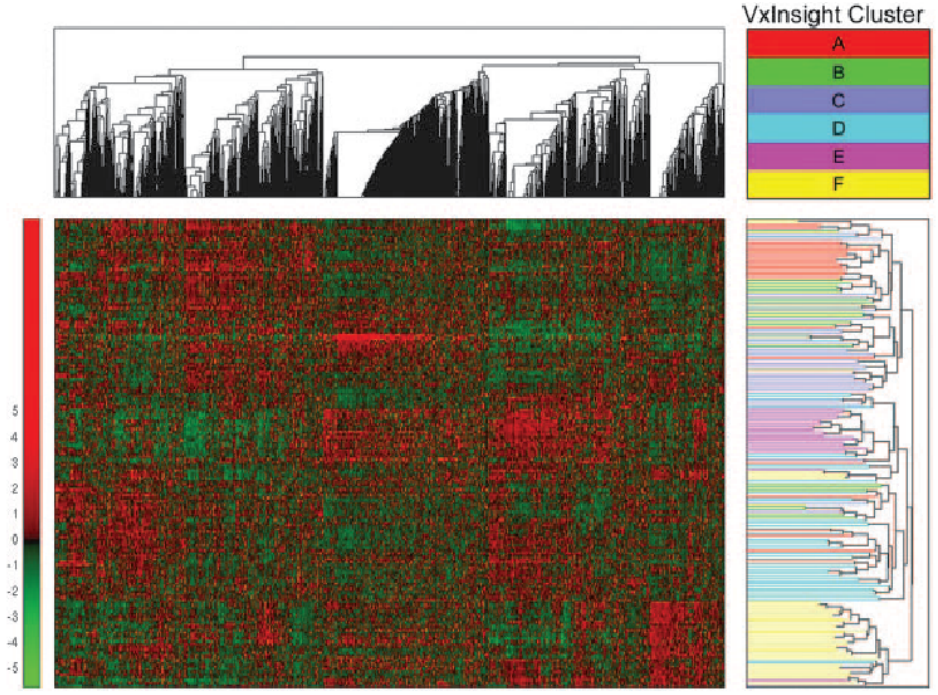


Figure 1.5. *Application of clustering on gene expression data.*

In Figure 1.5, a heat map based clustering on gene expression data is shown [36].

Also, clustering has long been employed in astronomy. The earliest related studies date back to theses published by the Royal Astronomical Society more than a hundred years ago. The main applications consist of the separation of stars and galaxies [37], the morphology of galaxy clustering [38] and young stellar clusters [39].

Other applications include analysis of commercial data, such as market structure analysis [40, 41], financial time series analysis [42, 43] and complex networks [44, 45].

1.4 STRUCTURE OF THE THESIS

The rest of the thesis is organized as follows: In Chapter 2 distance measures are discussed and a new divisive clustering is introduced. Data representation for clustering is discussed and an MST-based (minimum spanning tree based) clustering algorithm is introduced in Chapter 3. Chapter 4 analyzes the clustering mechanism and presents a split-and-merge based clustering algorithm. Chapter 5 describes a fast and approximate MST algorithm. The contributions of the thesis are summarized in Chapter 6 and conclusions are drawn in Chapter 7.

2 Distance Measure

2.1 TYPICAL DISTANCE MEASURES

In pattern recognition and machine learning, a distance measure, or alternatively a similarity/dissimilarity measure, is used to measure how similar is a pair of data points. In supervised and semi-supervised learning, the distance measure can be learned from labeled examples. For example, Weinberger and Saul showed how to learn a Mahalanobis distance metric for K nearest neighbor (KNN) classification from labeled examples [46]. In clustering, however, no labeled example is available, and the selection of a suitable distance measure is crucial for a clustering. Most typical measures are shown in Table 2.1.

Table 2.1 Classical Distance/Similarity Measures

Distance	Formulation	Description
Euclidean	$D(\mathbf{x}_i,\mathbf{x}_j)=\left(\sum_{l=1}^{d}\left\|x_{il}-x_{jl}\right\|^2\right)^{1/2}$	d is the dimensionality. It is the most used measure.
Minkowski	$D(\mathbf{x}_i,\mathbf{x}_j)=\left(\sum_{l=1}^{d}\left\|x_{il}-x_{jl}\right\|^p\right)^{1/p}$	It is the extension of Euclidean distance. If $0<p<1$, it is called fractional distance, which is more meaningful than Euclidean for high dimensional data [47].
Manhattan	$D(\mathbf{x}_i,\mathbf{x}_j)=\sum_{l=1}^{d}\left\|x_{il}-x_{jl}\right\|.$	It is also called City Block, and is the special case of p=1 of Minkowski distance. It is usually used in subspace clustering [48].

Distance	Formulation	Description
Mahalanobis	$D(\mathbf{x}_i,\mathbf{x}_j)=\sqrt{(\mathbf{x}_i-\mathbf{x}_j)^T\mathbf{S}^{-1}(\mathbf{x}_i-\mathbf{x}_j)}$	It is the weighted Euclidean distance [49], namely, each feature has a different contribution to the distance, and **S** is the covariance matrix of the data set distribution.
Pearson Correlation	$D(\mathbf{x}_i,\mathbf{x}_j)=1-r_{ij}$, $r_{ij}=\frac{\sum_{l=1}^{d}(x_{il}-\bar{\mathbf{x}}_i)(x_{jl}-\bar{\mathbf{x}}_j)}{\sqrt{\sum_{l=1}^{d}(x_{il}-\bar{\mathbf{x}}_i)^2(x_{jl}-\bar{\mathbf{x}}_j)^2}}$, $\bar{\mathbf{x}}_i=\frac{1}{d}\sum_{l=1}^{d}x_{il}$	It is widely used in gene expression analysis. Euclidean distance is sensitive to the average gene expression level, while Pearson Correlation is more robust [50].
Cosine	$S(\mathbf{x}_i,\mathbf{x}_j)=\frac{\mathbf{x}_i^T\mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|}$	Cosine distance is usually applied to text clustering [51].

In addition, in the literature, some new distance measures have been proposed for clustering, such as point symmetry distance [52] and path-based distance (which is also called minimax distance) [53–56]. For manifold learning, other distances such as geodesic distance have been used.

2.2 POINT SYMMETRY DISTANCE

For data set $\mathbf{X}=\{\mathbf{x}_1,\ldots,\mathbf{x}_i,\ldots,\mathbf{x}_N\}$, point symmetry distance is defined as [52]:

$$D(\mathbf{x}_i,\mathbf{c})=\min_{1\le l\le N,l\ne i}\frac{\|(\mathbf{x}_i-\mathbf{c})+(\mathbf{x}_l-\mathbf{c})\|}{(\|\mathbf{x}_i-\mathbf{c}\|+\|\mathbf{x}_l-\mathbf{c}\|)} \tag{2.1}$$

where, $\mathbf{c}$ is a reference point (for example, the center of a cluster), $D(\mathbf{x}_i, \mathbf{c})$ denotes the distance between $\mathbf{x}_i$ and $\mathbf{c}$. The idea of the distance is to detect whether a symmetrical point of $\mathbf{x}_i$ with respect to $\mathbf{c}$ exits. If the answer is yes, then $D(\mathbf{x}_i, \mathbf{c}) = 0$. In K-means, Euclidean distance measure is used to compute the distance between a data point to a cluster center. If Euclidean distance is replaced by the point symmetry distance, a partition of $\mathbf{x}_i$ will be determined by the closest symmetrical point of $\mathbf{x}_i$ with respect to the cluster centers. The point symmetry distance is demonstrated in Figure 2.1.

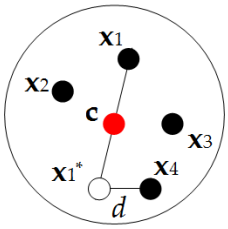


Figure 2.1. *Example of point symmetry distance. In data set* $\mathbf{X}=\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$ *the symmetry point of* $\mathbf{x}_1$ *in respect of cluster center* $\mathbf{c}$*, say* $\mathbf{x}_1^*$*, does not exist, while the nearest point of* $\mathbf{x}_1^*$ *is* $\mathbf{x}_4$*, and then the Euclidean distance between* $\mathbf{x}_4$ *and* $\mathbf{x}_4^*$ *is the numerator in Equation 2.1.*

The motivation of the point symmetry distance is based on the following observation [52]: In the real world, many objects are symmetrical, for instance, a human face, a cube, and so forth. Consequently, a cluster generally possesses symmetry. Su and Chou [52] combined the point symmetry distance with K-means and achieved good results on face detection.

However, the point symmetry distance has a drawback: Data points from different clusters may have small distances, which will lead to bad partitions (see Figure 2.2). Bandyopadhyay and Saha [53] improved the point symmetry distance as follows:

$$D(\mathbf{x}_i, \mathbf{c}) = \frac{(d_1 + d_2)}{2} \times D_e(\mathbf{x}_i, \mathbf{c}) \tag{2.2}$$

where d_1 and d_2 are distances to the nearest and the second nearest points of $\mathbf{x}_i$ with respect to $\mathbf{c}$, and D_e is the Euclidean distance of $\mathbf{x}_i$ and $\mathbf{c}$. The purpose of introducing d_1 and d_2 into Equation 2.2 is to avoid zero distance, while D_e indicates that $\mathbf{x}_i$ prefers to select a center relatively near.

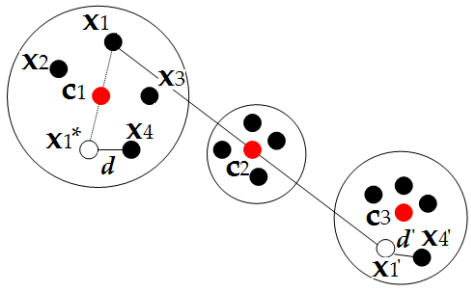


Figure 2.2. *A drawback of the point symmetry distance. Although* $\mathbf{x}_1$ *can find* $\mathbf{x}_4$ *via* $\mathbf{x}_1^*$ *according to the center of* $\mathbf{c}_1$*, it can also find* $\mathbf{x}_4'$ *via* $\mathbf{x}_1'$ *according to the center of* $\mathbf{c}_2$ *.*

From the design of the point symmetry distance and its improved version, one may perceive that the distance measure might disclose the hidden symmetry properties of points in the same cluster, which can benefit the clustering.

2.3 PATH-BASED DISTANCE

Path-based distance was proposed and successfully used to clustering by Fischer [54]. Then Chang and Yeung extended the distance in [55].

Suppose $G(V,E)$ is a complete graph of $\mathbf{X}=\{\mathbf{x}_1,\cdots,\mathbf{x}_i,\cdots,\mathbf{x}_N\}$, the N vertices correspond to the data points, and the pairwise edge weights correspond to the Euclidean distances. The path-based distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is defined as:

$$D_{ij}=\min_{p\in P_{ij}}\left(\max_{1\le i<|p|} d(p[i],p[i+1])\right) \tag{2.3}$$

where P_{ij} is the set of all paths from x_i to x_j on $G(V,E)$, and $d(p[i],p[i+1])$ is the Euclidean distance of ith edge of path p. Figure 2.3 shows an example. Suppose that three paths exist between vertices $\mathbf{x}_i$ and $\mathbf{x}_j$, and the maximum edge weights in the three paths are 5, 7 and 6, respectively. Then the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ is D_{ij}=min(5,6,7)=5.

The distance possesses transitivity, as illustrated in the two examples in Figure 2.4, and each has two clusters. According to Euclidean distance, $\mathbf{x}_1$ is near to $\mathbf{x}_3$ and $\mathbf{x}_2$ is near to $\mathbf{x}_5$, but according to path-based distance, $\mathbf{x}_1$ is near to $\mathbf{x}_4$ and $\mathbf{x}_2$ is near to $\mathbf{x}_6$. Intuitively, the two data sets can be well analyzed by a neighbor discriminant criterion:

"*My neighbor's neighbor can be also my neighbor*" [56], namely, the relationship of neighborhood is transitive. The path-based distance has this transitivity.

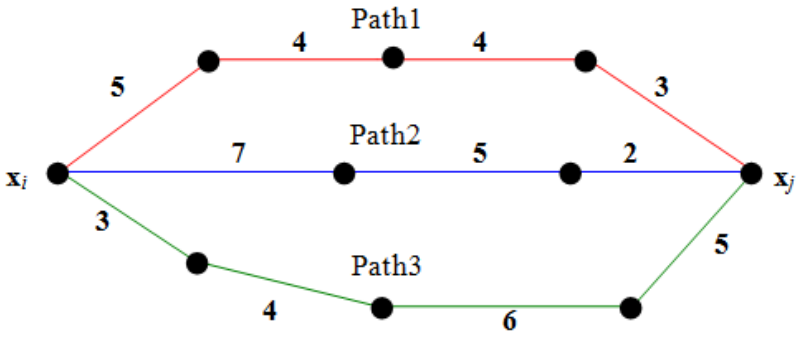


Figure 2.3. *An example of path-based distance.*

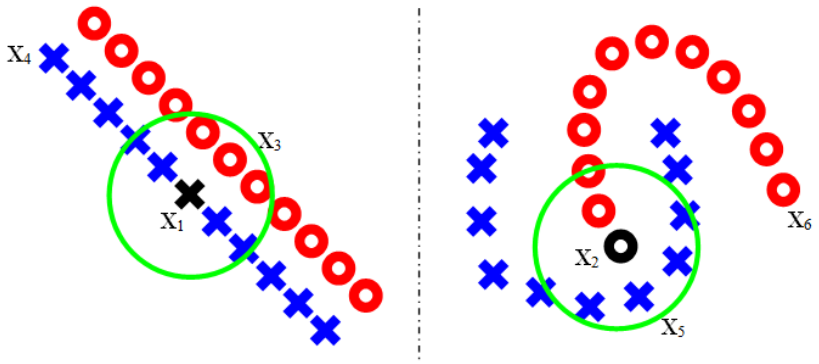


Figure 2.4. *Transitivity of path-based distance.*

The transitivity property is the main advantage of the distance. However, it is sensitive to outliers. As a result, clustering based on the path-based distance is fragile to outliers as can be seen in Figure 2.5.

Chang and Yeung [55] improved the path-based distance to be more robust to outliers. When computing the Euclidean distance of two points, they incorporated the density information to alleviate the effect of outliers. The improved path-based distance is defined as:

$$D'_{ij} = \min_{p \in P_{ij}} \left(\max_{1 \le i < |p|} \frac{d(p[i], p[i+1])}{\rho[i]\rho[i+1]} \right) \tag{2.4}$$

where $\rho[i]$ is the density of $\mathbf{x}_i$ and is computed from its K nearest neighbors, $d(p[i], p[i+1])$ is the Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$. If $\mathbf{x}_i$ has a sparse neighborhood, then $\rho[i]$ is small and the path-based distance from $\mathbf{x}_i$ is great.

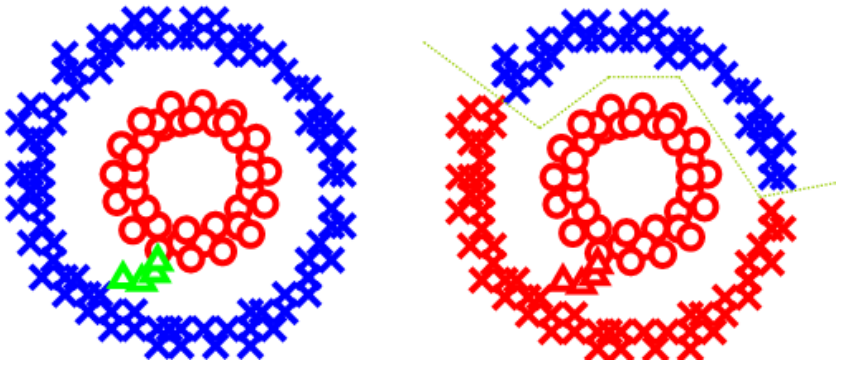


***Figure 2.5**. Left is a data set, which consists of two clusters (two circles) and some outliers represented by "△". Right is a clustering using the path-based distance. The clustering cannot discover the cluster structure because of the outliers.*

In summary, the two distances discussed above have been reported to produce satisfactory results on some data sets. These measures can disclose hidden information for clustering that is ignored by other measures.

2.4 FURTHEST REFERENCE POINT BASED DISTANCE

In this section, we discuss a new distance called *furthest reference point based distance* (FRPD) and a divisive clustering algorithm derived from this distance.

FRPD is based on the following observation: Data points in the same cluster have similar distances to a reference point, which is defined as the furthest point from the center of the cluster to be partitioned, while those in different clusters may have different distances to the same reference point. An example of FRPD-based partition is illustrated in Figure 2.6.

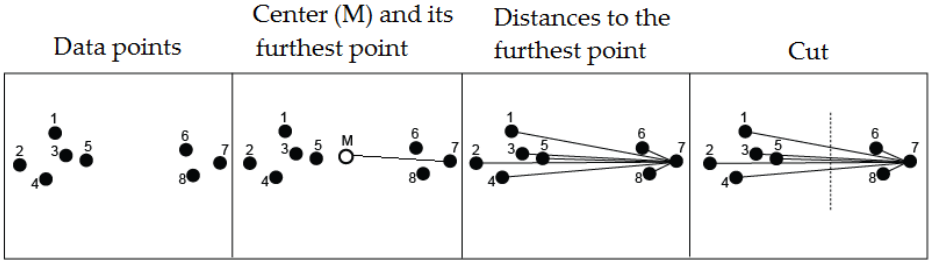


Figure 2.6. *FRPD-based partition.*

In Figure 2.6, data point 7 is the furthest point from the center M of the data set, and it is therefore selected as the reference point. Suppose that d_{ri} denotes the distance of the reference r to point i $(1 \le i \le 8)$. The distance sequence is, in ascending order, $< d_{r7}, d_{r8}, d_{r6}, d_{r5}, d_{r3}, d_{r1}, d_{r4}, d_{r2} >$. From the sequence, we determine the pair $< d_{r6}, d_{r5} >$ as the neighbor pair of which the difference $|d_{ri5}-d_{ri6}|$ is the largest. Therefore, the sequence is cut into two parts from this pair: $< d_{r7}, d_{r8}, d_{r6} >$ and $< d_{r5}, d_{r3}, d_{r1}, d_{r4}, d_{r2} >$. The data set is partitioned correspondingly into two groups: {7,8,6} and {5, 3, 1, 4, 2}.

However, the partition process does not always work. For example, when the furthest reference point is on or near to the perpendicular of two centers, the wanted partition is not achieved. Surprisingly, this is related to the detection of outliers.

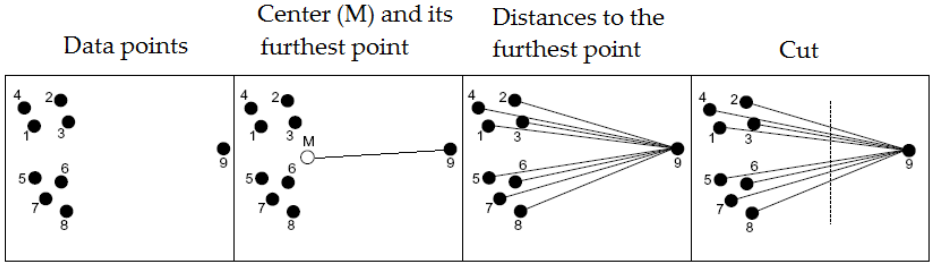


Figure 2.7*. The reference point is an outlier.*

In Figure 2.7, the data set consists of two clusters and an outlier. The ordered sequence is <d_{r9}, d_{r3}, d_{r6}, d_{r8}, d_{r2}, d_{r7}, d_{r5}, d_{r4}>, where < d_{r9}, d_{r3}> has the largest difference. Accordingly, the reference point 9 is cut off.

2.5 DIVFRP: A DIVISIVE CLUSTERING ALGORITHM

By employing the furthest reference based distance, a divisive clustering algorithm (DA) has been introduced in [**P1**] consisting of the following steps:

1. Select a cluster to be partitioned,
2. Partition the selected cluster.

When applied repeatedly, the final clustering result is obtained. For the first step, three design alternatives exist: 1) completely partition every cluster; 2) partition the cluster with the most data points; 3) partition the cluster with the highest variance. The first alternative is simple but it ignores the results of the sequence of partitioning. The second alternative tries to balance the sizes of clusters but it ignores the density information of a cluster, which is taken into consideration in the third alternative. The third alternative was, therefore, selected in DA.

In the second step, the distances to the furthest reference point are used to partition the data set. The idea is shown in Figure 2.6 and 2.7.

After the data set is partitioned, the optimal K is determined.

2.5.1 Divisive Algorithm

Suppose $CS = \{C_i : 0 \le i \le M-1\}$ is a set of clusters. DA will find a cluster and bi-partition it. Before describing DA formally, some definitions are given as follows.

Definition 2.1 Let rp(C_i)be the furthest point to the mean of C_i, namely the reference point in C_i, as in

$$\mathrm{rp}(C_i) = \arg\max \sum_{\mathbf{x} \in C_i} \left\| \mathbf{x} - \mu(C_i) \right\| \tag{2.5}$$

where $\mu(C_i) = 1/|C_i| \sum_{\mathbf{x} \in C_i} \mathbf{x}$.

Definition 2.2 Let $\mathrm{Rank}(C_i)$ be an ordered list and:

$$\mathrm{Rank}(C_i) = \left\langle \mathrm{near\_ref}(C_i) \circ \mathrm{Rank}(C_i - \{\mathrm{near\_ref}(C_i)\}) \right\rangle \tag{2.6}$$

where $\circ$ is concatenate operator, near_ref(C_i) is the nearest point to the reference point in C_i as in

$$\mathrm{near\_ref}(C_i) = \arg\min_{\mathbf{x} \in C_i} \left\| \mathbf{x} - \mathrm{rp}(C_i) \right\| \tag{2.7}$$

Definition 2.3 Assume Rank(C_i)=$\langle x_1, \mathbf{x}_2, \cdots, \mathbf{x}_{ni} \rangle$, C_i is to be split into C_{i1} and C_{i2}, where $|C_i| = n_i$, $C_{i1} = \langle \mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_j \rangle$, $C_{i2} = \langle \mathbf{x}_{j+1}, \mathbf{x}_{j+2}, \cdots \mathbf{x}_{ni} \rangle$ and $1 \le j < n_i$. The dissimilarity function $g(C_{i1}, C_{i2})$ is defined as

$$g(C_{i1}, C_{i2}) = \left\| \mathbf{x}_{j+1} - \mathrm{rp}(C_i) \right\| - \left\| \mathbf{x}_j - \mathrm{rp}(C_i) \right\| \tag{2.8}$$

This dissimilarity definition can be compared with the distance (dissimilarity) definition of single-linkage algorithm as in

$$\mathrm{dist}(C_i, C_j) = \min_{\mathbf{x}_i \in C_i, \mathbf{x}_j \in C_j} \left\| \mathbf{x}_i - \mathbf{x}_j \right\| \tag{2.9}$$

Because single-linkage is an agglomerative algorithm, the pair of clusters with minimum distance, $\mathrm{dist}(C_i, C_j)$, are merged at each step. Whereas DA is a divisive one, the bi-partitioned pair (C_{i1}, C_{i2}) of cluster C_i should maximize the dissimilarity function $g(C_{i1}, C_{i2})$.

Definition 2.4 Let next_split(CS) be the cluster with the maximum deviation with respect to its centroid:

$$\mathrm{next\_split}(CS) = \arg\max_{C_i \in CS} \left\| \mathrm{dev}(C_i) \right\| \tag{2.10}$$

where $\mathrm{dev}(C_i) = 1/|C_i| \sum_{\mathbf{x} \in C_i} \left\| \mathbf{x} - \mu(C_i) \right\|$. So next_split($CS$) is the cluster to be split at next step.

Definition 2.5 Let $D(C_i)$ be a set consisting of the differences between elements of all neighboring pairs in Rank(C_i):

$$D(C_i) = \{ d : d = \left\| \mathrm{next}(\mathbf{x}) - \mathrm{rp}(C_i) \right\| - \left\| \mathbf{x} - \mathrm{rp}(C_i) \right\| \} \tag{2.11}$$

where $\mathbf{x}$ is a component in Rank(C_i), next($\mathbf{x}$) is the next component to $\mathbf{x}$ in Rank(C_i).

Definition 2.6 Let $b(C_i)$ be the point in C_i with the maximum difference in $D(C_i)$:

$$b(C_i) = \arg\max_{\mathbf{x} \in C_i} (\left\| \mathrm{next}(\mathbf{x}) - \mathrm{rp}(C_i) \right\| - \left\| \mathbf{x} - \mathrm{rp}(C_i) \right\|) \tag{2.12}$$

The cluster C_i is finally bi-partitioned into C_1 and C_2 as in:

$$C_1 = \{ \mathbf{x} : \mathbf{x} \in C_i \wedge \left\| \mathbf{x} - \mathrm{rp}(C_i) \right\| \le \left\| b(C_i) - \mathrm{rp}(C_i) \right\| \} \tag{2.13}$$

$$C_2 = C_i - C_1$$

The divisive algorithm (DA) is formally stated as follows.

Step 1. The cluster to be split is determined by Eq. (2.10): $C_i = \mathrm{next\_split}(CS)$. The first cluster to be split is the initial data set which includes all points.

Step 2. Partition C_i into two clusters with Eqs. (2.11)-(2.13).
Step 3. Repeat Step 1 And 2 until each cluster has only one object.

Note that recording the partition process is for the later analysis of the number of clusters K.

2.5.2 Determine the number of clusters

We classify the splits into two categories: *essential* and *inessential*. The split in Figure 2.6 is essential, but the split in Figure 2.7 is inessential because only an outlier is detected with the split. The number of the essential splits equals the number of clusters minus 1. To determine an essential split, we analyze the change of sum-of-error peaks.

We observed that peaks generally become lower with the split process going on. Under this circumstance, sliding averages are employed to detect the peaks.

Definition 2.7 Let $J_e(i)$ be the sum-of-error after ith bipartition:

$$J_e(i) = \sum_{j=0}^{i} J_{ce}(C_j) \tag{2.14}$$

where $0 \le i \le N-1$, $J_{ce}(C_j)$ is an effective error of each cluster defined as

$$J_{ce}(C_i) = \sum_{\mathbf{x} \in C_i} \left\| \mathbf{x} - \mu(C_i) \right\| \tag{2.15}$$

Definition 2.8 Let *Diff* be a list that consists of the differences between neighboring sum-of-errors:

$$Diff = \left\langle d_1, d_2, \cdots d_i \right\rangle \tag{2.16}$$

where $1 \le i \le N-1$, $d_i = J_e(i-1) - J_e(i)$.

Figure 2.8a illustrates the bipartitions (only first two) of the data set in Figure. 2.7. Figure 2.8b shows $J_e(i)$ of each bipartitions. It seems difficult to perceive some information to detect the number of clusters K only from Figure 2.8b. But in Figure 2.8c two points A and B, which correspond to $J_e(0)$-$J_e(1)$ and $J_e(1)$-$J_e(2)$, respectively, are very different from the remaining points, because the first two bipartitions lead to large decreases of J_e.

Definition 2.9 Let $P = <p_0, p_1, \ldots, p_j, \ldots, p_{t-1}>$ be a peak list, $p_j \in \{d_i: d_i$ is an element of *Diff*$\}$.

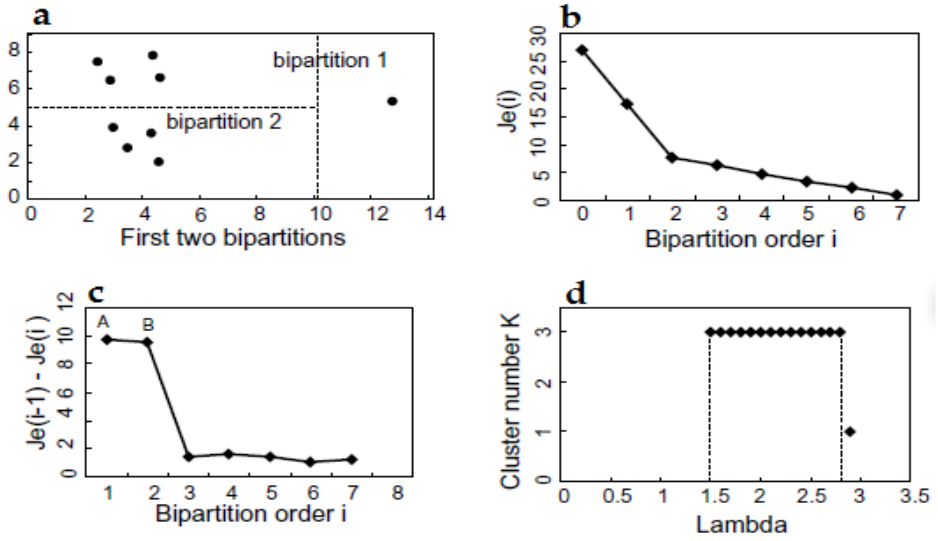


***Figure 2.8**. Illustration of the clustering process. (a) The bipartition process of data set in Figure 2.7; (b) the eight J_es for total eight bipartitions; (c) the differences of neighboring J_es, the points A and B are two potential peaks; and (d) the graph of function f.*

Note that if p_{j-1} and p_j in P correspond to d_u and d_v in *Diff*, respectively, then $v > u$ holds. The following fact exists: If the peak list P has t elements, the number of clusters should be t t+1.

The task of this phase is to construct the peak list P. Suppose that an element of *Diff*, say d_j, is selected as an element of P, say p_m, if the following holds:

$$d_j \geq \lambda \,\text{avgd}(m) \tag{2.17}$$

where λ is a parameter; avgd(m) is the average of the elements between de and d_j in *Diff*, d_e corresponds to the prior peak in P, namely, p_{m-1}. Two exceptions exist. When d_j is next to d_e in *Diff*, i.e. $j = e+1$, no elements exist between d_e and d_j; when p_m is first element in P, i.e. m=0, d_e does not exist. To remedy this, the previous average avgd(m-1) is used instead of avgd(m) in Eq. (2.17) for the former exception and the global average for the latter. Consequently, the sliding average is defined as:

$$\text{avgd}(m) = \begin{cases} \dfrac{1}{j-e-1}\displaystyle\sum_{f=e+1}^{j-1} d_f : \text{ if } m \neq 0 \text{ and } j > e+1 \\ \text{avgd}(m-1) \quad : \text{ if } m \neq 0 \text{ and } j = e+1 \\ \dfrac{1}{N-1}\displaystyle\sum_{f=1}^{N-1} d_f \quad : \text{ if } m = 0 \end{cases} \tag{2.18}$$

The window width of the sliding average is not fixed. In Figure 2.8c, when we consider point A and determine whether it is a peak, we will compare its value with global average since currently the peak list P is empty and no sliding average is available.

Definition 2.10 Let $LBD(S)$ be a list of binary relations of $(\lambda, f(\lambda))$ defined as

$$LBD(S) = \left\langle (\min_{\lambda \in S}(\lambda), f(\min_{\lambda \in S}(\lambda))) \circ LBD(S - \{\min_{\lambda \in S}(\lambda)\}) \right\rangle \tag{2.19}$$

The list $LBD(S)$ collects all binary relations of $(\lambda, f(\lambda))$ in an ascending order with respect to λ. We then consider how to eliminate the spurious clusters and consequently discover optimal cluster number K from the list $LBD(S)$.

Definition 2.11 Let SC be a set that consists of the K' clusters, $Q(SC)$= $\left\langle q_0, q_1, \cdots, q_i, \cdots, q_{K'-1} \right\rangle$ be an ordered list, where q_i $(0 \le i \le K' - 1)$ is the product of the cluster number and the sum-of-error with respect to a cluster in SC, $Q(SC)$ is defined:

$$Q(SC) = \left\langle \min_{C_i \in SC}(|C_i| \times J_{ce}(C_i)) \circ Q(SC - \{\arg\min_{C_i \in SC}(|C_i| \times J_{ce}(C_i))\}) \right\rangle \tag{2.20}$$

The criterion of identifying spurious cluster is given as follows. Suppose m spurious clusters exist in SC. Because a spurious cluster comprises a small number of objects and has a small effective error, the m spurious clusters are corresponding to the first m elements in $Q(SC)$, namely, $q_0, q_1, \cdots, q_{m-1}$. The element q_{m-1} must satisfy:

1. $q_m \ge \alpha q_{m-1}$,
2. If the cluster $C_{m\text{-}1}$ in SC is corresponding to $q_{m\text{-}1}$ in $Q(SC)$, then $\max_{C_i \in SC}|C_i| > \beta|C_{m-1}|$, where α and β are two real number parameters.

The first condition indicates a relative change ratio of the normal cluster and the spurious cluster near the boundary. The second one is an absolute constraint on spurious clusters. When we apply the criterion to SC, the spurious clusters are detected, but this is not final result since there may exist a number of candidates of K. As we know, a candidate of K and the corresponding set SC is decided by λ. We repeatedly increase λ by the step of σ and apply the criterion until it converges. For a candidate of K, K'_i, suppose that s_i spurious clusters are detected from K'_i clusters in terms of the criterion, the next candidate of K is K'_{i+1}, then the convergence is defined as:

1. According to the criterion of identifying spurious cluster, no spurious cluster in *SC* is detected, i.e. $s_i = 0$, or
2. After the spurious clusters being removed from *SC*, the number of normal clusters is equal to or greater than the next candidate of *K*, i.e., $K'_i - s_i \geq K'_{i+1}$ or
3. Candidate K'_{i+1} does not exist, i.e. K'_i is the last candidate of *K*.

For the first situation, all clusters in *SC* have relatively consistent object number and J_{ce}. For the second situation, if $K'_i - s_i < K'_{i+1}$, some of spurious clusters still exist in the K'_{i+1} clusters, and we must continue to consider the candidate K'_{i+1}; otherwise, all of spurious clusters are excluded from K'_{i+1} clusters, and it is meaningless to consider K'_{i+1} for removing spurious clusters. The last situation is obvious. Based on the definition of convergence, the spurious clusters detection mechanism (*SCD*) is formally presented as follows.

Repeat

1. Scan the list *LBD*(*S*) from the left to the right. Find out the pair of λ, λ_i and λ_j, which satisfy:
   (1) $f(\lambda_i) = f(\lambda_j)$, subject to $i < j$, $f(\lambda_i) > f(\lambda_{i-1})$ and $f(\lambda_j) < f(\lambda_{j+1})$;
   (2) $\Delta\lambda > \gamma$, where $\Delta\lambda = \lambda_j - \lambda_i$;
2. Construct the set *SC* which consists of K' clusters, and *Q*(*SC*), where $K' = f(\lambda_i) = f(\lambda_j)$;
3. Determine the spurious clusters according to the spurious cluster criterion;

Until convergence.

2.6 SUMMARY

In summary, distance measure is a crucial component in a clustering algorithm. First the typical distance measures were discussed and some new ones were introduced, and then the furthest reference point based distance and an efficient divisive clustering algorithm were proposed.

3 Graph-based Data Representation

3.1 HIDDEN INFORMATION FOR CLUSTERING

In machine learning and pattern recognition communities, researchers make great efforts to discover effective data representations, because such a representation may unveil some hidden information in the data set. Taking advantage of them can improve the performance of a clustering algorithm. In the previous chapter, it was verified that a novel distance measure might disclose hidden information for clustering and lead to a successful clustering algorithm. In this chapter, the argument will be discussed from the viewpoint of data representations.

Clustering ensemble combines multiple clustering results into a consolidated, stable and effective clustering [57–59]. In general, the relationship of data points and clusters forms a *binary cluster-association matrix* (BM), see Figure 3.1. Iam-On et al. [60–62] exploited the hidden information from BM further, and improved the performance of the ensemble algorithms. An entry in the BM denotes that, if a point belongs to a cluster, it is a hard relationship, namely, either "yes" or "no". This means that some potential relations are ignored. Iam-On et al. improved the situation.

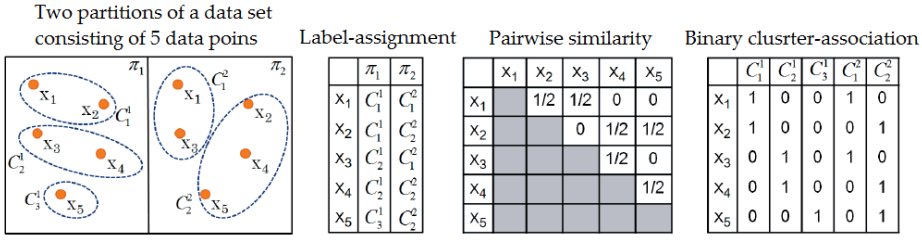
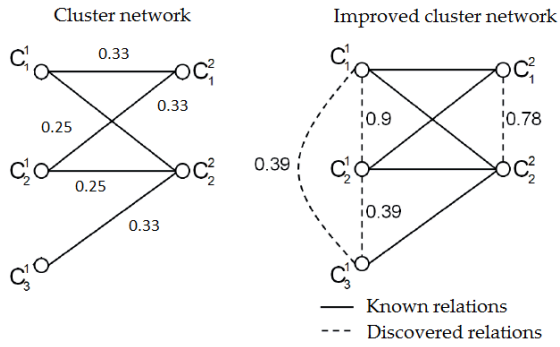


Figure 3.1. *The typical information used in a clustering ensemble* [61].



	C_1^1	C_2^1	C_3^1	C_1^2	C_2^2
x_1	1	0.9	0.39	1	0.78
x_2	1	0.9	0.39	0.78	1
x_3	0.9	1	0.39	1	0.78
x_4	0.9	1	0.39	0.78	1
x_5	0.39	0.39	1	0.78	1

***Figure 3.2**. Mining the hidden information* [61].

First, Iam-On et al. defined a cluster network (see Figure 3.2), where the weight of two clusters C_1 and C_2 is:

$$W(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|} \tag{3.1}$$

Then, the edges with zero weight were improved. If two clusters have no common data points, there will be no edge connecting the corresponding vertices directly. However, these vertices can be indirectly connected via other vertices, meaning that the connectivity of the vertices is transmittable. In the left of Figure 3.2, C_1^1 and C_2^1 are connected by C_1^2, and the evidence of the association between C_1^1 and C_2^1 is $E_1 = \min(W(C_1^1, C_1^2), W(C_2^1, C_1^2)) = 0.33$, and simultaneously, C_1^1 and C_2^1 are connected via C_2^2, the other association evidence being $E_2 = \min(W(C_1^1, C_2^2), W(C_2^1, C_2^2)) = 0.25$. The total evidence is $E_1 + E_2 = 0.58$. The weight of the edge between C_1^1 and C_2^1 is $(E_1 + E_2) / E_{\max} \times DC$, where $E_{\max}$ is the largest association evidence of all the vertex pairs without direct connection for normalizing the weight and, $DC \in [0,1]$ is the confidence level of accepting two non-identical clusters as being similar and here is 0.9. With this method, the dotted edges are generated (see the middle of Figure 3.2). Therefore the BM is refined and called *refined association* (RA) matrix (see the right of Figure 3.2).

Iam-On et al. [61] proposed also an alternative method to produce the matrix, but the details are omitted here. The work in [61] shows that discovering and using hidden information can be an effective way to

design a clustering algorithm. In the next section graph-based data representations, which can also disclose some hidden information, will be discussed.

3.2 GRAPH-BASED DATA REPRESENTATIONS

In a clustering algorithm, generally the data set is partitioned according to the similarities of the data points and an objective function. But when the clusters are of different sizes, shapes and densities, the similarities are not enough to produce a satisfactory clustering result. If we transfer the data beforehand into a representation that can describe the intrinsic structure of the data set, the performance of the clustering might be improved. In this chapter, the graph data representation based on K rounds of MSTs are discussed, and a new clustering algorithm using this representation is proposed.

3.2.1 Data representation based on KNN graph

For each data point, if it is connected to its KNN, the so-called KNN graph will be achieved, see Figure 3.3. A variant of this graph is obtained if each point is connected to those from which the distances to the point are not greater than a given value ε, and the variant is call ε-NN.

A KNN graph can be applied to classification, since the local data points may have similar class conditional probability [63]. It can also be used for clustering, as the local data points may reside in the same cluster. However, when the data set is high dimensional, the locality of KNN no longer holds. This can make the KNN-based methods useless. The use of KNN in high dimensional space has been studied [64, 65].

3.2.2 Clustering based on the KNN graph

Chameleon [66] is a clustering algorithm based on a KNN graph, see Figure. 3.4. It takes advantage of the three properties of a KNN graph. (1) Two data points far away from each other may reside in two disconnected sub-graphs, respectively, see Figure 3.5. Even if two points are in the same sub-graph, they may be separated in the

following partition steps. (2) A KNN graph discloses the densities of different regions. In the top-right of Figure 3.3, the densities of point A and B are different. The density can be expressed as the radius of the nearest neighbors, for example, the distance from the point to the furthest nearest neighbor. (3) Partitioning a KNN graph is computationally efficient compared to the corresponding complete graph. In fact, the first two properties mentioned above are derived from the hidden information captured by a KNN graph: A point and its nearest neighbor may be in the same cluster.

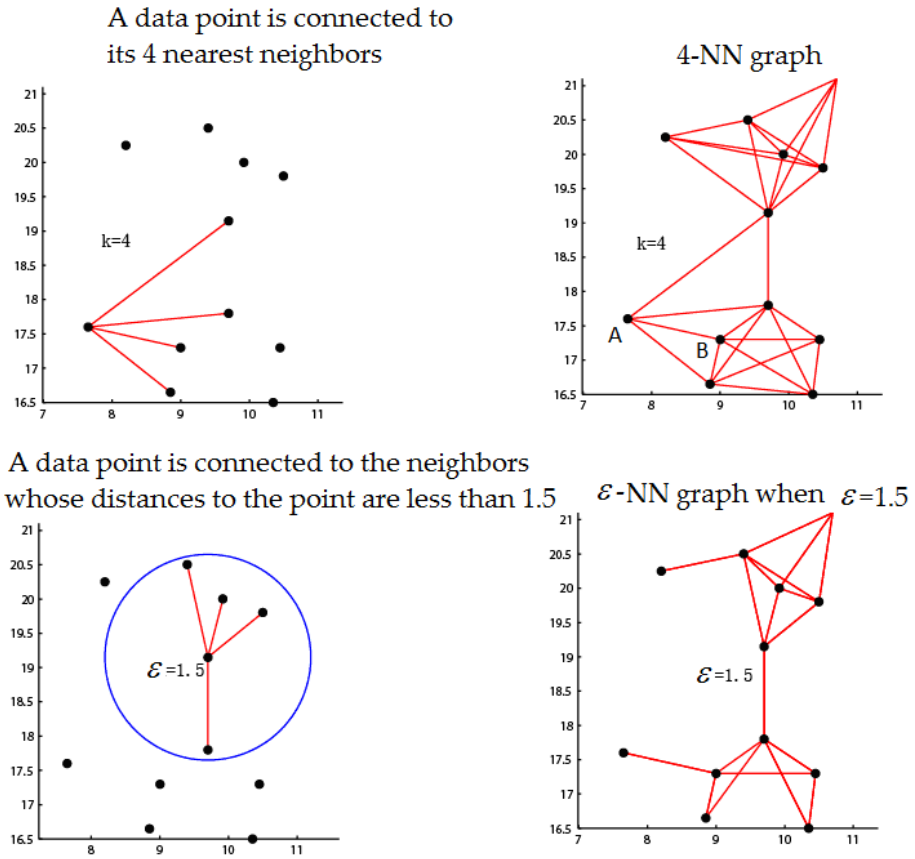


***Figure 3.3**. KNN and ε -NN.*

Chameleon first constructs a KNN graph of the data set, and applies *hMetis* [67] to partition the graph into clusters. It then defines the relative interconnectivity between two clusters and relative closeness within a cluster, and repeatedly merges clusters until the specified

number of clusters is achieved. The algorithm can detect clusters with arbitrary shapes.

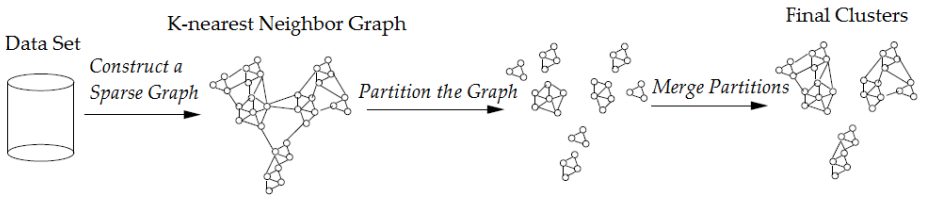


Figure 3.4 *Framework of Chameleon. It consists of three steps: constructing the KNN graph of the data set; partitioning the graph; combining the sub-graphs.*

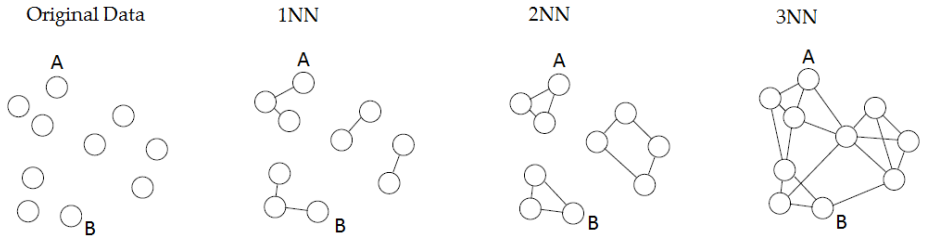


Figure 3.5 *In a 1NNgraph, points A and B are in different sub-graphs. In a 2NN graph, points A and B are in different sub-graphs. In a 3NN graph, points A and B are in the same graph.*

3.2.3 MST based data representation

For a given weighted undirected graph $G=(V,E)$, a *spanning tree* $ST=(V',E')$ is a tree that contains all the vertices of the graph. An MST is a spanning tree with the smallest total weight, an example of an MST is shown in Figure 3.6 [68].

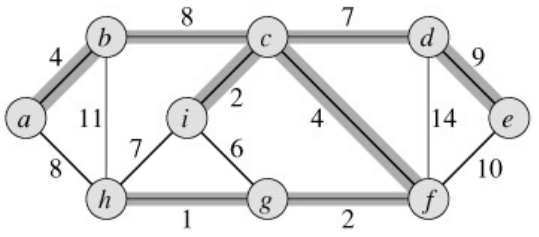


Figure 3.6 *The bolded edges forms an MST of the graph.*

Similar to a KNN graph, an MST can depict the intrinsic structure of the data set. This can be explained with the following relationship: $1NN \subseteq MST \subseteq KNN$. When K=1, all edges of the KNN are included in

the MST; when K increases, all edges of the MST are included in the KNN [69]. MST has been applied to clustering in [13, 70].

To apply MSTs to clustering, Zahn proposed several typical cluster problems [13], which are illustrated in Figure 3.7. Edge *ab* is defined as *inconsistent* if its weight is much greater than both the average weights of the edges connected to *a* and *b*. For well-separated data sets, for example those on the left and right at the bottom of Figure 3.7, one can obtain partitions by determining the inconsistent edges and removing them from the MST.

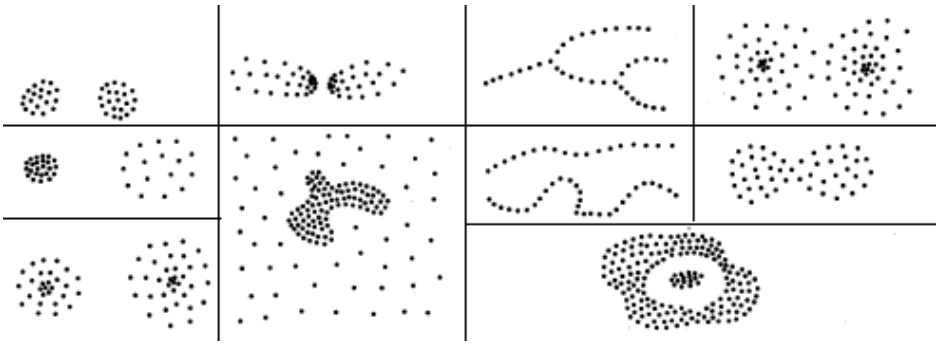


Figure 3.7 *Typical cluster problems [13].*

For data sets that are not well-separated, for instance those in the top-right of Figure 3.7, Zahn defined the *diameter* (path with the most number of edges) of the MST, and proposed an approach for clustering with the diameter.

In fact, both the inconsistent edge and the diameter of an MST describe the intrinsic structure of the data set via the MST. In other words, the two concepts can discover the hidden structure of the data set.

Clustering algorithms exist based on other data representations in the literature, such as clustering based on the relative neighbor graph [71], multipartite graphs [72], and so on.

3.3 K-MST-BASED GRAPH

Besides inconsistent edges and diameters, an MST possesses more useful information. For example, *K^{th}-order minimum spanning tree* (K-

MST) is an interesting representation for disclosing more information. It was first discussed in [73]. It is defined here as follows.

Definition 3.1 Suppose that edge set $T_1 = f_{mst}(V, E)$ denotes a minimum spanning tree of graph $G(V, E)$, the K[th]-order MST is defined as:

$$T_K = f_{mst}(V, E - \bigcup_{j=1}^{K-1} T_j) \tag{3.2}$$

where $f_{mst} : (V, E) \to T$ is a mapping from $G(V, E)$ to an MST. The above definition is not strict. For example, if T_1 has a vertex v and its degree is $|V| - 1$, then it is isolated in graph $G(V,E\text{-}T_1)$. This is because $T1$ contains all edges connected to v even if $G(V, E)$ is a complete graph. An edge connected to v in $G(V,E\text{-}T_1)$ does not exist, therefore in T_2 v is isolated. We solve this problem by selecting the longest edge of v, as isolation means v is unreachable, and the longest edge approaches the case more than the others.

Definition 3.2 KMST is a graph which is consists of K MSTs:

$$\text{KMST} = \bigcup_{i=1}^{K} T_i \tag{3.3}$$

KMST is also written as $G_{mst}(\mathbf{X}, K)$. We will discuss clustering based on K-MSTs in the next section and on KMST in the next chapter.

3.4 CLUSTERING BASED ON K-MSTS

3.4.1 Motivation

Although a lot of clustering algorithms have been proposed in the literature, each of them favors certain kinds of data sets. It means that a user needs to select a suitable algorithm and parameters for a specific data set. However, the user normally has no a priori knowledge about their data sets. This is the dilemma of clustering. Two techniques to alleviate this dilemma to some extent are *clustering ensemble* [59, 74, 75] and *multi-objective clustering* [76]. However, the clustering results of ensemble methods are generally not unique, while multi-objective clustering methods are too complex.

In [**P3**], a clustering method based on K-MST was proposed. The method employs K-MST as the data representation and deals with two kinds of typical data sets.

3.4.2 Typical cluster problems

Zahn in [13] described some typical cluster problems illustrated in Figure 3.7, and proposed different MST-based approaches to deal with those cluster problems.

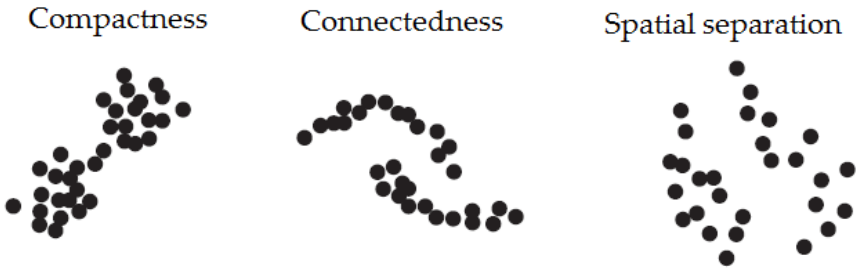


Figure 3.8 *Hand and Knowles [76] proposed three clustering criteria.*

Traditional clustering algorithms can be roughly categorized into hierarchical, partitioning, density-based and model-based [77]. Different from this taxonomy, Handl and Knowles [76] grouped them into three classes by the objective functions which are illustrated in Figure 3.8: (1) algorithms based on *compactness* such as K-means and average-linkage, which minimize the deviation of the intra-clusters; (2) algorithms based on *connectedness* such as path-based clustering [53–55] and single-linkage [94], which maximize the connectedness of the intra-clusters; and (3) algorithms based on spatial separation, usually combined with other objectives.

In this thesis, the cluster problems are classified into two groups: *separated* problem and *touching* problem. The former can be further divided into distance separated problem and density separated problem. More detailed definitions can be found in [**P3**]. In Figure 3.9a, the two classes have similar shape, size and density. Since the distance of a point pair in the same cluster is less than that of a point pair in different clusters, the data set is said to be distance separated. So are the

data sets in Figure 3.9b–e. The data set in Figure 3.9f is density separated, while those in Figure 3.9g–h are touching.

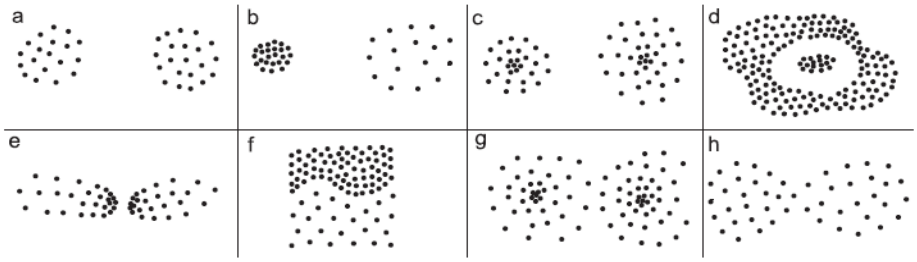


Figure 3.9 *Typical cluster problems. (a)–(e) distance separated cluster problems; (f) density separated cluster problem; (g) and (h) touching cluster problems.*

3.4.3 Clustering algorithm for separated problems

A clustering algorithm based on K-MST is proposed to simultaneously deal with both distance and density separated problems. For separated problems, the data set is first represented as a graph, which consists of T_1 and T_2, and one may partition the two trees to achieve the clusters. The main idea is to design the weight of an edge in the graph, and then iteratively remove the edges of the biggest weight until a partition of the graph is obtained. The weight of an edge is defined as in [**P**3]:

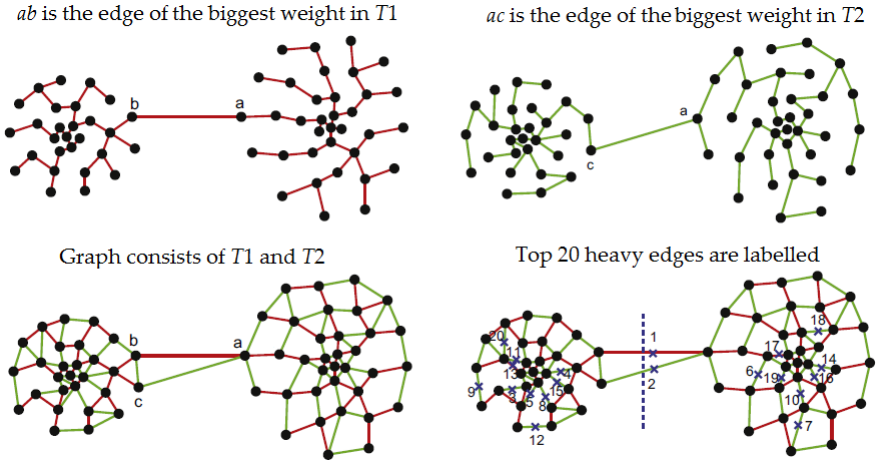


Figure 3.10 *Distance separated data set in Figure 3.9c is partitioned. The removal of the first two heavy edges leads to a partition, and the two edges come from T_1 and T_2, respectively. In the labeled 20 edges, 17 come from T_2 and 3 from T_1.*

Definition 3.3 Let $G_{mst}(\mathbf{X}) = (V, E_{mst})$ be a two-round-MST based graph, $e_{ab} \in E_{mst}$ and $a, b \in V$, $w(e_{ab})$ be the weight of e_{ab} as in

$$w(e_{ab}) = \frac{\rho(e_{ab}) - \min(\text{avg}(E_a - \{e_{ab}\}), \text{avg}(E_b - \{e_{ab}\}))}{\rho(e_{ab})} \tag{3.4}$$

where $E_a = \{e_{ij} \mid (e_{ij} \in E_{mst}) \wedge (i = a \vee j = a)\}$, $E_b = \{e_{ij} \mid (e_{ij} \in E_{mst}) \wedge (i = b \vee j = b)\}$, $\text{avg}(E) = (1/|E|) \times \sum_{e \in E} \rho(e)$ and $\rho(e)$ is the Euclidean distance of edge e.

Definition 3.4 Let $Rank(E_{mst})$ be a list of edges ordered descendingly by corresponding weights as in

$$Rank(E_{mst}) = \left\langle \text{topweight}(E_{mst}) \circ Rank(E_{mst} - \{\text{topweight}(E_{mst})\}) \right\rangle \tag{3.5}$$

where $\text{topweight}(E_{mst}) = \arg\max_{e \in E_{mst}}(w(e))$, $\circ$ is a concatenate operator.

Edge removing scheme: The edge with large weight has the priority to be removed, namely edges are removed in the order of $Rank(E_{mst})$. Since every removal of edge may lead to a graph cut (excluding the first removal), we must determine whether or not a new graph cut is achieved after each removal. The determination could be made by traversing the graph with either breadth-first search algorithm or depth-first search algorithm.

However, the partition may be invalid, and this can be determined by a criterion: If the number of the removed edges from T_1 is almost equal to that from T_2, then the partition is valid, otherwise, invalid.

A distance separated example is shown in Figure 3.10. First, the two longest edges *ab* and *ac* are removed and a partition is achieved. Even if one removes 18 more edges, no new partition is produced. In the 20 removed edges, the majority come from T_2.

A density separated example is shown in Figure 3.11. After the longest nine edges are removed, where five edges come from T_1 and four edges from T_2, a partition is produced.

Definition 3.5 Let E_{gcut} be a set of removed edges when a graph cut on a two-round-MST based graph is achieved, if the following holds:

$$Ratio(E_{gcut}) = \frac{\min(|E_{gcut} \cap T1|, |E_{gcut} \cap T2|)}{|E_{gcut}|} \geq \lambda \tag{3.6}$$

where λ is a threshold, then the graph cut is valid, otherwise it is invalid. If the first graph cut is valid, the cluster is said to be separated, otherwise, non-separated.

Although the above approach can detect the separated clusters, it cannot deal with the touch cluster problems in [**P3**]. Another K-MST based clustering method for this kind of data set is proposed. For a touching cluster problem, a touching region exists in the neighbor boundaries of two clusters, which is called *neck* here. T_1 and T_2 are analyzed simultaneously to detect the neck.

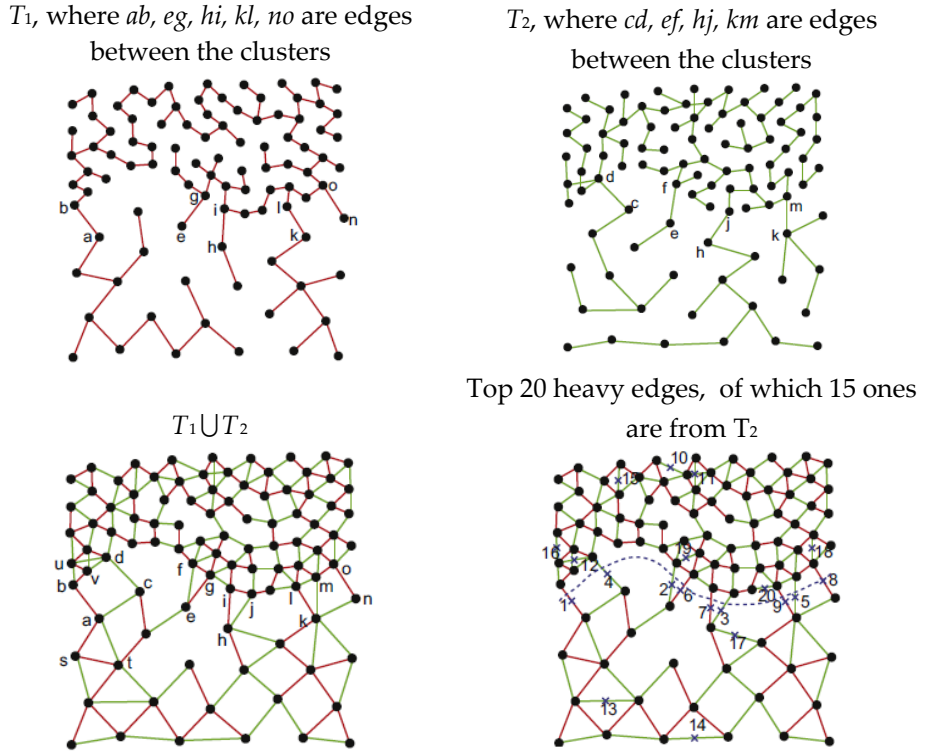


Figure 3.11 *Density separated data set in Figure 3.9f is partitioned.*

In Figure 3.12, cut1 and cut3 produce two partitions on T_1 and T_2, respectively. The two partitions are quite similar: only two data points (*a* and *c*) are inconsistent. The intuition behind this is that both cut1 and cut3 remove an edge from the neck. Similar phenomenon can be observed from cut2 and cut3, where only two data points (*b* and *c*) are inconsistent.

The details of the above two algorithms and the analysis of their performance and properties can be found in [**P3**].

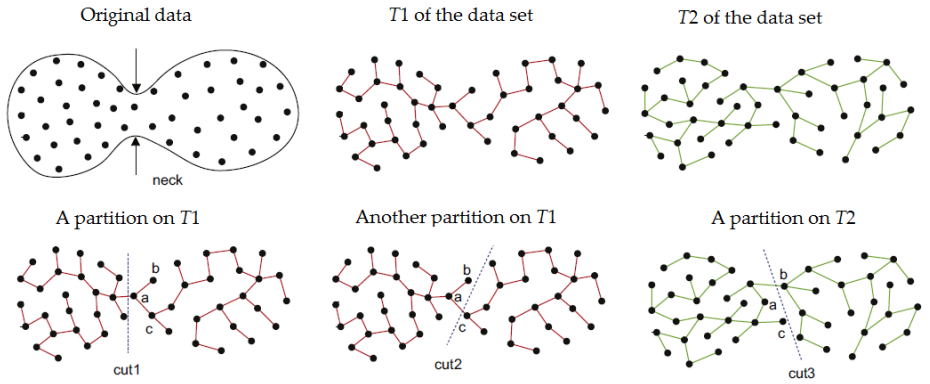


Figure 3.12 *Clustering on a touching problem*

3.5 SUMMARY

In this chapter, the design of a clustering algorithm from the point of view of hidden information was discussed. While different data representations can effectively discover the hidden information, graph-based data representations and corresponding clustering approaches were introduced, and a novel clustering method based on K-MST was presented.

4 Graph-based Split-and-Merge Algorithm

4.1 MECHANISM FOR CLUSTERING

In this chapter a KMST graph-based approach, which employs the split-and-merge mechanism, is introduced. In this thesis, a clustering mechanism refers to some operations and a convergence condition, which are repeatedly applied until a stable status is obtained. The clustering mechanism can be a crucial factor in the effectiveness of a clustering algorithm. Before introducing the split-and-merge based clustering, the mechanism of two well-known algorithms, K-means [1] and Affinity Propagation [78], are briefly analyzed.

4.1.1 Mechanism of K-means

Although K-means has been around for more than 50 years, it is still extensively employed because of its simplicity, effectiveness and efficiency. Among the top ten data mining algorithms, K-means is ranked number two [79].

The main idea is very simple. Initially, *K* data points are randomly selected as the cluster centers, and the following operations are then repeatedly performed until all the centers are fixed:

1) Assign each point to its nearest centers to form a partition;

2) Re-compute the center of each cluster.

In this process, a key component is ignored: what distance measure is used? A common choice is Euclidean distance. However, if K-means employs other measures, such as point symmetry distance [52], it can become more effective for special data sets.

The mechanism contained in K-means is very intuitive: The operations are the assignment of data points to the nearest center and update of the centers and the convergence condition is that all the centers remain unchanged. The criterion of assigning a point to its

nearest center embodies the basic idea of clustering: The data points in the same cluster are close, while those in different clusters are far away from each other. This can be formulized to minimize the sum of the squared errors (*SSE*):

$$SSE = \sum_{k=1}^{K} \sum_{i \in C_k} (\mathbf{x}_i - \overline{\mathbf{x}}^{(k)})^T (\mathbf{x}_i - \overline{\mathbf{x}}^{(k)}) \tag{4.1}$$

where $\overline{\mathbf{x}}^{(k)}$ is the mean of the k^{th} cluster, and K is the number of clusters. The process of updating the centers and re-assigning the data points in terms of new centers reduces the *SSE*. This is because if a data point in C_k exists so that the following condition is satisfied:

$$\frac{n_k}{n_k - 1}(\mathbf{x}_i - \overline{\mathbf{x}}^{(k)})^T (\mathbf{x}_i - \overline{\mathbf{x}}^{(k)}) > \frac{n_{k'}}{n_{k'} + 1}(\mathbf{x}_i - \overline{\mathbf{x}}^{(k')})^T (\mathbf{x}_i - \overline{\mathbf{x}}^{(k')}) \tag{4.2}$$

then $\mathbf{x}_i$ can be moved from C_k to $C_{k'}$ and *SSE* will be reduced [80].

In Figure 4.1, the process of convergence of K-means on a sample data set is illustrated. The partition gradually approaches the expected result and *SSE* is reduced.

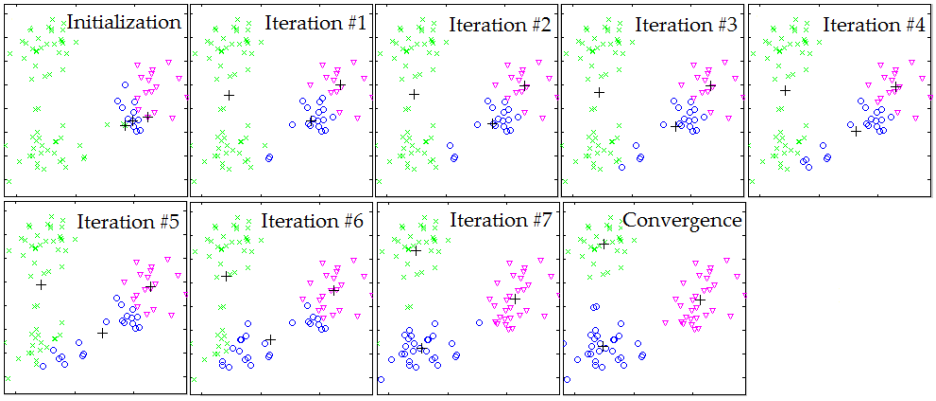


Figure 4.1 *An example of K-means. "+" denotes the centers.*

Although K-means is prone to converging at local optima, the mechanism contained in it is a good paradigm for designing a clustering algorithm.

4.1.2 Mechanism of Affinity Propagation

Similar to K-means [1], Affinity Propagation (AP) [78] is another clustering algorithm based on cluster representatives, even if K-means

uses the term *mean* while the later uses the term *exemplar*. The main idea of AP is that, initially, every data point is an exemplar, and all the exemplars are then iteratively refined by exchanging messages between the data points until the good exemplars emerge. Before we introduce the mechanism of AP, the following terms are defined:

- ✧ Similarity matrix $S = [s(i,k)]_{N \times N}$:

$$s(i,k) = \begin{cases} -\left\| \mathbf{x}_i - \mathbf{x}_k \right\|^2, & \text{if } i \neq k \\ \text{median}(\{s(i,j)\}), & \text{if } i = k, j \neq i \end{cases} \tag{4.3}$$

  When $i = k$, $s(k,k)$ denotes a preferred value at which k is taken as an exemplar, and is defined as the median of all the point pairs.

- ✧ Responsibility matrix $R = [r(i,k)]_{N \times N}$, $r(i,k)$ is a message sent from a data point i to a candidate exemplar k, which indicates the suitability of point k as the exemplar of the point i:

$$r(i,k) \leftarrow s(i,k) - \max_{k's.t.k' \neq k} \left\{ a(i,k') + s(i,k') \right\} \tag{4.4}$$

- ✧ Availability matrix $A = [a(i,k)]_{N \times N}$, $a(i,k)$ is a message sent from point k to i, which indicates its suitability in point i selecting point k as its exemplar:

$$a(i,k) \leftarrow \begin{cases} \min\{0, r(k,k) + \sum_{i's.t.i' \notin \{i,k\}} \max\{0, r(i',k)\}\}, & \text{if } i \neq k \\ \sum_{i's.t.i' \neq k} \max\{0, r(i',k)\}, & \text{if } i = k \end{cases} \tag{4.5}$$

Then, the exemplar of x_i is:

$$\arg\max_k (a(i,k) + r(i,k)) \tag{4.6}$$

The mechanism of AP can be described as:

1. Initialize similarity matrix S in terms of (4.3), $a(i,k) = 0$, $r(i,k) = 0$;
2. Update $a(i,k)$ and $r(i,k)$ according to (4.4) and (4.5), respectively;
3. Compute the exemplar of each data point;
4. Repeat Steps 2 and 3 until the exemplars remain unchanged in a specified number of consecutive iterations or the specified maximum number of iterations is reached.

The iteration process of the above mechanism is shown in Figure 4.2.

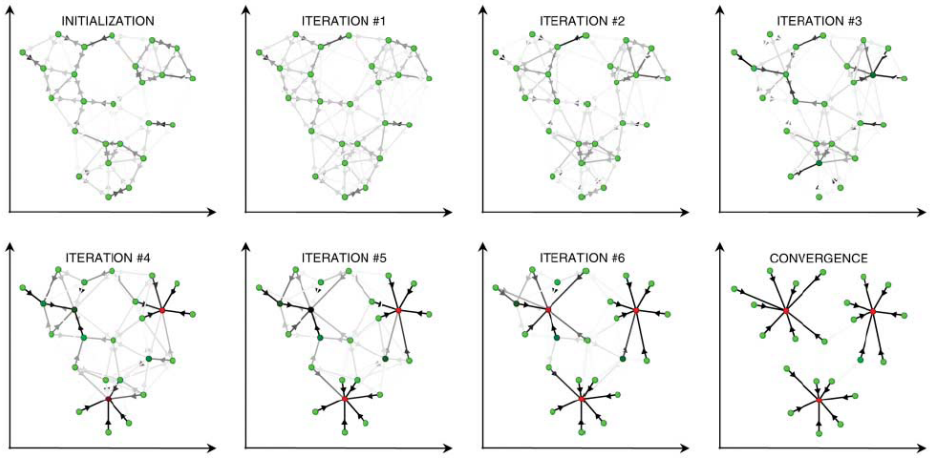


Figure 4.2 From initialization to convergence, six iterations are performed. The arrows indicate the directions of the passed messages.

Compared with K-means, AP contains a more complex mechanism, especially in the way in which the exemplars are produced.

4.1.3 Mechanism of split-and-merge

In general, for a given data set, data points within a local small region are also in the same cluster. This is the spirit of the split-and-merge mechanism based clustering, which is described as:

1. The data set is split into a number of sufficiently small subsets so that the data points in the same subset are as homogenous as possible.
2. Repeatedly combine the neighbor subsets with a certain merge criterion until the specified number of clusters is obtained.

Clustering algorithms based on the above mechanism decompose a complex problem into simpler ones, thus they are effective on complicated data sets.

Chameleon [66] and CSM [81] are two typical split-and-merge based clustering algorithms. Chameleon constructs a k-nearest-neighbor graph of the data set first, and uses hMetis [67], a graph-partitioning algorithm, to partition the graph into subclusters so that the edge cut is minimized. It then defines a cluster similarity, which consists of two

factors of relative interconnectivity and relative closeness, and merges the most similar subcluster pair repeatedly.

CSM applies K-means to partition the data set into a number of subclusters, and defines cluster cohesion to measure the cluster similarity and merges the subclusters. Both methods can deal with complex cluster problems.

4.2 GRAPH-BASED SPLIT-AND-MERGE CLUSTERING

In [**P4**], a clustering based on a split-and-merge mechanism, which employs KMST both in the split and merge stages, is proposed. The algorithm is composed of three stages, illustrated in Figure 4.3.

In the first stage, an MST of the data set **X** is constructed. According to this MST, **X** is pruned into **X**′ so that the leaf nodes in the MST are removed. This is because the leaf nodes usually have negative contributions to the clustering. Three rounds of MSTs are then performed to produce a graph, $G_{mst}(\mathbf{X}',3)$.

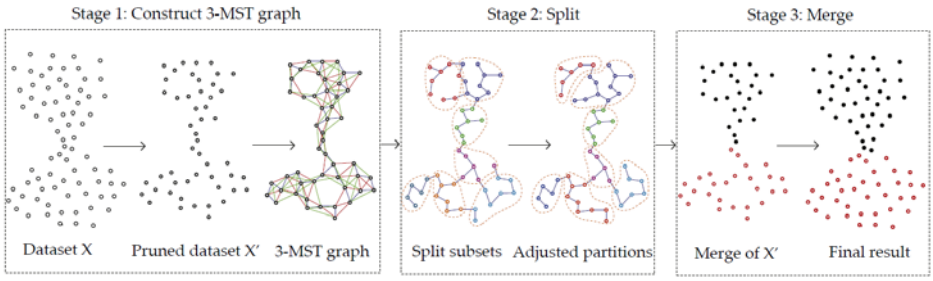


Figure 4.3 *Overview of the graph-based split-and-merge clustering.*

In the split stage, initial cluster centers are generated from $G_{mst}(\mathbf{X}',3)$, and K-means is then applied to the data set to achieve a partition. The partition is adjusted so that each cluster is a sub-tree of the MST of **X**′.

In the merge stage, a merge criterion is defined by using the information accumulated by $G_{mst}(\mathbf{X}',3)$, and the clusters in the partition are combined. At last, the removed leaf nodes are re-assigned to their nearest clusters. Before describing the algorithm, some definitions are given as follows.

Definition 4.1 Let **X**′ be the pruned version of **X** as in:

$$\mathbf{X}' = \mathbf{X} \setminus \{v_i \mid v_i \in V, \text{degree}(v_i) = 1\} \tag{4.7}$$

where degree(v_i) denotes the degree of vertex v_i in the MST of $\mathbf{X}$.

Definition 4.2 Let v_i be the ith prototype from $G_{mst}(\mathbf{X}', 3) = (V, E)$, $V_{i-1} = \{v_1, v_2, \cdots, v_{i-1}\}$, $E_{i-1} = \{(\mathbf{x}_i, \mathbf{x}_j) \mid (\mathbf{x}_i, \mathbf{x}_j) \in E \wedge (\mathbf{x}_i = v \vee \mathbf{x}_j = v) \wedge v \in V_{i-1}\}$, v_i is generated as:

$$v_i = \arg \max_{v \in V \setminus V_{i-1}} \mathrm{Car}(\{(\mathbf{x}_i, \mathbf{x}_j) \mid (\mathbf{x}_i, \mathbf{x}_j) \in (E \setminus E_{i-1} \wedge (\mathbf{x}_i = v \vee \mathbf{x}_j = v))\}) \tag{4.8}$$

where Car(S) denotes the cardinality.

The above definition determines the vertex with maximum degree as a prototype. After the data set is partitioned, a main tree in a cluster is defined and the partition is adjusted by repeatedly combining a non-main tree with a connected main tree.

Definition 4.3 Let $F_i = \{t_1, t_2, \cdots, t_j, \cdots, t_n\}$, and each t_j being a tree. The main tree of F_i is defined as:

$$\mathrm{Maintree}(F_i) = \arg \max_{t_j \in F_i} (\mathrm{Car}(t_j)) \tag{4.9}$$

where $\mathrm{Car}(t_j)$ denotes the edge number of tree t_j.

After $\mathbf{X}'$ has been split into K' subgroups, the merge stage is performed to obtain the final clusters.

Definition 4.4 Let *Pairs* be the set of neighboring pairs from $\mathbf{X}'$ as in:

$$Pairs = \{(C_i, C_j) \mid \exists (\mathbf{x}_p, \mathbf{x}_q) \in T_1, (\mathbf{x}_p \in C_i \wedge \mathbf{x}_q \in C_j) \vee (\mathbf{x}_p \in C_j \wedge \mathbf{x}_q \in C_i)\} \tag{4.10}$$

where $i \neq j$.

Definition 4.5 Suppose $(C_i, C_j) \in Pairs$. In $G_{mst}(\mathbf{X}', 3)$, let $E_{\mathrm{inter}}(C_i, C_j)$ be the set of edges across the partitions C_i and C_j:

$$E_{inter}(C_i, C_j) = \{(\mathbf{x}_p, \mathbf{x}_q) \mid (\mathbf{x}_p \in C_i \wedge \mathbf{x}_q \in C_j) \vee (\mathbf{x}_p \in C_j \wedge \mathbf{x}_q \in C_i)\} \tag{4.11}$$

Definition 4.6 Suppose that $(C_i, C_j) \in Pairs$. In $G_{mst}(\mathbf{X}', 3)$, set $V_{i,j} = \{\mathbf{x}_p \mid (\mathbf{x}_p, \mathbf{x}_q) \in E_{inter}(C_i, C_j) \wedge \mathbf{x}_p \in C_i\}$, $E_{i,j}$ is a set of edges within C_i where at least one endpoint is in $V_{i,j}$

$$E_{i,j} = \{(\mathbf{x}_p, \mathbf{x}_q) \mid \mathbf{x}_p, \mathbf{x}_q \in C_i \wedge (\mathbf{x}_p \in V_{i,j} \vee \mathbf{x}_q \in V_{i,j})\} \tag{4.12}$$

Definition 4.7 Suppose that $(C_i, C_j) \in Pairs$. In $G_{mst}(\mathbf{X}', 3)$, the connection span of C_i with respect to C_j is:

$$ConnSpan_{i,j} = \max_{\mathbf{x}_p, \mathbf{x}_q \in V_{i,j}} w(\mathbf{x}_p, \mathbf{x}_q) \tag{4.13}$$

Definition 4.8 Suppose that $(C_i, C_j) \in Pairs$. In $G_{mst}(\mathbf{X}', 3)$, the inter-connectivity(*IC*) of C_i and C_j is defined as:

$$IC(C_i,C_j)=\frac{\left|E_{inter}(C_i,C_j)\right|}{\min(\left|C_i\right|,\left|C_j\right|)}\times\frac{\min(\mathrm{Avg}(E_{i,j}),\mathrm{Avg}(E_{j,i}))}{\mathrm{Avg}(E_{inter}(C_i,C_j))}\times \max(ConnSpann_{i,j},ConnSpan_{j,i}) \tag{4.14}$$

where Avg(E) denotes the average weight of an edge set E.

Definition 4.9 Suppose that $(C_i,C_j)\in Pairs$, C_i is bisected into C_i^1 and C_i^2, C_j is bisected into $C_j^1$and C_j^2, the intra-similarity(*IS*) of the pair (C_i, C_j) is defined as:

$$IS(C_i,C_j)=\frac{1}{r_1\times r_2}\times\frac{\sqrt{\mathrm{Avg}(E_{inter}(C_i^1,C_i^2))\times\mathrm{Avg}(E_{inter}(C_j^1,C_j^2))}}{\mathrm{Avg}(E_{inter}(C_i^1,C_i^2))+\mathrm{Avg}(E_{inter}(C_j^1,C_j^2))} \tag{4.15}$$

where r_1 and r_2 are the numbers of edges of $E_{inter}(C_i^1, C_i^2)$ and $E_{inter}(C_j^1, C_j^2)$, respectively, and Avg(E) denotes the average weight of an edge set E.

Taking into account the inter-connectivity and the intra-similarity as a whole, we define the overall merge index as:

$$R(C_i,C_j)=IC(C_i,C_j)\times\mathrm{IS}(C_i,C_j) \tag{4.16}$$

The proposed algorithm is described as follows.

Step 1. Construct 3-MST graph
- 1.1 Construct an MST on **X**
- 1.2 Produce **X′** by pruning the leaves of the MST
- 1.3 Create three MSTs on **X′**: T_1, T_2 and T_3
- 1.4 Compute 3-MST graph based on **X′**: $G_{mst}(\mathbf{X}',3)\leftarrow T_1\cup T_2\cup T_3$

Step 2. Split the pruned dataset **X′** into clusters
- 2.1 Select K' highest degree nodes from $G_{mst}(\mathbf{X}',3)$ as initial prototypes
- 2.2 Apply K-means with the prototypes to produce K' partitions
- 2.3 For each of the partitions, find its main tree in T_1
- 2.4 For each of the sub-trees, repeatedly combine it with another sub-tree until it belongs to a main tree

Step 3. Merge the partitions into final clusters
- 3.1 Generate the set of neighboring partition pairs *Pairs*
- 3.2 Foe each pair $(C_i,C_j)\in Pairs$, calculate the merge criterion $R(C_i,C_j)$
- 3.3 Repeatedly merge the pair with maximum R()-

value until K clusters have been obtained.
3.4 Add the pruned leaves to the clustering using T_1

Step 4 If the convergence is achieved, stop; otherwise go to step 1.

4.3 SUMMARY

The mechanism is one of the important factors for devising clustering algorithms. Both K-means and Affinity Propagation contain well-designed mechanisms.

The split-and-merge mechanism divides a complicated cluster problem into a number of small homogeneous clusters, and then combines similar clusters. We propose an MST-based split-and-merge algorithm. It takes advantage of the information revealed by KMST, and incorporates it into the split-and-merge framework.

5 Fast Approximate MST Based on K-means

5.1 MOTIVATION AND INTRODUCTION

MST is used in the previous methods, but the construction process will be time-consuming if the data set is large. This chapter studies a fast approximate MST so that those methods can deal with large data sets.

For a given undirected and weighted graph, an MST is a spanning tree whose weights are minimized. Since MSTs can reflect the intrinsic structure of the data set, they have been extensively employed in image segmentation [82, 83], clustering [84], classification [85] and manifold learning [86, 87].

Prim's algorithm is a typical MST algorithm. Its idea is as follows: One randomly selects a node and considers it as a tree, then repeatedly adds an edge that connects one node from the tree to a new node so that the weight is the minimum. This is repeated until the tree contains all the nodes. The time complexity is O(Elog$V$), where E is the number of edges and V is the number of nodes. If a Fibonacci heap is used to implement the min-priority queue, the complexity becomes O(E+Vlog$V$) [66].

Kruskal's algorithm is another widely used algorithm for solving MSTs [88]. It sorts edges in ascending order by the weights, and initially considers each node as a tree. The shortest edges, except those that lead to a circle, are then used repeatedly to connect the trees until all the trees are combined into one. Its time complexity is O(Elog$V$).

However, for a complete graph $E = N*(N-1)/2$, and therefore Prim's and Kruskal's algorithms require $O(N^2)$ time, which makes them impractical for large data sets.

In [**P5**], a fast approximate MST algorithm was proposed. The algorithm employs the divide-and-conquer strategy that yields time complexity of $O(N^{1.5})$.

5.2 RELATED WORK

Related work can be found in the literature [89, 90]. Wang and Wilkes [89] applied the divide-and-conquer strategy to construct an approximate MST. However, their final goal was not to figure out an MST but to detect the longest edges in the early stage of clustering to improve the clustering efficiency. Initially, the data points were randomly stored in a list and each point was connected to its predecessor and successor. Intuitively, the list formed a *spanning tree* (ST). To optimize this ST, a hierarchical clustering algorithm was applied so that the data set was partitioned into a number of subsets. After the distances between the points in the same subset were computed, the ST was updated by replacing its longest edge with the shortest one between the two partitions which were generated by removing the longest. This procedure was iteratively performed until the difference of edge weights between two consecutive updates was less than a threshold.

Lai et al. [90] proposed an approximate MST based on the Hilbert curve and applied it to clustering. It is a two-stage algorithm. In the first stage, an approximate MST is constructed with a Hilbert curve. In the second stage, along with the approximate MST, the densities of data points are measured, and the data set is partitioned in terms of a given threshold of density. The procedure of constructing an approximate MST is an iterative process and each iteration contains a refinement step, where the number of iterations is d+1 and d is the dimensionality of the data set. In each iteration, a procedure similar to the Prim's algorithm is used to produce the approximate MST. The main difference is that when the minimum priority queue is maintained, the algorithm takes into account only the approximate MST in the last iteration and neighbors of the visited points determined by the sorted Hilbert table, while Prim's algorithm takes into account all the neighbors of visited points. However, the accuracy of Lai et al.'s algorithm depends on the order of the Hilbert curve.

5.3 PROPOSED FAST AND APPROXIMATE MST

To improve the efficiency of constructing an MST, an intuitive way is to reduce the redundant computation. When finding the nearest neighbors of a data point, it is not necessary to scan the whole data set but only those in its neighborhood. Similarly, when Kruskal's algorithm computes an MST from a complete graph, it is not necessary to sort all of the $N \times (N-1)/2$ edges but only the $(1+\alpha) \times N$ edges with the smallest weights, where $(N-3)/2 \geq \alpha \geq -1/N$.

Motivated by this observation, a divide-and-conquer method was proposed in [**P5**] to improve the MST algorithm.

In general, a divide-and-conquer method consists of the following three steps [66]:

1) Divide: A problem is divided into a number of sub-problems, which are similar to the original problem, but the scale is smaller.
2) Find a solution: Each sub-problem is solved and the sub-solutions are achieved.
3) Merge: All of the sub-solutions are combined into the final solution of the original problem.

According to the above procedure, a two-phase fast and approximate MST algorithm is devised as follows:

1) Divide-and-conquer:
   - ✧ Partition the data set into $\sqrt{N}$ subsets by K-means algorithm.
   - ✧ Construct an MST of each subset using Kruskal's or Prim's algorithm.
   - ✧ Merge the MSTs of the subsets in terms of a defined merge criterion.
2) Refinement:
   - ✧ Repartition the data set by assigning the data points to the nearest centers, which are selected from the previous cluster boundaries, so that the data points residing in the neighbor boundary regions of pairs of neighbor partitions are in the same cluster.
   - ✧ Construct the second approximate MST.

- ✧ Combine the two approximate MSTs into a graph, and apply the traditional MST to the graph to produce a more accurate approximate MST.

The process is illustrated in Figure 5.1. In the first stage, an approximate MST is constructed, but its accuracy is not always enough, because many of data points in the boundary regions are connected incorrectly, while the fundamental reason is that a neighbor pair in different subsets is not taken into account when the approximate MST is generated. To remedy this drawback, the refinement stage is designed.

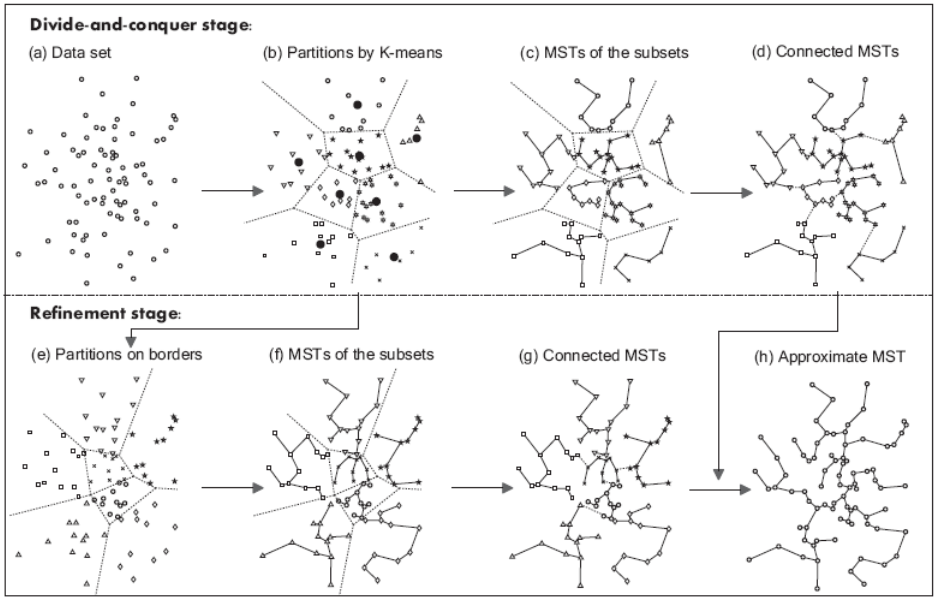


Figure 5.1 *Overview of the proposed fast approximate MST algorithm. (a) A given dataset. (b) The data set is partitioned into* $\sqrt{N}$ *subsets by K-means. (c) An exact MST algorithm is applied to each subset. (d) MSTs of the subsets are connected. (e) The data set is partitioned again so that the neighboring data points in different subsets of (b) are partitioned into identical partitions. (f) Exact MST algorithm is used again on the secondary partition. (g) MSTs of the subsets are connected. (h) A more accurate approximate MST is produced by merging the two approximate MSTs in (d) and (g) respectively.*

In the refinement stage, the data set is repartitioned aiming to group the data points that are near to each other and in different subsets of the first stage into the same subset. After the second approximate MST is obtained, the two MSTs are combined into a graph, which has at most

$2\times(N-1)$ edges. The final approximate MST is achieved by applying a traditional algorithm to this graph. The details of the proposed algorithm are in [**P5**].

5.4 SUMMARY

In this chapter a fast and approximate MST algorithm was studied. The motivation of this work was to speed up the construction of an MST, since traditional algorithms have time complexity of $O(N^2)$ and can be impractical for large data sets.

The proposed MST algorithm employs the divide-and-conquer strategy and its time complexity is $O(N^{1.5})$. Although the algorithm only produces an approximate MST, the effect of the sub-optimal result is expected to be insignificant in practical applications, such that clustering and manifold learning. The proposed algorithm can also be applied to the algorithms presented in Chapters 3 and 4.

6 Summary of the Contributions

6.1 CONTRIBUTIONS OF THE THESIS

[**P1**]: A dissimilarity measure based on the furthest reference point and a divisive clustering algorithm (DIVFRP) derived from this measure were proposed. Taking the dissimilarity measure as its fundamental component, DIVFRP is computationally efficient compared to traditional divisive clustering algorithms, such as single-link, and robust to outliers. Meanwhile, the proposed method can determine automatically the number of clusters, which is a challenge and interesting problem in clustering community. This is achieved by detecting the peaks of *SSE* but not by employing internal clustering validity indices, which is a general method in the literature but more subjective than the former since the estimated results depend on the internal indices selected. The experimental results of this performance are shown in Tables 6.1 and 6.2. The numbers of clusters estimated by the proposed method in the both data sets are more accurate than those by the others.

[**P2**]: This is a comment on a paper [91], which proposes a geodesic distance-based approach to build the neighborhood graph for isometric embedding. However, there is an error in the method. The local geodesic distance is estimated with the initial nearest neighbors determined by Euclidean distance, and in turn, optimizes a neighborhood graph. This optimization process violates the triangle inequality theorem. The error is pointed out and discussed in this paper.

[**P3**]: Generally, a clustering algorithm can deal with only certain types of data sets. However, data sets often are much different in cluster sizes, shapes and densities. Therefore, a clustering algorithm that can analyze all different data sets does not exist. In this paper, a graph-theoretical clustering method (2-MSTClus) was proposed and an

attempt was made to make it handle more kinds of data sets. It employed two rounds of MSTs as the data representation, and as a result the hidden information on separated and touching cluster problems is discovered. Since separated and touching cluster problems are complementary, the proposed clustering method is more universal than traditional ones. Figures 6.1 and 6.2 and Table 6.3 demonstrate this performance, where the data sets in the two figures are complex and synthetic and one in the table is real. The proposed method outperforms the others on the three data sets. Moreover, the data representation proposed can be used to other machine learning problems, such as community detection [95].

[**P4**]: This paper proposed the split-and-merge strategy based algorithm (SAM), which uses a graph generated by three rounds of MST. The main contribution is to represent the data set with the graph, and exploit the information of the graph in the split and merge processes so that complex data sets can be dealt with. The graph is used from two aspects: To partition the data set into subsets with K-means, the initial cluster centers are determined with the density information of the graph, which can be used as an initialization method of K-means, while random selection of cluster centers is a weak point of it. To design a merge criterion, the density and connection information of the graph is also used. As a result, complicated data sets can be analyzed.

[**P5**]: The clustering algorithms proposed in [**P3**] and [**P4**] are based on minimum spanning trees. However, the time complexity to construct an MST of a complete graph is $O(N^2)$, which is slow for large data sets. We propose a fast and approximate MST algorithm, which employs divide-and-conquer strategy and works in $O(N^{1.5})$. From the experimental results shown in Figure 6.3, we can see that proposed algorithm is computationally more efficient than Prim's algorithm. In this divide-and-conquer framework, any other exact or approximate MST algorithm can be applied to the sub-problems.

6.2 SUMMARY OF RESULTS

Experiment results of the papers are shown in this section. Table 6.1 provides a comparison of DIVFRP, DBScan and K-means. The comparison focuses on estimating cluster number *K* and the clustering accuracy. Here, DIVFRP, CH index-based DBScan and Silhouette index-based DBScan achieve the same results. They discover the cluster number *K* correctly, and three external indices give values close to the expected value 1 and outperform K-means results.

In Table 6.2, only DIVFRP estimates the number of clusters correctly. Silhouette index based DBScan and K-means achieve better results than CH index based DBScan and K-means.

Table 6.1 Performances of DIVFRP on R15[1]

Method	Estimated *K*	Rand	Jaccard	FM
DIVFRP [**P1**]	15	1.00	0.99	0.99
DBScan–CH [92]	15	1.00	0.99	0.99
DBScan–Silhouette [92]	15	1.00	0.99	0.99
K-means–CH [1]	18.7	0.99	0.87	0.93
K-means–Silhouette [1]	18.1	0.99	0.87	0.93

Table 6.2 Performances of DIVFRP on Wine

Method	Estimated *K*	Rand	Jaccard	FM
DIVFRP [**P1**]	3	0.72	0.42	0.59
DBScan–CH [92]	6	0.71	0.27	0.44
DBScan–Silhouette [92]	2	0.68	0.46	0.65
K-means–CH [1]	40.6	0.67	0.05	0.16
K-means–Silhouette [1]	2	0.67	0.46	0.65

In Table 6.3, 2-MSTClus is tested on Wine data. It performs better than all comparative algorithms because there are some outliers in the data set.

[1] The data sets can be found from: http://cs.uef.fi/sipu/datasets/.

Table 6.3 Performances of 2-MSTClus on Wine

Method	Rand	Adjusted rand	Jaccard	FM
K-means [1]	0.88	0.73	0.70	0.82
DBScan [92]	0.88	0.74	0.70	0.83
Single-linkage [93]	0.78	0.56	0.59	0.76
Spectral clustering [94]	0.80	0.55	0.53	070
2-MSTClus [**P3**]	0.93	0.85	0.82	0.90

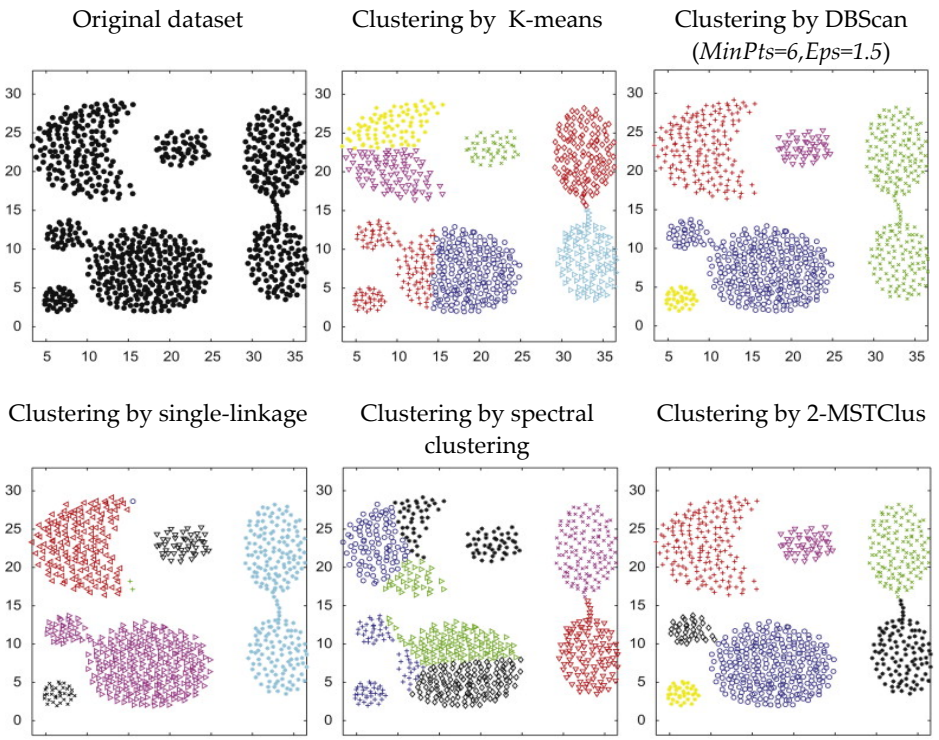


Figure 6.1 *Clustering results on Aggregation data set.*

In Figure 6.1, a data with a complex cluster structure is presented. It consists of seven clusters, and is a composite cluster problem. All of the four existing algorithms fail on this dataset, and only 2-MSTClus identifies the seven clusters accurately.

The data set in Figure 6.2 is composed of three sub data sets from [13]. The left top part is a touching problem; the left bottom one is a distance-separated problem; and the right one is a density-separated

problem. For this composite data set, 2-MSTClus can identify the six clusters, but the other four clustering methods k-means, single-linkage, DBScan and spectral clustering cannot.

The running time of FMST and Prim's algorithm on the four datasets is illustrated in the first row of Figure 6.3. From the results, we can see that FMST is computationally more efficient than Prim's algorithm, especially for the large datasets ConfLongDemo and MiniBooNE. The efficiency for MiniBooNE shown in the rightmost of the second and third row in Figure 6.3, however, deteriorates because of the high dimensionality.

The edge error rates and weight error rates of the four datasets are shown in the third row of Figure 6.3. We can see that both the edge error rate and the weight error rate decrease with the increase of the data size.

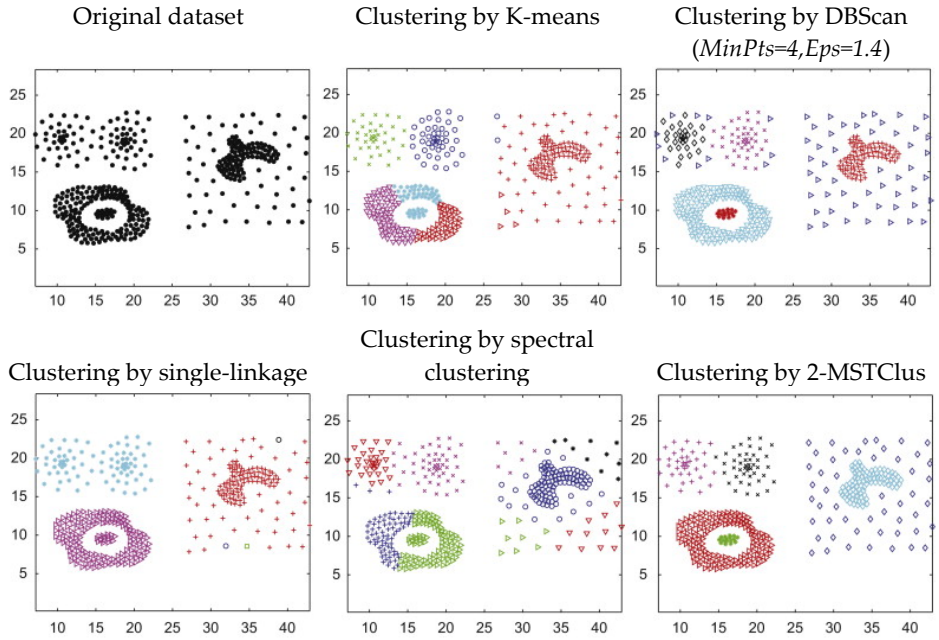


Figure 6.2 *Clustering results on Compound data set.*

For datasets with high dimension, the edge error rates are bigger. For example, the maximum edge error rates of MNIST are 18.5%, while those of t4.8k and ConfLongDemo are less than 3.2%. In contrast, the weight error rates decrease when the dimensionality increases. For instance, the weight error rates of MNIST are less than 3.9%.

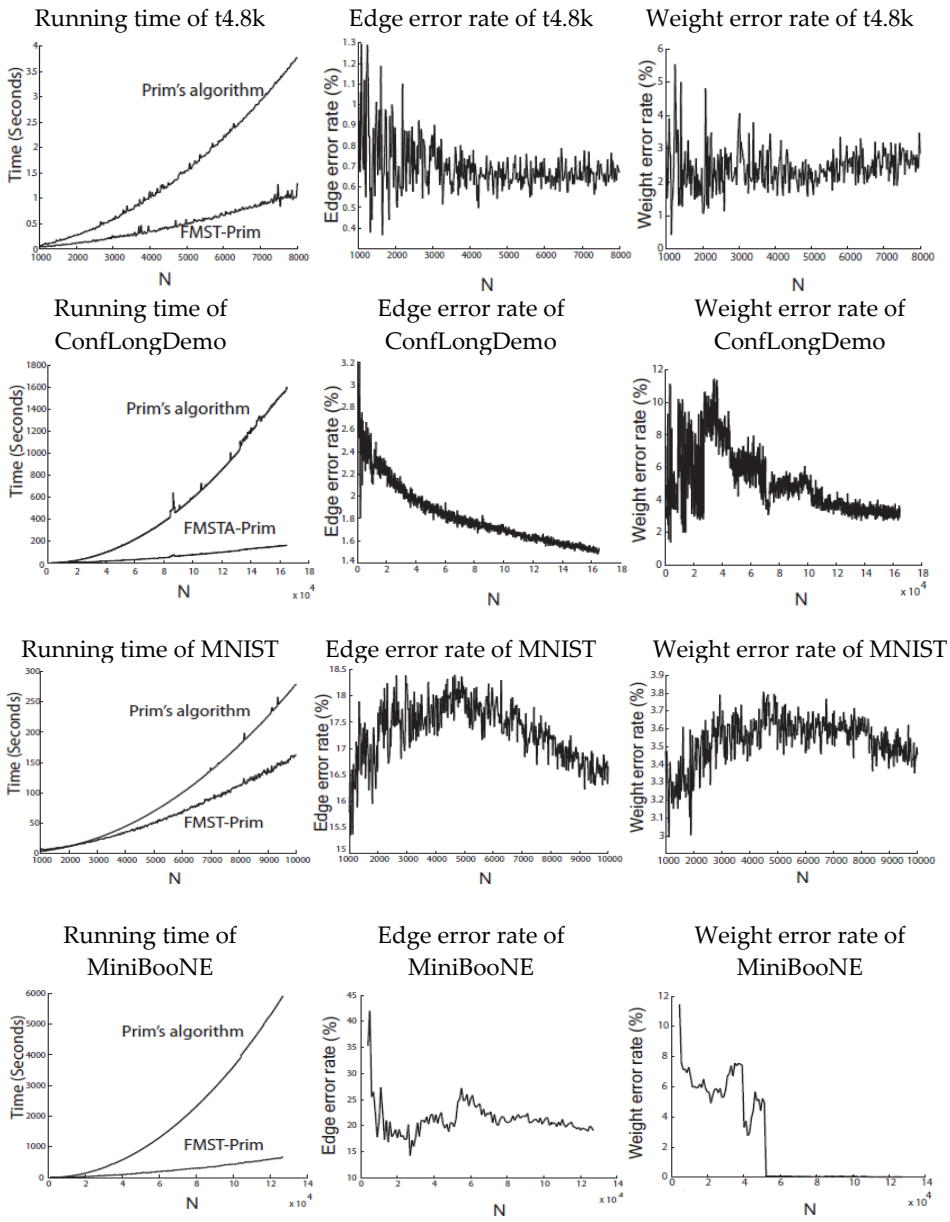


Figure 6.3 *The results of the test on the four datasets*[2].

[2] T4.8K is from HTTP://glaros.dtc.umn.edu/gkhome/cluto/cluto/download; ConfLongDemo is from HTTP://archive.ics.uci.edu/ml/; MNIST and MiniBooNE are from HTTP://yann.lecun.com/exdb/mnist.

7 Conclusions

In this thesis, we focus on graph-based clustering algorithm using a minimum spanning tree (MST) or K-MST. A fast approximate algorithm is introduced to construct an MST. The work on graph-based clustering algorithms is based on the clues provided by the components in designing a clustering algorithm, such as distance measure, data representation and mechanism.

A graph-based data representation may have a great effect on the performance of a clustering algorithm, because graph-based data representations can disclose more hidden and useful information for clustering, which cannot be perceived directly from its natural form. The 2-MSTClus clustering method proposed in this thesis is an example. This method represents the data set with first and second order MSTs, respectively. The motivation is to make the clustering more universal so that it can deal with more types of data sets. Coupled with the properties of the two rounds of MSTs, the clustering strategy in 2-MSTClus can achieve this goal.

The other sample that demonstrated the effect of a graph-based data representation is the algorithm SAM presented in this thesis. It makes use of the information discovered by a graph, which is composed of three rounds of MSTs, in its different phases so that complex data sets can be analyzed.

A proper mechanism is crucial for a clustering algorithm, as it involves how the information collected by distance measures and graph-based data representations are used. The split-and-merge strategy in SAM is a typical mechanism. The split process is designed to decompose complex structures of the data set, while merge process attempts to form valid clusters.

A distance measure can determine partially the performance of a clustering algorithm. A suitable measure can disclose the structure information of the data set, and make the algorithm more efficient. DIVFRP takes advantage of the furthest reference point based dissimilarity measure can detect clusters in spherical shapes, and is

efficient compared to traditional divisive clustering algorithms, such as single-link.

In future work, the focus will be on clustering algorithms that can deal with complex and large data sets, and methods that make a clustering algorithm more unified. For clustering complex and large data sets, resampling and ensemble techniques will be considered, while for the universality, a clustering algorithm selection framework is being studied.

References

[1] J. MacQueen, "Some methods for classification and analysis of multivariate observations," *Fifth Berkeley Symposium on Mathematics, Statistics and Probability*, University of California Press, 281–297, 1967.

[2] A.K. Jain, R.C. Dubes, "Algorithms for Clustering Data," Prentice-Hall, Englewood Cliffs, NJ, 1988.

[3] S.B. Everitt, "Cluster Analysis," John Wiley & Sons Inc., New York, 1974.

[4] A.K. Jain, M.N. Murty, P.J. Flynn, "Data clustering: A review," *ACM Computing Surveys*, 31(3), 264–323, 1999.

[5] L. Feng, M.H. Qiu, Y.X. Wang, Q.L. Xiang, and Y.F. Yang, "A fast divisive clustering algorithm using an improved discrete particle swarm optimizer," *Pattern Recognition Letters*, 31(11), 1216–1225, 2010.

[6] M. Chavent, Y. Lechevallier, O. Briant, "DIVCLUS-T: A monothetic divisive hierarchical clustering method," *Computational Statistics & Data Analysis*, 52(2), 687–701, 2007.

[7] C. Zhong, D. Miao, R. Wang, X. Zhou, "DIVFRP: An automatic divisive hierarchical clustering method based on the furthest reference points," *Pattern Recognition Letters*, 29(16), 2067–2077, 2008.

[8] K.C. Gowda, T.V. Ravi, "Divisive clustering of symbolic objects using the concepts of both similarity and dissimilarity," *Pattern Recognition*, 28(8), 1277–1282, 1995.

[9] J.H. Ward, "Hierarchical grouping to optimize an objective function," *Journal of the American Statistical Association*, 58(301), 236–244, 1963.

[10] J. Herrero, A. Valencia, J. Dopazo, "A hierarchical unsupervised growing neural network for clustering gene expression patterns," *Bioinformatics* 17 (2), 126–136, 2001.

[11] R.L. Cilibrasi, P.M.B. Vitanyi, "A fast quartet tree heuristic for hierarchical clustering," *Pattern Recognition*, 44(3), 662–677, 2011.

[12] J. Wang, M. Li, J. Chen, Y. Pan, "A fast hierarchical clustering algorithm for functional modules discovery in protein interaction networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(3), 607–620, 2011.

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

[14] J. Shi, J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), 888–905, 2000.

[15] C. Zhong, D. Miao, R. Wang, "A graph-theoretical clustering method based on two rounds of minimum spanning trees," *Pattern Recognition*, 43(3), 752–766, 2010.

[16] P.D. McNicholas, T.B. Murphy, "Model-based clustering of microarray expression data via latent Gaussian mixture models," *Bioinformatics*, 26(21) 2705–2712, 2010.

[17] J. Guo, E. Levina, G. Michailidis, "Pairwise variable selection for high-dimensional model-based clustering," *Biometrics*, 66(3), 793–804, 2010.

[18] D.B. Woodard, M. Goldszmidt, "Online model-based clustering for crisis identification in distributed

computing," *Journal of the American Statistical Association*, 106(493), 49–60, 2011.

[19] S.N. Sulaiman, N.A.M. Isa, "Adaptive fuzzy K-means clustering algorithm for image segmentation," *IEEE Transaction on Consumer Electronics*, 56(4), 2661–2668, 2010.

[20] H.Q. Liu, L.C. Jiao, F. Zhao, "Non-local spatial spectral clustering for image segmentation," *Neurocomputing*, 74(1–3), 461–471, 2010.

[21] X.H. Xu, J. Zhu, Q. Guo, "Fuzzy clustering based parameter clustering method in mandarin speech recognition," *Journal of the Chinese Institute of Engineers*, 28(5), 887–891, 2005.

[22] K. Hashimoto, H. Zen, Y. Nankaku, A. Lee, K. Tokuda, "Bayesian context clustering using cross validation for speech recognition," *IEICE Transactions on Information and Systems*, 94(3), 668–678, 2011.

[23] I. Lee, B.W. On, "An effective web document clustering algorithm based on bisection and merge," *Artificial Intelligence Review*, 36(1), 69–85, 2011.

[24] D. Cai, X.F. He, J.W. Han, "Locally consistent concept factorization for document clustering," *IEEE Transactions on Knowledge and Data Engineering*, 23(6), 902–913, 2011.

[25] E.D. Giacomo, W. Didimo, L. Grilli, G. Liotta, "Graph visualization techniques for web clustering engines," *IEEE Transactions on Visualization and Computer Graphics*, 13(2), 294–304, 2007.

[26] J.C. Mar, C.A. Wells, J. Quackenbush, "Defining an informativeness metric for clustering gene expression data," *Bioinformatics*, 27(8), 1094–1100, 2011.

[27] A.E. Baya, P.M. Granitto, "Clustering gene expression data with a penalized graph-based metric," *BMC Bioinformatics*, 12(2), 2011.

[28] W.F. Zhang, C.C. Liu, H. Yan, "Clustering of temporal gene expression data by regularized spline regression and an energy based similarity measure," *Pattern Recognition*, 43(12), 3969–3976, 2010.

[29] B. Chen, J. He, S. Pellicer, "Using hybrid hierarchical K-means (HHK) clustering algorithm for protein sequence motif super-rule-tree (SRT) structure construction," *International Journal of Data Mining and Bioinformatics*, 4(3), 316–330, 2010.

[30] J. Zhu, L. Xie, B. Honig, "Structural refinement of protein segments containing secondary structure elements: Local sampling, knowledge-based potentials, and clustering," *Proteins-Structure Function and Bioinformatics*, 65(2), 463–479, 2006.

[31] P.M.R. Tedder, J.R. Bradford, C.J. Needham, G.A. McConkey, A.J. Bulpitt, D.R. Westhead, "Gene function prediction using semantic similarity clustering and enrichment analysis in the malaria parasite Plasmodium falciparum," *Bioinformatics*, 26(19), 2431–2437, 2010.

[32] G.H. Xiao, W. Pan, "Consensus clustering of gene expression data and its application to gene function prediction," *Journal of Computational and Graphical Statistics*, 16(3), 733–51, 2007.

[33] S. Yoon, J.C. Ebert, E.Y. Chung, G.D. Micheli, R.B. Altman, "Clustering protein environments for function prediction: Finding PROSITE motifs in 3D," *BMC Bioinformatics*, 8(4), 2007.

[34] N. Mantel, "The detection of disease clustering and a generalized regression approach," *Cancer research*, 27(2), 209–220, 1967.

[35] C. Gray, T.J. MacGillivray, C. Eeley, et al., "Magnetic resonance imaging with K-means clustering objectively measures whole muscle volume compartments in sarcopenia/cancer cachexia," *Clinical Nutrition*,. 30(1), 106–111, 2011.

[36] C.S. Wilson, G.S. Davidson, S.B. Martin, et al., "Gene expression profiling of adult acute myeloid leukemia identifies novel biologic clusters for risk classification and outcome prediction", *Blood*, 108, 685–696, 2006.

[37] D., Munshi, A.L. Melott, P. Coles, "Generalized cumulant correlators and hierarchical clustering (of galaxies)," *Monthly Notices of the Royal Astronomical Society*, , 311(1), 149–160, 1900.

[38] P. Mähönen, T. Frantti, "Fuzzy classifier for star-galaxy separation," *The Astrophysical Journal*, 541(1), 61–263, 2000.

[39] R.A.C., Croft, et al., "Galaxy morphology, kinematics and clustering in a hydrodynamic simulation of a lambda cold dark matter universe," *Monthly Notices of the Royal Astronomical Society*, 400(1), 43–67, 2009.

[40] G. Punj, D.W. Stewart, "Cluster analysis in marketing research: Review and suggestions for application," *Journal of Marketing Research*, 20(2), 134–148, 1983.

[41] J.J. Huang, G.H. Tzeng, C.S. Ong, "Marketing segmentation using support vector clustering," *Expert Systems with Applications*, 32(2), 313–317, 2007.

[42] F. Pattarin, S. Paterlini, T. Minerva, "Clustering financial time series: An application to mutual funds style

analysis," *Computational Statistics & Data Analysis*, 47(2), 353–372, 2004.

[43] A. Krawiecki, J.A. Hołyst, D. Helbing, "Volatility clustering and scaling for financial time series due to attractor bubbling," *Physical Review Letters*, 89(15), 158701, 2002.

[44] S. Zhang, R.S. Wang, X.S. Zhang, "Identification of overlapping community structure in complex networks using fuzzy c-means clustering," *Physica A: Statistical Mechanics and its Applications*, 374(1), 483–490, 2007.

[45] S. Boccaletti, M. Ivanchenko, V. Latora, et al., "Detecting complex network modularity by dynamical clustering," *Physical Review E*, 75(4), 045102, 2007.

[46] K.Q. Weinberger, L.K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, 10, 207–244, 2009.

[47] C.C., Aggarwal, A. Hinneburg, D.A. Keim, "On the surprising behavior of distance metrics in high dimensional space," *Proc. Int'l Conf. Database Theory*, 420–434, 2001.

[48] C.C. Aggarwal, C. Procopiuc, J.L. Wolf, J.S. Park, "Fast algorithms for projected clustering," *ACM SIGMOD Proc. Int'l Conf. Management of Data*, 61-72, 1999.

[49] T.I. Netzeva, A.P. Worth, T. Aldenberg, et al., "Current status of methods for defining the applicability domain of (quantitative) structure-activity relationships," *Alternatives to Laboratory Animals*, 33(2), 1–19, 2005.

[50] P. D'haeseleer, "How does gene expression clustering work?," *Nature Biotechnology*, 23(12), 1499–1502, 2005.

[51] M. Steinbach, G. Karypis, V. Kumar, "A comparison of document clustering techniques," *ACM SIGKDD Proc. Int'l Conf. Knowledge Discovery and Data Mining*, 2000.

[52] M.C. Su, C.H. Chou, "A modified version of the K-means algorithm with a distance based on cluster symmetry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6), 674–680, 2001.

[53] S. Bandyopadhyay, S. Saha, "GAPS: A clustering method using a new point symmetry-based distance measure," *Pattern Recognition*, 40(12), 3430-3451, 2007.

[54] B. Fischer, J.M. Buhmann, "Bagging for path-based clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11), 1411–1415, 2003.

[55] H. Chang, D.Y. Yeung, "Robust path-based spectral clustering," *Pattern Recognition*, 41(1), 191–03, 2008.

[56] K.H. Kim, S. Choi, "Neighbor search with global geometry: A minimax message passing algorithm," *In ICML Proc. Int'l Conf. Machine learning*, 2007.

[57] T. Li, M. Ogihara, S. Ma, "On combining multiple clusterings: An overview and a new perspective," *Applied Intelligence*, 33(2), 207–219, 2010.

[58] A.L.N. Fred, A.K. Jain, "Combining multiple clusterings using evidence accumulation". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6), 835–850, 2005.

[59] A. Topchy, A.K. Jain, W. Punch, "Clustering ensembles: Models of consensus and weak partitions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12), 1866–1881, 2005.

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

[61] N. Iam-On, B. Tossapon, G. Simon, et al., "A link-based approach to the cluster ensemble problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12), 2396-2409, 2011.

[62] N. Iam-On, B. Tossapon, G. Simon, et al., "A link-based cluster ensemble approach for categorical data clustering," *IEEE Transactions on Knowledge and Data Engineering*, 24(3), 413–425, 2012.

[63] C. Domeniconi, J. Peng, D. Gunopulos, "Locally adaptive metric nearest-neighbor classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9), 1281–1285, 2002.

[64] A. Hinneburg, C.C. Aggarwal, D.A. Keim, "What is the nearest neighbor in high dimensional spaces?," *Proc. Int'l Conf. Very Large Data Bases*, 2000.

[65] K. Beyer, J. Goldstein, R. Ramakrishnan, et al., "When is 'nearest neighbor' meaningful?," *Int'l Conf. Database Theory*, 217-235, 1999.

[66] G. Karypis, E.H. Han, V. Kumar, "Chameleon: hierarchical clustering using dynamic modeling," *IEEE Transactions on Computer*, 32(8), 68–75, 1999.

[67] G. Karypis, V. Kumar, "hMETIS 1.5: A hypergraph partitioning package," *Tech. Report, Dept. of Computer Science, Univ. of Minnesota*, 1998.

[68] T.H. Corman, C.E. Leiserson, R.L. Rivest, C. Stein, "Introduction to Algorithms," *Second Edition*, *MIT press*, *Cambridge*, *MA*, 2001.

[69] J.M. González-Barrios, A.J. Quiroz, "A clustering procedure based on the comparison between the k nearest neighbors graph and the minimal spanning tree," *Statistics & Probability Letters*, 62(1), 23–34, 2003.

[70] Y. Xu, V. Olman, D. Xu, "Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees," *Bioinformatics*, 18(4), 536–545, 2002.

[71] S. Bandyopadhyay, "An automatic shape independent clustering technique," *Pattern Recognition*, 37(1), 33–45, 2004.

[72] A. Vashist, C.A. Kulikowski, I. Muchnik, "Ortholog clustering on a multipartite graph," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(1), 17–27, 2007.

[73] J.H Friedman, L.C. Rafsky, "Graph-theoretic measures of multivariate association and prediction," *The Annals of Statistics*, 11(2), 377–391, 1983.

[74] H.G. Ayad, M.S. Kamel, "Cumulative voting consensus method for partitions with variable number of clusters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1), 160–173, 2008.

[75] A. Gionis, H. Mannila, P. Tsaparas, "Clustering aggregation," *ACM Transactions on Knowledge Discovery from Data*, 1(1), 1–30, 2007.

[76] J. Handl, J. Knowles, "An evolutionary approach to multiobjective clustering," *IEEE Transactions on Evolutionary Computation*, 11(1), 56–76, 2007.

[77] J. Han, M. Kamber, J. Pei, "Data Mining: Concepts and Techniques," *Morgan Kaufmann*, San Francisco, CA, 2011.

[78] B. J. Frey, D. Dueck, "Clustering by passing messages between data points," *Science,* 315(5814), 972–976, 2007.

[79] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip, "Top 10 algorithms in data mining," *Knowledge and Information Systems,* 14(1), 1–37, 2008.

[80] D. Steinley, "K-means clustering: A half-century synthesis," *British Journal of Mathematical and Statistical Psychology*, 59(1), 1–34, 2006.

[81] C.-R. Lin, M.-S. Chen, "Combining partitional and hierarchical algorithms for robust and efficient data clustering with cohesion self-merging," *IEEE Transactions on Knowledge and Data Engineering*, 17(2), 145–159, 2005.

[82] L. An, Q.-S. Xiang, S. Chavez, "A fast implementation of the minimum spanning tree method for phase unwrapping," *IEEE Transactions on Medical Imaging*, 19(8), 805–808, 2000.

[83] Y. Xu, E. C. Uberbacher, "2D image segmentation using minimum spanning trees," Image and Vision Computing, 15(1), 47–57, 1997.

[84] Y. Xu, V. Olman, D. Xu, "Clustering gene expression data using a graph-theoretic approach: An application of minimum spanning trees," *Bioinformatics*, 18(4), 536–545, 2002

[85] P. Juszczak, D.M.J. Tax, R.P.W. Duin, "Minimum spanning tree based one-class classifier," *Neurocomputing*, 72(7–9), 1859–1869, 2009.

[86] L. Yang, "K-edge connected neighborhood graph for geodesic distance estimation and nonlinear data projection," *Proceedings of the 17th International Conference on Pattern Recognition*, 196–199, 2004.

[87] L. Yang, "Building k edge-disjoint spanning trees of minimum total length for isometric data embedding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10), 1680–1683, 2005.

[88] J.B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical society*, 7(1), 48–50, 1956.

[89] X. Wang, D.M. Wilkes, "A divide-and-conquer approach for minimum spanning tree-based clustering," *IEEE Transactions on Knowledge and Data Engineering*, 21(7), 945–958, 2009.

[90] C. Lai, T. Rafa, D.E. Nelson, "Approximate minimum spanning tree clustering in high-dimensional space," *Intelligent Data Analysis*, 13(4), 575–597, 2009.

[91] G. Wen, L. Jiang, J. Wen, "Using locally estimated geodesic distance to optimize neighborhood graph for isometric data embedding," *Pattern Recognition*, 41(7), 2226–2236, 2008.

[92] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *Int'l Conf. Knowledge Discovery and Data Mining*, 226–231, 1996.

[93] P.H.A. Sneath, "The application of computers to taxonomy," *Journal of General Microbiology*, 17(1), 201–226, 1957.

[94] S. Jianbo, M. Jitendra, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), 888–905, 2000.

[95] J. Wu, X. Lia, L. Jiao, X. Wang, B. Sun, "Minimum spanning trees for community detection," Physica A: Statistical Mechanics and its Applications, 392(9), 2265–2277, 2013.

Appendix: Original Publications

1

Paper **P1**

Reprinted from Pattern Recognition Letters, Volume 29, Issue 16, C. Zhong, D. Miao, R. Wang, X. Zhou, "DIVFRP: An automatic divisive hierarchical clustering method based on the furthest reference points," Pages 2067–2077, Copyright (2008), with permission from Elsevier.

Pattern Recognition Letters 29 (2008) 2067–2077

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

DIVFRP: An automatic divisive hierarchical clustering method based on the furthest reference points

Caiming Zhong [a,b,*], Duoqian Miao [a,c], Ruizhi Wang [a,c], Xinmin Zhou [a]

[a] School of Electronics and Information Engineering, Tongji University, Shanghai 201804, PR China
[b] College of Science and Technology, Ningbo University, Ningbo 315211, PR China
[c] Tongji Branch, National Engineering and Technology Center of High Performance Computer, Shanghai 201804, PR China

ARTICLE INFO

Article history:
Received 25 September 2007
Received in revised form 21 May 2008
Available online 22 July 2008

Communicated by L. Heutte

Keywords:
Divisive clustering
Automatic clustering
Furthest reference point
Dissimilarity measure
Peak
Spurious cluster

ABSTRACT

Although many clustering methods have been presented in the literature, most of them suffer from some drawbacks such as the requirement of user-specified parameters and being sensitive to outliers. For general divisive hierarchical clustering methods, an obstacle to practical use is the expensive computation. In this paper, we propose an automatic divisive hierarchical clustering method (DIVFRP). Its basic idea is to bipartition clusters repeatedly with a novel dissimilarity measure based on furthest reference points. A sliding average of sum-of-error is employed to estimate the cluster number preliminarily, and the optimum number of clusters is achieved after spurious clusters identified. The method does not require any user-specified parameter, even any cluster validity index. Furthermore it is robust to outliers, and the computational cost of its partition process is lower than that of general divisive clustering methods. Numerical experimental results on both synthetic and real data sets show the performances of DIVFRP.

1. Introduction

Clustering is an unsupervised classification technique in pattern analysis (Jain et al., 1999). It is defined to divide a data set into clusters without any prior knowledge. Objects in a same cluster are more similar to each other than those in different clusters. Many clustering methods have been proposed in the literature (Xu and Wunsch, 2005; Jain et al., 1999). These methods can be roughly classified into following categories: hierarchical, partitional, density-based, grid-based and model-based methods. However, the first two methods are the most significant algorithms in clustering communities. The hierarchical clustering methods can be further classified into agglomerative methods and divisive methods. Agglomerative methods start with each object as a cluster, recursively take two clusters with the most similarity and merge them into one cluster. Divisive methods, proceeding in the opposite way, start with all objects as one cluster, at each step select a cluster with a certain criterion (Savaresi et al., 2002) and bipartition the cluster with a dissimilarity measure.

In general, partitional clustering methods work efficiently, but the clustering qualities are not as good as those of hierarchical methods. The *K*-means (MacQueen, 1967) clustering algorithm is one of well-known partitional approaches. Its time complexity is $O(NKId)$, where N is the number of objects, K is the number of clusters, I is the number of iterations required for convergence, and d is the dimensionality of the input space. In practice, K and d are usually far less than N, it runs in linear time on low-dimensional data. Even though it is computationally efficient and conceptually simple, *K*-means has some drawbacks, such as no guarantee of convergence to the global minimum, the requirement of the number of clusters as an input parameter provided by users, and sensitivity to outliers and noise. To remedy these drawbacks, some variants of *K*-means have been proposed: PAM (Kaufman and Rousseeuw, 1990), CLARA (Kaufman and Rousseeuw, 1990), and CLARANS (Ng and Han, 1994).

To the contrary, hierarchical clustering methods can achieve good clustering results, but only at the cost of intensive computation. Algorithm single-linkage is a classical agglomerative method with time complexity of $O(N^2 \log N)$. Although algorithm CURE (Guha et al., 1998), one improved variant of single-linkage, can produce good clustering quality, the worst-case time complexity of CURE is $O(N^2 \log_2 N)$. Compared to agglomerative methods, divisive methods are more computationally intensive. For bipartitioning a cluster C_i with n_i objects, a divisive method will produce a global optimal result if all possible $2^{n_i-1}-1$ bipartitions are considered. But clearly, the computational cost of the complete enumeration is prohibitive. This is the very reason why divisive methods are seldom applied in practice. Some improved divisive

* Corresponding author. Address: School of Electronics and Information Engineering, Tongji University, Shanghai 201804, PR China. Tel.: +86 21 69589867; fax: +86 21 69589359.
E-mail addresses: zhongcaiming@nbu.edu.cn, charman_zhong@hotmail.com (C. Zhong).

0167-8655/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2008.07.002

methods do not consider unreasonable bipartitions identified by a pre-defined criterion in order to reduce the computational cost (Gowda and Ravi, 1995). Chavent et al. (2007) in a monothetic divisive algorithm use a monothetic approach to reduce the number of admissible bipartitions.

Most traditional clustering methods, such as *K*-means, DBScan (Ester et al., 1996), require some user-specified parameters. Generally, however, the required parameters are unknown to users. Therefore, automatic clustering methods are expected in practical applications. Some clustering methods of this kind have been presented in the literature (Wang et al., 2007; Tseng and Kao, 2005; Garai and Chaudhuri, 2004; Bandyopadhyay and Maulik, 2001; Tseng and Yang, 2001). Roughly these methods can be categorized into two groups: clustering validity index-based methods (Wang et al., 2007; Tseng and Kao, 2005) and genetic scheme-based methods (Garai and Chaudhuri, 2004; Bandyopadhyay and Maulik, 2001; Tseng and Yang, 2001). Wang et al. (2007) iteratively apply the local shrinking-based clustering method with different cluster number *K*s. In the light of CH index and Silhouette index, the qualities of all clustering results are measured. The optimal clustering result with the best cluster quality is selected. Tseng and Kao (2005) use Hubert's Γ index to measure a cluster strength after each adding (or removing) of objects to (or from) the cluster. For genetic scheme-based clustering methods, it is crucial to define a reasonable fitness function. Bandyopadhyay and Maulik (2001) take some validity indices as fitness functions directly. In the methods of Garai and Chaudhuri (2004) and Tseng and Yang (2001), although validity indices are not used directly, the fitness functions are very close to validity indices essentially. So genetic scheme-based methods, in different extents, are dependent on the clustering validity indices. However, clustering validity indices are not a panacea since an index that can deal with different shapes and densities is not available.

Robustness to outliers is an important property for clustering algorithms. Clustering algorithms that are vulnerable to outliers (Patan and Russo, 2002) may use some outlier detection mechanisms (Aggarwal and Yu, 2001; Ramaswamy et al., 2000; Breunig et al., 2000; Knorr and Ng, 1998) to eliminate the outliers in data sets before clustering proceeds. However, since this is an extra task, users prefer to clustering algorithms robust to outliers.

In this paper, we propose an efficient divisive hierarchical clustering algorithm with a novel dissimilarity measure (DIVFRP). Based on the furthest reference points, the dissimilarity measure makes the partition process robust to outliers and reduces the computational cost of partitioning a cluster C_i to $O(n_i \log n_i)$. After a data set being partitioned completely, the algorithm employs a sliding average of differences between neighboring pairs of sum-of-errors to detect potential peaks and determine the candidates of the cluster number. Finally, spurious clusters are removed and the optimal cluster number *K* is achieved. Our experiments demonstrate these performances. The remaining sections are organized as follows: algorithm DIVFRP is presented in Section 2. Section 3 presents experimental results. The performances are studied in Section 4. Section 5 concludes the paper.

2. The clustering algorithm

We begin our discussion of the clustering algorithm DIVFRP by considering the concept of general clustering algorithm.

Let $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_i, \ldots, \boldsymbol{x}_N\}$ be a data set, where $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots, x_{id})^T \in \Re^d$ is a feature vector, and x_{ij} is a feature. A general clustering algorithm attempts to partition the data set $\boldsymbol{X}$ into *K* clusters: $C_0, C_1, \ldots, C_{K-1}$ and one outlier C_{outlier} set according to the similarity or dissimilarity measure of objects. Generally, $C_i \neq \emptyset$, $C_i \cap C_j = \emptyset$, $\boldsymbol{X} = C_0 \cup C_1 \cup \ldots \cup C_{K-1} \cup C_{\text{outlier}}$, where $i = 0, 1, \ldots, K-1$, $j = 0, 1, \ldots, K-1$, $i \neq j$.

The algorithm DIVFRP comprises three phases:

1. Partitioning a data set.
2. Detecting the peaks of differences of sum-of-errors.
3. Eliminating spurious clusters.

2.1. Partitioning a data set

2.1.1. The dissimilarity measure based on the furthest reference points

Similarity or dissimilarity measures are essential to a clustering scheme, because the measures determine how to partition a data set. In a divisive clustering method, let C_i be the cluster to be bipartitioned at a step of the partitioning process, $g(C_x, C_y)$ be a dissimilarity function. If the divisive method bipartitions C_i into C_{i1} and C_{i2}, the pair (C_{i1}, C_{i2}) will maximize the dissimilarity function *g* (Theodoridis and Koutroumbas, 2006). According to this definition of dissimilarity, we design our dissimilarity measure as follows.

For a data set consisting of two spherical clusters, our dissimilarity measure is on the basis of the observation: the distances between points in a same cluster and a certain reference point are approximative. We call the distances a representative. For the two clusters, two representatives exist with respect to a same reference point. Assume that there exits a point on the line that passes through the two cluster mean points, and both clusters are on the same side of the point. Taking the point as the reference point, one will get the maximum value of the difference between the two representatives. On the contrary, if the reference point is on the perpendicular bisector of the line segment that ends at the two cluster mean points, one will get the minimum value. However, it is difficult to get the ideal reference point since the cluster structure is unknown. We settle for the furthest point from the centroid of the whole data set instead, because it never lies between the two cluster mean points and two clusters must be on the same side of it. Fig. 1 illustrates the dissimilarity measure based the furthest point and how the cluster being split.

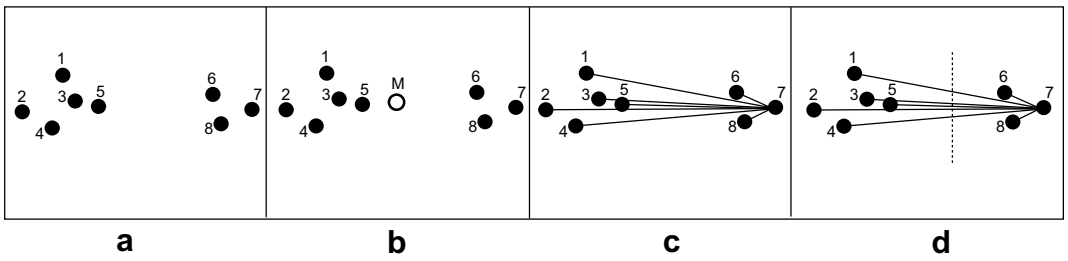


Fig. 1. Illustration of the dissimilarity measure and a split. In (a), a data set with two spherical clusters is shown. In (b), the hollow point *M* is the mean point of the data set; point 7 is the furthest point to the mean and selected as the reference point. In (c), distances from all points including the reference point to the reference are computed. In (d), the neighboring pair $< d_{r6}, d_{r5} >$ with maximum difference between its two elements is selected as the boundary, with which the cluster is split.

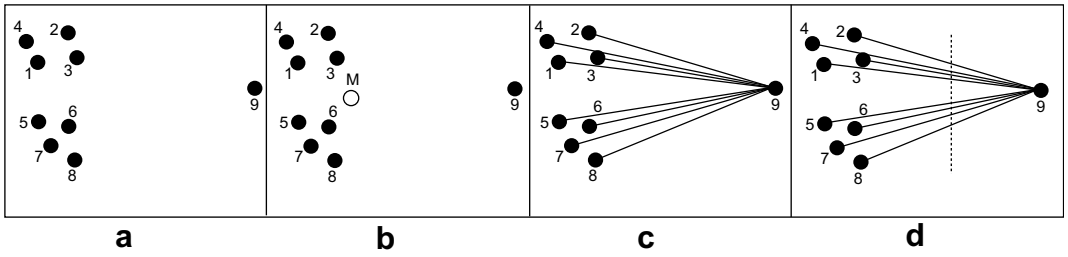


Fig. 2. Illustration of the reference point as an outlier. In (a), two spherical clusters with an outlier is shown. In (b), the hollow point M is the mean point of the data set; point 9 is the furthest point to the mean and selected as the reference point. In (c), distances from all points to the reference are computed. In (d), the neighboring pair $< d_{r9}, d_{r3} >$ with maximum difference between its two elements is selected as the boundary, with which the reference point itself is peeled off.

In Fig. 1b, point 7 is the furthest point to the data set centroid (hollow point M). So it is selected as the reference point r. Let d_{ri} stand for the distance between the reference point r and a point i ($1 \leqslant i \leqslant 8$), the distances are sorted ascendantly as $\langle d_{r7}, d_{r8}, d_{r6}, d_{r5}, d_{r3}, d_{r1}, d_{r4}, d_{r2}\rangle$. Considering all neighboring pairs in the list, we observe that the difference between the elements of the neighboring pair $\langle d_{r6}, d_{r5}\rangle$ is the maximum and select the pair as the boundary. This is the very dissimilarity measure. In accordance with the boundary, the list is then split into two parts: $\langle d_{r7}, d_{r8}, d_{r6}\rangle$ and $\langle d_{r5}, d_{r3}, d_{r1}, d_{r4}, d_{r2}\rangle$. Correspondingly, two clusters are formed: {7,8,6} and {5,3,1,4,2}.

Note that the dissimilarity measure based the furthest reference point does not always discern two clusters well. As aforementioned, when the furthest point is on or close to the perpendicular bisector of the line segment that takes two cluster mean points as its two endpoints, respectively, the dissimilarity measure fails to split the two clusters. Surprisingly, however, the dissimilarity measure acts as an outlier detector in this situation. This property, discussed in Section 4, endows our algorithm with robustness to outliers. In Fig. 2, the sorted list of distances from all of the points to the reference is $\langle d_{r9}, d_{r3}, d_{r6}, d_{r8}, d_{r2}, d_{r7}, d_{r1}, d_{r5}, d_{r4}\rangle$. The neighboring pair $\langle d_{r9}, d_{r3}\rangle$ has the maximum difference and functions as the boundary. So the reference point 9 is peeled off as an outlier.

Then we formally define the dissimilarity function. Suppose C_i is one of the valid clusters, $\boldsymbol{x} \in C_i$, $|C_i| = n_i$.

Definition 1. Let $rp(C_i)$ be the furthest point to the mean of C_i, namely the reference point in C_i, as in

$$rp(C_i) = \arg\sum_{\boldsymbol{x}\in C_i} \|\boldsymbol{x} - \mu(C_i)\| \tag{1}$$

where $\mu(C_i) = \frac{1}{|C_i|}\sum_{\boldsymbol{x}\in C_i}\boldsymbol{x}$

Definition 2. Let $\mathrm{Rank}(C_i)$ be an ordered list as in

$$\mathrm{Rank}(C_i) = \langle \mathrm{near\_ref}(C_i) \circ \mathrm{Rank}(C_i - \{\mathrm{near\_ref}(C_i)\})\rangle \tag{2}$$

where $\circ$ is concatenate operator, $\mathrm{near\_ref}(C_i)$ is the nearest point to the reference point in C_i as in

$$\mathrm{near\_ref}(C_i) = \arg\min_{\boldsymbol{x}\in C_i} \|\boldsymbol{x} - rp(C_i)\| \tag{3}$$

Definition 3. Assume $\mathrm{Rank}(C_i) = \langle \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{n_i}\rangle$, C_i is to be split into C_{i1} and C_{i2}, where $C_{i1} = \langle \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_j\rangle$, $C_{i2} = \langle \boldsymbol{x}_{j+1}, \boldsymbol{x}_{j+2}, \ldots, \boldsymbol{x}_{n_i}\rangle$ and $1 \leqslant j < n$. The dissimilarity function $g(C_{i1}, C_{i2})$ is defined as

$$g(C_{i1}, C_{i2}) = \|\boldsymbol{x}_{j+1} - rp(C_i)\| - \|\boldsymbol{x}_j - rp(C_i)\| \tag{4}$$

This dissimilarity definition can be compared with the distance (dissimilarity) definition of single-linkage algorithm as in

$$\mathrm{dist}(C_i, C_j) = \min_{\boldsymbol{x}_i\in C_i, \boldsymbol{x}_j\in C_j} \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \tag{5}$$

Because single-linkage is an agglomerative algorithm, the pair of clusters with minimum distance, $\mathrm{dist}(C_i, C_j)$, are merged at each step. Whereas the presented method DIVFRP is a divisive algorithm, the bipartitioned pair (C_{i1}, C_{i2}) of cluster C_i should maximize the dissimilarity function $g(C_{i1}, C_{i2})$.

2.1.2. The partition algorithm

A divisive clustering problem can be divided into two sub-problems (Savaresi et al., 2002):

Problem 1. Selecting which cluster must be split.
Problem 2. How to split the selected cluster.

For Problem 1, generally, the following three criteria can be employed for selecting the cluster to be split at each step (Savaresi et al., 2002):

(1) complete split: every cluster is split;
(2) the cluster with the largest number of objects;
(3) the cluster having maximum variance with respect to its centroid.

Since every cluster is split, criterion (1) is simple, but it does not consider the effect of splitting sequence on the quality of the clusters. Criterion (2) attempts to balance the object numbers of the clusters, but it ignores the "scatter" property. Criterion (3) considers the "scatter" property well, so we use maximum deviation which is similar to criterion (3) in DIVFRPs.

Suppose there are totally M clusters at a certain step, namely $C_0, C_1, \ldots, C_{M-1}$. One of the clusters will be selected for the further bipartition.

Definition 4. Let next_split(CS) be the cluster with the maximum deviation with respect to its centroid, $\mathrm{CS} = \{C_i : 0 \leqslant i \leqslant M-1\}$:

$$\mathrm{next\_split(CS)} = \arg\max_{C_i\in \mathrm{CS}} \|\mathrm{dev}(C_i)\| \tag{6}$$

where $\mathrm{dev}(C_i) = \frac{1}{|C_i|}\sum_{\boldsymbol{x}\in C_i}\|\boldsymbol{x} - \mu(C_i)\|$.

So next_split(CS) is the cluster to be split at next step.

An optimal partition will give the maximum value of the dissimilarity function g (Theodoridis and Koutroumbas, 2006). Bearing this in mind, we can bipartition a cluster as follows.

Definition 5. Let $D(C_i)$ be a set consisting of the differences between elements of all neighboring pairs in $\mathrm{Rank}(C_i)$:

$$D(C_i) = \{d : d = \|\mathrm{next}(\boldsymbol{x}) - rp(C_i)\| - \|\boldsymbol{x} - rp(C_i)\|\} \tag{7}$$

where $\boldsymbol{x}$ is a component in $\mathrm{Rank}(C_i)$, $\mathrm{next}(\boldsymbol{x})$ is the next component to $\boldsymbol{x}$ in $\mathrm{Rank}(C_i)$.

Definition 6. Let $b(C_i)$ be the point in C_i with the maximum difference in $D(C_i)$:

$$b(C_i) = \arg\max_{\boldsymbol{x}\in C_i}(\|\mathrm{next}(\boldsymbol{x}) - rp(C_i)\| - \|\boldsymbol{x} - rp(C_i)\|) \tag{8}$$

The cluster is bipartitioned into C_{i1} and C_{i2} as in:

$$C_{i1} = \{\boldsymbol{x} : \boldsymbol{x} \in C_i \wedge \|\boldsymbol{x} - rp(C_i)\| \leqslant \|b(C_i) - rp(C_i)\|\} \quad (9)$$

$$C_{i2} = C_i - C_{i1} \quad (10)$$

The divisive algorithm (DA) is formally stated as follows.

Divisive algorithm (DA)

Step 1. From Eq. (6), the cluster to be split is determined: $C_i = \text{next\_split}(S)$. The first cluster to be split is the initial data set which includes whole points.
Step 2. Partition C_i into two clusters with Eqs. (8)–(10), record the partition process.
Step 3. If each cluster has only one object, stop; otherwise go to step 1.
Note that recoding the partition process is for the later analysis of the optimal K.

2.2. Detecting the peaks of sum-of-error differences

After partitioning the whole data set, we will figure out the proper clusters. We classify splits into two categories: essential splits and inessential splits. The split in Fig. 1 is essential, because two expected clusters are achieved after the split. Correspondingly, the split in Fig. 2 is inessential, because only an outlier is detected with the split. Intuitively, the number of the proper essential splits is equal to the optimal number of clusters minus 1. Accordingly, the task of finding the optimal number of clusters is equivalent to that of determining the number of essential splits. By inspecting the split process, we find that the difference of sum-of-errors between an essential split and the prior inessential split has a local maximum. We call the local maximum a peak. Then the problem of determining the essential splits can be transformed to that of detecting the peaks.

As what we observed, peaks generally become lower with the split process going on. Under this circumstance, the sliding averages are employed to detect the peaks.

Definition 7. Let $J_e(i)$ be the sum-of-error after ith bipartition:

$$J_e(i) = \sum_{j=0}^{i} J_{ce}(C_j) \quad (11)$$

where $0 \leqslant i \leqslant N-1$, $J_{ce}(C_i)$ is an effective error of each cluster defined as

$$J_{ce}(C_i) = \sum_{\boldsymbol{x} \in C_i} \|x - \mu(C_i)\| \quad (12)$$

Definition 8. Let Diff be a list that consists of the differences between neighboring sum-of-errors:

$$\text{Diff} = \langle d_1, d_2, \ldots, d_i \rangle \quad (13)$$

where $1 \leqslant i \leqslant N-1$, $d_i = J_e(i-1) - J_e(i)$.

Fig. 3a illustrates the bipartitions (only first two) of the data set in Fig. 2. Fig. 3b shows $J_e(i)$ of each bipartitions. It seems difficult to perceive some information to detect cluster number K only from Fig. 3b. But in Fig. 3c two points A, B, which correspond to $J_e(0) - J_e(1)$ and $J_e(1) - J_e(2)$, respectively, are very different from the remaining points, because the first two bipartitions lead to large decreases of J_e.

Definition 9. Let $P = \langle p_0, p_1, \ldots, p_j, \ldots, p_{t-1} \rangle$ be a peak list, $p_j \in \{d_i : d_i$ is an element of Diff$\}$.

Note that if p_{j-1} and p_j in P correspond to d_u and d_v in Diff, respectively, then $v > u$ holds. Obviously, the following fact exists:

Fact 1: If the peak list P has t elements, the optimal cluster number K should be $t+1$.

The task of this phase is to construct the peak list P.

Suppose that an element of Diff, say d_j, is selected as an element of P, say p_m, if the following holds:

$$d_j \geqslant \lambda \,\text{avg}\, d(m) \quad (14)$$

where λ is a parameter; avg$d(m)$ is the average of the elements between d_e and d_j in Diff, d_e corresponds to the prior peak in P, namely, p_{m-1}. Two exceptions exist. When d_j is next to d_e in Diff, i.e. $j = e+1$, no elements exist between d_e and d_j; when p_m is first element in P, i.e. $m = 0$, d_e does not exist. As remedies, the previous average avg$d(m-1)$ is used instead of avg$d(m)$ in Eq. (14) for the former exception and the global average for the latter. Consequently, the sliding average is defined as

$$\text{avg}d(m) = \begin{cases} \frac{1}{j-e-1} \sum_{f=e+1}^{j-1} d_f & : \text{if } m \neq 0 \text{ and } j > e+1 \\ \text{avg}d(m-1) & : \text{if } m \neq 0 \text{ and } j = e+1 \\ \frac{1}{N-1} \sum_{f=1}^{N-1} d_f & : \text{if } m = 0 \end{cases} \quad (15)$$

Clearly, the window width of the sliding average is not fixed. In Fig. 3c, when we consider point A and determine whether it is a peak, we will compare its value with global average since currently the peak list P is empty and no sliding average is available.

The parameter λ in Eq. (14) is a crucial factor for detecting the peaks. Different values of λ lead to different peak lists. However, it is difficult to select a proper value for the parameter λ, because it would be different for data sets with different density distributions. Since for a given λ it is computational simple (the computational cost is linear to N) to construct a peak list, we can construct the peak lists greedily with all the possible values of λ. Intuitively, the parameter must be great than 1, and less than the value which results in the whole data set taken as one cluster. Being a real number, the λ can take a small increment, say σ, when we construct the peak lists iteratively.

Consider a function $f : S \to T$ with the Fact 1 in mind. T is a set of possible values of K, while S is a set of possible values of λ. Then $S = \{\lambda : \lambda = \omega + \sigma \times \eta, \lambda < f^{-1}(1), \omega \geqslant 1, \eta \in \mathbb{N}\}$, $T = \{k : 1 \leqslant k \leqslant N, k \in \mathbb{N}\}$, ω is the initial value of λ and σ is the increment in the construction of the peak lists. We will discuss the values of the parameters σ and ω in Section 3.1. Fig. 3d illustrates the graph of function f on the same data set.

In general, the function f is monotonically decreasing. When λ is small, the element number of the corresponding peak list is large. When the element number is greater than the optimal K, some of discovered clusters are spurious.

Definition 10. Let LBD(S) be a list of binary relations of $(\lambda, f(\lambda))$ defined as

$$\text{LBD}(S) = \langle (\min_{\lambda \in S}(\lambda), f(\min_{\lambda \in S}(\lambda))) \circ \text{LBD}(S - \{\min_{\lambda \in S}(\lambda)\}) \rangle \quad (16)$$

The list LBD(S) collects all binary relations of $(\lambda, f(\lambda))$ in an ascending order with respect to λ. In next subsection, we will consider how to eliminate the spurious clusters and consequently discover optimal cluster number K from the list LBD(S).

2.3. Eliminating the spurious clusters

By inspecting the list LBD(S), we find that some different λs produce the same number of clusters K. We call this local stability. Suppose λ_i and λ_j are the first and the last value, respectively, that lead to $f(\lambda_i) = f(\lambda_j) = K'$ in LBD(S). The local stability can be mea-

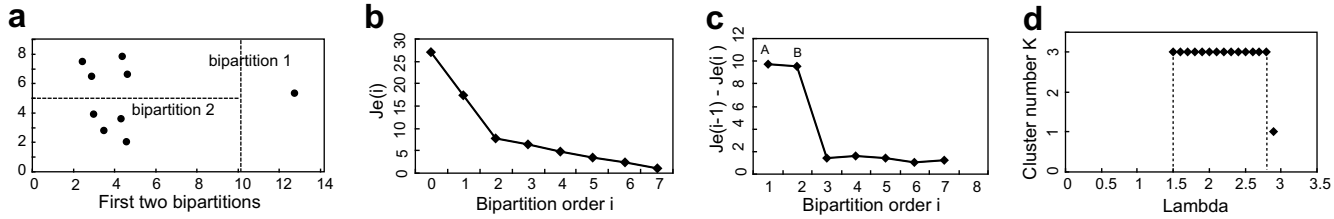


Fig. 3. Illustration of the clustering process. (a) The bipartition process of data set in Fig. 2; (b) the eight J_es for total eight bipartitions; (c) the differences of neighboring J_es, the points A and B are two potential peaks; and (d) the graph of function f.

sured by $\Delta\lambda = \lambda_j - \lambda_i$. In Fig. 3d, $f(1.5) = f(2.8) = 3$, and $\Delta\lambda = 1.3$. If the $\Delta\lambda$ is great than a threshold, say γ, the corresponding K' is regarded as a candidate of K. However, some spurious clusters may exist for a candidate of K. There are two kinds of spurious clusters. The one are the clusters consisting of outliers, the other one are the clusters partitioned from the normal clusters when K' is great than the real number K. Compared to a normal cluster, in general, a spurious cluster consists of a small number of objects and has a small effective error J_{ce} no matter how dense or sparse it is. Under this observation, we define a criterion to identify the spurious clusters.

Note that after the divisive algorithm (DA) is applied, every object is a cluster. Suppose K' is a candidate of K, the $(K'-1)$th peak in P corresponds to d_j in Diff. When we detect spurious clusters from K' clusters, the last $N-j-1$ partitioning operations are not needed. Otherwise, every cluster contains one object, it is meaningless to detect spurious clusters under this situation. For instance, considering $K' = 3$ in Fig. 3d. The 2th peak in Fig. 3c corresponds to d_2 (this is a special case, here $j = K' - 1 = 2$, generally $j \geqslant K' - 1$). When we determine that if there exist spurious clusters in the $K' = 3$ clusters which are produced by the first two bipartition (Fig. 3a), we need the object number and J_{ce} of the three clusters, but these information is not available after complete partitioning. There are two solutions for this problem. One is that the last $N-j-1 = 6$ partitions are rolled back, the other one is to record all needed information in the process of bipartitioning. For the sake of decreasing space complexity, we employ the first one, that is, the last $N-j-1$ partitioning operations are undone.

Definition 11. Let SC be a set that consists of the K' clusters, $Q(\mathrm{SC}) = \langle q_0, q_1, \ldots, q_i, \ldots, q_{K'-1}\rangle$ be an ordered list, where q_i $(0 \leqslant i \leqslant K'-1)$ is the product of the cluster number and the sum-of-error with respect to a cluster in SC, $Q(\mathrm{SC})$ is defined to be

$$Q(\mathrm{SC}) = \langle \min_{C_i \in \mathrm{SC}}(|C_i| \times J_{ce}(C_i)) \circ Q(\mathrm{SC} - \{\arg\min_{C_i \in \mathrm{SC}}(|C_i| \times J_{ce}(C_i))\})\rangle \tag{17}$$

The criterion of identifying spurious cluster is given as follows. Suppose m spurious clusters exist in SC. Because a spurious cluster comprises a small number of objects and has a small effective error, the m spurious clusters are corresponding to the first m elements in $Q(\mathrm{SC})$, namely, $q_0, q_1, \ldots, q_{m-1}$. The element q_{m-1} must satisfy:

1. $q_m \geqslant \alpha q_{m-1}$,
2. If the cluster C_{m-1} in SC is corresponding to q_{m-1} in $Q(\mathrm{SC})$, then $\max_{C_i \in \mathrm{SC}}|C_i| > \beta|C_{m-1}|$,

where α and β are two real number parameters.The criterion includes the above two conditions, which are similar to the definition of large or small clusters (He et al., 2003). The first condition indicates a relative change ratio of the normal cluster and the spurious cluster near the boundary. The second one is an absolute constraint on spurious clusters.When we apply the criterion to SC, the spurious clusters are detected, but this is not final result since there may exist a number of candidates of K. As we know, a candidate of K and the corresponding set SC is decided by λ. We repeatedly increase λ by the step of σ and apply the criterion until it converges. For a candidate of K, K'_i, suppose that s_i spurious clusters are detected from K'_i clusters in terms of the criterion, the next candidate of K is K'_{i+1}, then the convergence is defined as:

1. according to the criterion of identifying spurious cluster, no spurious cluster in SC is detected, i.e. $s_i = 0$, or
2. after the spurious clusters being removed from SC, the number of normal clusters is equal to or greater than the next candidate of K, i.e. $K'_i - s_i \geqslant K'_{i+1}$, or
3. the candidate K'_{i+1} does not exist, i.e. K'_i is the last candidate of K.

For the first situation, all clusters in SC have relatively consistent object number and J_{ce}. For the second situation, if $K'_i - s_i < K'_{i+1}$, some of spurious clusters still exist in the K'_{i+1} clusters, and we must continue to consider the candidate K'_{i+1}; otherwise, all of spurious clusters are excluded from K'_{i+1} clusters, and it is meaningless to consider K'_{i+1} for removing spurious clusters. The last situation is obvious. Based on the definition of convergence, the spurious clusters detection mechanism (SCD) is formally presented as follows. The parameters α, β, γ will be discussed in Section 3.1.

Spurious clusters detection algorithm (SCD)

Step 1. Scan the list LBD(S) from the left to the right. Find out the pair of λ, λ_i and λ_j, which satisfy:
  (1) $f(\lambda_i) = f(\lambda_j)$, subject to $i < j$, $f(\lambda_i) > f(\lambda_{i-1})$ and $f(\lambda_j) < f(\lambda_{j+1})$;
  (2) $\Delta\lambda > \gamma$, where $\Delta\lambda = \lambda_j - \lambda_i$;
Step 2. Construct the set SC which consists of K' clusters, and $Q(\mathrm{SC})$, where $K' = f(\lambda_i) = f(\lambda_j)$;
Step 3. Determine the spurious clusters according to the spurious cluster criterion;
Step 4. If the convergence is achieved, stop; otherwise go to step 1.

3. Numerical experiments

3.1. Setting parameters

From Eq. (14), the parameter λ directly determines the class number K. Since discovering the number K automatically is one of our objectives, the proper selected λ is vital for us. Greedily, we inspect all possible values of λ to find optimal K. The parameter ω decides the start point of λ. It is meaningless to assign λ a value less than or equal to 1, because each one object will be a cluster in this situation. For covering more λs, we can easily set a value to ω, usually a value between 1.2 and 2.0 is selected. In LBD(S), the parameter σ adjusts the interval between the previous λ and the next one, and consequently determines different change rate of the number K. Generally, if the interval is small, the number K will change slowly. However, a too small interval will be not helpful to reflect the change of K but result in time-consuming. The parameter

σ is set to 0.1 in all experiments. Our experiments indicate that if it is set to a smaller value, for instance 0.05, the clustering result does not change at all. The parameter γ is always 0.5. Empirically, the two parameters α and β are set to 2 and 3, respectively. In (He et al., 2003), large clusters are defined as those containing 90% of objects, or the minimum ratio of the object number of a cluster in the large group to that of a cluster in small group is 5. In our divisive method, however, the objects in the outer shell of clusters are peeled off gradually in the process of partition. Only the kernels remain. In general, the difference of the object numbers of two kernels is not as large as that of the object numbers of the two corresponding clusters. So the two parameters are relatively small. All the parameters remain unchanged in the following four experiments.

3.2. Comparing DIVFRP with other methods

In this section, two performances of DIVFRP will be compared to other algorithms. The one is the difference between the discovered cluster number K and the real K. The other is how the partitions produced by the algorithms are consistent with the real partitions. To measure the consistency, three external validity criteria, namely Fowlkes and Mallows (FM) index, Jaccard coefficient and Rand statistic, are employed. However, the three external criteria do not consider the outliers. In order to use the criteria, we redistribute each object partitioned as an outlier in DIVFRP into the corresponding cluster of which the mean is closest to the object. The two well-known clustering algorithms DBScan and K-means are selected for the comparison. To obtain the cluster number K with the two algorithms, we combine DBScan and K-means with two relative validity criteria: Calinski–Harabasz (CH) index and Silhouette index. For a same clustering method with the different input parameters, the best clustering scheme can be determined by the relative criteria (Halkidi et al., 2002), therefore the optimal cluster number K is discovered. We can change the input parameters Eps and MinPts for DBScan and the cluster number K for K-means to produce the best clustering scheme. Note that DBScan can produce outliers too. Similarly, the outliers are redistributed into the nearest clusters for the purpose of comparison.

CH Index is defined as follows

$$\mathrm{CH} = \frac{Tr(B)/(K-1)}{Tr(W)/(N-K)} \tag{18}$$

where K is the cluster number and N is the object number, and

$$Tr(B) = \sum_{i=1}^{K} |C_i| \times \|\mu(C_i) - \mu(C)\|^2 \tag{19}$$

where $\mu(C) = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i$;

$$Tr(W) = \sum_{k=1}^{K} \sum_{i=1}^{|C_i|} \|\boldsymbol{x}_i - \mu(C_i)\|^2 \tag{20}$$

Silhouette index is defined as follows

$$S = \frac{1}{N}\sum_{i=1}^{N} \frac{b_i - a_i}{\max(a_i, b_i)} \tag{21}$$

where K is the cluster number and N is the object number and

$$a_i = \frac{1}{|C_j| - 1} \sum_{\boldsymbol{y}\in C_j,\ \boldsymbol{y}\neq \boldsymbol{x}_i} \|\boldsymbol{y} - \boldsymbol{x}_i\| \tag{22}$$

$$b_i = \min_{l\in H,\ l\neq j} \frac{1}{|C_l|} \sum_{\boldsymbol{y}\in C_l} \|\boldsymbol{y} - \boldsymbol{x}_i\| \tag{23}$$

where $\boldsymbol{x}_i \in C_j,\ H = \{h : 1 \leqslant h \leqslant K,\ h \in \mathbb{N}\}$.

First we demonstrate the performances of DIVFRP with two synthetic 2D data sets R15 and D31 (Veenman et al., 2002) of which the coordinates are transformed linearly and slightly. The data set R15 which is shown in Fig. 4a has 15 Gaussian clusters which are arranged in rings, and each cluster contains 40 objects. Fig. 4b shows the clustering result of DIVFRP with the outliers redistributed into the nearest clusters. Fig. 5 shows the curve of the cluster number K and the λ based on LBD(S). Clearly, only horizontal line segments in the curve may denote the candidates of K, because a horizontal line segment denotes $f(\lambda_i) = f(\lambda_j) = K'$, where (λ_i, K') and (λ_j, K') are the two endpoints of the line segment. In general, the first candidate of K lies on the knee of the curve. In Fig. 5, the first horizontal line segment $\overline{AB}$ is selected to produce the initial candidate of K, as the length of horizontal line segment (namely $\Delta\lambda$) is 12.7 and much greater than γ. The K' determined by

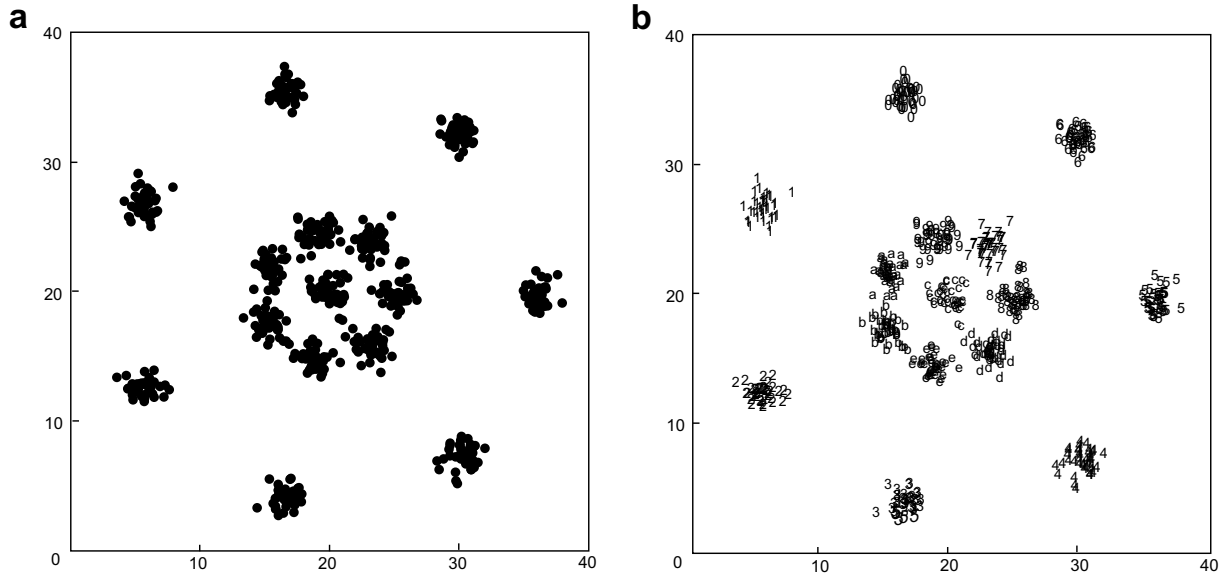


Fig. 4. (a) The original data set of R15 and (b) the clustering result of DIVFRP for R15 with the outliers redistributed into the closest clusters.

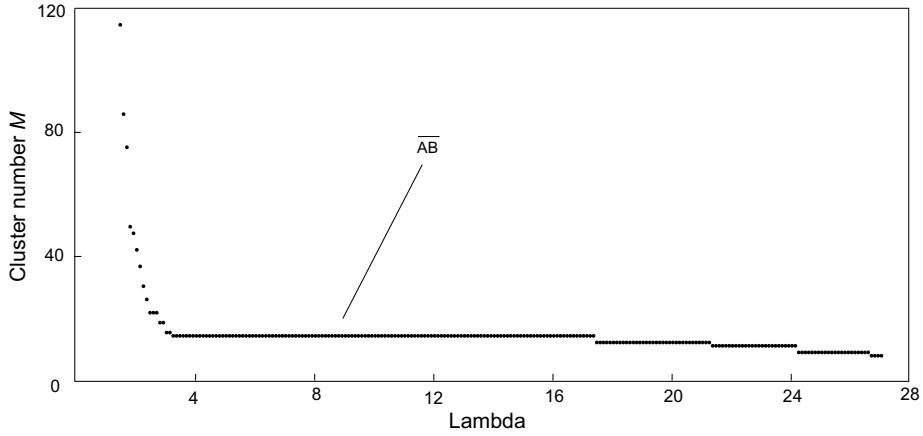


Fig. 5. The curve of λ and the cluster number K for the data set R15. The horizontal line segment $\overline{AB}$ is qualified to produce a candidate of K, and the algorithm SCD converges at this line segment.

$\overline{AB}$ is 15. According to the criterion of spurious cluster, no spurious cluster exists in the corresponding $Q(SC)$. Consequently, the spurious clusters detection algorithm (SCD) converges after first iteration, and the final cluster number K is 15. Then we consider DBScan algorithm. To achieve optimal clustering result, we run DBScan algorithm repeatedly with the change of the parameter Eps from 0.1 to 10 and the parameter MinPts from 5 to 50. Two increments for the changes are 0.1 and 1, respectively. For K-means algorithm, we change the parameter K from 2 to $N-1$ with increment of 1 in all experiments. Even for a same parameter K, the clustering results of K-means probably are quit different because of the initial cluster centers selected randomly. Accordingly, we consider the average performance of 10 runs of K-means in all the comparisons. CH index and Silhouette index are computed in each run of both DBScan and K-means algorithms. The optimal results have maximum CH index or Silhouette index values. Table 1 provides the comparison of DIVFRP, DBScan and K-means. The comparison focuses on two performances mentioned above, namely the cluster number K and partition consistency. Clearly, DIVFRP, CH index-based DBScan and Silhouette index-based DBScan achieve same perfect results. They discover the cluster number K correctly, and three external indices are close to the expected value 1. However, because K-means algorithm may converge at local minimum, both CH index-based K-means and Silhouette index-based K-means have unsatisfied results.

Table 1
Performances of applying the methods to R15 data set

Method	Estimated K	Rand	Jaccard	FM
DIVFRP	15	0.9991	0.9866	0.9932
DBScan–CH	15	0.9991	0.9866	0.9932
DBScan–Silhouette	15	0.9991	0.9866	0.9932
K-means–CH	18.7	0.9917	0.8728	0.9333
K-means–Silhouette	18.1	0.9908	0.8688	0.9302

The second synthetic data set D31 comprises 31 Gaussian clusters, and each cluster has 100 objects. Fig. 6a shows the original data set and clustering result of DIVFRP is shown in Fig. 6b. In Fig. 7, the curve of the cluster number and λ is shown. $\overline{AB}$ is the first horizontal line segment with the length greater than γ. As a result, the K' determined by $\overline{AB}$ is the initial candidate of K. The succeeding horizontal line segments $\overline{CD}$ and $\overline{EF}$ are qualified to produce the

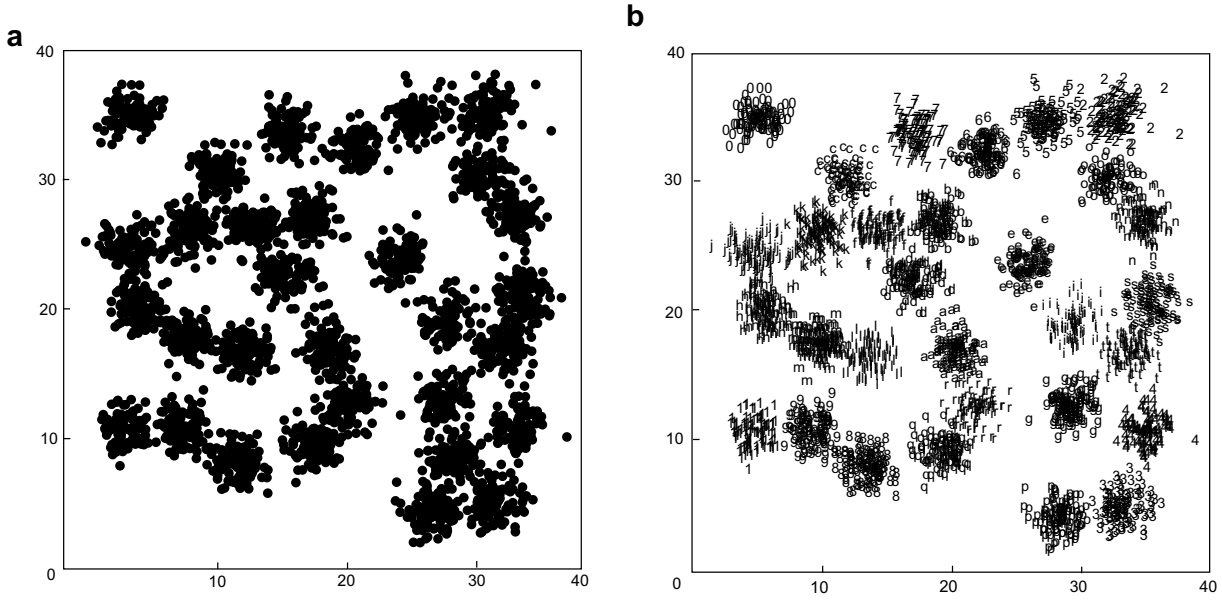


Fig. 6. (a) The original data set of D31 and (b) the clustering result of DIVFRP for D31 with the outliers redistributed into the closest clusters.

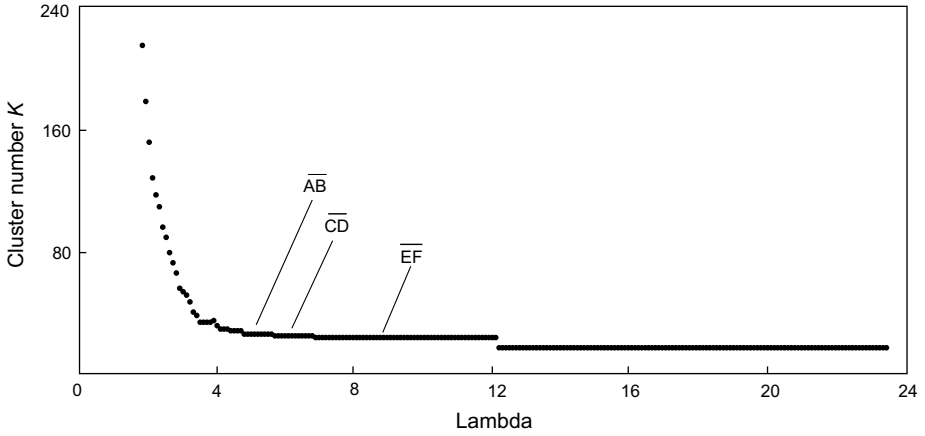


Fig. 7. The curve of λ and the cluster number K for the data set D31. The horizontal line segment $\overline{AB}$, $\overline{CD}$ and $\overline{EF}$ are qualified to produce the candidates of K. The algorithm SCD converges at line segment $\overline{EF}$.

candidates of K, since the lengths of the two line segments are 1.2 and 54, respectively. We apply SCD to the candidates of K. For the first $K' = 33$ denoted by $\overline{AB}$, there are two spurious clusters detected. We move to the line segment $\overline{CD}$, the corresponding K' is 32. Similarly, one spurious cluster is discerned by SCD. Then we consider the line segment $\overline{EF}$ with $K' = 31$. Since this iteration of SCD gives no spurious cluster, SCD converges at this line segment, and accordingly the final number of clusters is 31. As for DBScan, we set the parameter Eps on D31 from 0.1 to 5.0 with increment 0.1, MinPts from 10 to 40 with increment 1, and run it repeatedly. From the Table 2, we observe that both DIVFRP and DBScan determine the cluster number K precisely, but K-means algorithms do not. Moreover, the three external indices demonstrate that the partition qualities of the two DBScan algorithms are slimly better than that of DIVFRP, and at the same time the Silhouette index-based DBScan outperforms CH index-based DBScan slightly.

Next, we experiment on two real data sets IRIS and WINE. The well-known data set IRIS has three clusters of 50 objects each, and each object has four attributes. The curve about K and λ, which is produced by applying DIVFRP to IRIS data set, is displayed in Fig. 8a. Both horizontal line segment $\overline{AB}$ and $\overline{CD}$ in the curve have the length of 1.3 which is greater than γ. Therefore, the two K's denoted by $\overline{AB}$ and $\overline{CD}$, respectively, are the two candidates of K. As SCD identifies one spurious cluster from K' ($K' = 4$) clusters corresponding to $\overline{AB}$ and no spurious cluster from K' ($K' = 3$) clusters corresponding to $\overline{CD}$, the final cluster number is determined to 3.

Table 2
Performances of applying the methods to D31 data set

Method	Estimated K	Rand	Jaccard	FM
DIVFRP	31	0.9969	0.9083	0.9520
DBScan–CH	31	0.9970	0.9116	0.9537
DBScan–Silhouette	31	0.9971	0.9127	0.9543
K-means–CH	38.9	0.9909	0.7442	0.8541
K-means–Silhouette	28.7	0.9803	0.5854	0.7469

Table 3
Performances of applying the methods to IRIS data set

Method	Estimated K	Rand	Jaccard	FM
DIVFRP	3	0.8859	0.7071	0.8284
DBScan–CH	3	0.8797	0.6958	0.8208
DBScan–Silhouette	2	0.7762	0.5951	0.7714
K-means–CH	2.9	0.7142	0.4768	0.6570
K-means–Silhouette	2	0.7636	0.5723	0.7504

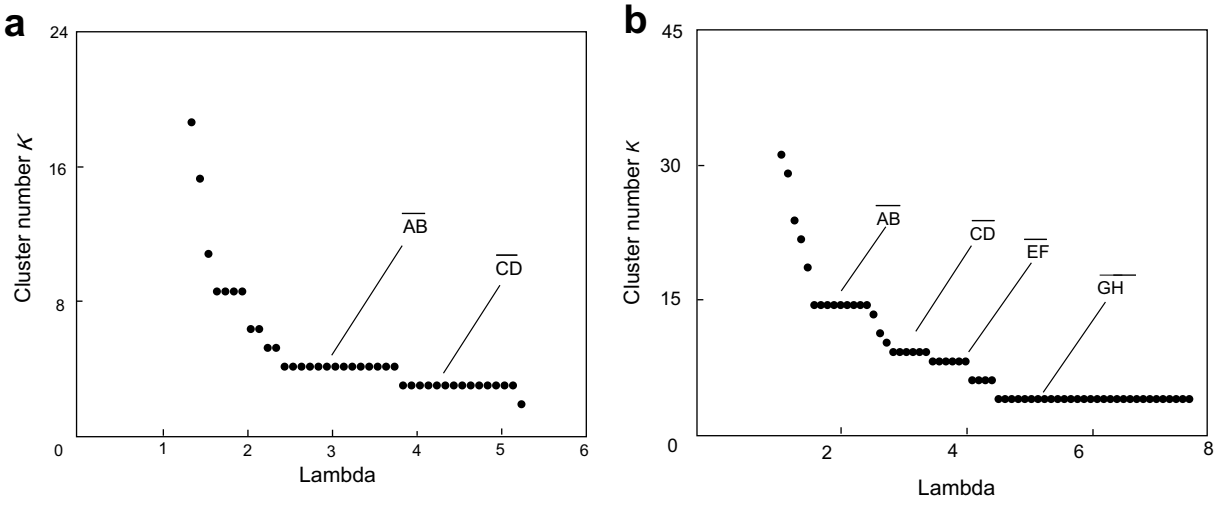


Fig. 8. (a) The curve of λ and the cluster number K for the data set IRIS. The algorithm SCD converges at line segment $\overline{CD}$. (b) The curve of λ and the cluster number K for the data set WINE. The algorithm SCD converges at line segment $\overline{EF}$.

Table 4
Performances of applying the methods to WINE data set

Method	Estimated K	Rand	Jaccard	FM
DIVFRP	3	0.7225	0.4167	0.5883
DBScan–CH	6	0.7078	0.2656	0.4467
DBScan–Silhouette	2	0.6836	0.4623	0.6474
K-means–CH	40.6	0.6660	0.0453	0.1639
K-means–Silhouette	2	0.6688	0.4638	0.6549

Table 3 depicts the comparison of the performances of the algorithms. The parameter ranges and increments of DBScan for IRIS data set are the same as for D31 data set. DIVFRP and CH index base DBScan achieve similar results and outperform the three remaining methods.

The data set WINE contains the chemical analyses of 178 kinds of wines from Italy on 13 aspects. The 178 entries are categorized into three clusters with 48, 59, and 71 entries for each cluster. SCD

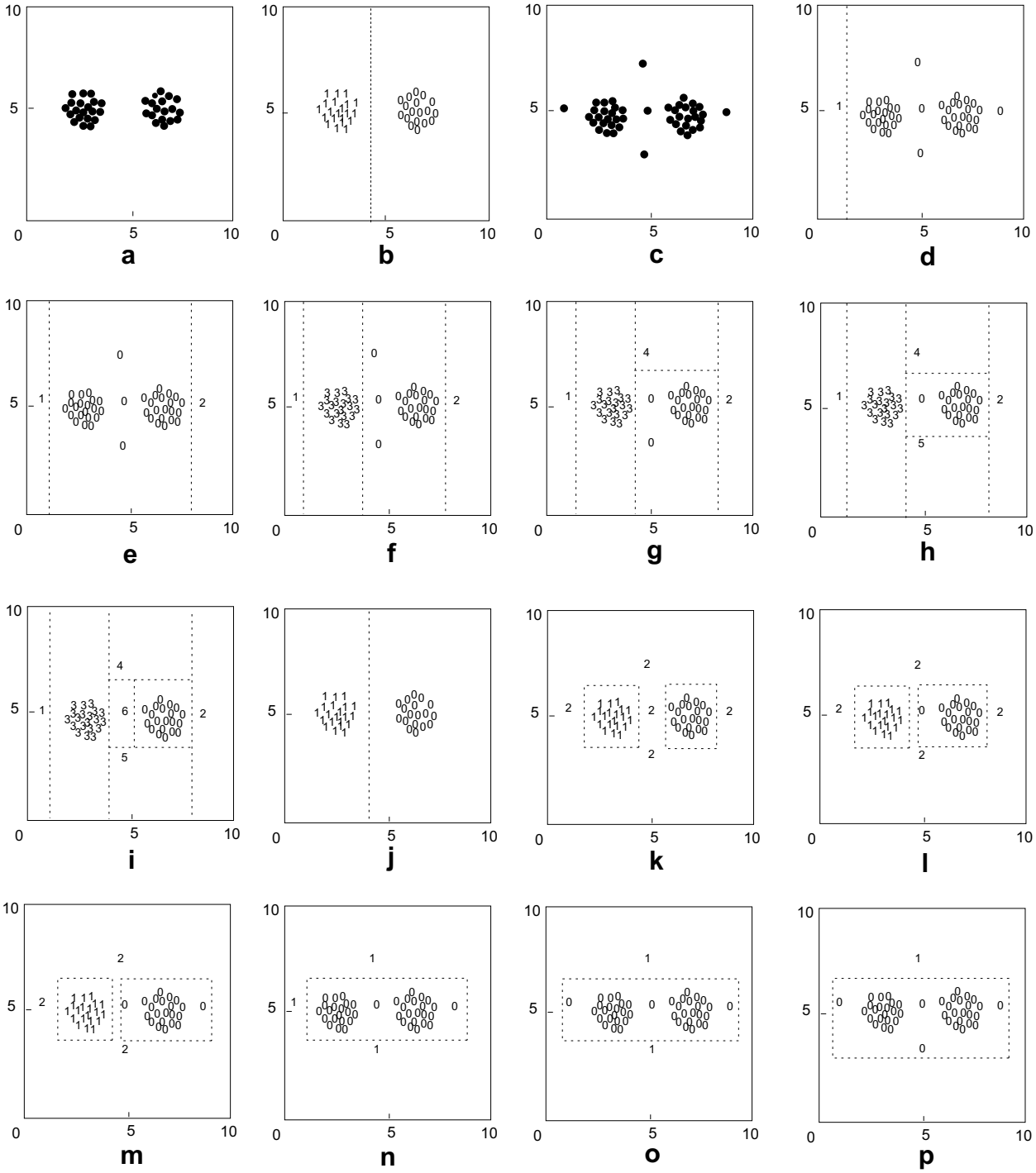


Fig. 9. The figures illustrate the effect of outliers on DIVFRP and DBScan. In (a), a data set contains two clusters. In (b), the first partition of DIVFRP identifies the two clusters. In (c), a data set contains five outliers and two clusters. In (d, e) and (g–i), each partition peels off one outlier, while an expected cluster is partitioned in Fig. 9f. In (j), DBScan detects the two clusters with MinPts = 5 and $0.60 \leqslant \text{Eps} \leqslant 2.10$. In (k), the outliers are identified by DBScan with MinPts = 5 and $0.60 \leqslant \text{Eps} \leqslant 1.01$. In (l), DBScan with MinPts = 5 and $1.02 \leqslant \text{Eps} \leqslant 1.20$. In (m), DBScan with MinPts = 5 and $1.21 \leqslant \text{Eps} \leqslant 1.24$. In (n), DBScan with MinPts = 5 and $1.25 \leqslant \text{Eps} \leqslant 1.29$. In (o), DBScan with MinPts = 5 and $1.30 \leqslant \text{Eps} \leqslant 1.72$. In (p), DBScan with MinPts = 5 and $1.73 \leqslant \text{Eps} \leqslant 2.10$.

converges at the horizontal line segment $\overline{GH}$ to which the corresponding K' is 4, see Fig. 8b. Although SCD identifies a spurious cluster, $K' = 4$ is the last candidate of K in this data set. In terms of item three of convergence criterion, SCD converges and the final estimated number of clusters is $K = K' - 1 = 3$. In Table 4, only DIVFRP estimates the cluster number correctly. Silhouette index-based DBScan and K-means achieve better results than CH index-based DBScan and K-means.

4. Discussion

From the four experiments, we observe that DIVFRP figures out the optimum cluster number and achieves good results compared to validity indices-based DBScan and K-means. In addition, DIVFRP is robust to outliers.

Outliers in a data set are those random objects that are very different from others and do not belong to any clusters (Lin and Chen, 2005). In general, the presence of outliers may deteriorate the result of a clustering method. For this kind of clustering methods, some outlier detection mechanisms can be employed to remove the outliers before the partition process. However, a clustering method robust to outliers is more expected. DIVFRP is equipped with this property. Fig. 9 shows the partition processes of DIVFRP on two similar data sets. The one data set contains two clusters only and is shown in Fig. 9a. The other one combines the two clusters with another five outliers and is shown in Fig. 9c. For the data set in Fig. 9a, one split is enough for partitioning the two clusters with DIVFRP, see Fig. 9b. When there exist extra five outliers in the data set, six splits are needed, which are depicted from Fig. 9d to Fig. 9i. Actually, the effect of five splits of the six is to peel off the five outliers. In other words, the existence of outliers does not change final clustering result but only leads to more partitions, and none parameters are involved in the whole partitioning process. Note that some adjacent outliers peeled off at one partition may be regarded as a cluster in the peak detection process. However, it can be removed as a spurious cluster in SCD. Fig. 9j–p illustrates the clustering of DBScan on the same data sets. When no outliers exist, in Fig. 9j, DBScan can discover the two clusters with MinPts $= 5$ and $0.60 \leqslant \text{Eps} \leqslant 2.10$. For the data set with five outliers, although DBScan can still discover the two clusters with MinPts $= 5$ in Fig. 9k, the range of possible values of Eps is reduced to $[0.60, 1.01]$. Fig. 9l–p shows that when MinPts keeps unchanged, the remaining values of Eps, $[1.02, 2.10]$, result in poor clusters. The experiments on DBScan indicate that the parameters of DBScan are sensitive to outliers. Therefore, compared with DBScan, we can say that DIVFRP is robust to outliers.

Generally, the computational cost of a divisive clustering method is very expensive, because there are $2^{n_i-1} - 1$ possibilities to bipartition a cluster C_i with n_i objects in a divisive algorithm. However, DIVFRP employs furthest reference points to bipartition a cluster optimally, and does not need to consider the complete enumeration of all possible bipartitions. With this property, the computational cost of the partition scheme of DIVFRP is lower than that of general divisive clustering methods, even some agglomerative methods. For a data set with N objects, for example, the computational cost of DIVFRP is lower than that of single-linkage ($O(N^2 \log N)$). Because in DIVFRP, totally there are $N - 1$ bipartitions after every object becomes a cluster, and the computational cost of each bipartition is $O(n_i \log n_i)$, where n_i is the object number of a cluster to be bipartitioned and it is less than N except the first bipartition in which n_i is N.

However, DIVFRP has some drawbacks. It can only find out clusters in spherical shape. This drawback results from that DIVFRP uses Euclidean distance and takes the furthest points as reference points. With different reference point, clusters in different shapes may be detected. For example, if we take the mean points as reference points instead of the furthest points, the clusters in ring shape with close centers can be identified. Another drawback is that when some valid clusters are quit dense, they may be mis-detected as spurious clusters. If lots of objects locate at an almost identical position, their $|C_i| \times J_{ce}(C_i)$ would be very small, which is employed to construct $Q(\text{SC})$. In the future work, we will explore and improve the criterion of detecting spurious cluster to overcome this drawback. In addition, similar to classical hierarchical clustering methods, DIVFRP is also incapable of dealing with large-scale data sets because of its quadratic computational cost. For high-dimensional data sets, distance-based clustering algorithms are ineffective (Xu and Wunsch, 2005). DIVFRP is very a distance-based algorithm, but we can employ dimensionality reduction methods to preprocess high-dimensional data, and then apply DIVFRP to the dimensionality reduced data sets.

5. Conclusion

In this paper, we present an automatic divisive hierarchical algorithm based on furthest reference points (DIVFRP). It contains three phases: partitioning data set, detecting the peaks of sum-of-error differences and eliminating spurious clusters. In partitioning data set phase, other than general divisive hierarchical clustering algorithms, DIVFRP employs a novel dissimilarity function which takes the furthest points as reference points. With the dissimilarity function, the computational cost of partitioning data set is less than $O(N^2 \log N)$. Sliding average is used to detect the peaks in second phase. In the third phase, the spurious clusters are removed and the optimal cluster number K is determined. The experiments on both artificial and real data sets show that DIVFRP can automatically and precisely detect the number of clusters and achieve a good clustering quality. In addition, the presence of outliers does not degrade the quality of clustering results, since DIVFRP can peel off the outliers bit by bit.

Acknowledgements

The paper is supported by the National Natural Science Foundation of China, Grant No. 60775036, and Ningbo University, Grant No. 200460. The authors are grateful to Jian Yu, Cor J. Veenman and Yuntao Qian for their valuable comments and suggestions.

References

Aggarwal, C., Yu, P., 2001. Outlier detection for high dimensional data. In: Proc. SIGMOD'01, Santa Barbara, CA, USA, pp. 37–46.

Bandyopadhyay, S., Maulik, U., 2001. Nonparametric genetic clustering: Comparison of validity indices. IEEE Trans. Systems Man Cybernet. – Part C: Appl. Rev. 31 (1).

Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J., 2000. LOF: Identifying density-based local outliers. In: Proc. SIGMOD'00, Dallas, Texas, pp. 427–438.

Chavent, M., Lechevallier, Y., Briant, O., 2007. DIVCLUS-T: A monothetic divisive hierarchical clustering method. Comput Statist. Data Anal. 52 (2), 687–701.

Ester, M., Kriegel, H.P., Sander, J., Xu, X.,1996. A density based algorithm for discovering clusters in large spatial databases. In: Proc. KDD'96, Portland OR, USA, pp. 226–231.

Garai, G., Chaudhuri, B.B., 2004. A novel genetic algorithm for automatic clustering. Pattern Recognition Lett. 25, 173–187.

Gowda, K.C., Ravi, T.V., 1995. Divisive clustering of symbolic objects using the concept of both similarity and dissimilarity. Pattern Recognition 28 (8), 1277–1282.

Guha, S., Rastogi, R., Shim, K., 1998. CURE: An efficient clustering algorithm for large databases. In: Proc. ACM SIGMOD Internat. Conf. Management Data. ACM Press, New York, pp. 73–84.

Halkidi, M., Batistakis, Y., Vazirgiannis, M., 2002. Clustering validity checking methods: Part II. ACM SIDMOD Record 31 (3), 19–27.

He, Z., Xu, X., Deng, S., 2003. Discovering cluster-based local outliers. Pattern Recognition Lett. 24, 641–1650.

Jain, A.K., Murty, M.N., Flynn, P.J., 1999. Data clustering: A review. ACM Comput. Surveys 31 (3), 264–323.

Kaufman, L., Rousseeuw, P.J., 1990. Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, New York.

Knorr, E.M., Ng, R.T., 1998. Algorithms for mining distance based outliers in large datasets. In: Proc. VLDB'98, New York, USA, pp. 392–403.

Lin, C.R., Chen, M.S., 2005. Combining partitional and hierarchical algorithms for robust and efficient data clustering with cohesion self-merging. IEEE Trans. Knowledge Data Eng. 17 (2), 145–159.

MacQueen, J.B., 1967. Some methods for classification and analysis of multivariate observations. In: Proc. 5th Berkeley Symposium Mathematical Statistical Probability, vol. 1, pp. 281–297.

Ng, R.T., Han, J., 1994. Efficient and effective clustering methods for spatial data mining. In: Bocca, J.B., Jarke, M., Zaniolo, C. (Eds.), Proc. 20th Internat. Conf. on Very Large Data Bases (VLDB'94). Morgan Kaufmann, Santiago, pp. 144–155.

Patan, G., Russo, M., 2002. Fully automatic clustering system. IEEE Trans. Neural Networks 13 (6).

Ramaswamy, S., Rastogi, R., Kyuseok, S., 2000. Efficient algorithms for mining outliers from large data sets. In: Proc. SIGMOD'00, Dallas, Texas, pp. 93–104.

Savaresi, S.M., Boley, D.L., Bittanti, S., Gazzaniga, G., 2002. Cluster selection in divisive clustering algorithms. In: Proc. 2nd SIAM ICDM, Arlington, VA, pp. 299–314.

Theodoridis, S., Koutroumbas, K., 2006. Pattern Recognition, third ed. Academic Press Inc., Orlando, FL.

Tseng, V.S., Kao, C.P., 2005. Efficiently mining gene expression data via a novel parameterless clustering method. IEEE/ACM Trans. Comput. Bioinformatics 2 (4) (October–December).

Tseng, L.Y., Yang, S.B., 2001. A genetic approach to the automatic clustering problem. Pattern Recognition 34, 415–424.

Veenman, C.J., Marcel, J.T., Reinders, M.J.T., Backer, E., 2002. A maximum cluster algorithm. IEEE Trans. Pattern Anal. Machine Intell. 24 (9), 1273–1280.

Wang, X., Qiu, W., Zamar, R.H., 2007. CLUES: A non-parametric clustering method based on local shrinking. Comput. Statist. Data Anal. 52 (1), 286–298.

Xu, R., Wunsch II, D., 2005. Survey of clustering algorithms. IEEE Trans. Neural Networks 16 (3), 645–678.

2

Paper **P2**

Reprinted from Pattern Recognition, Volume 42, Issue 5, C. Zhong, D. Miao, "A comment on 'Using locally estimated geodesic distance to optimize neighborhood graph for isometric data embedding'," Pages 1012–1013,

Pattern Recognition 42 (2009) 1012 – 1013

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

A comment on "Using locally estimated geodesic distance to optimize neighborhood graph for isometric data embedding"

Caiming Zhong[a,b,c,*], Duoqian Miao[a,c]

[a]School of Electronics and Information Engineering, Tongji University, Shanghai 201804, PR China
[b]College of Science and Technology, Ningbo University, Ningbo 315211, PR China
[c]Tongji Branch, National Engineering and Technology Center of High Performance Computer, Shanghai 201804, PR China

ARTICLE INFO

Article history:
Received 21 July 2008
Accepted 11 November 2008

Keywords:
Triangle inequality
Geodesic distance
Euclidean distance

ABSTRACT

A geodesic distance-based approach to build the neighborhood graph for isometric embedding is proposed to deal with the highly twisted and folded manifold by Wen et al. [Using locally estimated geodesic distance to optimize neighborhood graph for isometric data embedding, Pattern Recognition 41 (2008) 2226–2236]. This comment is to identify the error in their example and the ineffectiveness of their algorithm.

Wen et al. recently proposed an approach which deals with highly twisted and folded manifold for isometric data embedding [1]. The approach employs locally estimated geodesic distances to optimize a neighborhood graph which is usually constructed with Euclidean distances in some isometric embedding methods such as Isomap [2]. Unfortunately, the example given in Ref. [1] is incorrect, and the algorithm OptimizeNeighborhoodbyGeod(X, k, m, d) is ineffective. This comment aims at identifying the errors.

In Ref. [1], the initial neighborhood is determined by Euclidean distance, and then the local geodesic distance is estimated. Fig. 1, which is corresponding to Fig. 2 in Ref. [1], illustrates the process of the estimation. Let $N(x)$ be a set of Euclidean distance based three nearest neighbors of a data point x, then $N(x)=\{x_1,x_2,x_3\}$, and $N(x_1)=\{x_{11},x_{12},x_{13}\}$. Let $d(x,y)$ be the Euclidean distance between data point x and y. As x_1 is a neighbor of x, and x_{11} is neighbor of x_1, applying triangle inequality theorem, we have

$$d(x,x_{11}) \leqslant d(x,x_1)+d(x_1,x_{11}). \tag{1}$$

Furthermore, x_{11} is not a neighbor of x, that implies

$$d(x,x_{11}) > d(x,x_i), \quad i=1,2,3. \tag{2}$$

From (1) and (2), we obtain

$$d(x,x_1)+d(x_1,x_{11}) > d(x,x_i), \quad i=1,2,3. \tag{3}$$

That is to say, in Fig. 1, $d(x,x_1)=2$, $d(x_1,x_{11})=5$ and $d(x,x_3)=12$ cannot exist simultaneously. Accordingly, x_3 cannot be optimized into x_{11}, and for the same reason, x_2 cannot be optimized into x_{12}.

Based on the above analysis, the algorithm OptimizeNeighborhoodbyGeod(X, k, m, d) given in Ref. [1] is ineffective. The algorithm is as follows.

Algorithm 1. OptimizeNeighborhoodbyGeod(X, k, m, d).

/* $X=x_i$ be the high dimensional data set, k be the neighborhood size, m be the scope for locally estimating geodesic distances, d be the dimension of the embedding space, and $m<k$. The output is the optimized neighborhood set $N=\{N(x_i)\}$ for all points */

(1) Calculate the neighborhood $N(x_i)$ for any point x_i using Euclidean distance d_e, where $N(x_i)$ is sorted ascendingly. Let $d_g(x_i,x_j)=d_e(x_i,x_j)$ for the pairs of all points.
(2) For $i=1$ to $|X|$, where $|X|$ is the number of points in X.
(3) · For $j=1$ to k
(4) ·· Select jth point from $N(x_i)$, denoted as x_{ij}
(5) ·· For $p=1$ to m
(6) ··· Select pth point from $N(x_{ij})$, denoted as x_{ijp}
(7) ···If $d_g(x_i,x_{ij})+d_g(x_{ij},x_{ijp})<d_g(x_i,x_{ik})$ and $x_{ijp}\notin N(x_i)$ and parent $(x_{ijp})\in N(x_i)$
(8) ····· Delete x_{ik} from $N(x_i)$
(9) ····· Insert x_{ijp} into $N(x_i)$ ascendingly

DOI of original articles: 10.1016/j.patcog.2007.12.015, 10.1016/j.patcog.2008.11.002.
*Corresponding author at: School of Electronics and Information Engineering, Tongji University, Shanghai 201804, PR China. Tel.: +86 21 69589867.
E-mail addresses: charman_zhong@hotmail.com, zhongcaiming@nbu.edu.cn (C. Zhong).

0031-3203/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2008.11.007

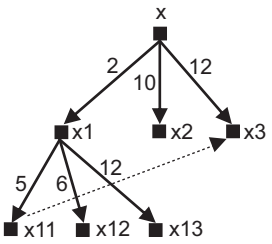


Fig. 1. Example to optimizing the neighborhood of the point x.

(10) $\cdots\cdots d_g(x_i, x_{ijp}) = d_g(x_i, x_{ij}) + d_g(x_{ij}, x_{ijp})$
(11) $\cdots\cdots$ Let $j = 1$ and break
(12) $\cdots$ End
(13) $\cdot\cdot$ End
(14) $\cdot$ End
(15) End
(16) $N = N(x_i)$ be the optimized neighborhood for all points in X

In the step 1, since $d_g(x_i, x_j) = d_e(x_i, x_j)$ for the pairs of all points, i.e. all the local geodesic distances are initialized to corresponding Euclidean distances. According to the analysis on the example, the condition of step 7 is never satisfied, and the block from steps 8 to 11 is never executed. Consequently, the algorithm is ineffective.

References

[1] G. Wen, L. Jiang, J. Wen, Using locally estimated geodesic distance to optimize neighborhood graph for isometric data embedding, Pattern Recognition 41 (2008) 2226–2236.
[2] J.B. Tenenbaum, V. de Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (2000) 2319–2323.

About the Author—CAIMING ZHONG is currently pursuing his Ph.D. at Tongji University, Shanghai, China. His research interests include cluster analysis, manifold learning and image segmentation.

About the Author—DUOQIAN MIAO is a professor of Department of Computer Science and Technology at Tongji University, Shanghai, China. He has published more than 40 papers in international proceedings and journals. His research interests include soft computing, rough sets, pattern recognition, data mining, machine learning and granular computing.

3

Paper **P3**

Reprinted from Pattern Recognition, Volume 43, Issue 3, C. Zhong, D. Miao, R. Wang, "A graph - theoretical clustering method based on two rounds of minimum spanning trees, " Pages 752 – 766, Copyright (2010), with permission from Elsevier.

Pattern Recognition 43 (2010) 752–766

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

A graph-theoretical clustering method based on two rounds of minimum spanning trees

Caiming Zhong[a,b,c], Duoqian Miao[a,b,*], Ruizhi Wang[a,b]

[a]Department of Computer Science and Technology, Tongji University, Shanghai 201804, PR China
[b]Key Laboratory of Embedded System & Service Computing, Ministry of Education of China, Shanghai 201804, PR China
[c]College of Science and Technology, Ningbo University, Ningbo 315211, PR China

ARTICLE INFO

Article history:
Received 22 January 2009
Received in revised form 19 July 2009
Accepted 24 July 2009

Keywords:
Graph-based clustering
Well-separated cluster
Touching cluster
Two rounds of MST

ABSTRACT

Many clustering approaches have been proposed in the literature, but most of them are vulnerable to the different cluster sizes, shapes and densities. In this paper, we present a graph-theoretical clustering method which is robust to the difference. Based on the graph composed of two rounds of minimum spanning trees (MST), the proposed method (2-MSTClus) classifies cluster problems into two groups, i.e. separated cluster problems and touching cluster problems, and identifies the two groups of cluster problems automatically. It contains two clustering algorithms which deal with separated clusters and touching clusters in two phases, respectively. In the first phase, two round minimum spanning trees are employed to construct a graph and detect separated clusters which cover distance separated and density separated clusters. In the second phase, touching clusters, which are subgroups produced in the first phase, can be partitioned by comparing cuts, respectively, on the two round minimum spanning trees. The proposed method is robust to the varied cluster sizes, shapes and densities, and can discover the number of clusters. Experimental results on synthetic and real datasets demonstrate the performance of the proposed method.

1. Introduction

The main goal of clustering is to partition a dataset into clusters in terms of its intrinsic structure, without resorting to any a priori knowledge such as the number of clusters, the distribution of the data elements, etc. Clustering is a powerful tool and has been studied and applied in many research areas, which include image segmentation [1,2], machine learning, data mining [3], and bioinformatics [4,5]. Although many clustering methods have been proposed in the recent decades, there is no universal one that can deal with all cluster problems, since in the real world clusters may be of arbitrary shapes, varied densities and unbalanced sizes [6,7]. In addition, Kleinberg [8] presented an impossibility theorem to indicate that it is difficult to develop a universal clustering scheme. However, in general, users have not any a priori knowledge on their datasets, which makes it a tough task for them to select suitable clustering methods. This is the dilemma of clustering.

Two techniques have been proposed and studied to alleviate the dilemma partially, i.e. clustering ensemble [9–11] and multiobjective clustering [12]. The basic idea of a clustering ensemble is to use different data representation, apply different clustering methods with varied parameters, collect multiple clustering results, and discover a cluster with better quality [13]. Fred and Jain [13] proposed a co-association matrix to depict and combine the different clustering results by exploring the idea of evidence accumulation. Topchy et al. [10] proposed a probabilistic model of consensus with a finite mixture of multinomial distributions in a space of clusterings, and used the EM algorithm to find the combined partitions. Taking advantage of correlation clustering [14], Gionis et al. [11] presented a clustering aggregation framework, which can find a new clustering that minimizes the total number of disagreements with all the given clusterings. Being different from a clustering ensemble which is limited to the posteriori integration of the solutions returned by the individual algorithms, multiobjective clustering considers the multiple clustering objective functions simultaneously, and trades off solutions during the clustering process [12]. Compared with the individual clustering approach, both clustering ensembles and multiobjective clustering can produce more robust partitions and higher cluster qualities. In addition, some of other clustering methods can automatically cope with arbitrary shaped and non-homogeneous clusters [15].

* Corresponding author at: Department of Computer Science and Technology, Tongji University, Shanghai 201804, PR China. Tel.: +86 21 69589867.
E-mail addresses: charman_zhong@hotmail.com (C. Zhong), miaoduoqian@163.com (D. Miao).

0031-3203/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2009.07.010

Recently more attention has been paid to graph-based clustering methods. Being an intuitive and effective data representation approach, graphs have been employed in some clustering methods [16–25]. Obviously, the tasks of these kinds of methods include constructing a suitable graph and partitioning the graph effectively. Most graph-based clustering methods construct the graphs using k nearest neighbors [16,17]. Karypis et al. in CHAMELEON [16] represented a dataset with k nearest neighbor graph, and used relative interconnectivity and relative closeness to partition the graph and merge the partitions so that it can detect arbitrary shaped and connected clusters. This varies from data representation of CHAMELEON, in which a vertex denotes a data item, Fränti et al. employed a vertex to represent a cluster so as to speed up the process of clustering [17]. Other graph-based methods take advantage of minimum spanning trees (MST) to represent a dataset [18,19]. Zahn [18] divided a dataset into different groups in terms of their intrinsic structures, and conquered them with different schemes. Xu et al. [19] provided three approaches to cluster datasets, i.e. clustering through removing long MST-edges, an iterative clustering algorithm, and a globally optimal clustering algorithm. Although the methods of Zahn and Xu are effective for datasets with specific structures, users do not know how to select reasonable methods since they have no information about the structures of their datasets. From a statistical viewpoint, González-Barrios [20] identified clusters by comparing k nearest neighbor-based graph and the MST of a dataset. The limitation of González-Barrios's method is that only i.i.d. data are considered. Päivinen [21] combined a scale-free structure with MST to form a scale-free minimum spanning tree (SFMST) of a dataset, and determined the clusters and branches from the SFMST. Spectral clustering is another group of graph-based clustering algorithms [22]. Usually, in a spectral clustering, a fully connected graph is considered to depict a dataset, and the graph is partitioned in line with some cut off criterion, for instance, normalized cut, ratio cut, minmax cut, etc. Lee et al.[23]recently presented a novel spectral clustering algorithm that relaxes some constraints to improve clustering accuracy whilst keeping clustering simplicity. In addition, relative neighbor graphs can be used to cluster data [24,25].

For the purpose of relieving the dilemma of users such as choice of clustering method, choice of parameters, etc., in this paper, we propose a graph-theoretical clustering method based on two rounds of minimum spanning trees (2-MSTClus). It comprises two algorithms, i.e. a separated clustering algorithm and a touching clustering algorithm, of which the former can partition separated clusters but has no effect on touching clusters, whereas the latter acts in the opposite way. From the viewpoint of data intrinsic structure, since the concepts of separated and touching are mutually complement as will be discussed in Section 2.1, clusters in any dataset can be either separated or touching. As the two algorithms are adaptive to the two groups of clusters, the proposed method can partially alleviate the user dilemma aforementioned. The main contributions are as follows:

(1) A graph representation, which is composed of two rounds of minimum spanning tree, is proposed and employed for clustering.
(2) Two mutually complementary algorithms are proposed and merged into a scheme, which can roughly cope with clustering problems with different shapes, densities and unbalanced sizes.

The rest of this paper is organized as follows. Section 2 depicts the typical cluster problems. In terms of the typical cluster problems, a graph-based clustering method is presented in Section 3. Section 4 demonstrates the effectiveness of the proposed method on synthetic and real datasets. Section 5 discusses the method and conclusions are drawn in Section 6.

2. Typical cluster problems

2.1. Terms of cluster problems

Since there does not exist a universal clustering algorithm that can deal with all cluster problems [7], it is significant for us to clarify what typical cluster problems are and which typical cluster problem a clustering algorithm favors. Some frequently used terms about cluster problem in the paper are defined as follows.

Definition 1. For a given distance metric, a *well-separated cluster* is a set of points such that the distance between any pair of points in the cluster is less than the distance between any point in the cluster and any point not in the cluster.

The above definition of a *well-separated cluster* is similar to the one in [27]. However, it is also similar to the second definition of a *cluster* presented in [28]. That implies a *cluster* is well-separated for a given distance metric.

Definition 2. For a given density metric and a distance metric, a pair of *separated clusters* is two sets of points such that (1) the closest point regions between the two clusters are with high densities compared to the distance between the two closest points from the two regions, respectively, or (2) the closest point regions between the two clusters are different in density.

For the former situation the separated clusters are called *distance-separated clusters*, while for the later called *density-separated clusters*. Obviously, the separated clusters defined above are not transitive. For instance, if A and B are a pair of separated clusters, and B and C are another pair of separated clusters, then A and C are not necessarily a pair of separated clusters.

Definition 3. A pair of *touching clusters* is two sets of points that are joined by a small neck whose removal produces two separated clusters which are substantially large than the neck itself.

Generally, a threshold, which is dependent on a concrete clustering method, is employed to determine how small a small neck is.

Definition 4. For a given distance metric, a *compact cluster* is a set of points such that the distance between any point in the cluster and the representative of the cluster is less than the distance between the point and any representative of other clusters.

In general, a centroid or a medoid of a cluster can be selected as the representative. The difference between the two representative candidates is that a centroid of a cluster is not necessarily a member point of the cluster, while a medoid must be a member point.

Definition 5. For a given distance metric, a *connected cluster* is a set of points such that for every point in the cluster, there exists at least one other point in the cluster, the distance between them is less than the distance between the point and any point not in the cluster.

The definitions of a compact cluster and a connected cluster are similar to those of center-based cluster and contiguouscluster in [27], respectively.

2.2. Cluster problem samples described by Zahn

Some typical cluster problems are described in Fig. 1 by Zahn [18]. Fig. 1(a) illustrates two clusters with similar shape, size and density.

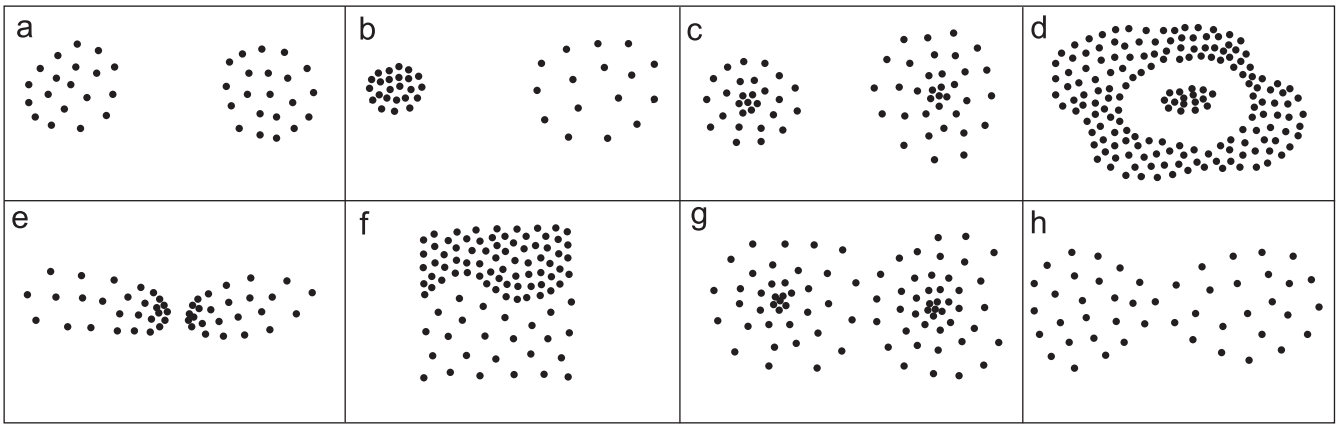


Fig. 1. Typical cluster problems from Ref. [18]. (a)–(e) are distance-separated cluster problems; (f) is density-separated cluster problem; (g) and (h) are density-separated cluster problems.

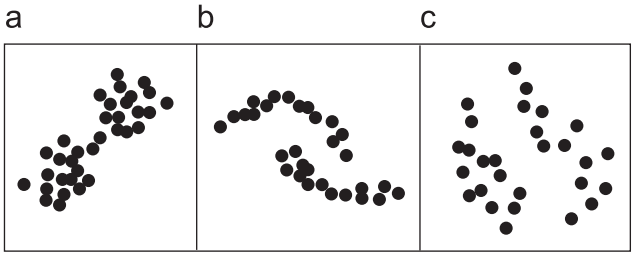


Fig. 2. The three patterns by Handl [12]. In (a), the compactness pattern illustrates the compactness objective which is suitable to deal with spherical datasets. In (b), the connectedness pattern depicts the connectedness objective which handles datasets of arbitrary shape; (c) is a spacial pattern.

It can be distinctly perceived that the two clusters are separated, since the inter-cluster density is very high compared to the intra-cluster pairwise distance. The principal feature of Fig. 1(b), (c) and (d) is still distance-separated, even if the shapes, sizes and/or densities of two clusters in each figure are diverse. The density of the two clusters in Fig. 1(e) are gradually varied, and become highest in their adjacent boundaries. From the global viewpoint of the rate of density variation, however, the separability remains prominent. Intuitively, the essential difference between the two clusters represented in Fig. 1(f) lies in density, rather than distance, and Zahn [18] called it density gradient. Fig. 1(g) and (h) are quite different from those aforementioned figures, because the two clusters touch each other slightly. Zahn [18] classified the cluster problems in Fig. 1(g) and (h) as touching cluster problems.

2.3. Cluster problems implied by clustering algorithms

Traditionally, clustering algorithms can be categorized into hierarchical, partitioning, density-based and model-based methods [3]. Being different from the traditional taxonomy, however, Handl and Knowles [12,26] classified clustering algorithms into three categories with different clustering criteria illustrated in Fig. 2: (1) algorithms based on the concept of compactness, such as k-means, average-linkage, etc., which make an effort to minimize the intra-cluster variation and are suitable for handling spherical datasets; (2) algorithms based on the concept of connectedness, such as path-based clustering algorithm [29], single-linkage, etc., which can detect the clusters with high intra-connectedness; (3) algorithms based on spatial separation criterion, which is opposed to connectedness criterion and generally considered incorporated with other criteria rather than independently. Actually, the clustering approach taxonomy in [12] is cluster-problem-based, as a clustering algorithm is categorized by the cluster problem which it favors, since the criteria of compactness, connectedness and spatial separation delineate the dataset structures instead of algorithms themselves. In accordance with the classification of clustering algorithm in [12], therefore, the cluster problems fall mainly into two groups: compact cluster problems and connected cluster problems.

2.4. Cluster problems classified in this work

In this paper, we classify cluster problems into two categories: separated problems and touching problems. The former includes distance-separated problems and density-separated problems. In terms of Definition 2, for example, we call the cluster problems depicted in Fig. 1(a)–(e) distance-separated, while the cluster problem depicted in Fig. 1(f) density-separated. Cluster problems in Fig. 1(g) and (h) are grouped, similarly in [18], as touching problems according to Definition 3. Since separated problem and touching problem are mutually supplemental, they may cover all kinds of datasets. This taxonomy of cluster problems ignores the compactness and connectedness. In fact, separated clusters can be compact or connected, and touching clusters can also be compact or connected. Based on our taxonomy, Fig. 3(a) and (b) are touching problems, Fig. 3(c) and (d) are separated problems; while in terms of clustering criteria in [12], Fig. 3(a) and (c) are compact problems, Fig. 3(b) and (d) are connected problems.

With the two-round-MST based graph representation of a dataset, we propose a separated clustering algorithm and a touching clustering algorithm, and encapsulate the two algorithms into a same method.

3. The clustering method

3.1. Problem formulation

Suppose that $X=\{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_i,\ldots,\mathbf{x}_N\}$ is a dataset, $\mathbf{x}_i=(x_{i1}, x_{i2},\ldots,x_{ij},\ldots,x_{id})^T\in\Re^d$ is a feature vector, and x_{ij} is a feature. Let $G(X)=(V,E)$ denote a weighted and undirected complete graph with vertex set $V=X$ and edge set $E=\{(\mathbf{x}_i,\mathbf{x}_j)|\mathbf{x}_i,\mathbf{x}_j\in X, i\neq j\}$. Each edge $e=(\mathbf{x}_i,\mathbf{x}_j)$ has a length $\rho(\mathbf{x}_i,\mathbf{x}_j)$, and generally the length can be Euclidean distance, Mahalanobis distance, City-block distance, etc. [7]. In this paper, Euclidean distance is employed. A general clustering algorithm attempts to partition the dataset X into K clusters: $C_1,C_2,\ldots,C_K$, where $C_i\neq\emptyset$, $C_i\cap C_j=\emptyset$, $X=C_1\cup C_2\cdots\cup C_K$, $i=1:K$,

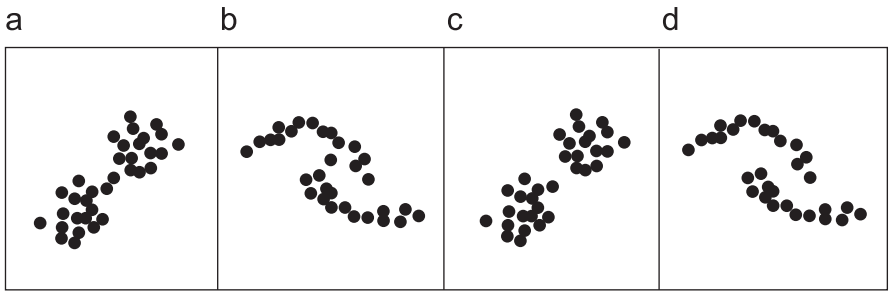


Fig. 3. The different taxonomies of cluster problems. The patterns in (a) and (b) are touching problems, and the patterns in (c) and (d) are separated problems in this paper. The patterns in (a) and (c) are compact problems, and the patterns in (b) and (d) are connected problems by Handl [12].

$j=1:K$, $i\neq j$. Correspondingly, the associated graph will be cut into K subgraphs.

A minimum spanning tree (MST) of graph $G(X)$ is an acyclic subset $T\subseteq E$ that connects all the vertices in V and whose total lengths $W(T)=\sum_{\mathbf{x}_i,\mathbf{x}_j\in T}\rho(\mathbf{x}_i,\mathbf{x}_j)$ is minimum.

3.2. Algorithm for separated cluster problem

As mentioned above, separated cluster problems are either distance-separated or density-separated. Zahn [18] employed two different algorithms to deal with the two situations, respectively. For the purpose of automatic clustering, we try to handle distance-separated problem and density-separated problem with one algorithm.

3.2.1. Two-round-MST based graph

Compared with KNN-graph-based clustering algorithms [16,17], MST-based clustering algorithms [18,19] have two disadvantages. The first one is that only information about the edges included in MST is made use of to partition a graph, while information about the other edges is lost. The second one is that for MST-based approaches every edge's removal will result in two subgraphs. This may lead to a partition without sufficient evidence. With these observations in mind, we consider using second round of MST for accumulating more evidence and making MST-based clustering more robust. It is defined as follows.

Definition 6. Let $T_1=f_{mst}(V,E)$ denote the MST of $G(X)=(V,E)$. The second round MST of $G(X)$ is defined as

$$T_2=f_{mst}(V,E-T_1) \tag{1}$$

where $f_{mst}:(V,E)\rightarrow T$ is a function to produce MST from a graph.

If there exists a vertex, say v, in T_1 such that the degree of v is $|V|-1$, v is isolated in $G(V,E-T_1)$. Hence T_2 cannot be generated in terms of Definition 6. To remedy the deficiency simply, the edge connected to v and with the longest length in T_1 is preserved for producing T_2.

Combining T_1 and T_2, a two-round-MST based graph, say $G_{mst}(X)=(V,T_1+T_2)=(V,E_{mst})$, is obtained. The two-round-MST based graph is inspired by Yang [30]. Yang used k MSTs to construct k-edge connected neighbor graph and estimate geodesic distances in high dimensional datasets. Fig. 4(a) and (b), respectively, represent the T_1 and T_2 of Fig. 1(c), in which the dataset is distance-separated. Fig. 4(c) represents the corresponding two-round-MST based graph.

The lengths of edges from T_1 and T_2 have a special relationship (see Theorem 3), which can be used to partition two-round-MST based graph.

Lemma 1. *Let $T(V_T,E_T)$ be a tree. If $T'(V'_T,E'_T)$ is maximum tree such that $V'_T\subseteq V_T$, $E'_T\cap E_T=\emptyset$, then either $|E'_T|=|E_T|-1$ or $|E'_T|=|E_T|$.*

Proof. If $|V_T|-1$ vertices of $T(V_T,E_T)$ have degree 1, and the other vertex, say v, has degree $|V_T|-1$. In T, from any vertex with degree 1, there exist at most $|V_T|-2$ edges connected to other vertices except its neighbor, i.e. v, and no more edge is available to construct $T'(V'_T,E'_T)$. At this moment, $V'_T=V_T\backslash\{v\}$, and $|E'_T|=|V_T|-2=|E_T|-1$.

Otherwise, suppose vertex v_0 has degree of 1, its neighbor is v_1. From v_0, $|V_T|-2$ edges can be used to construct $T'(V'_T,E'_T)$. In addition, there must exist an edge between vertex v_1 and its non-neighbor vertex. At this moment, $V'_T=V_T$, and $|E'_T|=|V_T|-1=|E_T|$. □

Corollary 2. *Let $F(V_F,E_F)$ be an acyclic forest. Suppose $F'(V'_F,E'_F)$ is maximum acyclic forest such that $V'_F\subseteq V_F$, $E'_F\cap E_F=\emptyset$, and for any $e\in E'_F$, $F(V_F,E_F\cup\{e\})$ is cyclic, then $|E'_F|\leq|E_F|$.*

Theorem 3. *Suppose T_1 and T_2 are first round and second round MST of $G(V,E)$, respectively. If edges of T_1 and edges of T_2 are ordered ascendingly by their weights as $e_1^1,e_1^2,\ldots,e_1^i,\ldots,e_1^{|V|-1}$, $e_2^1,e_2^2,\ldots,e_2^i,\ldots,e_2^{|V|-1}$, then $\rho(e_1^i)\leq\rho(e_2^i)$, where i denotes the sequence number of ordered edges, and $1\leq i\leq|V|-1$.*

Proof. Suppose there exists j such that $\rho(e_1^j)>\rho(e_2^j)$. Obviously $\rho(e_1^j)>\rho(e_2^j)\geq\rho(e_2^{j-1})\geq\cdots\geq\rho(e_2^1)$, in terms of Kruskal's algorithm of constructing a MST, the reason why $e_2^1,e_2^2,\ldots,e_2^j$ are not selected in the j th step of constructing T_1 is that the combination of any one of these edges with $e_1^1,e_1^2,\ldots,e_1^{j-1}$ would produce a cycle in T_1. Let $e_1^1,e_1^2,\ldots,e_1^{j-1}$ form $F(V_F,E_F)$ and $e_2^1,e_2^2,\ldots,e_2^j$ form $F'(V'_F,E'_F)$, then the two forests are acyclic since $e_1^1,e_1^2,\ldots,e_1^{j-1}$ and $e_2^1,e_2^2,\ldots,e_2^j$ are the part of T_1 and T_2, respectively. Because if any edge of $F'(V'_F,E'_F)$ is added into $F(V_F,E_F)$, $F(V_F,E_F)$ would be cyclic, we have $V'_F\subseteq V_F$. However, $|E_F|=j-1$ and $|E'_F|=j$, this contradicts Corollary 2. □

For a tree, any removal of edge will lead to a partition. Whereas to partition a two-round-MST based graph, at least two edges must be removed, of which at least one edge comes from T_1 and T_2, respectively. Accordingly, compared with a cut on MST, a two-round-MST based graph cut requires more evidence and may result in a more robust partition.

Generally, for a given dataset, MST is not unique because two or more edges with same length may exist. However, the non-uniqueness of MST does not influence the partition of a graph for clustering [18], and the clustering induced by removing long edges is independent of the particular MST [31].

3.2.2. Two-round-MST based graph cut

After a dataset is represented by a two-round-MST based graph, the task of clustering is transformed to partitioning the graph with a

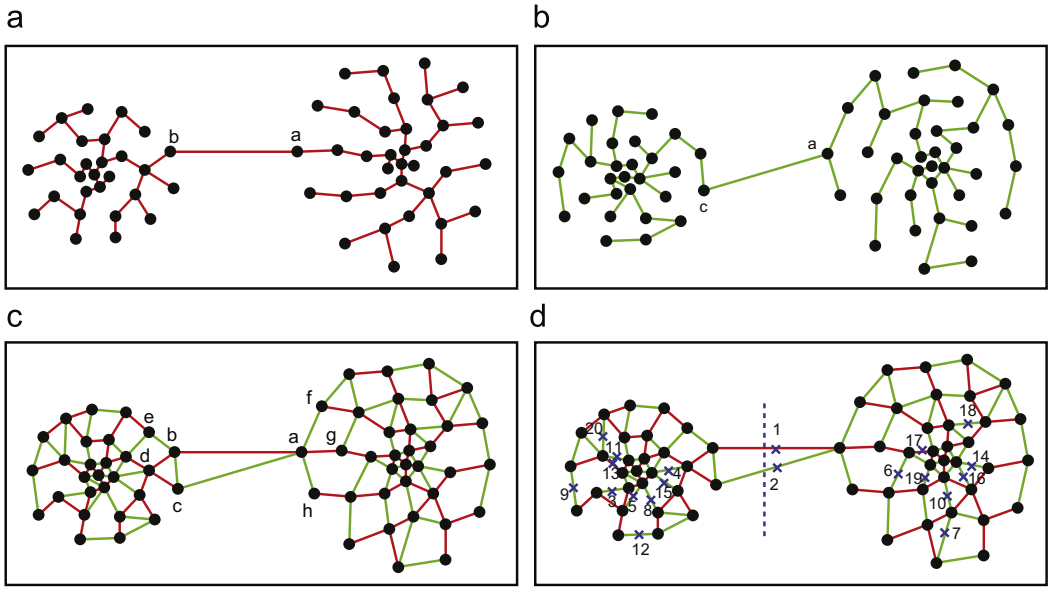


Fig. 4. Two-round-MSTs of the dataset in Fig. 1(c). (a) is the first round of MST, i.e. T_1, and ab is the significant and expected edge to be removed in traditional MST based clustering methods. (b) is the second round of MST, i.e. T_2, and ac is another significant edge. (c) illustrates the two-round-MST based graph. To partition the graph, ab and ac are expected edges to be cut. (d) depicts the top 20 edges with large weights based on Definition 2. The first two edge removals result in a valid graph cut. From the top 20 edges, 17 edges come from T_2, and 3 from T_1.

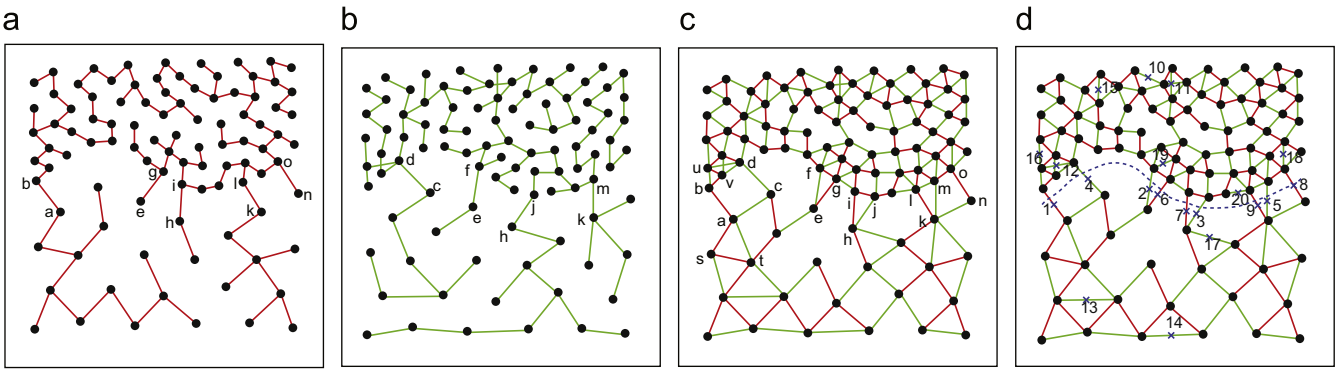


Fig. 5. Density-separated cluster problem taken from [18]. (a) illustrates the first round of MST; (b) depicts the second round of MST. In (c), the two-round-MST based graph is illustrated; bu and bv are edges connected to ab by vertex b, while as and at are connected to ab by a. The average lengths of the two groups are quite different. (d) illustrates the graph cut. The dashed curve is the graph cut which is achieved by the removals of the top nine edges based on Definition 2, the $Ratio(E_{gcut})$ is 0.444 and greater than the threshold λ.

partitioning criterion. In general, a partitioning criterion plays a pivot role in a clustering algorithm. Therefore, the next task is to define an effective partitioning criterion. Fig. 4(a) is the MST of a distance-separated dataset illustrated in Fig. 1(c). Obviously, ab is the very edge to be removed and lead to a valid partition for MST-based methods. Zahn [18] defined an edge inconsistency to detect the edge. That is, the edge, whose weight is significantly larger than the average of nearby edge weights on both sides of the edge, should be deleted. However, this definition is only relevant for the distance-separated cluster problem, for instance, Fig. 4(a). For density-separated cluster problem illustrated in Fig. 5(a), which is called density gradient problem in [18], Zahn first determined the dense set and the sparse set with a histogram of edge lengths, then singled out five inter-cluster edges ab, eg, hi, kl and no. Although Zahn's method for density-separated problem is feasible, it is somewhat complex. In brief, Zahn used two partitioning criteria to deal with distance-separated cluster problems and density-separated cluster problems, respectively. Our goal, however, is to handle the two situations with one partitioning criterion.

From Figs. 4(c) and 5(c), we observe that the main difference between a distance-separated cluster problem and a density-separated cluster problem is whether the average lengths of edges connected to two sides of an inter-cluster edge are similar or not. For distance-separated clusters in Fig. 4(c), the average length of edges connected to end point a of edge ab is similar to that of edges connected to the other end of ab, while for density-separated clusters in Fig. 5(c), the average lengths of two sets of edges connected, respectively, to two ends of ab are quite different. Accordingly, for the purpose of identifying an inter-cluster edge with one criterion for both distance-separated clusters and density-separated clusters, we compare the length of the inter-cluster edge with the minimum of the average lengths of the two sets of edges which are connected to its two ends, respectively. First, we define the weight of an edge as follows:

Definition 7. Let $G_{mst}(X)=(V,E_{mst})$ be a two-round-MST based graph, $e_{ab}\in E_{mst}$ and $a,b\in V$, $w(e_{ab})$ be the weight of e_{ab} as in

$$w(e_{ab})=\frac{\rho(e_{ab})-\min(avg(E_a-\{e_{ab}\}),avg(E_b-\{e_{ab}\}))}{\rho(e_{ab})} \quad (2)$$

where $E_a = \{e_{ij} | (e_{ij} \in E_{mst}) \wedge (i = a \vee j = a)\}$, $E_b = \{e_{ij} | (e_{ij} \in E_{mst}) \wedge (i = b \vee j = b)\}$,

$$avg(E) = \frac{1}{|E|} \sum_{e \in E} \rho(e)$$

and $\rho(e)$ is the Euclidean distance of edge e.

Analyzing two-round-MST based graphs of some separated datasets and the corresponding weights defined above, we find that two-round-MST based graphs and the weights have three good features: (1) generally, the weights of inter-cluster edges are quite larger than those of intra-cluster edges. (2) The inter-cluster edges are approximately equally distributed to T_1 and T_2. (3) Except inter-cluster edges, most of edges with large weights come from T_2, and this is supported by Theorem 3. Fig. 4(d) depicts the top 20 weights of the distance-separated dataset in Fig. 1(c). The two inter-cluster edges are those with top two weights, respectively, and one is from T_1 and the other one is from T_2. Among the next 18 edges, 16 edges come from T_2 and only two edges come from T_1. Fig. 5(d) describes the top 20 weights of the density-separated dataset in Fig. 1(f). The top nine weights are from the very nine inter-cluster edges, of which five are from T_1 and four are from T_2, and all of the remaining 11 edges belong to T_2.

In terms of the first feature, a desired two-round-MST based graph cut can be achieved by removing the edges with largest weight one by one. The next two features indicate that whether or not a graph cut is valid can be determined by analyzing the distribution of removed edges.

Definition 8. Let $Rank(E_{mst})$ be a list of edges ordered descendingly by corresponding weights as in

$$Rank(E_{mst}) = \langle edge\_topweight(E_{mst}) \circ Rank(E_{mst} - \{edge\_topweight(E_{mst})\}) \rangle \quad (3)$$

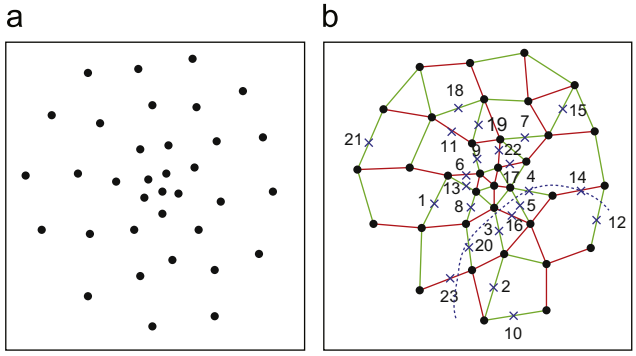


Fig. 6. A cluster cannot be partitioned any more. (a) illustrates the sub-dataset of Fig. 1(c); (b) separated clustering algorithm is applied to the sub-dataset. When a graph cut is obtained, which is indicated by the dashed curve, the $Ratio(E_{gcut})$ is 0.304 and less than the threshold λ.

where $edge\_topweight(E_{mst}) = \arg\max_{e \in E_{mst}}(w(e))$, $\circ$ is a concatenate operator.

Edge removing scheme: The edge with large weight has the priority to be removed, namely edges are removed in the order of $Rank(E_{mst})$. Since every removal of edge may lead to a graph cut (excluding the first removal), we must determine whether or not a new graph cut is achieved after each removal. The determination could be made by traversing the graph with either breadth-first search algorithm or depth-first search algorithm.

Definition 9. Let E_{gcut} be a set of removed edges when a graph cut on a two-round-MST based graph is achieved, if the following holds:

$$Ratio(E_{gcut}) = \frac{\min(|E_{gcut} \cap T_1|, |E_{gcut} \cap T_2|)}{|E_{gcut}|} \geq \lambda \quad (4)$$

where λ is a threshold, then the graph cut is valid, otherwise it is invalid. If the first graph cut is valid, the cluster is said to be separated, otherwise, non-separated.

Figs. 4(d) and 5(d) illustrate that both two first graph cuts are valid, because the $Ratio(E_{gcut})$'s are 0.500 and 0.440, respectively, greater than the threshold $\lambda = 0.333$ which is discussed in Section 4. Consequently, the datasets in Figs. 4 and 5 are separated, and are partitioned by removing the first two and first nine edges, respectively. Fig. 6(a) represents a subgraph produced by applying the scheme on the dataset in Fig. 4, while Fig. 6(b) indicates this subdataset is no-separated, since the $Ratio(E_{gcut})$ for the first cut is 0.304 and less than the threshold. However, the scheme is not always effective, because two exceptions exist.

Exception 1. In a density-separated dataset, there exist two (or more) inter-cluster edges which have a common vertex close to dense part. For example, in Fig. 7(a), inter-cluster edge e_{fg} and e_{hg} have a common vertex g which belongs to the dense part of the dataset. The dashed curve is the expected graph cut. But the weight of e_{fg} is less than those of e_{ab}, e_{cd} and e_{ef}, because when we compute the weight of e_{fg}, another inter-cluster edge e_{hg} is concerned according to Definition 7. As a result, more removed edges are from T_2 when the first graph cut is achieved, and the probability of the cut being valid decreases. The straightforward solution is to ignore the longest neighbor edge. For example, when the weight of e_{fg} is computed, edge e_{hg} should be ruled out from E_g.

Exception 2. The weight defined in Definition 7 is a ratio. If there exists an edge which is quite small in length, and the vertices connected to its one end are extremely close, then its weight is relatively large. In Fig. 7(c), vertices e, f, g are very close. For edge e_{hf}, because $avg(E_f - \{e_{hf}\})$ is extremely small, $w(e_{hf})$ is top 1 even though its length is far less than those of e_{ab} and e_{ac}. To remedy this exception, the edge length can be considered as a penalty.

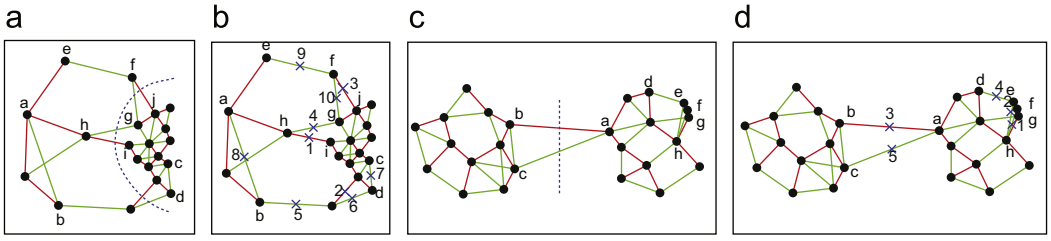


Fig. 7. Two exceptions for Definition 2. (a) is a density-separated cluster problem, the dashed curve is the expected graph cut. (b) illustrates the top 10 edges. As hg is considered for evaluating the weight of fg in terms of Definition 2, fg has a less weight than ab, ef and cd do. (c) is a distance-separated cluster problem, the dashed line is the expected graph cut. (d) illustrates another exception: since e, f, g are too close, he and hf have greater weights than ab and ac do.

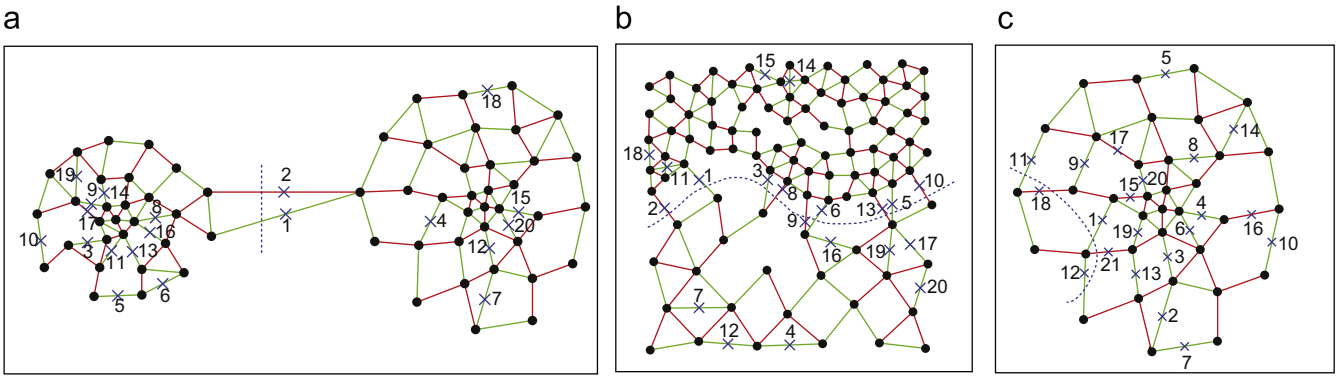


Fig. 8. Graph cuts with the improved definition of weight. In (a), the removals of the top two edges lead to the graph cut on dataset illustrated in Fig. 4, and corresponding $Ratio(E_{gcut})$ is 0.500. In (b), the graph cut on dataset presented in Fig. 5 is obtained by removing the top 13 edges, and the corresponding $Ratio(E_{gcut})$ is 0.385. In (c), the graph cut is different from that illustrated in Fig. 6, and the corresponding $Ratio(E_{gcut})$ is 0.238.

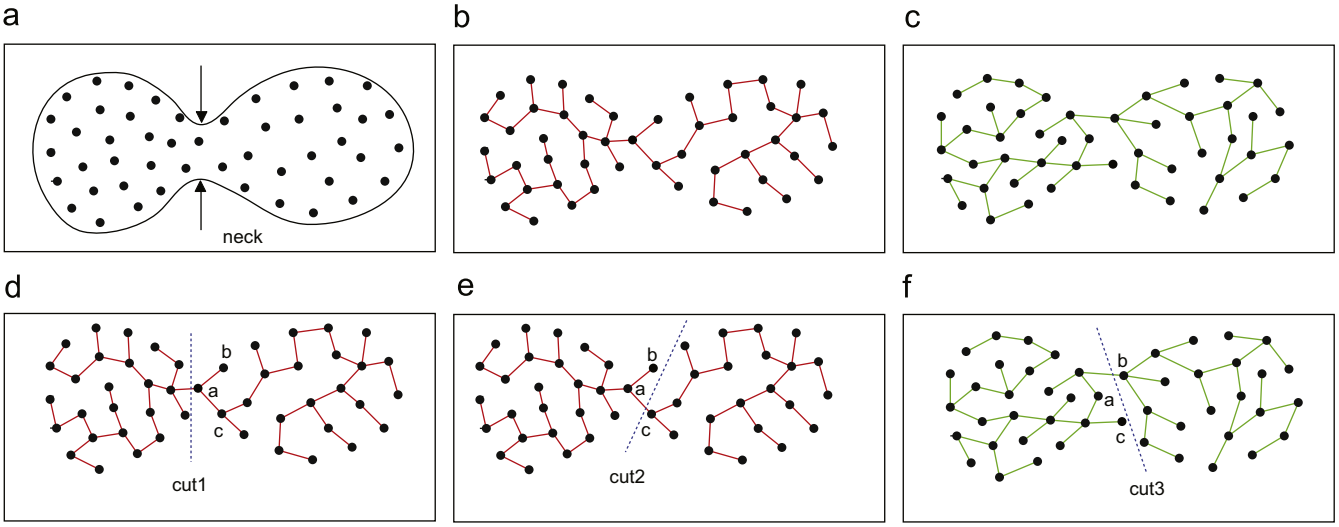


Fig. 9. Clustering a touching problem. (a) is the dataset from Fig. 1. (b) illustrates the first round of MST. (c) represents the second of MST. Comparing *cut*1 of T_1 in (d) and *cut*3 of T_2 in (f), two inconsistent vertices (a,c) exist, while between *cut*2 in (e) and *cut*3 in (f), there also exist two inconsistent vertices (b,c).

Therefore, the weight of e_{ab} in Definition 7 is redefined as

$$w(e_{ab})=\delta\times\frac{\rho(e_{ab})-\min(avg(E'_a-\{e'_a\}),avg(E'_b-\{e'_b\}))}{\rho(e_{ab})}+(1-\delta)\times\rho(e_{ab}) \tag{5}$$

where $E'_a = E_a - \{e_{ab}\}$, $e'_a = \arg\max_{e\in E'_a}(\rho(e))$, $E'_b = E_b - \{e_{ab}\}$, $e'_b = \arg\max_{e\in E'_b}(\rho(e))$, δ is a penalty factor and $0\le\delta\le 1$. $E'_a - \{e'_a\}$ and $E'_b - \{e'_b\}$ ignore the longest neighbor edges, while penalty factor δ gives a tradeoff between the ratio and the edge length.

Fig. 8 illustrates the first graph cut of applying redefined weight on the three datasets in Figs. 4–6. The corresponding $Ratio(E_{gcut})$'s are 0.500, 0.380, 0.240, respectively. According to the discussion of λ in Section 4, the first two graph cuts are still valid and the third one is still invalid.

A subgraph partitioned from a separated problem may be another separated problem. Therefore, we must apply the graph cut method to every produced subgraph iteratively to check whether or not it can be further partitioned until no subgraphs are separated.

Algorithm 1. Clustering separated cluster problems

Input: $G(X)=(V,E)$, the graph of the dataset to be partitioned
Output: S, the set of partitioned subdatasets.
Step 1. Compute T_1 and T_2 of $G(X)$, and combine the two MSTs to construct the two-round-MST based graph $G_{mst}(X)$, and put it into a table named *Open*; create another empty table named *Closed*.
Step 2. If table *Open* is empty, sub-datasets corresponding to subgraphs in *Closed* table are put into S; return S.
Step 3. Get a graph $G'(X')=(V',E')$ out of *Open* table, calculate the weights of edges in $G'(X')$ with Eq. (5), and build the list $Rank(E')$.
Step 4. Remove the edges of $G'(X')$ in the order of $Rank(E')$ until a cut is achieved.
Step 5. If the cut is valid in terms of Definition 9, put the two subgraphs produced by the cut into *Open* table; otherwise put graph $G'(X')$ into *Closed* table.
Step 6. Go to Step 2.

Algorithm 1iteratively checks subgraphs, and partitions the separated ones until there exists no separated subgraphs. At the same

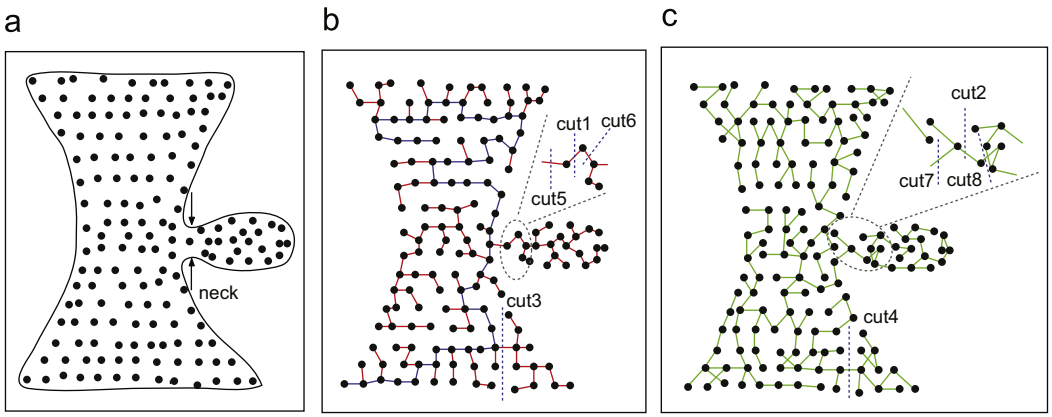


Fig. 10. A teapot dataset as a touching problem. (a) is a teapot dataset with a neck. In (b), the diameter defined by Zahn [18] is illustrated by the blue path. *cut*5 in (b) and *cut*7 in (c) are similar ($\beta = 1$), so are *cut*1 and *cut*2, *cut*6 and *cut*2, *cut*1 and *cut*8, *cut*3 and *cut*4.

time, the algorithm takes no action on non-separated subgraphs, namely touching subgraphs. In fact, the reason why Algorithm 1 can identify separated clusters is the definition of edge weight in Definition 7. The weight of an edge e_{ab} reflects the relation between the length of e_{ab} and two neighbor region densities of vertices a and b, respectively, where the neighbor region density is measured by the average length of the edges in the region. If the weight of e_{ab} is large, the densities of the two neighbor regions are high compared to the length $\rho(e_{ab})$, or the two densities are very different. For a pair of touching clusters, as a neck exists and lengths of edges in the neck are small compared to the neighbor region densities, namely the weights of edges in the neck are small, Algorithm 1 cannot detect the touching clusters.

3.3. Algorithm for touching cluster problem

Although Algorithm 1 can identify separated clusters, it becomes disabled for touching clusters. After the algorithm is applied to a dataset, each induced sub-dataset will be either a touching cluster or a basic cluster which cannot be partitioned further.

For a touching cluster, a neck exists between the two connected subclusters. The neck of touching cluster problem in Fig. 1(h) is illustrated in Fig. 9(a). Zahn [18] defined a diameter of MST as a path with the most number of edges, and detected the neck using diameter histogram. However, a diameter does not always pass through a neck. Fig. 10 illustrates an exception. We identify the neck by considering T_1 and T_2 simultaneously. The two-round-MSTs of Fig. 1(h) are depicted by Fig. 9(b) and (c), respectively. The task in this phase is to detect and remove these edges crossing the neck, and discover touching clusters. Based on the two-round-MSTs, an important observation is as follows:

Observation 1. A partition resulted from deleting an edge crossing the neck in T_1 is similar to a partition resulted from deleting an edge crossing the neck in T_2.

On the contrary, for the two cuts from T_1 and T_2, respectively, if one cut does not cross the neck, the two corresponding partitions will be generally quite different from each other. Comparing the partition on T_1 in Fig. 9(d) with the partition on T_2 in Fig. 9(f), we notice that only two vertices (a and c) belong to different group, and is called *inconsistent vertices*. Similarly, only two *inconsistent vertices* (b and c) exist between the cuts in Fig. 9(e) and (f).

For the purpose of determining whether two cuts are similar, the number of *inconsistent vertices* must be given out as a constraint, i.e. if the number of *inconsistent vertices* between two cuts is not greater than a threshold, say β, the two cuts are similar. For the previous example in Fig. 9, $\beta = 2$ is reasonable. However, some unexpected pairs of cuts which do not cross the neck of a dataset may conform to the criterion and are determined to be similar. For example, the *cut*3 in Fig. 10(b) and the *cut*4 in Fig. 10(c) are similar if $\beta = 1$, however, the two cuts are unexpected. Fortunately, the following observation can remedy this bad feature.

Observation 2. With a same threshold β, the number of pairs of similar cuts which cross the neck is generally greater than that of pairs of similar cuts which do not cross the neck.

In Fig. 10(b) and (c), suppose $\beta = 1$, it is easy to find another pair of similar cuts which cross the necks other than *cut*1 and *cut*2, for instance, *cut*5 and *cut*7, *cut*6 and *cut*2, *cut*1 and *cut*8, while there exists no other pair of similar cuts near *cut*3 and *cut*4. Therefore, *cut*3 and *cut*4 are discarded because the similar evidence is insufficient. With the observation in mind, we can design the algorithm for touching cluster problems as follows.

Definition 10. Let P^{T_1} be the list of $N-1$ partitions on T_1 as in

$$P^{T_1} = \langle (p_{11}^{T_1}, p_{12}^{T_1}), (p_{21}^{T_1}, p_{22}^{T_1}), \ldots, (p_{(N-1)1}^{T_1}, p_{(N-1)2}^{T_1}) \rangle \quad (6)$$

where pair $(p_{i1}^{T_1}, p_{i2}^{T_1})$ denotes the partition which results from removing the i th edge in T_1, $p_{i1}^{T_1}$ and $p_{i2}^{T_1}$ are subsets of vertices, $p_{i1}^{T_1} \cup p_{i2}^{T_1} = V$, $|p_{i1}^{T_1}| \leq |p_{i2}^{T_1}|$. Similarly, the list of $N-1$ partitions on T_2, P^{T_2}, is defined as

$$P^{T_2} = \langle (p_{11}^{T_2}, p_{12}^{T_2}), (p_{21}^{T_2}, p_{22}^{T_2}), \ldots, (p_{(N-1)1}^{T_2}, p_{(N-1)2}^{T_2}) \rangle \quad (7)$$

Obviously, some partitions both on T_1 and T_2 can be very skewed. However, a valid partition is expected to generate two subsets with relatively balanced element numbers in some typical graph partition methods, such as ratio cut [32]. Therefore, the partition lists P^{T_1} and P^{T_2} can be refined by ignoring skewed partitions so as to reduce the number of comparisons.

Definition 11. Let RP^{T_1} and RP^{T_2} be the refined lists, all of the elements of RP^{T_1} come from P^{T_1} , and all of the elements of RP^{T_2} come

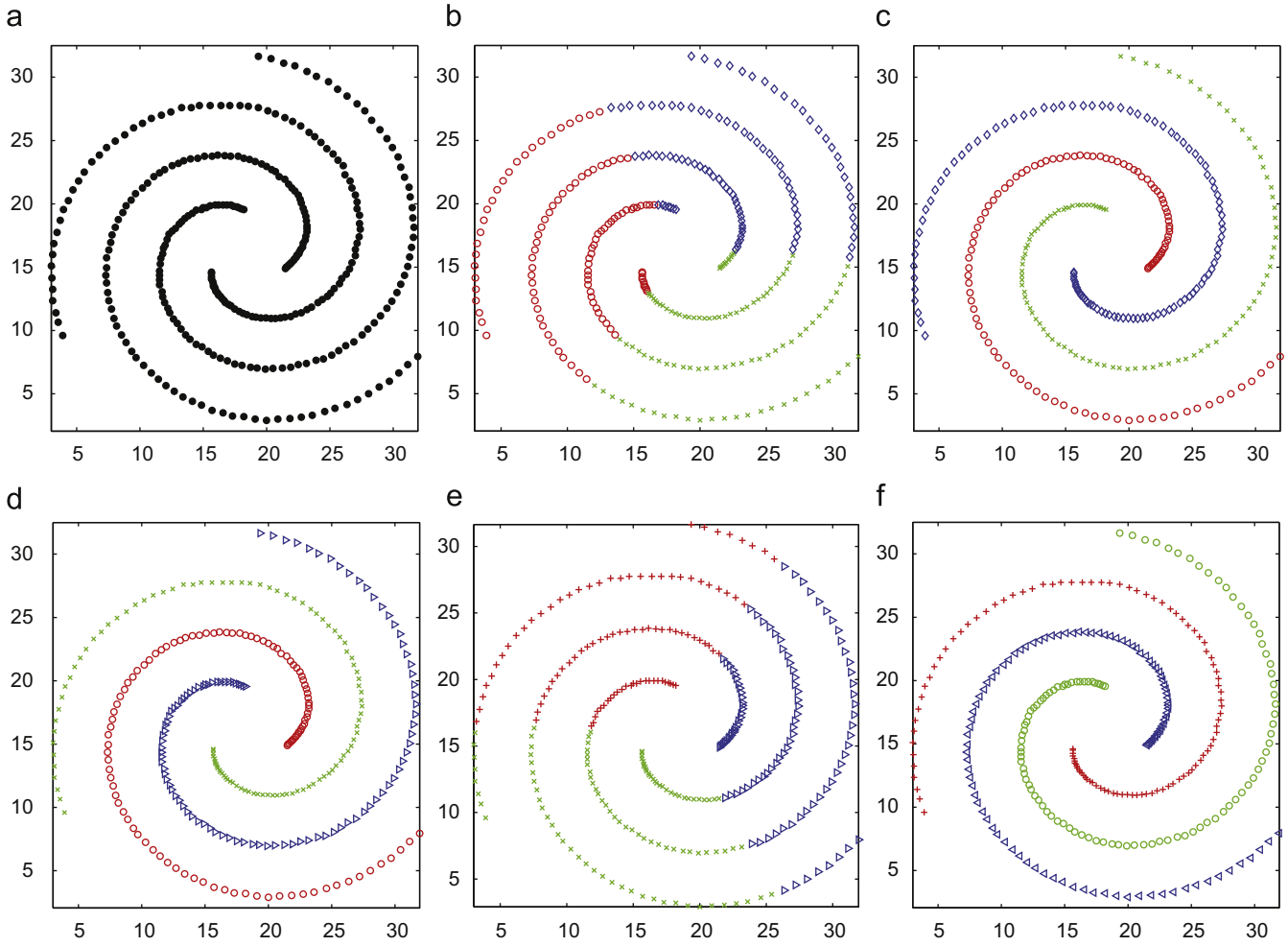


Fig. 11. Clustering results on DS1. (a) is the original dataset; (b) is the clustering result of k-means; (c) is the clustering result of DBScan (MinPts = 3, Eps = 1.6); (d) is the clustering result of single-linkage; (e) is the clustering result of spectral clustering; (f) is the clustering result of 2-MSTClus.

from P^{T_2}, as in

$$RP^{T_1} = \langle (rp_{11}^{T_1}, rp_{12}^{T_1}), (rp_{21}^{T_1}, rp_{22}^{T_1}), \ldots, (rp_{L1}^{T_1}, rp_{L2}^{T_1}) \rangle \quad (8)$$

$$RP^{T_2} = \langle (rp_{11}^{T_2}, rp_{12}^{T_2}), (rp_{21}^{T_2}, rp_{22}^{T_2}), \ldots, (rp_{M1}^{T_1}, rp_{M2}^{T_2}) \rangle \quad (9)$$

where $L \leq N-1$, $\varepsilon \leq \min(|rp_{i1}^{T_1}|, |rp_{i2}^{T_1}|)/\max(|rp_{i1}^{T_1}|, |rp_{i2}^{T_1}|)$; $M \leq N-1$, $\varepsilon \leq \min(|rp_{i1}^{T_2}|, |rp_{i2}^{T_2}|)/\max(|rp_{i1}^{T_2}|, |rp_{i2}^{T_2}|)$, $\varepsilon \ll N$, ε will be discussed in Section 4.

In the next step, partitions in RP^{T_1} will be compared with those in RP^{T_2}. As the number of *inconsistent vertices* between two cuts must be less than or equal to the threshold β, if $||rp_{i1}^{T_1}| - |rp_{j1}^{T_2}|| > \beta$, the comparison between two partitions $(rp_{i1}^{T_1}, rp_{i2}^{T_1})$ and $(rp_{j1}^{T_2}, rp_{j2}^{T_2})$ can be skipped.

For the purpose of saving the computational cost, we can further combine the two lists RP^{T_1} and RP^{T_2}, and order them ascendingly by the element numbers of the left parts of the pairs. Only pairs which come from different MSTs and of which element number of left parts have differences not more than β will be compared.

Definition 12. Let SP be a set which consists of all the elements of RP^{T_1} and RP^{T_2}:

$$SP = \{(rp_{11}^{T_1}, rp_{12}^{T_1}), \ldots, (rp_{L1}^{T_1}, rp_{L2}^{T_1}), (rp_{11}^{T_2}, rp_{12}^{T_2}), \ldots, (rp_{M1}^{T_1}, rp_{M2}^{T_2})\} \quad (10)$$

Definition 13. For a $sp \in SP$, let $left(sp)$ denote the left part of sp, the source of sp is defined as

$$source(sp) = \begin{cases} 1 & \text{if } sp \text{ comes from } T_1 \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

For example, if $sp = (rp_{11}^{T_1}, rp_{12}^{T_1})$, then $left(sp) = rp_{11}^{T_1}$ and $source(sp) = 1$.

Definition 14. Let $CP(SP) = \langle (cp_{11}, cp_{12}), \ldots, (cp_{(L+M)1}, cp_{(L+M)2}) \rangle$ be the ordered list as in

$$CP(SP) = \langle part\_\min(SP) \circ CP(SP - \{part\_\min(SP)\}) \rangle \quad (12)$$

where $part\_\min(SP) = \arg\min_{sp \in SP} |left(sp)|$, and $\circ$ is a concatenate operator.

Definition 15. Two partitions (cp_{i1}, cp_{i2}) and (cp_{j1}, cp_{j2}) are said to be similar, where $i \leq j$, if the followings hold:

(a) $source((cp_{i1}, cp_{i2})) \neq source((cp_{j1}, cp_{j2}))$;
(b) $|cp_{i1} - cp_{j1}| + |cp_{j1} - cp_{i1}| \leq \beta$ or $|cp_{i1} - cp_{j2}| + |cp_{j2} - cp_{i1}| \leq \beta$.

The first condition indicates that the two partitions come from different MSTs, while the second condition reveals that the number of inconsistent vertices between the two partitions is sufficient small.

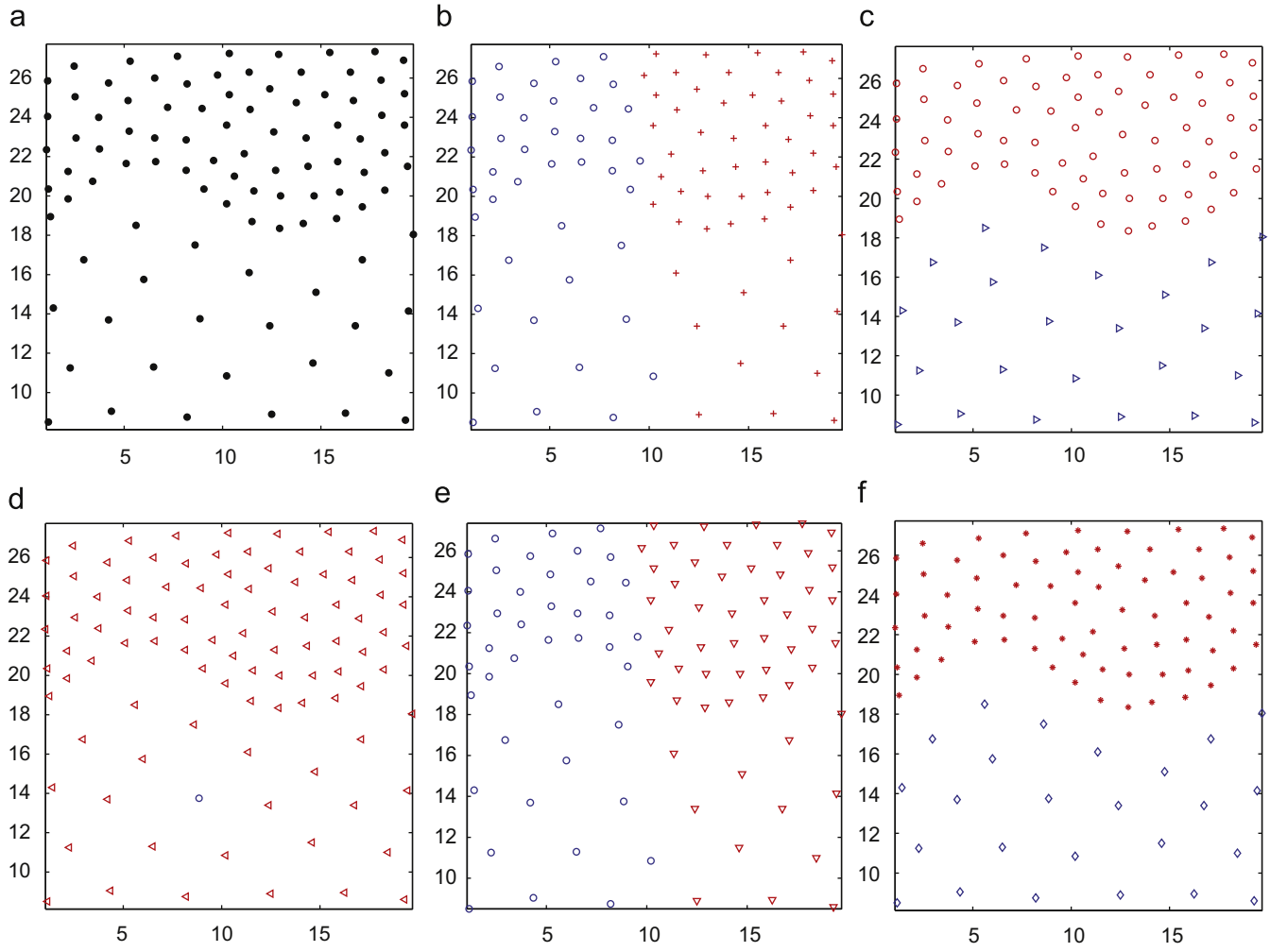


Fig. 12. Clustering results on DS2. (a) is the original dataset; (b) is the clustering result of k-means; (c) is the clustering result of DBScan (MinPts = 3, Eps = 1.9); (d) is the clustering result of single-linkage; (e) is the clustering result of spectral clustering; (f) is the clustering result of 2-MSTClus.

Algorithm 2. Clustering touching problems

Input: T_1 and T_2, the two rounds of MST of a sub-dataset generated by **Algorithm 1**.
Output: S, the set of expected partitions.
Step 1. Construct the ordered list $CP(SP)$ with T_1 and T_2; create two empty set S' and S.
Step 2. For each $(cp_{i1}, cp_{i2}) \in CP(SP)$, it is compared with $(cp_{j1}, cp_{j2}) \in CP(SP)$ and $j > i$. According to Definition 15, if there exists a partition (cp_{j1}, cp_{j2}) which is similar to (cp_{i1}, cp_{i2}), (cp_{i1}, cp_{i2}) is put into S'.
Step 3. For each $s \in S'$, if there exists a $t \in S'$, $t \neq s$, and t is similar to s, s is put into S.
Step 4. Combine similar partitions in S.

In Algorithm 2, Step 3 is to remove the unexpected partitions in terms of Observation 2. For simplicity, only those partitions without similar others are removed. In Step 3, when determining the similarity between t and s, we ignore whether they come from different MSTs or not, since at this stage only the number of inconsistent vertices are concerned. Step 4 combines the similar partitions. This can be achieved by assigning *inconsistent vertices* to two groups in terms of the evidence (support rate) accumulated from the similar partitions. Algorithm 2 can identify touching clusters except overlapping ones.

3.4. The combination of the two algorithms

As mentioned above, cluster problems are categorized into separated problems and touching problems in this paper, and the two cluster problems roughly cover all the cluster problems since they are mutual complementary. As Algorithm 1 automatically identifies separated clusters and has no effect on touching clusters, Algorithms 1 and 2 can be easily combined to deal with any cluster problem. When every subset partitioned by Algorithm 1 is fed to Algorithm 2, we will obtain the final clustering results. Therefore, the two algorithms can be easily combined to form the method 2-MSTClus.

Many traditional clustering algorithms are vulnerable to the different cluster sizes, shapes and densities. However, since the separated clusters and touching clusters can roughly cover all kinds of clusters (except overlapping clusters) in terms of definition of separated cluster and touching cluster regardless of cluster size, shape and density, the combination of Algorithms 1 and 2 is robust to diversifications of sizes, shapes and densities of clusters.

3.5. Computational complexity analysis

The computational complexity of Algorithm 1 is analyzed as follows. For a graph $G(X) = (V, E)$, if Fibonacci heaps are used to implement the min-priority queue, the running time of Prim's MST

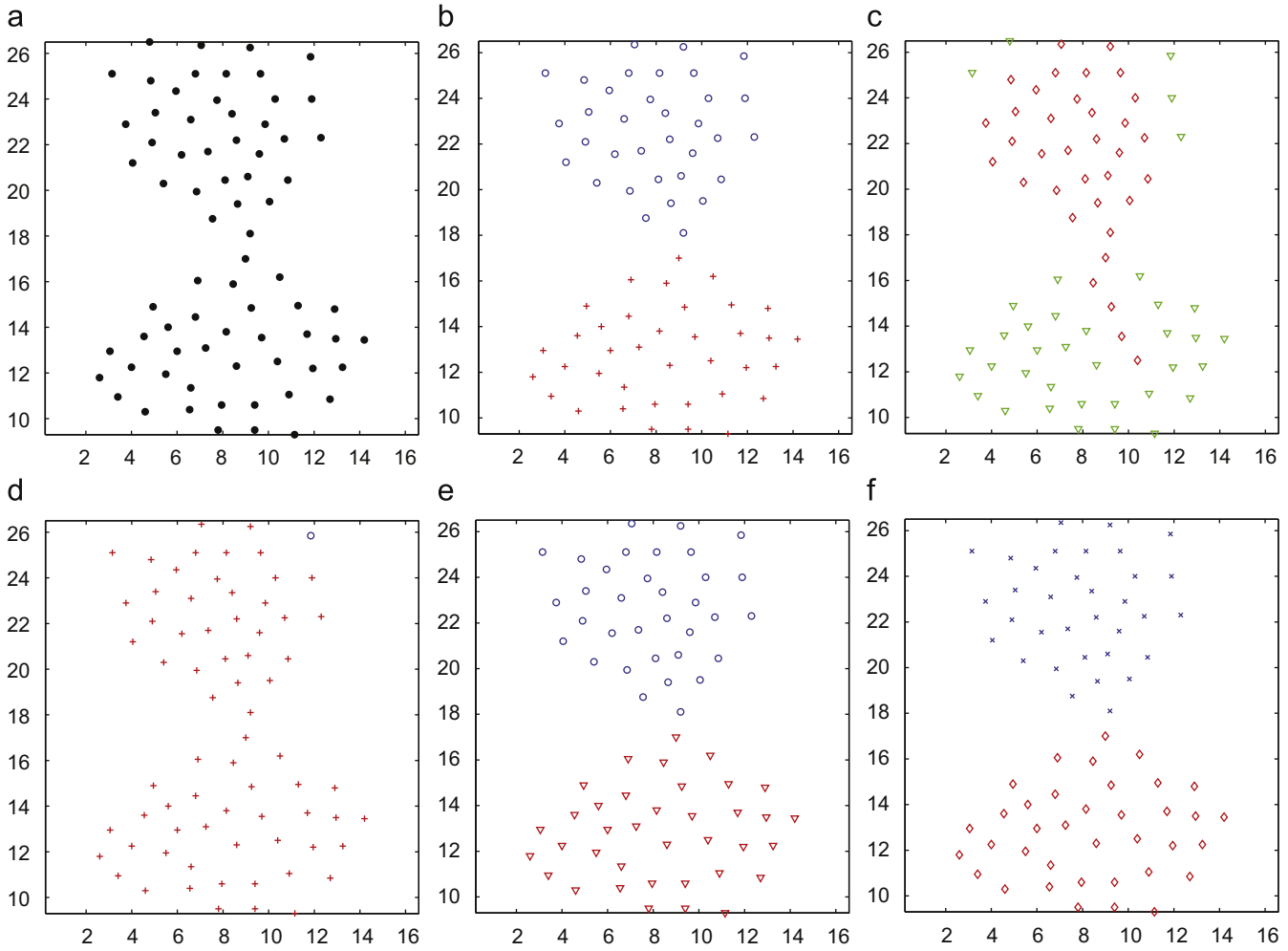


Fig. 13. Clustering results on DS3. (a) is the original dataset; (b) is the clustering result of k-means; (c) is the clustering result of DBScan (MinPts = 5, Eps = 1.5); (d) is the clustering result of single-linkage; (e) is the clustering result of spectral clustering; (f) is clustering result of 2-MSTClus.

algorithm is $O(|E|+|V|\log|V|)$ [35]. As the MST in a graph-based clustering method is generally constructed from a complete graph, $|E|$ is equal to $|V|^2$ and the computational complexity of Prim's algorithm is $O(|V|^2)$. In Step 1, accordingly, T_1 and T_2 are generated in $O(N^2)$, while Step 3 sorts the list $Rank(E')$ in $O(N\log N)$. Step 4 repeatedly removes an edge of $G'(X')$ and checks if a cut is achieved in $O(|X'|\log|X'|)$, where $|X'|\leq N$. The iteration time from Step 2 to Step 6 is the number of separated clusters in dataset, which is generally far less than N. Therefore, the time complexity of Algorithm 1 is $O(N^2)$.

In Step 1 of Algorithm 2, the constructing SP takes $O(N^2)$, and sorting $CP(SP)$ takes $O(N\log N)$. As a result, the Step 1 can be done in $O(N^2)$. The iteration in Step 2 of Algorithm 2 can be finished in $O(N\log N)$. Both Steps 3 and 4 in Algorithm 2 are executed in $O(N)$. The computational complexity of Algorithm 2 is $O(N^2)$.

Obviously, since the method 2-MSTClus is composed of Algorithms 1 and 2, its overall time complexity is $O(N^2)$.

4. Experimental results

4.1. Parameter setting

In the two proposed algorithms, although four parameters exist, they are all set to fixed values in all our experiments. The parameter λ determines whether a graph cut is valid or not when the framework deals with separated cluster problems. For the first graph cut on a separated problem, the numbers of edges removed from T_1 and T_2, respectively, are almost equal. The ideal situation is that two clusters are far away from each other, and when the first graph cut is achieved the number of the edges removed from T_1 is equal to that of the edges from T_2, i.e. $\lambda=0.5$. For non-separated problems, on the contrary, the edges removed from T_2 are in the majority, and that leads to significantly skewed ratio. However, a change of local density may disturb the ratio. Accordingly the parameter λ is relaxed to some degree. In all of our experiments, the parameter is set to 0.330. Specially, suppose that three edge removals result in a cut, although the $Ratio(E_{gcut})$ is 0.333 and only slightly greater than the parameter, the absolute difference is very small (only 1).

When the weight of an edge is computed, the parameter δ is employed to balance the relative weight (the ratio of lengths) and the absolute weight (the length of the edge). As the relative weight is crucial, δ is generally set to 0.9.

In touching problem algorithm, the parameter β is the margin of the number of *inconsistent vertices*. If it is too small, some neck-crossed partitions may be identified as invalid, while if it is too large, some non-neck-crossed partitions may be regarded as valid. As overlapped clusters are beyond the scope of this paper, and only slightly touching clusters are considered, β is set to a relatively small value, for instance, 2 in all our experiments. In addition, some extreme skewed partitions can be ignored by the parameter ε to save

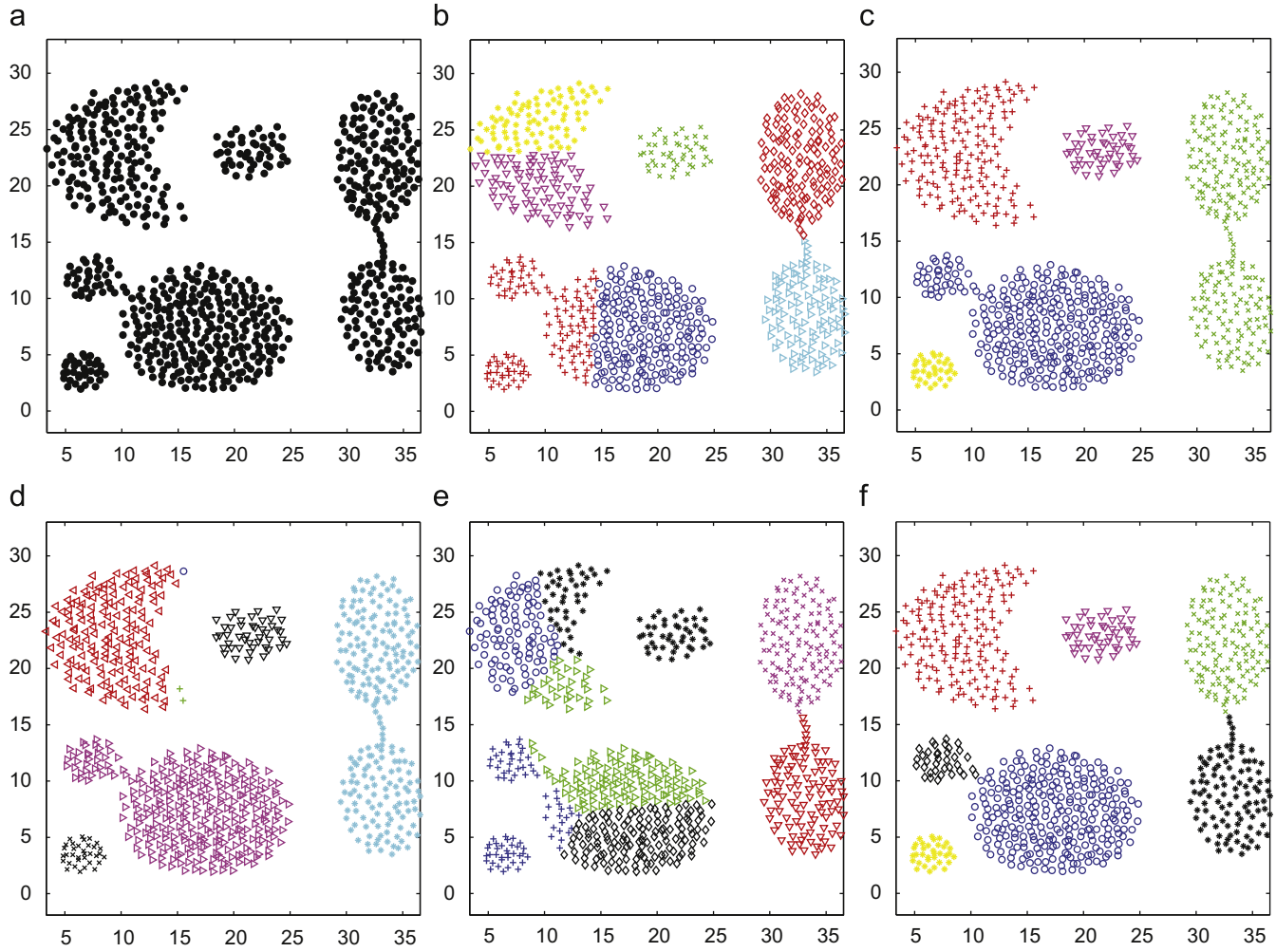


Fig. 14. Clustering results on DS4. (a) is the original dataset; (b) is the clustering result of k-means; (c) is the clustering result of DBScan (MinPts = 6, Eps = 1.5); (d) is the clustering result of single-linkage; (e) is the clustering result of spectral clustering; (f) is clustering result of 2-MSTClus.

the computational cost in touching problem algorithm. For example, cuts on some short branches (called hairs by [18]) are meaningless. Although the short branches can be identified adaptively, for simplicity we set $\varepsilon \in [0.01, 0.05]$ in the following experiments. If the number of vertices contained in a short branch is less than ε vertices, cuts on the short branch will be ignored.

4.2. Experiments on synthetic and real datasets

The proposed method 2-MSTClus is tested on five 2-D synthetic datasets, DS1–DS5, and two UCI datasets, Iris and Wine, and compared with four typical clustering algorithms, namely k-means, DBScan, single-linkage and spectral clustering, in which normalized cut is used. For DBScan, in the all experiments, the parameters are selected with the best clustering result.

The dataset DS1 is taken from [34] and illustrated in Fig. 11(a). It contains three spiral clusters, which are separated from each other in distance, therefore it is a distance-separated cluster problem. However, in terms of Handl's taxonomy [12], DS1 falls into the group of connectedness. Fig. 11(b)–(f) depict the clustering results of the four methods and 2-MSTClus. As DBScan and single-linkage prefer the datasets with connectedness, they can discover the three actual clusters, but k-means and spectral clustering cannot. 2-MSTClus can easily deal with DS1 as a separated problem and detect the three clusters.

Fig. 12 illustrates the clustering results of DS2, which is from [18]. It is a typical density-separated cluster problem. Since DBScan is a density-based and identifies clusters with the concept of density-reachable, it partitions DS2 well. However, k-means, single-linkage and spectral clustering are ineffective on DS2. While 2-MSTClus still produces ideal result by its separated algorithm.

The dataset DS3 is also taken from [18] and illustrated in Fig. 13. It is composed of two clusters, which are compact and slightly touched. k-means, which favors this kind of dataset, and spectral clustering have good performance on D3, whereas single-linkage and DBScan perform badly. Instead of the separated algorithm of 2-MSTClus, the touching algorithm of 2-MSTCLus can detect the two clusters.

In Fig. 14(a), the dataset DS4 is taken from [11]. Compared with the former three datasets DS1–DS3, this dataset is more complex. It consists of seven clusters, and is a composite cluster problem. All of the four algorithms, k-means, DBScan, single-linkage, spectral clustering fail on this dataset. However, 2-MSTClus identifies the seven clusters accurately.

In DS4, three clusters are distance-separated from others, while two pairs are internal touched. When the dataset is fed to 2-MSTClus, Algorithm 1 is first applied to it. Fig. 15(a) represents the first cut when top two weight edges are removed. As the $Ratio(E_{gcut})$ is 0.333, and greater than the threshold λ, the cut is valid. Next cut analysis is on the remaining six clusters. In Fig. 15(b), the removals of the top

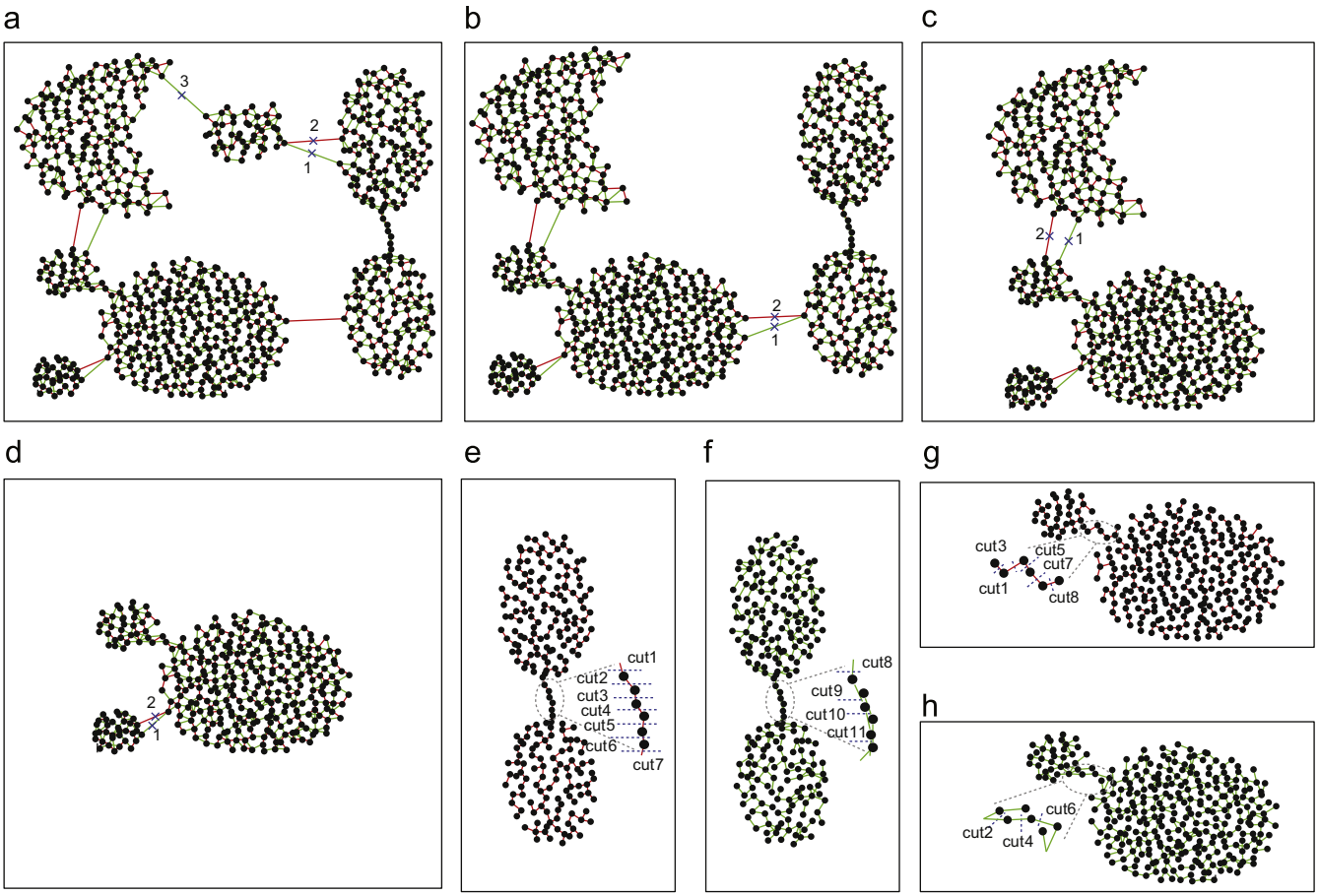


Fig. 15. Clustering process of 2-MSTClus on DS4. (a)–(d) illustrate the four graph cuts by the separated algorithm. In (e) and (f), Algorithm 2 is applied, *cut*1 and *cut*8, *cut*2 and *cut*9, *cut*3 and *cut*9, *cut*4 and *cut*9, *cut*5 and *cut*10, *cut*6 and *cut*11, *cut*7 and *cut*11,are similar ($\beta = 2$). In (g) and (h), *cut*1 and *cut*2, *cut*3 and *cut*6, *cut*5 and *cut*2, *cut*7 and *cut*2, *cut*4 and *cut*4, etc. are similar ($\beta = 2$).

two edges result in a valid cut, which partitions the six clusters into two groups. Similarly, the separated sub-dataset in Fig. 15(c) and (d) is further partitioned. Afterwards, Algorithm 1 could not partition the clusters any more. Then all the clusters are checked by Algorithm 2. Two touching clusters problems are figured out in Fig. 15(e)–(h), even though in Fig. 15(g) and (h) the two clusters have significant difference in their sizes.

The dataset DS5 is composed of three datasets from [18]. The left top dataset in Fig. 16(a) is a touching problem; the left bottom one is a distance-separated problem; while the right one is a density-separated problem. For this composite dataset, 2-MSTClus can identify the six clusters, but the four clustering methods *k*-means, single-linkage, DBScan and spectral clustering cannot.

In Fig. 17(a), (b) and (d), the distance-separated problems are identified with the removals of top two edges, respectively. With diverse densities, the two clusters in Fig. 17(c) are partitioned by Algorithm 1, and the corresponding $Ratio(E_{gcut})$ is 0.417, hence the graph cut is valid. As for the touching problem in the top left of Fig. 16(a), Algorithm 1 is ineffective. The similar cuts in Fig. 17(e) and (f) are detected by Algorithm 2.

Two real datasets from UCI are employed to test the proposed method. The first one is IRIS, which is a well-known benchmark for machine learning research. The dataset consists of three clusters with 50 samples each, and the one is well separated from the other two clusters, while the two clusters are slightly touched to each other. Similar to DS4 and DS5, it is also a composite clustering problem. When the dataset is fed to the 2-MSTClus, Algorithm 1 in the first round cuts off 50 samples, which constitute the separated cluster. Then the algorithm produces no clusters further. When Algorithm 2 is applied to the two subsets, only the cluster that is composed of 100 samples has some similar cuts between its T_1 and T_2, therefore, this cluster is further partitioned.

The performance comparison on IRIS is presented in Table 1. Four frequently-used external clustering validity indices are employed to evaluate the clustering results: Rand, Adjusted rand, Jaccard and FM. From Table 1, it is evident that 2-MSTClus performs best, since all of indices of 2-MSTClus are ranked first.

The second real dataset is WINE. It is composed of 178 samples, which fall into three clusters. From Table 2, 2-MSTClus performs only better than single-linkage. Compared with the former datasets, the performance of 2-MSTClus on WINE is slightly weakened. This is because some outliers exist in the dataset. In Algorithm 1, the graph cut criterion is a heuristic, however, the existence of outlier may affect the heuristic.

5. Discussion

Traditional MST-based clustering methods [18,19,24,33] make use of an MST to partition a dataset. A general way of partitioning is to remove the edges with relative large lengths, and one removal leads to a bipartition. Within an MST, although some crucial information of a dataset are collected, some are missed. T_1 and T_2 are combined to form a graph for the purpose of accumulating more evidence to partition datasets. In a two-round-MST based graph, a graph cut requires at least two edge removals. Only the evidence from T_1 and T_2 being consistent, is the graph cut valid. For analyzing

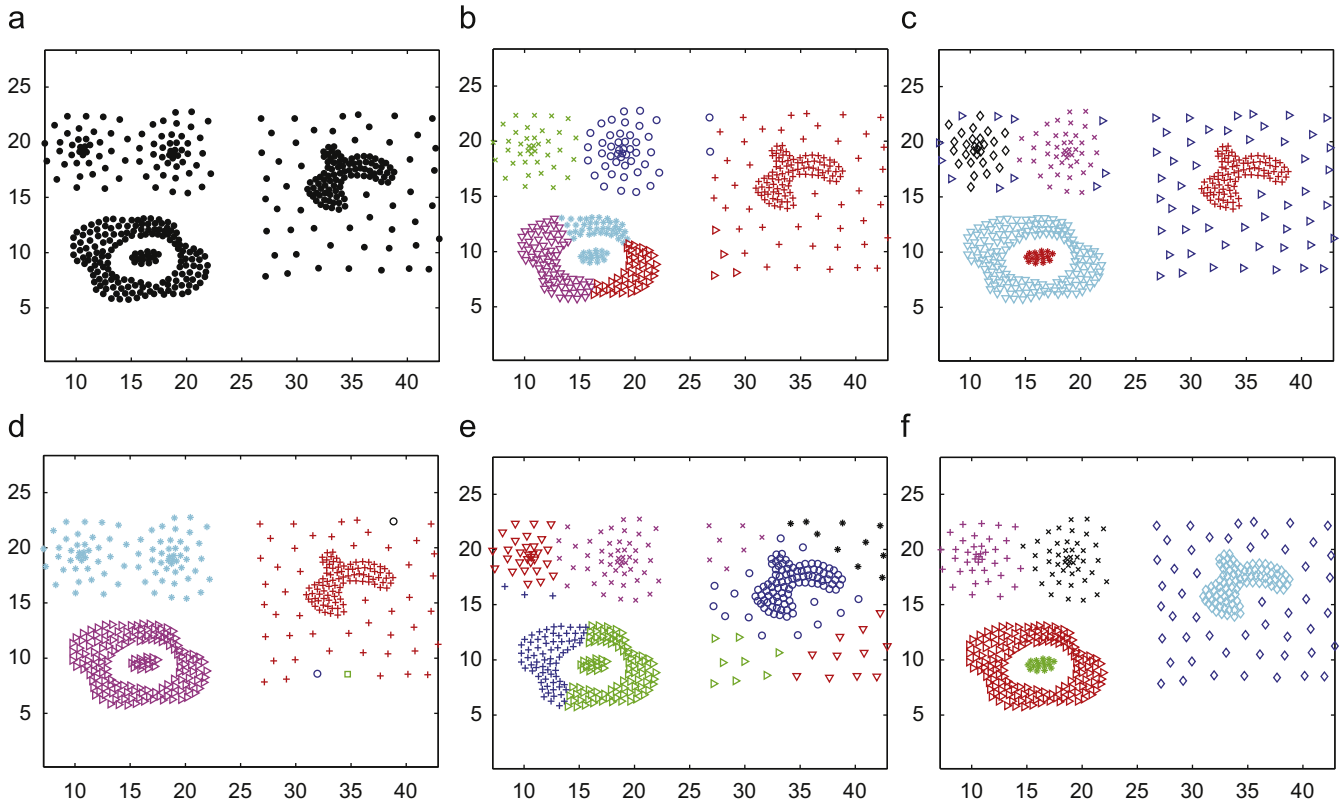


Fig. 16. Clustering results on DS5. (a) is the original dataset; (b) is the clustering result of *k*-means; (c) is the clustering result of DBScan (MinPts = 4, Eps = 1.4); (d) is the clustering result of single-linkage; (e) is the clustering result of spectral clustering; (f) is the clustering result of 2-MSTClus.

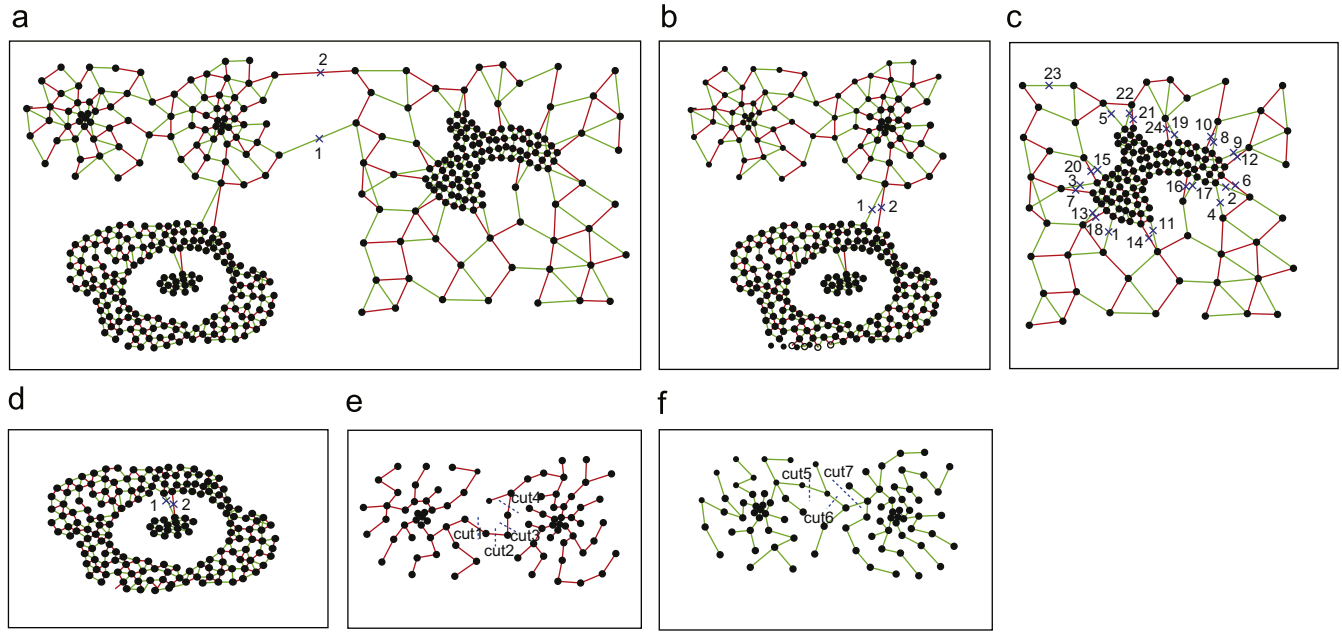


Fig. 17. Clustering process of 2-MSTClus on DS5. In (a), (b) and (d), the separated clustering method is applied. Only two edges are removed for each dataset and three valid graph cuts are achieved. (c) illustrates the partitioning process of the density-separated cluster problem. Totally 24 edges are removed for the graph cut, from which 14 edges come from T_2 and 10 from T_1. In (e) and (f), *cut*1 and *cut*5, *cut*1 and *cut*6, *cut*2 and *cut*7, *cut*3 and *cut*7, *cut*4 and *cut*7 are similar ($\beta = 2$).

the touching problems in 2-MSTClus, the concept of *inconsistent vertices* delivers the same idea.

The proposed method 2-MSTClus deals with a dataset in terms of which cluster problem it belongs to, separated problem or touching problem. The diversifications of sizes, shapes as well as densities of clusters have no effect on the clustering process.

A drawback of 2-MSTClus is that it is not robust to outliers. Although some outlier detection methods can be used to preprocess a dataset and remedy this drawback, we will discover more robust mechanism to outliers based on two-round-MST based graph in the future. In addition, the proposed method cannot detect the overlapping clusters. If a dataset composed of two overlapping clusters is

Table 1
Performance comparison on IRIS data.

Method	Rand	Adjusted rand	Jaccard	FM
k-Means	0.8797	0.7302	0.6959	0.8208
DBScan	0.8834	0.7388	0.7044	0.8268
Single-linkage	0.7766	0.5638	0.5891	0.7635
Spectral clustering	0.7998	0.5468	0.5334	0.6957
2-MSTClus	0.9341	0.8512	0.8188	0.9004

Table 2
Performance comparison on WINE data.

Method	Rand	Adjusted rand	Jaccard	FM
k-Means	0.7183	0.3711	0.4120	0.7302
DBScan	0.7610	0.5291	0.5902	0.7512
Single-linkage	0.3628	0.0054	0.3325	0.5650
Spectral clustering	0.7644	0.4713	0.4798	0.6485
2-MSTClus	0.7173	0.3676	0.4094	0.5809

dealt with 2-MSTClus, the two clusters will be recognized as one cluster.

If more MSTs are combined, for instance, $T_1, T_2, T_3, \ldots, T_k$, $k \leq N/2$, does the performance of the proposed method become better? In other words, how is a suitable k selected for a dataset? This is an interesting problem for the future work.

6. Conclusions

In this paper, a two-round-MST based graph is utilized to represent a dataset, and a clustering method 2-MSTClus is proposed. The method makes use of the good properties of the two-round-MST based graph, automatically differentiates separated problems from touching problems, and deals with the two kinds of cluster problem. It does not request user-defined cluster number, and is robust to different cluster shapes, densities and sizes. Our future work will focus on improving the robustness of 2-MSTClus to outliers and selecting a reasonable k for constructing k-MST.

Acknowledgments

We would like to thank the anonymous reviewers whose thoughtful comments and suggestions improved the quality of this paper. The paper is supported by the National Natural Science Foundation of China, Grant No. 60775036, No. 60475019, and the Research Fund for the Doctoral Program of Higher Education: No. 20060247039.

References

[1] W. Cai, S. Chen, D. Zhang, Fast and robust fuzzy c-means clustering algorithms incorporating local information for image segmentation, Pattern Recognition 40 (2007) 825–838.
[2] Z. Wu, R. Leahy, An optimal graph theoretic approach to data clustering: theory and its application to image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 15 (1993) 1101–1113.
[3] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan-Kaufman, San Francisco, 2006.
[4] Z. Yu, H.S. Wong, H. Wang, Graph-based consensus clustering for class discovery from gene expression data, Bioinformatics 23 (2007) 2888–2896.
[5] S. Bandyopadhyay, A. Mukhopadhyay, U. Maulik, An improved algorithm for clustering gene expression data, Bioinformatics 23 (2007) 2859–2865.
[6] A.K. Jain, M.C. Law, Data clustering: a user's dilemma, Pattern Recognition and Machine Intelligence, Lecture Notes in Computer Science, 3776, Springer, Berlin, Heidelberg, 2005, pp. 1–10.
[7] R. Xu, D. Wunsch II, Survey of clustering algorithms, IEEE Transactions on Neural Networks 16 (2005) 645–678.
[8] J. Kleinberg, An Impossibility Theorem for Clustering, MIT Press, Cambridge, MA, USA, 2002.
[9] H.G. Ayad, M.S. Kamel, Cumulative voting consensus method for partitions with variable number of clusters, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (2008) 160–173.
[10] A. Topchy, A.K. Jain, W. Punch, Clustering ensembles: models of consensus and weak partitions, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (2005) 1866–1881.
[11] A. Gionis, H. Annila, P. Tsaparas, Clustering aggregation, ACM Transactions on Knowledge Discovery from Data 1 (2007) 1–30.
[12] J. Handl, J. Knowles, An evolutionary approach to multiobjective clustering, IEEE Transactions on Evolutionary Computation 11 (2007) 56–76.
[13] A.L. Fred, A.K. Jain, Combining multiple clusterings using evidence accumulation, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (2005) 835–850.
[14] N. Bansal, A. Blum, S. Chawla, Correlation clustering, Machine Learning 56 (2004) 89–113.
[15] G.V. Nosovskiy, D. Liu, O. Sourina, Automatic clustering and boundary detection algorithm based on adaptive influence function, Pattern Recognition 41 (2008) 2757–2776.
[16] G. Karypis, E.H. Han, V. Kumar, CHAMELEON: a hierarchical clustering algorithm using dynamic modeling, IEEE Transactions on Computers 32 (1999) 68–75.
[17] P. Fränti, O. Virmajoki, V. Hautamäki, Fast agglomerative clustering using a k-nearest neighbor graph, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (2006) 1875–1881.
[18] C.T. Zahn, Graph-theoretical methods for detecting and describing gestalt clusters, IEEE Transactions on Computers C-20 (1971) 68–86.
[19] Y. Xu, V. Olman, D. Xu, Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning tree, Bioinformatics 18 (2002) 536–545.
[20] J.M. González-Barrios, A.J. Quiroz, A clustering procedure based on the comparison between the k nearest neighbors graph and the minimal spanning tree, Statistics & Probability Letters 62 (2003) 23–34.
[21] N. Päivinen, Clustering with a minimum spanning tree of scale-free-like structure, Pattern Recognition Letters 26 (2005) 921–930.
[22] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (2000) 888–905.
[23] C.H. Lee, et al., Clustering high dimensional data: a graph-based relaxed optimization approach, Information Sciences 178 (2008) 4501–4511.
[24] S. Bandyopadhyay, An automatic shape independent clustering technique, Pattern Recognition 37 (2004) 33–45.
[25] G.T. Toussaint, The relative neighborhood graph of a finite planar set, Pattern Recognition 12 (1980) 261–268.
[26] J. Handl, J. Knowles, D.B. Kell, Computational cluster validation in post-genomic data analysis, Bioinformatics 21 (2005) 3201–3212.
[27] M. Steinbach, L. Ertöz, V. Kumar, The Challenges of Clustering High Dimensional Data, New Directions in Statistical Physics: Bioinformatics and Pattern Recognition, L.T. Wille (Ed.), Springer, Berlin, 2002, pp. 273–307.
[28] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[29] B. Fischer, J.M. Buhmann, Path-based clustering for grouping of smooth curves and texture segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (2003) 513–518.
[30] L. Yang, K-edge connected neighborhood graph for geodesic distance estimation and nonlinear data projection, in: Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04), 2004.
[31] U. Brandes, M. Gaertler, D. Wagner, Engineering graph clustering: models and experimental evaluation, ACM Journal of Experimental Algorithmics 12 (2007).
[32] L. Hagen, A.B. Kahng, New spectral methods for ratio cut partitioning and clustering, IEEE Transactions on Computer-Aided Design 11 (1992) 1074–1085.
[33] Y. Li, A clustering algorithm based on maximal θ-distant subtrees, Pattern Recognition 40 (2007) 1425–1431.
[34] H. Chang, D.Y. Yeung, Robust path-based spectral clustering, Pattern Recognition 41 (2008) 191–203.
[35] T.H. Corman, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, second ed., MIT press, Cambridge, MA, 2001.

About the Author—CAIMING ZHONG is currently pursuing his Ph.D. in Computer Sciences at Tongji University, Shanghai, China. His research interests include cluster analysis, manifold learning and image segmentation.

About the Author—DUOQIAN MIAO is a professor of Department of Computer Science and Technology at Tongji University, Shanghai, China. He has published more than 40 papers in international proceedings and journals. His research interests include soft computing, rough sets, pattern recognition, data mining, machine learning and granular computing.

About the Author—RUIZHI WANG is a Ph.D. candidate of Computer Sciences at Tongji University of China. Her research interests include data mining, statistical pattern recognition, and bioinformatics. She is currently working on biclustering algorithms and their applications to various tasks in gene expression data analysis.

4

Paper **P4**

Reprinted from Information Sciences, Volume 181, Issue 16, C. Zhong, D. Miao, P. Fränti., “Minimum spanning tree based split - and - merge: A hierarchical clustering method, ” Pages 3397 – 3410, Copyright (2011), with permission from Elsevier.

Information Sciences 181 (2011) 3397–3410

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Minimum spanning tree based split-and-merge: A hierarchical clustering method

Caiming Zhong [a,b,c], Duoqian Miao [a,*], Pasi Fränti [b]

[a] Department of Computer Science and Technology, Tongji University, Shanghai 201804, PR China
[b] Department of Computer Science, University of Eastern Finland, P.O. Box: 111, FIN-80101 Joensuu, Finland
[c] College of Science and Technology, Ningbo University, Ningbo 315211, PR China

ARTICLE INFO

Article history:
Received 6 December 2009
Received in revised form 4 April 2011
Accepted 6 April 2011
Available online 13 April 2011

ABSTRACT

Most clustering algorithms become ineffective when provided with unsuitable parameters or applied to datasets which are composed of clusters with diverse shapes, sizes, and densities. To alleviate these deficiencies, we propose a novel split-and-merge hierarchical clustering method in which a minimum spanning tree (MST) and an MST-based graph are employed to guide the splitting and merging process. In the splitting process, vertices with high degrees in the MST-based graph are selected as initial prototypes, and *K*-means is used to split the dataset. In the merging process, subgroup pairs are filtered and only neighboring pairs are considered for merge. The proposed method requires no parameter except the number of clusters. Experimental results demonstrate its effectiveness both on synthetic and real datasets.

1. Introduction

Clustering plays an important role in data mining, pattern recognition, and machine learning. It aims at grouping N data points into K clusters so that data points within the same cluster are similar, while data points in diverse clusters are different from each other. From the machine learning point of view, clustering is unsupervised learning as it classifies a dataset without any a priori knowledge. A large number of clustering algorithms [20,37,46] have been presented in the literature since *K*-means [32], one of the most popular clustering algorithms, was published. The algorithms can be grouped into hierarchical clustering [12,15,16,19,27], partitional clustering [2,6,17,24,32,35], density-based clustering [10,18], grid-based clustering [1,43], model-based clustering [11,33], and graph-based clustering [29,38,40,45,47,49]. Hierarchical and partitional clustering are the two most common groups [30].

Generally, a hierarchical clustering algorithm partitions a dataset into various clusters by an agglomerative or a divisive approach based on a dendrogram. Agglomerative clustering starts by considering each point as a cluster, and iteratively combines two most similar clusters in terms of an objective function. In contrast, divisive clustering starts with only one cluster including all data points. It iteratively selects a cluster and partitions it into two subclusters. The main advantage of hierarchical clustering is that it produces a nested tree of partitions and can therefore be more informative than non-hierarchical clustering. Furthermore, with the dendrogram, the optimal number of clusters can be determined. However, hierarchical clustering has a relatively high computational cost. Single linkage [39] and complete linkage [26] are two well-known examples of hierarchical clustering algorithms, and they take $O(N^2 \log N)$ time.

Partitional clustering splits a dataset at once using an objective function. *K*-means is one of the most popular examples of partitional clustering. It employs mean-squared-error as its objective function. Its main advantage is that it runs efficiently: its computational complexity is $O(NKId)$, where I is the number of iterations for convergence, and d is the dimensionality of

* Corresponding author. Tel.: +86 21 69589867.
E-mail address: miaoduoqian@163.com (D. Miao).

0020-0255/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2011.04.013

the dataset. Since K and d are usually far less than N, the algorithm runs in a linear time on low-dimensional data. However, there does not exist a universal objective function that can be used to discover all different intrinsic structures of datasets. Therefore, partitional clustering produces inaccurate results when the objective function used does not capture the intrinsic structure of the data. This is the reason why partitional clustering algorithms are incapable of dealing with clusters of arbitrary shapes, different sizes and densities.

Clustering algorithms that combine the advantages of hierarchical and partitional clustering have been proposed in the literature [5,23,25,28,30,31]. This kind of hybrid algorithms analyzes the dataset in two stages. In the first stage, the dataset is split into a number of subsets with a partitioning criterion. In the second stage, the produced subsets are merged in terms of a similarity measure. Different split and merge approaches have been designed in several hybrid algorithms. CSM [30] first applies K-means to partition the dataset into K' subsets, where K' is an input parameter. Afterwards, single linkage, which uses a dedicated cohesion function as the similarity measure, is utilized to iteratively merge the K' subsets until K subsets are achieved. In the split stage, as K-means may produce different partitions in different runs, the final results may be unstable.

CHAMELEON [23] is another example of a hybrid clustering algorithm. It constructs a K-nearest neighbor graph, and employs a graph cut scheme to partition the graph into K' subsets. Relative inter-connectivity and relative closeness are defined to merge the subsets. Liu et al. [31] proposed a multi-prototype clustering algorithm, which can also be considered as a hybrid method. The method uses a convergence mechanism, and repeatedly performs split and merge operations until the prototypes remain unchanged. However, many empirical parameters are involved. Kaukoranta et al. [25] proposed a split-and-merge algorithm, where the objective function is to minimize the mean squared error.

A minimum spanning tree (MST) is a useful graph structure, which has been employed to capture perceptual grouping [21]. Zahn defined several criteria of edge inconsistency for detecting clusters of different shapes [49]. However, for datasets consisting of differently shaped clusters, the method lacks an adaptive selection of the criteria. Xu et al. [47] proposed three MST-based algorithms: removing long MST-edges, a center-based iterative algorithm, and a representative-based global optimal algorithm. But for a specific dataset, users do not know which algorithm is suitable.

In this paper, we propose a minimum spanning tree based split-and-merge method (SAM). It works on numerical data and assumes that the graph can be calculated in a vector space. In the splitting stage, three iterations of MSTs are used to construct a neighborhood graph called 3-MST graph. The vertices with high degrees in the graph are selected as the initial prototypes, and K-means is then applied. In the merge stage, the neighboring subsets with respect to the MST are filtered out and considered for merge.

The rest of the paper is organized as follows: In Section 2, the proposed algorithm is described. The experimental results are presented in Section 3, and conclusions are drawn in Section 4.

2. The proposed method

2.1. Problem formulation

Given a set of data points $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_N\}$, where $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots, x_{id})^T \in \Re^d$ is a feature vector, the goal of clustering is to partition the set X into K clusters: $C_1, C_2, \ldots, C_K$, where $C_i \neq \emptyset$, $C_i \cap C_j = \emptyset$, $X = C_1 \cup C_2 \cdots \cup C_K$, $i = 1{:}\ K$, $j = 1{:}\ K$, $i \neq j$. An undirected graph has been employed to represent a dataset for clustering [38,49]. Let $G(X) = (V, E)$ denote the undirected complete graph of X, the weights of the edges can be calculated with function $w : E \to \mathbb{R}$, where $V = X$, $E = \{(\mathbf{x}_i, \mathbf{x}_j) | \mathbf{x}_i, \mathbf{x}_j \in X, i \neq j\}$, and $w(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j)}$.

2.2. Overview of the split-and-merge algorithm

The algorithm consists of three main stages as illustrated in Fig. 1. First, an MST of the given dataset is constructed to guide the clustering process. In the splitting stage, K-means is applied to split the dataset into subsets, which are then adjusted according to the MST. In the merge stage, the final clusters are obtained by performing a carefully designed criterion-based merge that aims at maximizing intra-cluster similarity and minimizing inter-connectivity between the clusters.

2.3. The stage of constructing a 3-MST graph

2.3.1. Pruning leaves

A minimum spanning tree (MST) of graph $G(X)$ is a spanning tree T such that $W(T) = \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in T} w(\mathbf{x}_i, \mathbf{x}_j)$ is minimum. The leaves of an MST, called hairs in [49], are the vertices of degree 1. The leaves usually locate outside of kernels or skeletons of a dataset. For splitting two neighboring clusters, the data points in the neck [49] will have a negative effect on the clustering process. To alleviate the effect, we design a pruning step of removing the leaves so that the clusters are analyzed only on the essential data points. In [49], the skeleton of an MST was detected by repeatedly pruning until the number of removed hairs in two successive iterations remains the same. However, this may remove an entire cluster, and therefore, we conservatively perform only one pruning.

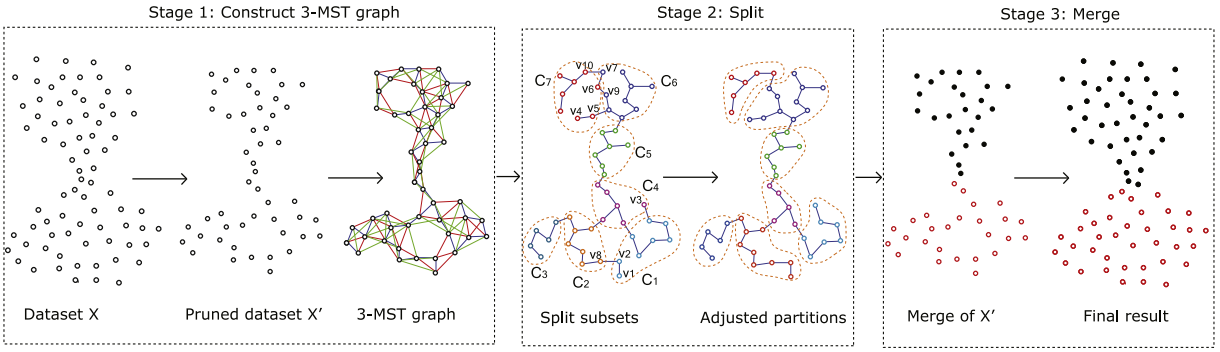


Fig. 1. The overview of split-and-merge. In Stage 1, the dataset X is pruned into X' according to the MST of X, and three iterations of MSTs of X' are computed and combined into a 3-MST graph. In Stage 2, X' is partitioned by K-means, where the initial prototypes are generated from the 3-MST graph. The partitions are then adjusted so that each partition is a subtree of the MST of X'. In Stage 3, the partitions are merged into the desired number of clusters and the pruned data points are distributed to the clusters.

Definition 1. Let X' be the pruned version of X as in

$$X' = X \setminus \{v_i | v_i \in V,\ degree(v_i) = 1\}, \tag{1}$$

where $degree(v_i)$ denotes the degree of vertex v_i in the MST of X.

2.3.2. Constructing a 3-MST graph

An MST describes the intrinsic skeleton of a dataset and accordingly can be used for clustering. In our proposed method, we use it to guide the splitting and merging processes. However, a single MST loses some neighborhood information that is crucial for splitting and merging. To overcome this drawback, we combine several MSTs and form a graph $G_{mst}(X,k)$ as follows:

Definition 2. Let $T_1 = f_{mst}(V,E)$ denote the MST of $G(X) = (V,E)$. The following iterations of an MST are defined as:

$$T_i = f_{mst}\left(V, E \setminus \cup_{j=1}^{i-1} T_j\right), \tag{2}$$

where f_{mst}: $(V,E) \to T$ is a function to compute an MST from graph $G(X) = (V,E)$, and $i \geqslant 2$.

In theory, the above definition of T_i is not rigorous because $E \setminus \cup_{j=1}^{i-1} T_j$ may produce isolated subgraph. For example, if there exists a vertex v in T_1 and the degree of v is $|V| - 1$, v will be isolated in $G(V, E \setminus T_1)$. Hence, the second MST (T_2) cannot be completed in terms of Definition 2. In practice, this is not a problem because the first MST T_1 is still connected and we can simply ignore it as a minor artefact, because it has no noticeably effect on the performance of the overall algorithm. However, for the sake of completeness, we solve this minor problem by always connecting such an isolated subgraph with an edge randomly selected from those connecting the isolated subgraph in T_1.

Let $G_{mst}(X,k)$ denote the k-MST graph, which is defined as a union of the k MSTs: $G_{mst}(X,k) = T_1 \cup T_2 \cup \cdots \cup T_k$. In this paper, we use $G_{mst}(X',k)$ to determine the initial prototypes in the split stage and to calculate the merge index of a neighboring partition pair in the merge stage. Here, k is set to 3 in terms of the following observation: 1 round of MST is not sufficient for the criterion-based merge but 3 iterations are. The number itself is a small constant and can be justified from computational point of view. Additional iterations do not add much to the quality, but only increase processing time. A further discussion concerning the selection of k can be found in Section 3.4.

2.4. Split stage

In the split stage, initial prototypes are selected as the nodes of highest degree in the graph $G_{mst}(X',k)$. K-means is then applied to the pruned dataset using these prototypes. The produced partitions are adjusted to keep the clusters connected with respect to the MST.

2.4.1. Application of K-means

The pruned dataset is first split by K-means in the original Euclidean space, where the number of partitions K' is set to $\sqrt{|X'|}$. This is done under the assumption that the number of clusters in a dataset is smaller than the square root of the number of patterns in the dataset [3,36]. If $\sqrt{|X'|} \leqslant K$, K' can be set to $K + \lambda(|X'| - K)$ to grantee that K' is greater than K, where $0 < \lambda < 1$. Since this is not a normal situation, we do not discuss the parameter λ in this paper. Moreover, if $|X'| \leqslant K$, the split-and-merge scheme will degenerate into a traditional agglomerative clustering.

However, to determine the K' initial prototypes is a tough problem, and a random selection would give an unstable splitting result. For example, the method proposed in [30] uses K-means with randomly selected prototypes in its split stage, and the final clustering results are not unique. We therefore utilize the MST-based graph $G_{mst}(X',3)$ to avoid this problem.

Definition 3. Let v_i be the ith prototype from $G_{mst}(X',3)=(V,E)$, $V_{i-1}=\{v_1,v_2,\ldots,v_{i-1}\}$, $E_{i-1}=\{(\mathbf{x}_i,\mathbf{x}_j)|(\mathbf{x}_i,\mathbf{x}_j)\in E\wedge(\mathbf{x}_i=v\vee\mathbf{x}_j=v)\wedge v\in V_{i-1}\}$, v_i is generated as:

$$v_i=\arg\max_{v\in V\setminus V_{i-1}} Car\big(\{(\mathbf{x}_i,\mathbf{x}_j)|(\mathbf{x}_i,\mathbf{x}_j)\in(E\setminus E_{i-1})\wedge(\mathbf{x}_i=v\vee\mathbf{x}_j=v)\}\big), \tag{3}$$

where $Car(S)$ denotes the cardinality of set S.

The above definition determines the vertex with the maximum degree as a prototype. It is a recursive definition, when we determine the ith prototype v_i, the edges connected to the existing $i-1$ prototypes will not be considered for counting the degrees of vertices. However, there may exist two or more vertices in $G_{mst}(X',3)$ simultaneously having the maximum degree. In this case, the vertex with the minimum sum of weights of its edges is to be selected.

After the K' initial prototypes have been determined, K-means is applied. The seven subsets $(C_1,\ldots,C_7)$ in stage 2 of Fig. 1 are the partitions on T_1 produced by K-means.

2.4.2. Adjusting partitions

In traditional MST-based clustering methods such as [47,49], each partition is a subtree of the MST rather than a forest. This is a reasonable observation, because data points in the same cluster are usually in the same branch of the MST. However, the partitions in stage 2 of Fig. 1 may not coincide with the observation. For example, the four subsets (C_1,C_4,C_6,C_7) are forests but not trees.

Furthermore, the partitions in stage 2 of Fig. 1 reduce the ability of the MST to guide the merging process. This is because only neighboring partitions will be considered to be merged, and the neighboring partitions can be easily determined by the MST. However, if the partitions are forests, the MST would lose the ability.

Therefore, a process of redistributing the vertices is designed to transform a forest into a tree. The process is described as follows. Suppose that T_1 of X' is partitioned into K' forests, which are denoted as $F_1,F_2,\ldots,F_{K'}$. For redistributing the vertices, the main tree in each forest is defined.

Definition 4. Let $F_i=\{t_1,t_2,\ldots,t_j,\ldots,t_n\}$, and each t_j being a tree. The main tree of F_i is defined as:

$$Maintree(F_i)=\arg\max_{t_j\in F_i} Car(t_j), \tag{4}$$

where $Car(t_j)$ denotes the edge number of the tree t_j.

The vertices which are not included in any main tree will be re-allocated so that each subset is a subtree of T_1 of X'. Suppose $F_1,\ldots,F_7$ are the forests of the subsets $C_1,\ldots,C_7$ in Fig. 1, respectively. To adjust the seven subsets into seven subtrees, vertices $v1$ and $v2$ can be re-allocated to $Maintree(F_2)$, $v3$ to $Maintree(F_1)$, $v4$, $v5$, $v6$ to $Maintree(F_6)$, and $v7$ to $Maintree(F_7)$. The re-allocation process is described as follows.

Let SV denote the set of vertices that are not in any main tree, ST the set of the main trees. The redistribution operation is defined as: for an edge e_{ab} from T_1 of X', if $a\in SV$, and $\exists T(T\in ST\wedge b\in T)$, then the vertex a is redistributed to T. For example, $v2$ is re-allocated to $Maintree(F_2)$, since $e_{v2v8}\in T_1$, $v2\in SV$, and $v8\in Maintree(F_2)$. We iteratively perform this operation until all of the vertices in SV are re-allocated to the main trees.

The above operation may produce non-unique redistributions. Take $v6$ and $v7$ for example, if e_{v6v9} and e_{v7v6} are considered before e_{v7v10}, then $v6$ and $v7$ will be redistributed to $Maintree(F_6)$. Similarly, $v6$ and $v7$ may be redistributed to $Maintree(F_7)$, or $v6$ to $Maintree(F_6)$ and $v7$ to $Maintree(F_7)$. However, the non-unique redistribution does not noticeably affect the final clustering result for the two reasons: one is that the subtrees in a subset are usually smaller than the main tree of the subset, the other one is that the non-uniqueness most often happens inside rather than between the expected clusters.

2.5. Merge stage

After X' has been split into K' subgroups, the merge stage is performed to obtain the final clusters. In the merging process, the crucial problem is to determine which pairs of subgroups should be merged. By brute force, there are $K'\times(K'-1)/2$ candidate pairs for merge. In this paper, a stepwise refinement method is taken to address the problem. At the beginning, T_1 is employed to filter out the pairs. On the basis of MST-based clustering [47,49], unconnected subtrees cannot be merged. For example, the subgroups C_3 and C_7 in Fig. 1 will not be considered as a pair for merge. Consequently, only the neighboring pairs with respect to T_1 are the candidates.

2.5.1. Neighboring pairs

Definition 5. Let *Pairs* be the set of neighboring pairs from X' as in:

$$Pairs=\{(C_i,C_j)|\exists(\mathbf{x}_p,\mathbf{x}_q)\in T_1,\ (\mathbf{x}_p\in C_i\wedge\mathbf{x}_q\in C_j)\vee(\mathbf{x}_p\in C_j\wedge\mathbf{x}_q\in C_i)\}, \tag{5}$$

where $i\neq j$.

For any two subgroups, if there exists an edge in T_1 connecting them, then the two subgroups belong to *Pairs*.

For selecting the best pair to be merged, we define an *inter-connectivity index* and an *intra-similarity index* to rank the pairs.

2.5.2. Inter-connectivity index

Definition 6. Suppose $(C_i, C_j) \in Pairs$. In $G_{mst}(X', 3)$, let $E_{inter}(C_i, C_j)$ be the set of edges across the partitions C_i and C_j:

$$E_{inter}(C_i, C_j) = \{(\mathbf{x}_p, \mathbf{x}_q) | ((\mathbf{x}_p \in C_i \wedge \mathbf{x}_q \in C_j) \vee (\mathbf{x}_p \in C_j \wedge \mathbf{x}_q \in C_i))\}. \quad (6)$$

In Fig. 2, $E_{inter}(C_6, C_7) = \{(a,g), (a,h), (b,g), (c,i), (d,i), (d,j), (e,i), (e,j), (e,l), (e,k)\}$, i.e. the edges crossing the dotted curve.

Definition 7. Suppose that $(C_i, C_j) \in Pairs$. In $G_{mst}(X', 3)$, set $V_{i,j} = \{\mathbf{x}_p | (\mathbf{x}_p, \mathbf{x}_q) \in E_{inter}(C_i, C_j) \wedge \mathbf{x}_p \in C_i\}$, $E_{i,j}$ is a set of edges within C_i where at least one endpoint is in $V_{i,j}$:

$$E_{i,j} = \{(\mathbf{x}_p, \mathbf{x}_q) | \mathbf{x}_p, \mathbf{x}_q \in C_i \wedge (\mathbf{x}_p \in V_{i,j} \vee \mathbf{x}_q \in V_{i,j})\}. \quad (7)$$

In Fig. 2, for example, $V_{6,7} = \{g,h,i,j,k,l\}$, $V_{7,6} = \{a,b,c,d,e\}$, $E_{6,7}$ includes all the internal edges of C_6 (i.e. the edges whose both endpoints are from C_6) except $\{(n,m), (n,o), (n,p), (m,o), (m,p), (m,q), (p,o), (p,q)\}$, and $E_{7,6}$ includes all the internal edges of C_7. Actually, $E_{i,j}$ is defined as the edges in the region of C_i that is close to C_j.

Connection span is then defined with respect to $G_{mst}(X', 3)$ as a factor of measuring the similarity of two clusters based on the width of their connection. The intuition behind this index is that it estimates the size of the common border of the two clusters. The larger the common border is, the higher the priority for merging these clusters will be. It depends only on the distances in vector space and makes no assumption on the dimensionality.

Definition 8. Suppose that $(C_i, C_j) \in Pairs$. In $G_{mst}(X', 3)$, the connection span of C_i with respect to C_j is:

$$ConnSpan_{i,j} = \max_{\mathbf{x}_p, \mathbf{x}_q \in V_{i,j}} w(\mathbf{x}_p, \mathbf{x}_q). \quad (8)$$

In Fig. 3, the connection span of C_6 and C_7 is marked by the dotted edges (g,k) and (a,e), respectively.

Definition 9. Suppose that $(C_i, C_j) \in Pairs$. In $G_{mst}(X', 3)$, the inter-connectivity (IC) of C_i and C_j is defined as:

$$IC(C_i, C_j) = \frac{|E_{inter}(C_i, C_j)|}{\min(|C_i|, |C_j|)} \times \frac{\min(Avg(E_{i,j}), Avg(E_{j,i}))}{Avg(E_{inter}(C_i, C_j))} \times \max\left(ConnSpan_{i,j}, ConnSpan_{j,i}\right), \quad (9)$$

where $Avg(E)$ denotes the average weight of an edge set E.

From Eq. (9), the inter-connectivity index IC is a composite of three factors. The factor $|E_{inter}(C_i, C_j)|/\min(|C_i|, |C_j|)$ describes that the more edges straddling a pair of clusters, the stronger is the connective strength between the two clusters. In the second factor, $\min(Avg(E_{i,j}), Avg(E_{j,i}))$ is the minimum of the average weights of the edges that are in the two close regions from C_i and C_j, respectively. $Avg(E_{inter}(C_i, C_j))$ is the average weight of the edges straddling C_i and C_j. This factor reflects that the ratio of the average weight of the straddling edges to the minimum average weight in the two close regions is inversely proportional to the connectivity of the two clusters. In fact, this is based on the observation that if the density between the two clusters is low compared with those of the close regions, the two clusters have a small probability to be merged. The third factor $\max(ConnSpan_{i,j}, ConnSpan_{j,i})$ indicates that a pair of clusters with large connection span has strong connectivity.

2.5.3. Intra-similarity index

For describing the intra-similarity of pairs of clusters, a strategy inspired by Karypis et al. [23] is employed. Each cluster of a neighboring pair is bisected in terms of T_1, and the corresponding inter-edges with respect to $G_{mst}(X', 3)$ are used to evaluate the intra-similarity between the two clusters. The process is described as follows.

Suppose C_i is the cluster to be bisected, and T_{i1} is the subtree of T_1 restricted to nodes and edges of C_i. The bisecting edge $e_{bisect} \in T_{i1}$ can be determined as:

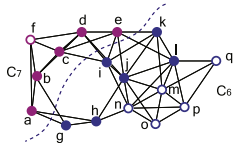


Fig. 2. Illustration of the inter edges between subgroups C_6 and C_7.

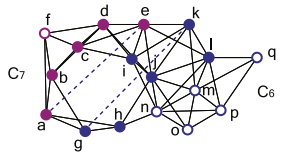


Fig. 3. Illustration of *connSpan* between subgroups C_6 and C_7.

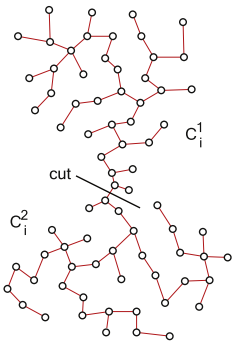


Fig. 4. Illustration of the bisection process of a cluster C_i. The data set is bisected into C_i^1 and C_i^2 by the cut so that the difference of cardinalities of C_i^1 and C_i^2 is minimal.

$$e_{bisect} = \arg\min_{e_j \in T_{i1}} \left| Car(t_{j1}) - Car(t_{j2}) \right|, \tag{10}$$

where t_{j1} and t_{j2} are two subtrees of T_{i1} generated by removing e_j from T_{i1}, and $Car(t)$ denotes the number of edges in a tree t. An example of the bisection is shown in Fig. 4. Correspondingly, C_i is bisected into two subsets: C_i^1 and C_i^2. The inter-edge set $E_{inter}(C_i^1, C_i^2)$ with respect to $G_{mst}(C_i, 3)$ is obtained by Definition 6. The intra-similarity of C_i and C_j is then defined as follows:

Definition 10. Suppose that $(C_i, C_j) \in Pairs$, C_i is bisected into C_i^1 and C_i^2, C_j is bisected into C_j^1 and C_j^2, the *intra-similarity* (*IS*) of the pair (C_i, C_j) is defined as:

$$IS(C_i, C_j) = \frac{1}{r_1 \times r_2} \times \frac{\sqrt{Avg\left(E_{inter}\left(C_i^1, C_i^2\right)\right) \times Avg\left(E_{inter}\left(C_j^1, C_j^2\right)\right)}}{Avg\left(E_{inter}\left(C_i^1, C_i^2\right)\right) + Avg\left(E_{inter}\left(C_j^1, C_j^2\right)\right)}, \tag{11}$$

where r_1 and r_2 are the numbers of edges of $E_{inter}(C_i^1, C_i^2)$ and $E_{inter}(C_j^1, C_j^2)$, respectively, and $Avg(E)$ denotes the average weight of an edge set E.

Eq. (11) implies that the intra-similarity between the pair (C_i, C_j) is high when the two averages $Avg(E_{inter}(C_i^1, C_i^2))$ and $Avg(E_{inter}(C_j^1, C_j^2))$ are close to each other. For the two numbers r_1 and r_2, unbalanced and small sizes indicate that the two clusters are more likely to be merged.

Taking into account the inter-connectivity and the intra-similarity as a whole, we define the overall merge index as:

$$R(C_i, C_j) = IC(C_i, C_j) \times IS(C_i, C_j). \tag{12}$$

The neighboring pair with the highest $R()$ value is merged first. After one neighboring pair is merged, *Pairs* and corresponding $R()$ values are updated. When K partitions are achieved, the merge process stops. Because a pruned leaf $\in X$ is connected to exact one vertex $\mathbf{x}_i \in X'$ in the MST of X, it is assigned the same label as $\mathbf{x}_i$.

Finally, the proposed algorithm is described as follows:

Algorithm 1: Split-and-merge (SAM)

Input: Dataset X, number of clusters K
Output: K clusters $C_1, C_2, \ldots, C_K$

1. Construct 3-MST graph
   - 1.1 Construct an MST on X
   - 1.2 Produce X' by pruning the leaves of the MST
   - 1.3 Create three MSTs on X': T_1, T_2 and T_3
   - 1.4 Compute 3-MST graph based on X': $G_{mst}(X', 3) \leftarrow T_1 \cup T_2 \cup T_3$
2. Split the pruned dataset X' into clusters
   - 2.1 Select K' highest degree nodes from $G_{mst}(X', 3)$ as initial prototypes
   - 2.2 Apply K-means with the prototypes to produce K' partitions
   - 2.3 For each of the partitions, find its main tree in T_1
   - 2.4 For each of the subtrees, repeatedly combine it with another subtree until it belongs to a main tree
3. Merge the partitions into final clusters
   - 3.1 Generate the set of neighboring partition pairs *Pairs*
   - 3.2 For each pair $(C_i, C_j) \in Pairs$, calculate the merge criterion $R(C_i, C_j)$
   - 3.3 Repeatedly merge the pair with maximum $R()$-value until K clusters have been obtained
   - 3.4 Add the pruned leaves to the clustering using T_1

2.6. Computational complexity

To construct an MST, if Fibonacci heaps are employed in Prim's algorithm, the running time is $O(|E| + |V|\log|V|)$ [9]. In this paper, an MST is computed from a complete graph, $|E| = O(|V|^2) = O(N^2)$, and therefore, the complexity of constructing an MST is $O(N^2)$.

In Step 1.1, the MST is constructed in $O(N^2)$ time, and in Step 1.2, the complexity of pruning the leaves from the MST is $O(N)$. In Step 1.3, to generate the three MSTs of X' takes $O(N^2)$ time. $G_{mst}(X', 3)$ is obtained in $O(N)$ time in Step 1.4. In Step 2.1, the initial prototypes are determined in $O(N)$, the time complexity of K-means on X' is $O(K'NdI)$, where d is the dimensionality of X', and I is the number of iterations. Since $K' \leqslant \sqrt{N}$, the total complexity of Step 2.1 is $O(N^{3/2}dI)$. The main trees can be determined in $O(N)$ in Step 2.2, and re-allocation can also be achieved in $O(N)$ in Step 2.3. Step 3.1 takes $O(N)$ to generate *Pairs*, and Step 3.2 takes $O(N)$ to compute the merge index $R()$ for every pair in *Pairs*. As the merging of a pair can be achieved in constant time, the maximum complexity of updating the merge index $R()$ is $O(N)$, and the number of iterations of Step 3.3 is $O(\sqrt{N})$, the worst case complexity of this Step is therefore $O(N^{3/2})$. Step 3.4 can be processed in constant time.

To sum up, the computational complexity of the proposed method (SAM) is $O(N^2)$, which is dominated by the construction of the 3-MST graph. If also the factor of dimensionality d is considered, the exact complexity would be $O(N^2d)$.

3. Experimental results

The clustering performance of the proposed method is evaluated on six synthetic and four real datasets. The first four synthetic datasets DS1–DS4 are taken from the literature [22,4,14,13], and the next two DS5, DS6 are from [42]. These datasets are illustrated in Fig. 5. The four real world instances are taken from the UCI datasets [50], including IRIS, WINE, Breast Cancer Wisconsin (WBC), and Breast Cancer Wisconsin Diagnostic (WDBC). The descriptions of these datasets are shown in Table 1.

The proposed method SAM is compared to the following six clustering algorithms:

1. K-means [32].
2. DBScan [10].
3. Single linkage [39].
4. Spectral clustering [38].
5. CSM [30].
6. CHAMELEON [23].

The first four algorithms are traditional clustering methods, whereas the next two are hybrid ones. In addition, three variants of SAM are performed to demonstrate the importance of the various steps and design choices of the algorithm:

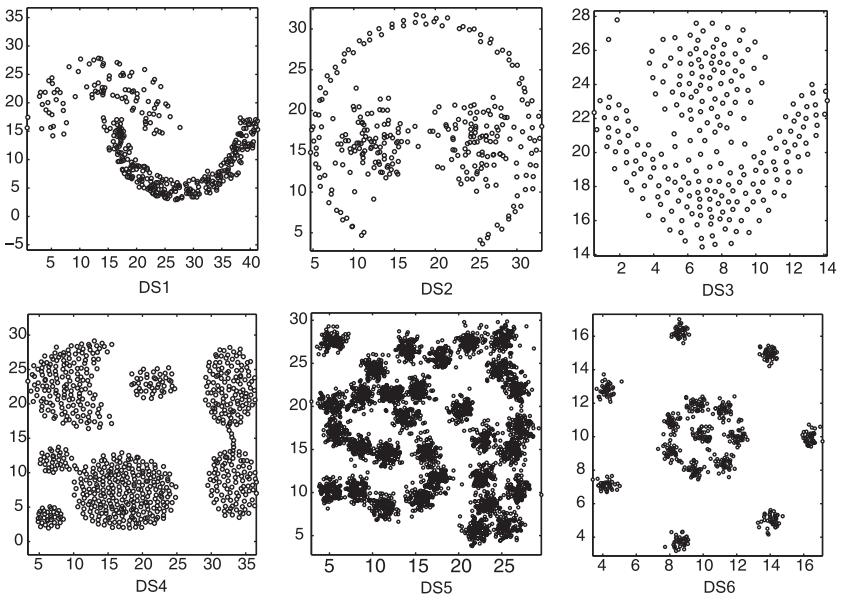


Fig. 5. Illustration of the six original synthetic datasets.

Table 1
Descriptions of used datasets.

Data set	Data size (N)	Dimensionality (d)	Number of clusters (K)
DS1	373	2	2
DS2	300	2	3
DS3	240	2	2
DS4	788	2	7
DS5	3100	2	31
DS6	600	2	15
Iris	150	4	3
Wine	178	13	3
WBC	683	9	2
WDBC	569	30	2

1. SAM-No-Pruning: split-and-merge without the pruning step.
2. SAM-2-MST: split-and-merge but using 2-MST instead of 3-MST.
3. SAM-SL: split-and-merge but using single linkage within the merge stage instead of the criterion based on $R()$-values.

As K-means and CSM may produce different partitions in different runs, in the following experiments we take the best clustering result out of 10 trial runs performed for each dataset in terms of one of the most used external clustering validity indices, Adjusted Rand (see Section 3.1). For DBScan, since we have no a priori knowledge about the parameters (*MinPts* and *Eps*), proper values were selected by trial and error.

3.1. External clustering validity indexes

An external clustering validity index is defined to measure the degree of correspondence between a clustering result and the prespecified partitions [41]. We will employ the four popular external indexes, Rand, FM, Jaccard, and Adjusted Rand ([34,41,44,48]), to evaluate the quality of the clustering results. Briefly, these four indexes are described as follows.

Suppose $\{P_1,\ldots,P_L\}$ is the set of prespecified partitions, $\{C_1,\ldots,C_K\}$ is a set of partitions produced by a clustering. For a pair of vectors $(\mathbf{x}_u,\mathbf{x}_v)$, it is referred as: (a) SS pair if $\exists I(\mathbf{x}_u \in P_I \wedge \mathbf{x}_v \in P_I)$ and $\exists J(\mathbf{x}_u \in C_J \wedge \mathbf{x}_v \in C_J)$, (b) SD pair if $\exists I(\mathbf{x}_u \in P_I \wedge \mathbf{x}_v \in P_I)$ and $\nexists J(\mathbf{x}_u \in C_J \wedge \mathbf{x}_v \in C_J)$, (c) DS pair if $\nexists I(\mathbf{x}_u \in P_I \wedge \mathbf{x}_v \in P_I)$ and $\exists J(\mathbf{x}_u \in C_J \wedge \mathbf{x}_v \in C_J)$, (d) DD pair if $\nexists I(\mathbf{x}_u \in P_I \wedge \mathbf{x}_v \in P_I)$ and $\nexists J(\mathbf{x}_u \in C_J \wedge \mathbf{x}_v \in C_J)$. Let a, b, c, d denote the numbers of SS, SD, DS, and DD pairs, respectively, and M the number of total pairs, the four indexes are defined as:

1. Rand

$$R = (a+d)/M. \tag{13}$$

2. Jaccard

$$J = a/(a+b+c). \tag{14}$$

3. FM

$$FM = \sqrt{\frac{a}{a+b}\frac{a}{a+c}}. \tag{15}$$

4. Adjusted Rand

$$AR = \frac{2(Ma-(a+b)(a+c))}{M(2a+b+c)-2(a+b)(a+c)}. \tag{16}$$

3.2. Results on synthetic datasets

DS1: This instance contains two datasets shaped like a crescent with different densities. The clustering results are illustrated in Fig. 6. SAM and SAM-No-Pruning, DBScan, CSM and CHAMELEON can partition the dataset properly. Both SAM-2-MST and SAM-SL fail. Since K-means favors spherical clusters, it fails on DS1. Single linkage clustering produces unsatisfactory partitions because it measures the distance between two clusters as the minimum distance between the pairs of data points in the two clusters. In the spectral clustering algorithm, the similarity matrix is created by a Gaussian kernel function with Euclidean distances, but the clustering result of this algorithm is similar to that of K-means. Although DBScan produces the expected partitions, it was difficult to tune the two parameters to achieve the proper result.

DS2: The set is composed of two Gaussian distributed clusters and one unclosed ring cluster surrounding the first two. Fig. 7 illustrates the clustering results. SAM and SAM-2-MST provide satisfactory clustering results, whereas SAM-No-Pruning

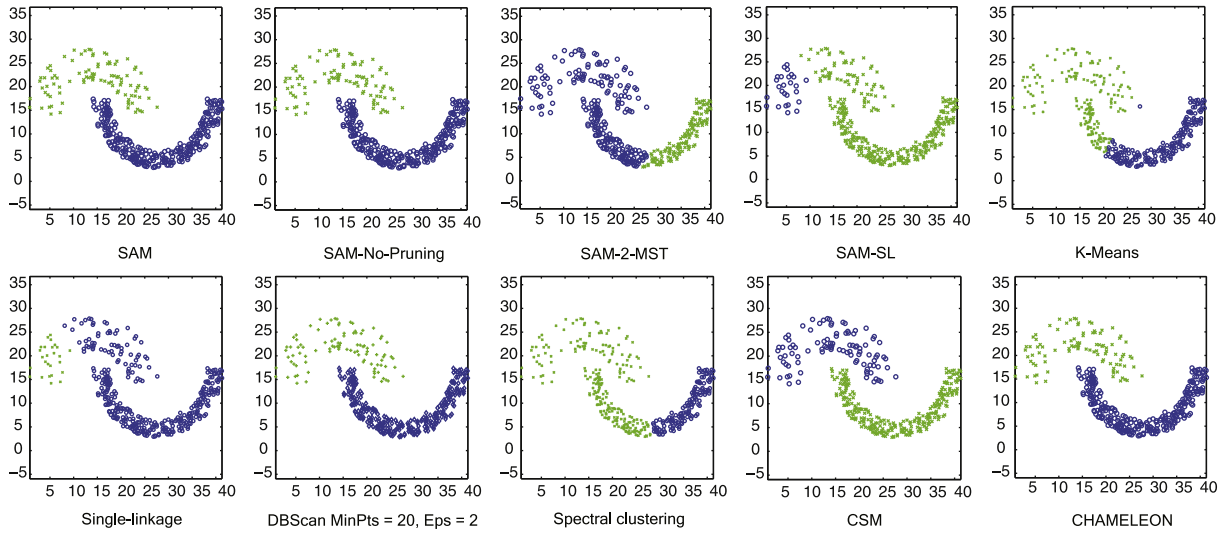


Fig. 6. Illustration of clustering results on DS1.

and SAM-SL produce improper results. CSM can sometimes identify the proper clusters, but not always. The results of DBScan and spectral clustering are not perfect, but better than those of *K*-means, single linkage, and CHAMELEON.

DS3: This dataset contains a spherical cluster and a half ring shaped cluster. The clustering results are shown in Fig. 8. All the algorithms except *K*-means and single linkage discover the expected clusters.

DS4: This dataset consists of 7 Gaussian distributed clusters. Fig. 9 illustrates the clustering results. SAM, its variant SAM-No-Pruning and SAM-2-MST, spectral clustering, CSM, and CHAMELEON find the expected clusters. SAM-SL and *K*-means provide partitions with low quality.

DS5: This dataset consists of 31 Gaussian distributed clusters. The clustering results are illustrated in Fig. 10. Spectral clustering, CSM, and CHAMELEON produce the expected partitions. SAM and SAM-SL are also successful except they fail to detect the cluster on the rightmost part. SAM-No-Pruning and SAM-2-MST perform much worse, whereas *K*-means, single linkage, and DBScan fail to detect almost all the expected clusters.

DS6: The 15 Gaussian distributed clusters in this dataset are arranged in two concentric circles. Fig. 11 describes the clustering results. SAM, SAM-SL, spectral clustering, CSM, and CHAMELEON produce the proper partitions, but SAM-No-Pruning, SAM-2-MST, *K*-means, single linkage, and DBScan do not.

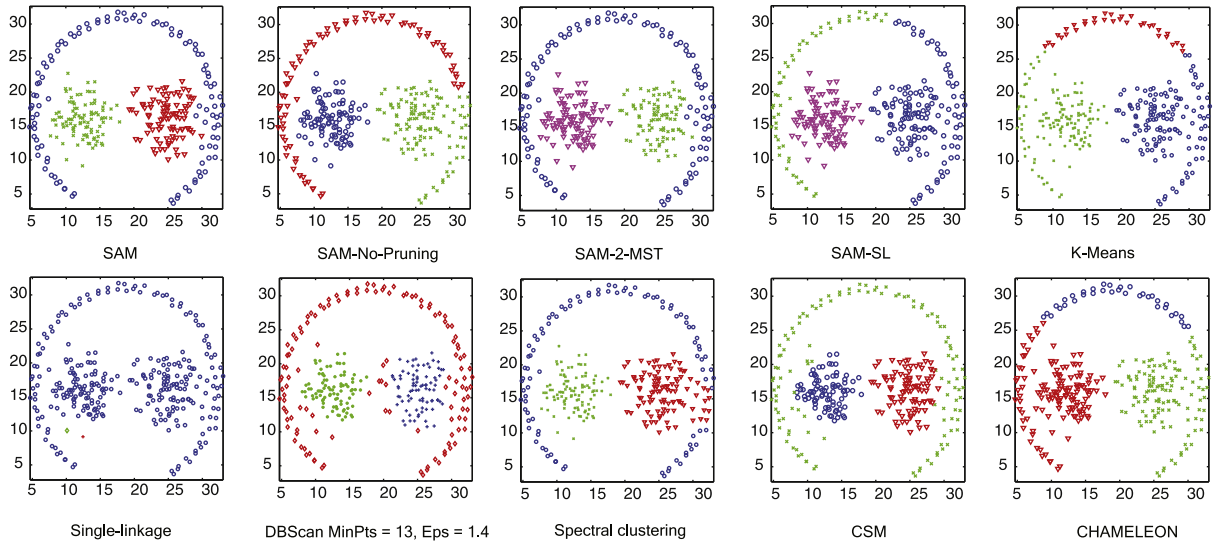


Fig. 7. Illustration of clustering results on DS2.

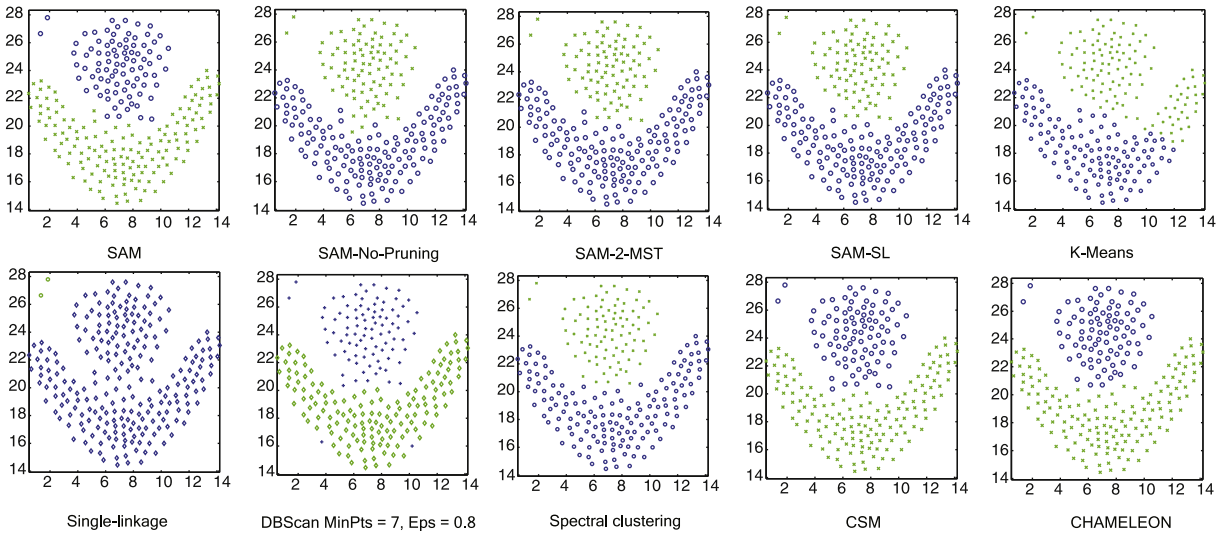


Fig. 8. Illustration of clustering results on DS3.

The corresponding Adjusted Rand index values of the clustering results on these six synthetic datasets are shown in Table 2.

3.3. Results on real datasets

The performance on each of the four UCI datasets is evaluated using the four common external clustering validity indices: Rand, Adjusted-Rand, Jaccard coefficient, and Fowlkes and Mallows (FM). The evaluation results on the four datasets are shown in Tables 3–6, respectively. The parameters of DBScan for IRIS, WINE, WBC, and WDBC are set to (MinPts = 8, Eps = 0.4), (MinPts = 3, Eps = 0.3), (MinPts = 9, Eps = 2), (MinPts = 4, Eps = 32.3), respectively.

For the IRIS dataset, Table 3 indicates that SAM, SAM-2-MST, and SAM-SL have the same performance and outperform the others. SAM-No-Pruning, CSM, DBScan, and *K*-means also provide partitions with relative high quality. In the case of the WINE dataset, the corresponding clustering qualities are shown in Table 4. *K*-means has the best performance, single linkage and spectral clustering provide more proper partitions than the proposed method SAM. It can be seen from Table 5 that SAM outperforms the other algorithms except *K*-means on the WBC dataset. As for the WDBC dataset in Table 6, DBScan has the

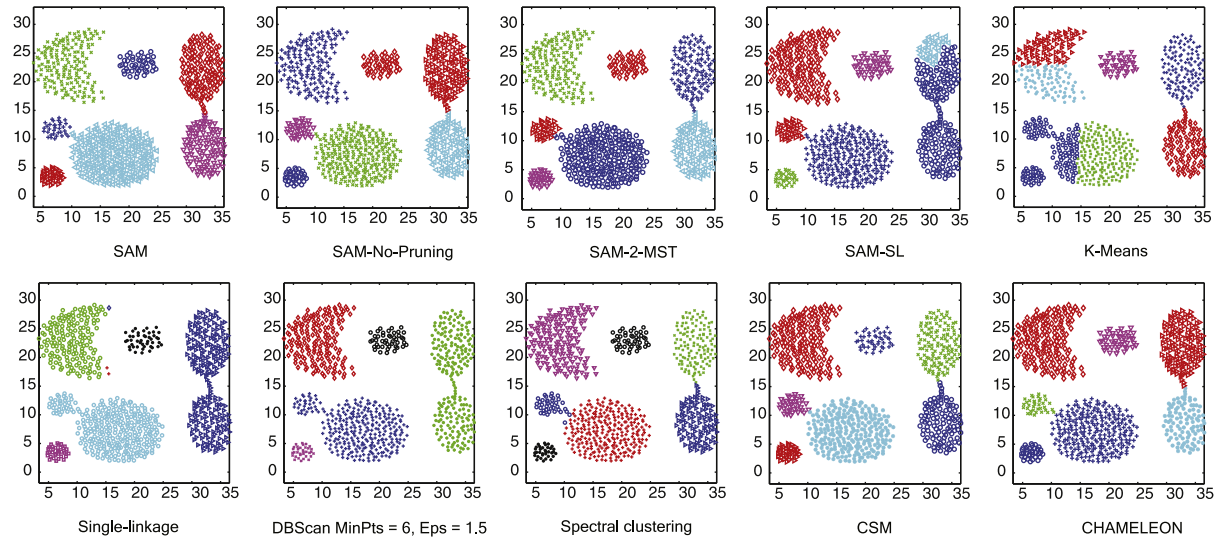


Fig. 9. Illustration of clustering results on DS4.

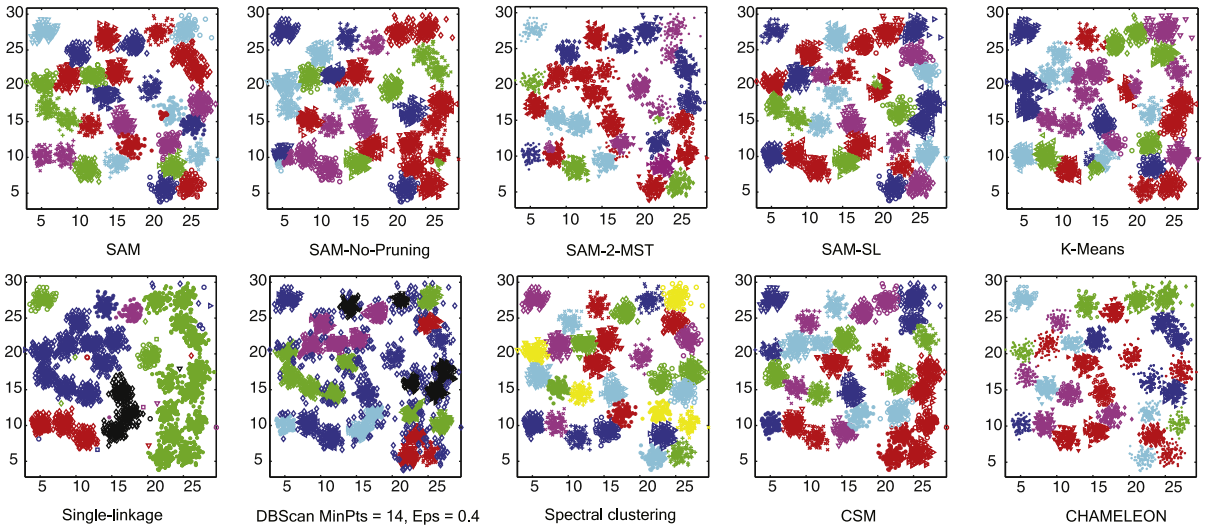


Fig. 10. Illustration of clustering results on DS5.

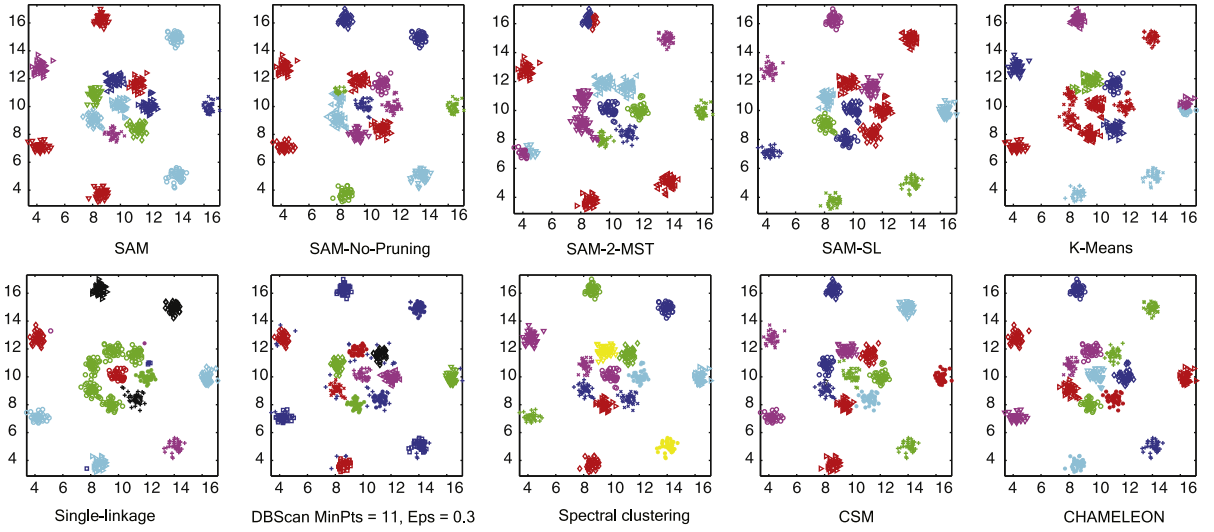


Fig. 11. Illustration of clustering results on DS6.

Table 2
Adjusted Rand index values of clustering performances on the six synthetic datasets.

Method	DS1	DS2	DS3	DS4	DS5	DS6
SAM	1.0000	0.9597	0.9339	0.9835	0.8896	0.9928
SAM-No-Pruning	1.0000	0.7272	0.9180	0.9920	0.8051	0.8723
SAM-2-MST	0.3181	0.9597	0.9178	0.9902	0.8183	0.8375
SAM-SL	0.2563	0.6130	0.9178	0.8743	0.8755	0.9928
K-means	0.5146	0.4739	0.4312	0.7186	0.8218	0.8055
Single linkage	0.2563	0.0004	0.0103	0.7996	0.1739	0.5425
DBScan	1.0000	0.8213	0.8859	0.8043	0.4321	0.8804
Spectral clustering	0.3178	0.8757	0.9178	0.9919	0.9522	0.9928
CSM	1.0000	0.9118	0.9667	0.9978	0.9196	0.9857
CHAMELEON	1.0000	0.4756	0.9617	1.0000	0.9274	0.9928

Table 3
Clustering performances on IRIS.

Method	Rand	Adjusted Rand	Jaccard	FM
SAM	0.9495	0.8858	0.8578	0.9234
SAM-No-Pruning	0.9075	0.7436	0.7298	0.8548
SAM-2-MST	0.9495	0.8858	0.8578	0.9234
SAM-SL	0.9495	0.8858	0.8578	0.9234
K-means	0.8797	0.7302	0.6959	0.8208
Single linkage	0.7766	0.5638	0.5891	0.7635
DBScan	0.8834	0.7388	0.7044	0.8268
Spectral clustering	0.8115	0.5745	0.5571	0.7156
CSM	0.8859	0.7455	0.7119	0.8321
CHAMELEON	0.7783	0.5492	0.5680	0.7369

Table 4
Clustering performances on WINE.

Method	Rand	Adjusted Rand	Jaccard	FM
SAM	0.7334	0.4007	0.4294	0.6009
SAM-No-Pruning	0.7012	0.3842	0.4113	0.5819
SAM-2-MST	0.7334	0.4007	0.4294	0.6009
SAM-SL	0.6976	0.3956	0.4702	0.6521
K-means	0.8797	0.7302	0.6959	0.8208
Single linkage	0.7766	0.5638	0.5891	0.7635
DBScan	0.6878	0.3171	0.3866	0.5582
Spectral clustering	0.7655	0.4741	0.4821	0.6506
CSM	0.6742	0.3757	0.4708	0.6618
CHAMELEON	0.7364	0.4769	0.5266	0.7049

Table 5
Clustering performances on WBC.

Method	Rand	Adjusted Rand	Jaccard	FM
SAM	0.9026	0.8033	0.8372	0.9114
SAM-No-Pruning	0.8876	0.7682	0.8061	0.8953
SAM-2-MST	0.8922	0.7820	0.8222	0.9025
SAM-SL	0.5565	0.0337	0.5453	0.7346
K-means	0.9240	0.8465	0.8703	0.9307
Single linkage	0.5453	0.0025	0.5444	0.7375
DBScan	0.8767	0.7529	0.7913	0.8838
Spectral clustering	0.5218	0.0246	0.4142	0.5867
CSM	0.5658	0.0585	0.5468	0.7333
CHAMELEON	0.5235	0.0279	0.5107	0.7007

best performance, while the proposed method SAM is better than *K*-means, single linkage, spectral clustering, CSM as well as its variants.

3.4. Discussion about the SAM variants

Pruning gives a small but consistent improvement (SAM vs. SAM-No-Pruning) based on the numeric results (Tables 2–5). In the case of DS2, pruning is critical to obtain the correct clustering whereas for the other test sets its effect is more like fine-tuning.

For the merge criterion, the index $R(C_i, C_j)$ is more complicated to evaluate than some simple alternatives like the single linkage criterion. Its effect on the clustering quality, however, is significant and in most cases the proposed approach (SAM) outperforms the single linkage variant (SAM-SL) using the single linkage algorithm in the merge stage. We consider the criterion-based merge critical for the performance of the algorithm.

The role of the *k*-MST is important. The value $k = 3$ was fixed already in preliminary tests using DS1-DS6, but let us discuss other choices. In SAM, 1-MST cannot be used in the merge stage as the criterion requires more information about the neighborhood than a simple spanning tree can provide, as the results of SAM-SL already showed. The question about the exact value of *k*, however, is less critical. In most cases, SAM and SAM-2-MST provide the same result, and only in two cases (DS1 and DS2) the 2-MST variant fails.

Table 6
Clustering performances on WDBC.

Method	Rand	Adjusted Rand	Jaccard	FM
SAM	0.8138	0.6269	0.6981	0.8223
SAM-No-Pruning	0.8003	0.6091	0.6772	0.8101
SAM-2-MST	0.8012	0.6190	0.6811	0.8197
SAM-SL	0.5308	0.0032	0.5219	0.7162
K-means	0.7504	0.4914	0.6499	0.7915
Single linkage	0.5326	0.0024	0.5315	0.7286
DBScan	0.8691	0.7367	0.7828	0.8782
Spectral clustering	0.7479	0.4945	0.6133	0.7604
CSM	0.5860	0.1335	0.5392	0.7201
CHAMELEON	0.8365	0.6703	0.7406	0.8514

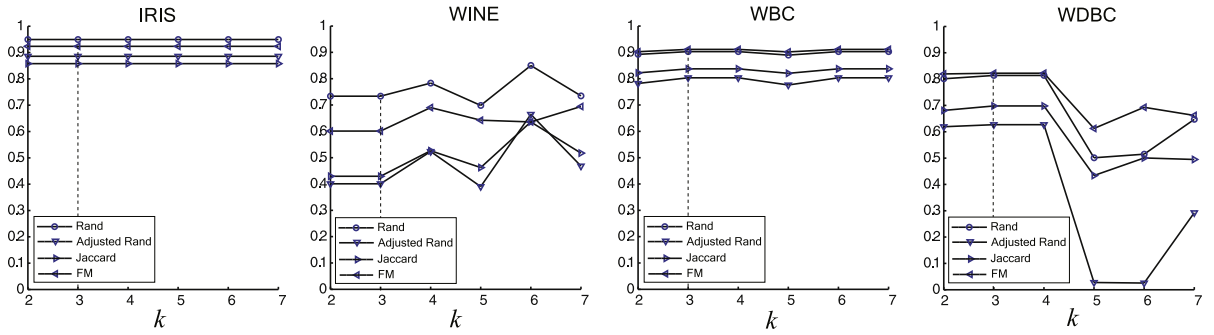


Fig. 12. The quality of clustering results for different values of k to compute k-MSTs.

Higher values than $k = 3$ were also tested, see Fig. 12. In most cases, the improvements due to higher values for k are insignificant and just increase the processing time. A contradicting example is the WDBC test set where higher values turn out to be harmful. Therefore, it is reasonable that the value of k is set to 3.

4. Conclusion

The proposed method employs minimum spanning trees in different stages. Before a dataset is split into different clusters, the hairs (leaves together with the connecting edges) of the first MST computed for the whole instance are pruned.

In the splitting stage, more than the desired K clusters are created by K-means. Three MSTs on an iteratively refined graph are computed and combined to determine the initial prototypes for K-means, because randomly selected initial prototypes would lead to unstable partitions.

After splitting, the initial MST is employed to adjust the partitions to make each subgroup corresponding to a subtree. Finally, in the merge step only neighboring pairs with respect to the same MST are considered to be merged.

Experiments have demonstrated the importance of each step of the algorithm. Except the number of clusters, there are no parameters left to be tuned by the user.

In summary, various MSTs are utilized during the whole split-and-merge algorithm because they can capture the intrinsic structure of a dataset. However, the computational complexity of constructing an MST is close to $O(N^2)$. The expensive computational cost obstructs the application of an MST to large scale data sets. One of our future work is to find a fast algorithm to construct an approximate MST.

One drawback of the proposed method is that the universality of the definitions of inter-connectivity and intra-similarity is insufficient. Although there does not exist a universal clustering method that can deal with all kinds of clustering problems, we try to improve the definition of inter-connectivity and intra-similarity to make the proposed method suitable for as many clustering problems as possible.

Although the proposed method does not have any direct restrictions for being applied to datasets with high dimensions, it is assumed to have all the same weaknesses as the other distance-based methods. Because to determine the intrinsic structure of this kind of datasets, different dimensions may have varied importance, whereas the proposed method equally considers all of dimensions of an input dataset. Thus, subspace clusterings [7,8] or other methods tailored for high dimensional data are expected to work better for high-dimensional datasets.

Acknowledgments

The authors thank the anonymous reviewers for their constructive comments and suggestions which helped improve the quality of this paper. The work of C. Zhong was supported by Zhejiang Provincial Natural Science Foundation of China, No.

Y1090851, and the Center for International Mobility (CIMO). The work of D. Miao was supported by the National Natural Science Foundation of China, No. 60970061, No. 61075056, and the Research Fund for the Doctoral Program of Higher Education, No. 20060247039.

References

[1] R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan, Automatic subspace clustering of high dimensional data for datamining applications, in: Proceedings of ACM-SIGMOD Conference on the Management of Data, 1998, pp. 94–105.
[2] M. Bereta, T. Burczynski, Immune *k*-means and negative selection algorithms for data analysis, Inform. Sci. 179 (2009) 1407–1425.
[3] J.C. Bezdek, N.R. Pal, Some new indexes of cluster validity, IEEE Trans. Syst. Man Cybern. B. Cybern. 28 (1998) 301–315.
[4] H. Chang, D.Y. Yeung, Robust path-based spectral clustering, Pattern Recognit. 41 (2008) 191–203.
[5] D. Cheng, R. Kannan, S. Vempala, G. Wang, A divide-and-merge methodology for clustering, ACM Trans. Database Syst. 31 (2006) 1499–1525.
[6] M.C. Chiang, C.W. Tsai, C.S. Yang, A time-efficient pattern reduction algorithm for *k*-means clustering, Inform. Sci. 181 (2011) 716–731.
[7] Y. Chu, J. Huang, K. Chuang, D. Yang, M. Chen, Density conscious subspace clustering for high-dimensional data, IEEE Trans. Knowl. Data Eng. 22 (2010) 16–30.
[8] Y. Chu, Y. Chen, D. Yang, M. Chen, Reducing redundancy in subspace clustering, IEEE Trans. Knowl. Data Eng. 21 (2009) 1432–1446.
[9] T.H. Corman, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, second ed., MIT Press, Cambridge, MA, 2001.
[10] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial data sets with noise, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Portland, OR, 1996, pp. 226–231.
[11] M. Figueiredo, A.K. Jain, Unsupervised learning of finite mixture models, IEEE Trans. Pattern Anal. Mach. Intell. 24 (2002) 381–396.
[12] P. Fränti, O. Virmajoki, V. Hautamäki, Fast agglomerative clustering using a k-nearest neighbor graph, IEEE Trans. Pattern Anal. Mach. Intell. 28 (2006) 1875–1881.
[13] L. Fu, E. Medico, FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data, BMC Bioinformatics 8 (3) (2007).
[14] A. Gionis, H. Annila, P. Tsaparas, Clustering aggregation, ACM Trans. Knowl. Disc. Data 1 (2007) 1–30.
[15] S. Guha, R. Rastogi, K. Shim, CURE: an efficient clustering algorithm for large databases, in: Proceedings of the 1998 ACM-SIGMOD International Conference Management of Data (SIGMOD'98), 1998, pp. 73–84.
[16] S. Guha, R. Rastogi, K. Shim, ROCK: a robust clustering algorithm for categorical attributes, in: Proceedings of the IEEE Conference on Data Engineering, 1999, pp. 512–521.
[17] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Data Mining, Inference and Prediction, Springer, New York, 2001.
[18] A. Hinneburg, D.A. Keim, An efficient approach to clustering in large multimedia databases with noise, in: Knowledge Discovery and Data Mining, 1998, pp. 58–65.
[19] C.-C. Hsu, C.-L. Chen, Y.-W. Su, Hierarchical clustering of mixed data based on distance hierarchy, Inform. Sci. 177 (2007) 4474–4492.
[20] A.K. Jain, M. Murthy, P. Flynn, Data clustering: a review, ACM Comput. Surv. 31 (1999) 264–323.
[21] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[22] A.K. Jain, M.C. Law, Data clustering: a user's dilemma, in: Pattern Recognition and Machine Intelligence, LNCS, vol. 3776, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 1–10.
[23] G. Karypis, E.H. Han, V. Kumar, CHAMELEON: a hierarchical clustering algorithm using dynamic modeling, IEEE Trans. Comput. 32 (1999) 68–75.
[24] L. Kaufman, P.J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, Wiley, New York, 1990.
[25] T. Kaukoranta, P. Fränti, O. Nevalainen, Iterative split-and-merge algorithm for VQ codebook generation, Opt. Eng. 37 (1998) 2726–2732.
[26] B. King, Step-wise clustering procedures, J. Am. Stat. Assoc. 69 (1967) 86–101.
[27] J.Z.C. Lai, T.J. Huang, An agglomerative clustering algorithm using a dynamic *k*-nearest-neighbor list, Inform. Sci. 181 (2011) 1722–1734.
[28] J.S. Lee, S. Olafsson, Data clustering by minimizing disconnectivity, Inform. Sci. 181 (2011) 732–746.
[29] C.H. Lee, O.R. Zaïane, H.-H. Park, J. Huang, R. Greiner, Clustering high dimensional data: a graph-based relaxed optimization approach, Inform. Sci. 178 (2008) 4501–4511.
[30] C.R. Lin, M.S. Chen, Combining partitional and hierarchical algorithms for robust and efficient data clustering with cohesion self-merging, IEEE Trans. Knowl. Data Eng. 17 (2005) 145–159.
[31] M. Liu, X. Jiang, A.C. Kot, A multi-prototype clustering algorithm, Pattern Recognit. 42 (2009) 689–698.
[32] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: Fifth Berkeley Symposium on Mathematics, Statistics and Probability, 1967, pp. 281–297.
[33] G. McLachlan, K. Basford, Mixture Models: Inference and Application to Clustering, Marcel Dekker, New York, 1988.
[34] G.W. Milligan, S.C. Soon, L.M. Sokol, The effect of cluster size, dimensionality, and the number of clusters on recovery of true cluster structure, IEEE Trans. Pattern Anal. Mach. Intell. 5 (1983) 40–47.
[35] R.T. Ng, J. Han, CLARANS: a method for clustering objects for spatial data mining, IEEE Trans. Knowl. Data Eng. 14 (2002) 1003–1016.
[36] M.R. Rezaee, B.P.F. Lelieveldt, J.H.C. Reiber, A new cluster validity index for the fuzzy *c*-mean, Pattern Recognit. Lett. 19 (1998) 237–246.
[37] S.E. Schaeffer, Graph clustering, Comput. Sci. Rev. 1 (2007) 27–64.
[38] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (2000) 888–905.
[39] P.H.A. Sneath, R.R. Sokal, Numerical Taxonomy, Freeman, San Francisco, London, 1973.
[40] G.T. Toussaint, The relative neighborhood graph of a finite planar set, Pattern Recognit. 12 (1980) 261–268.
[41] S. Theodorodis, K. Kouttoumbas, Pattern Recognition, fourth ed., Academic Press, Amsterdam, 2009.
[42] C.J. Veenman, M.J.T. Reinders, E. Backer, A maximum variance cluster algorithm, IEEE Trans. Pattern Anal. Mach. Intell. 24 (2002) 1273–1280.
[43] W. Wang, J. Yang, M. Muntz, STING: a statistical information grid approach to spatial datamining, in: Proceedings of the International Conference on Very Large Data Bases, 1997, pp. 186–195.
[44] M.J. Warrens, On the equivalence of Cohen's kappa and the Hubert–Arabie Adjusted Rand index, J. Classif. 25 (2008) 177–183.
[45] Z. Wu, R. Leahy, An optimal graph theoretic approach to data clustering: theory and its application to image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 15 (1993) 1101–1113.
[46] R. Xu, D. Wunsch II, Survey of clustering algorithms, IEEE Trans. Neural Netw. 16 (2005) 645–678.
[47] Y. Xu, V. Olman, D. Xu, Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning tree, Bioinformatics 18 (2002) 536–545.
[48] K.Y. Yeung, C. Fraley, A. Murua, A.E. Raftery, Model-based clustering and data transformations for gene expression data, Bioinformatics 17 (2001) 977–987.
[49] C.T. Zahn, Graph-theoretical methods for detecting and describing gestalt clusters, IEEE Trans. Comput. C-20 (1971) 68–86.
[50] <http://www.ics.uci/mlearn/MLRespository.html>.

5

Paper **P5**

C. Zhong, M. Malinen, D. Miao. P. Fränti. “A fast minimum spanning tree algorithm based on K-means, ” Manuscript (Submitted).

A Fast Minimum Spanning Tree Algorithm Based on K-means

Caiming Zhong[a,b,c], Mikko Malinen[b], Duoqian Miao[a,*], Pasi Fränti[b]

[a] *Department of Computer Science and Technology, Tongji University, Shanghai 201804, PR China*
[b] *Department of Computer Science, University of Eastern Finland, P.O. Box 111, FIN-80101 Joensuu, Finland*
[c] *College of Science and Technology, Ningbo University, Ningbo 315211, PR China*

Abstract

Minimum spanning trees (MSTs) have long been used in data mining, pattern recognition and machine learning. However, it is difficult to apply traditional MST algorithms to a large dataset since the time complexity of the algorithms is quadratic. In this paper, we present a fast MST algorithm on the complete graph of N points. The proposed algorithm employs a divide-and-conquer scheme to produce an approximate MST with theoretical time complexity of $O(N^{1.5})$, which is faster than the conventional MST algorithms with $O(N^2)$. It consists of two stages. In the first stage, called the divide-and-conquer stage, K-means is employed to partition a dataset into $\sqrt{N}$ clusters. Then an exact MST algorithm is applied to each cluster and the produced $\sqrt{N}$ MSTs are connected in terms of a proposed criterion to form an approximate MST. In the second stage, called the refinement stage, the clusters produced in the first stage form $\sqrt{N}-1$ neighboring pairs, and the dataset is repartitioned into $\sqrt{N}-1$ clusters with the purpose of partitioning the neighboring boundaries of a neighboring pair into a cluster. With the $\sqrt{N}-1$ clusters, another approximate MST is constructed. Finally, the two approximate MSTs are combined into a graph and a more accurate MST is generated from it. The proposed algorithm can be regarded as a framework, since any exact MST algorithm can be incorporated into the framework to reduce its running time. Experimental results show that the proposed approximate MST algorithm is computational efficient, and the approximation is sufficiently close to the true MST such that the performance in practical applications hardly suffers.

Keywords: Minimum spanning tree, divide-and-conquer, K-means.

1. Introduction

Given an undirected and weighted graph, the problem of MST is to find a spanning tree such that the sum of weights is minimized. Since an MST can roughly estimate the intrinsic structure of a dataset, it has been broadly applied in image segmentation [1], [47], cluster analysis [46], [51], [52], [53], classification [27], manifold learning [48], [49], density estimation [30], diversity estimation [33], and some applications of the variant problems of MST [10], [43], [36]. Since the pioneering algorithm of computing an MST was proposed by Otakar Borůvka in 1926 [6], the studies of the problem have been focused on finding the optimal exact MST algorithm, fast and approximate MST algorithms, distributed MST algorithms and parallel MST algorithms.

The beginning of the studies on constructing an exact MST is Borůvka's algorithm [6]. This algorithm begins with each vertex of a graph being a tree. Then for each tree it iteratively selects the shortest edge connecting the tree to the rest, and combines the edge into the forest formed by all the trees, until the forest is connected. The computational complexity of this algorithm is $O(E \log V)$, where E is the number of edges, and V is the number of vertices in the graph. Similar algorithms have been invented by Choquet [13], Florek et al. [19] and Sollin [42], respectively.

One of the most typical examples is Prim's algorithm, which was proposed independently by Jarník [26], Prim [39] and Dijkstra [15]. It first arbitrarily selects a vertex as a tree, then repeatedly adds the shortest edge that connects a new vertex to the tree, until all the vertices are included. The time complexity of Prim's algorithm is $O(E \log V)$. If Fibonacci heap is employed to implement a min-priority queue for searching for the shortest edge, the computational time is reduced to $O(E + V \log V)$ [14].

Kruskal's algorithm is another widely used exact MST algorithm [32]. In this algorithm, all the edges are sorted by their weights in non-decreasing order. It starts with each vertex being a tree, and iteratively combines the trees by adding edges in the sorted order excluding those leading to a cycle, until all the trees are combined into one tree. The running time of Kruskal's algorithm is $O(E \log V)$.

Several fast MST algorithms have been proposed. For a sparse graph, Yao [50], and Cheriton and Tarjan [11] independently proposed algorithms with $O(E \log\log V)$ time. Fredman and Tarjan [20] proposed Fibonacci heap as a data structure of implementing the priority queue for constructing an exact MST. With the heaps, the computational complexity is reduced to $O(E\beta(E, V))$, where $\beta(E, V) = \min\{i | \log^{(i)} V \leq E/V\}$. Gabow et al. [21] incorporated the idea of *Packets* [22] into Fibonacci heap, and reduced the complexity to $O(E \log\beta(E, V))$.

Recent progress on exact MST algorithm was made by Chazelle [9]. He discovered a new heap structure called soft heap to implement the priority queue, and reduced the time com-

*Corresponding author. Tel.:+86-21-69589867
Email address: miaoduoqian@163.com (Duoqian Miao)

plexity to $O(E\alpha(E,V))$, where α is the inverse of the Ackermann function. March et al. [35] proposed a dual-tree Boruvka on a kd-tree and a dual-tree Boruvka on a cover-tree for constructing MST, it is claimed that the time complexity is $O(N \log N\alpha(N)) \approx O(N \log N)$.

Although a lot of work has been done on the exact MST problem, distributed MST and parallel MST algorithms have also been studied in the literature. The first algorithm of distributed MST problem was presented by Gallager et al. [23]. The algorithm supposes that a processor exits at each vertex and knows initially only the weights of the adjacent edges. It runs in $O(V \log V)$ time. Several faster $O(V)$ time distributed MST algorithms have been proposed by Awerbuch [3] and Abdel-Wahab et al. [2], respectively. Peleg and Rubinovich [37] presented a lower bound of time complexity $O(D + \sqrt{V}/\log V)$ for constructing distributed MST on a network, where $D = \Omega(\log V)$ is the diameter of the network. Moreover, Khan et al. [29] proposed a distributed approximate algorithm of MST on networks and its complexity is $\tilde{O}(D+L)$, where L is the local shortest path diameter.

Chong et al. [12] presented a parallel algorithm to construct an MST in $O(\log V)$ time by employing a linear number of processors. Pettie and Ramachandran [38] proposed a randomized parallel algorithm to compute a minimum spanning forest, which also runs in logarithmic time. Bader and Cong [4] presented four parallel algorithms, of which three algorithms are variants of Borůvka. For different graphs, their algorithms can find MSTs 4 to 6 times faster using 8 processors than the sequential algorithms.

Several approximate MST algorithms have been proposed. The algorithms in [44], [7] are composed of two steps. In the first step, a sparse graph is extracted from the complete graph, and then in the second step, an exact MST algorithm is applied on the extracted graph. In these algorithms, different methods for extracting sparse graphs have been employed. For example, Vaidya [44] used a group of grids to partition a dataset into cubical boxes of identical size. For each box, a representative point was determined. Any two representatives of two cubical boxes were connected if the corresponding edge length was between two given thresholds. Within a cubical box, points were connected to the representative. Callahan and Kosaraju [7] applied well-separated pair decomposition of the dataset to extract a sparse graph.

Recent work for finding an approximate MST and applying it to clustering can be found in [45, 34]. Wang et al. [45] employ divide-and-conquer scheme to construct an approximate MST. However, their goal was not to find the MST but merely to detect the long edges of the MST at an early stage for clustering. Initially, data points are randomly stored in a list, and each data point is connected to its predecessor (or successor), and a spanning tree is achieved. At the same time, the weight of each edge from a data point to its predecessor (or successor) are assigned, and a spanning tree is formed by the randomly stored sequence of data points. To optimize the spanning tree, the dataset is divided into a collection of subsets with a divisive hierarchical clustering algorithm (DHCA). The distance between any pair of data points within a subset can be found by a brute force nearest neighbor search, and with the distances, the spanning tree is updated. The algorithm is performed repeatedly and the spanning tree is optimized further after each run.

Lai et al. [34] proposed an approximate MST algorithm based on Hilbert curve for clustering. It is a two-phase algorithm: the first phase is to construct an approximate MST of a given dataset with Hilbert curve, and the second phase is to partition the dataset into subsets by measuring the densities of points along the approximate MST with a specified density threshold. The process of constructing an approximate MST is iterative followed by a stepwise refinement. The number of iterations is $(d + 1)$, where d is the number of dimensions of the dataset. In each iteration, an approximate MST is generated similarly as in Prim's algorithm. The main difference is that Lai's method maintains a min-priority queue by considering the approximate MST produced in last iteration and the neighbors of visited points determined by a Hilbert sorted linear list, while Prim's algorithm by considering all the neighbors of a visited points. However, the accuracy of Lai's method depends on the order of Hilbert Curve and the number of neighbors of a visited point in the linear list.

In general, the computational cost of building an MST is the bottleneck of the efficiency for a practical problem that involves an MST of a large dataset, since so far the computational complexity of the best exact MST algorithm has been $O(E\alpha(E,V))$. This complexity roughly equals $O(N^2)$ for a complete graph of a dataset, in which $V = N$ and $E = N \times (N-1)/2$. In this paper, we present a fast approximat MST algorithm (FMST). It consists of two stages: divide-and-conquer and refinement. In the divide-and-conquer stage, the dataset is partitioned by K-means into $\sqrt{N}$ clusters, and the exact MSTs of all the clusters are constructed and merged. In the refinement stage, boundaries of the clusters are considered. It runs in $O(N^{1.5})$ time when Prim's or Kruskal's algorithm is used in its divide-and-conquer stage, and does not reduce the quality compared to an exact MST for a practical use.

The rest of this paper is organized as follows. In Section 2, the new fast divide-and-conquer MST algorithm is presented. Time complexity of the proposed method is analyzed in Section 3, and experiments on the efficiency and accuracy of the proposed algorithm are given in Section 4. Finally, we conclude this work in Section 5.

2. Proposed method

2.1. Overview of the proposed method

To improve the efficiency of solutions for problems such as constructing MST or K nearest neighbors (KNN), an intuitive way is to reduce the unnecessary comparisons. For example, to find the K nearest neighbor of a point in a dataset, it is not necessary to search the entire dataset but a small local portion. In the same way, to find the MST with Kruskal's algorithm in a complete graph, it is not necessary to sort all $N(N-1)/2$ edges in the graph but to find $(1+\alpha)N$ edges with least weights, where $(N-3)/2 \gg \alpha \geq -1/N$. With this observation in mind, we employ a divide-and-conquer technique to achieve improvement.

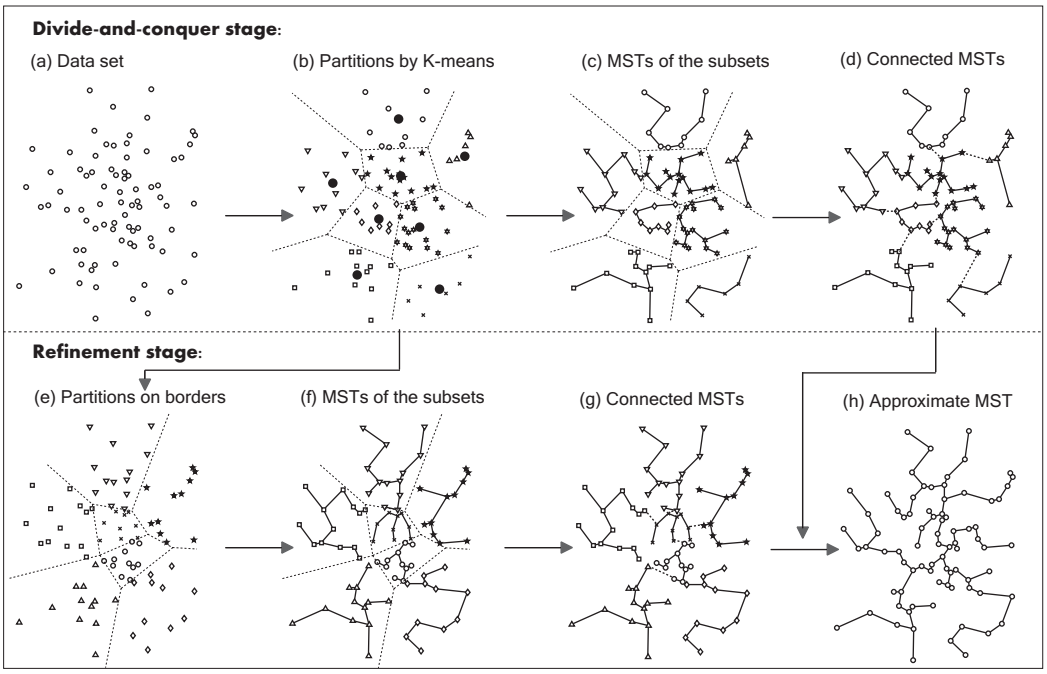


Figure 1: The scheme of the proposed fast MST algorithm. (a) A given dataset. (b) The dataset is partitioned into $\sqrt{N}$ subsets by K-means. The dashed lines form the corresponding Voronoi graph with respect to cluster centers (the big grey circles). (c) An exact MST algorithm is applied to each subset. (d) MSTs of the subsets are connected. (e) The dataset is partitioned again so that the neighboring data points in different subsets of (b) are partitioned into identical partitions. (f) Exact MST algorithm is used again on the secondary partition. (g) MSTs of the subsets are connected. (h) A more accurate approximate MST is produced by merging the two approximate MSTs in (d) and (g) respectively.

In general, a divide-and-conquer paradigm consists of three steps according to [14]:

1. Divide step. The problem is divided into a collection of subproblems that are similar to the original problem but smaller in size.
2. Conquer step. The subproblems are solved separately, and corresponding subresults are achieved.
3. Combine step. The subresults are combined to form the final result of the problem.

Following this divide-and-conquer paradigm, we construct a two-stage fast approximate MST method as follows:

1. Divide-and-conquer stage
   1.1. Divide step. For a given dataset of N data points, K-means is applied to partition the dataset into $\sqrt{N}$ subsets.
   1.2. Conquer step. An exact MST algorithm such as Kruskal's or Prim's algorithm is employed to construct an exact MST for each subset.
   1.3. Combine step. $\sqrt{N}$ MSTs are combined using a connection criterion to form a primary approximate MST.
2. Refinement stage
   2.1. Partitions focused on borders of the clusters produced in the previous stage are constructed.
   2.2. Secondary approximate MST is constructed with the conquer and combine steps in the previous stage.
   2.3. The two approximate MSTs are merged and a new more accurate is obtained by using an exact MST algorithm.

The process is illustrated in Fig. 1. In the first stage, an approximate MST is produced. However, its accuracy is insufficient compared to the corresponding exact MST, because may of the data points that are located in the boundaries of the subsets are connected incorrectly in the MST. This is because an exact MST algorithm is applied only to data points within a subset but not to those crossing the boundaries of the subsets. To compensate the drawback, a refinement stage is designed.

In the refinement stage, we re-partition the dataset so that the neighboring data points from different subsets will belong to the same partition. After this, the two approximate MSTs are merged, and the number of edges in the combined graph is at most $2(N-1)$. The final MST is built from this graph by an exact MST algorithm. The details of the method will be described in the following subsections.

2.2. Partition dataset with K-means

In general, a data point in an MST is connected to its nearest neighbors, which implies that the connections have a locality property. In the divide step, it is therefore expected that the subsets preserve this locality. Since K-means can partition local neighboring data points into the same group, we employ K-means to partition the dataset.

K-means requires to know the number of clusters and to determine the initial center points, we will discuss these two problems as follows.

2.2.1. The number of clusters K

In our method, the number of clusters K is set to $\sqrt{N}$. There are two reasons for this determination. One is that the maxi-

mum number of clusters in some clustering algorithms is often set to $\sqrt{N}$ as a rule of thumb [5], [41]. That means if a dataset is partitioned into $\sqrt{N}$ subsets, each subset will consist of data points coming from an identical genuine cluster, which satisfies the requirement of the locality property when constructing an MST.

The other reason is that the overall time complexity of the proposed approximate MST algorithm is minimized if K is set to $\sqrt{N}$, assuming that the data points are equally divided into the clusters. This choice will be theoretically and experimentally studied in more detail in Section 3 and 4, respectively.

2.2.2. Initialization of K-means

Clustering results of K-means are sensitive to the initial cluster centers. A bad selection of the initial cluster centers may have negative effects on the time complexity and accuracy of the proposed method. However, we still randomly select the initial centers due to the following considerations.

First, although a random selection may lead to a skewed partition, such as linear partition, the time complexity of the proposed method is still $O(N^{1.5})$, see Theorem 2 in Section 4. Second, in the proposed method, a refinement stage is designed to cope with the data points in the cluster boundaries. This process makes the accuracy relatively stable, and random selection of initial cluster centers is reasonable.

2.2.3. Divide and conquer algorithm

After the dataset is divided into $\sqrt{N}$ subsets by K-means, the MSTs of the subsets are constructed with an exact MST algorithm, such as Prim's or Kruskal's algorithm. This corresponds to the conquer step in the divide and conquer scheme, it is trivial and illustrated in Fig. 1(c). The algorithm of K-means based divide and conquer is described as follows:

Divide and Conquer Using K-means (DAC)
Input: Dataset X;
Output: MSTs of the subsets partitioned from X

Step 1. Set the number of subsets $K = \sqrt{N}$.

Step 2. Apply K-means to X to achieve K subsets $S = \{S_1, \ldots, S_K\}$, where the initial centers are randomly selected.

Step 3. Apply an exact MST algorithm to each subset in S, and an MST of S_i, denoted by $MST(S_i)$, is obtained, where $1 \le i \le K$.

The next step is to combine the MSTs of the K subsets into a whole MST.

2.3. Combine MSTs of the K subsets

An intuitive solution to combine MSTs is brute force: for the MST of a cluster, the shortest edge between it and MSTs of other clusters is computed. But this solution is time consuming, and therefore a fast MST-based effective solution is also presented. The two solutions are discussed below.

2.3.1. Brute force solution

Suppose we combine a subset S_l with another subset, where $1 \le l \le K$. Let x_i, x_j be data points and $x_i \in S_l$, $x_j \in X - S_l$. The edge that connects S_l to another subset can be found by brute force:

$$e = \arg\min_{e_i \in E_l} \rho(e_i) \tag{1}$$

where $E_l = \{e(x_i, x_j) | x_i \in S_l \wedge x_j \in X - S_l\}$, $e(x_i, x_j)$ is the edge between vertices x_i and x_j, $\rho(e_i)$ is the weight of edge e_i. The whole MST is obtained by iteratively adding e into MSTs and finding the new connecting edge between the merged subset and the remaining part. This process is similar to the single-link clustering [21].

However, the computational cost of the brute force method is high. Suppose that each subset has an equal size of N/K, and K is an even number. The running time T_c of combining the K trees into the whole MST is:

$$\begin{aligned} T_c =& 2 \times \{\frac{N}{K} \times \frac{(K-1) \times N}{K} + \frac{2 \times N}{K} \times \frac{(K-2) \times N}{K} \\ &+ \ldots + \frac{(K/2) \times N}{K} \times \frac{(K/2) \times N}{K}\} \\ =& (\frac{K^2}{6} + \frac{K}{4} - \frac{1}{6}) \times \frac{N^2}{K} \\ =& O(KN^2) \\ =& O(N^{2.5}) \end{aligned} \tag{2}$$

Consequently, a more efficient combining method is needed.

2.3.2. MST-based solution

The efficiency of combining process can be improved in two aspects. Firstly, in each combining iteration only one pair of neighboring subsets is considered for finding the connecting edge. Intuitively, it is not necessary to take into account subsets that are far from each other, because no edge in an exact MST connects the subsets. This consideration will save some computations. Secondly, to determine the connecting edge of a pair of neighboring subsets, the data points in the two subsets will be scanned only once. The implementation of the two techniques is discussed in detail.

Determine the neighboring subsets. As the aforementioned brute force solution runs in the same way as single-link clustering [24] and all the information required by single-link can be provided by the corresponding MST of the same data, we make use of the MST to determine the neighboring subsets and improve the efficiency of the combination process.

If each subset has one representative, an MST of the representatives of the K subsets can roughly indicate which pairs of subsets could be connected to. For simplicity, the center of a subset is selected as its representative. After an MST of the centers (MST_{cen}) is constructed, each pair of subsets whose centers are connected by an edge of MST_{cen} is combined. Although not all of the neighboring subsets can be discovered by MST_{cen},

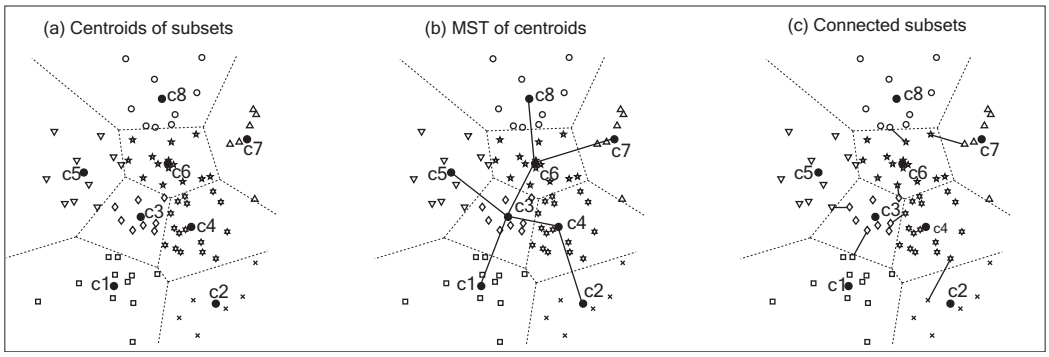


Figure 2: The combine step of MSTs of the proposed algorithm. In (a), centers of the partitions (c1, ..., c8) are calculated. In (b), a MST of the centers, MST_{cen}, is constructed with an exact MST algorithm. In (c), each pair of subsets whose centers are neighbors with respect to MST_{cen} in (b) is connected.

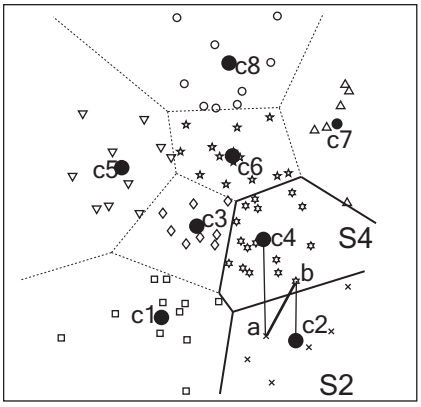


Figure 3: Detecting the connecting edge between S_4 and S_2.

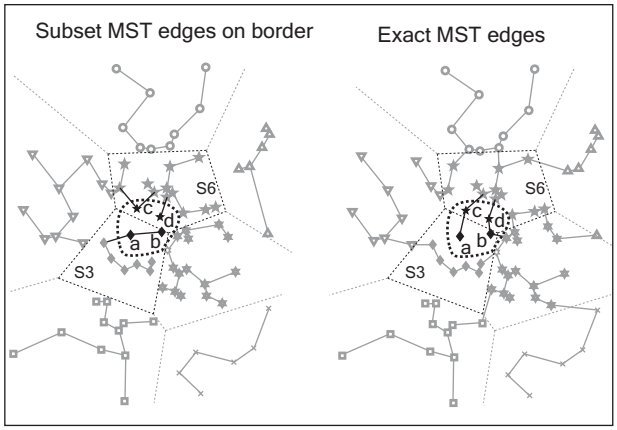


Figure 4: The data points on the subset boundaries are prone to be misconnected.

the dedicated refinement stage could remedy this drawback into some extent.

The centers of the subsets in Fig. 1(c) are illustrated as the solid points in Fig. 2(a), and MST_{cen} is composed of the dashed edges in Fig. 2(b).

Determine the connecting edges. For combining MSTs of a pair of neighboring subsets, an intuitive way is to find the shortest edge between the two subsets and connect the MSTs by this edge. Under the condition of average partition, finding the shortest edge between two subsets takes N steps, and therefore, the time complexity of the whole connection process is $O(N^{1.5})$. Although this does not increase the total time complexity of the proposed method, the absolute running time is still somewhat high.

To make the connecting process faster, a novel way to detect the connecting edges is illustrated in Fig. 3. Here, c_2 and c_4 are the centers of the subset S_2 and S_4, respectively. Suppose a is the nearest point to c_4 from S_2, and b is the nearest point to c_2 from S_4. The edge $e(a,b)$ is selected as the connecting edge between S_2 and S_4. The computational cost of this is low. Although the edges found are not always optimal, it can be compensated by the refinement stage.

Consequently, the algorithm of combining MSTs of subsets is summarized as follows:

Combine Algorithm (CA)

Input: MSTs of the subsets partitioned from X: $MST(S_1), \cdots, MST(S_K)$.
Output: Approximate MST of X, denoted by MST_1, and MST of the centers of $S_1, \cdots, S_K$, denoted by MST_{cen};

Step 1. Compute the center c_i of subset S_i, $1 \le i \le K$.

Step 2. Construct an MST, MST_{cen}, of $c_1, \cdots, c_K$ by an exact MST algorithm.

Step 3. For each pair of subsets (S_i, S_j) that their centers c_i and c_j are connected by an edge $e \in MST_{cen}$, discover the edge by **DCE** that connects $MST(S_i)$ and $MST(S_j)$.

Step 4. Add all the connecting edges discovered in Step 3 to $MST(S_1), \cdots, MST(S_K)$, and MST_1 is achieved.

Detect the Connecting Edge (DCE)
Input: A pair of subsets to be connected, (S_i, S_j);
Output: The edge connecting $MST(S_i)$ and $MST(S_j)$;

Step 1. Find the data point $a \in S_i$ such that the distance between a and the center of S_j is minimized.

Step 2. Find the data point $b \in S_j$ such that the distance between b and the center of S_i is minimized.

Step 3. Select edge $e(a,b)$ as the connecting edge.

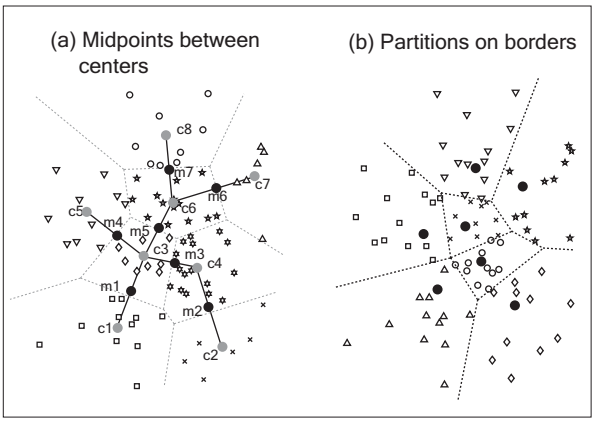


Figure 5: Boundary-based partition. In (a), the black solid points, $m_1, \cdots, m_7$, are the midpoints of the edges of MST_{cen}. In (b), each data point is assigned to its nearest midpoint, and the dataset is partitioned by the midpoints. The corresponding Voronoi graph is with respect to the midpoints.

2.4. Refine the MST focusing on boundaries

However, the accuracy of the approximate MST achieved so far is far from the exact MST. The reason is that, when the MST of a subset is built, the data points that lie in the boundary of the subset are considered only within the subset, but not across the boundaries. In Fig. 4, subsets S_6 and S_3 have a common boundary, and their MSTs are constructed independently. In the MST of S_3, point a and b are connected to each other. But in the exact MST they are connected to the points in S_6 rather than in S_3. Therefore, data points located on the boundaries are prone to be misconnected. Based on this observation, the refinement stage is designed.

2.4.1. Partition dataset focusing on boundaries

In this step, another complimentary partition is constructed so that the clusters would locate at the boundary areas of the previous K-means partition. We first calculate the midpoints of each edge of MST_{cen}. These midpoints generally lie near the boundaries, and are therefore employed as the initial cluster centers. The dataset is then partitioned by K-means. The partition process of this stage is different from that of the first stage. In this stage, the initial cluster centers are specified and the maximum number of iterations is set to 1 for the purpose of focusing on the boundaries. Since MST_{cen} has $\sqrt{N} - 1$ edges, there will be $\sqrt{N} - 1$ clusters in this stage. The process is illustrated in Fig. 5.

In Fig. 5(a), the midpoints of edges of MST_{cen} are computed as $m_1, \ldots, m_7$. In Fig. 5(b), the dataset is partitioned with respect to these 7 midpoints.

2.4.2. Build secondary approximate MST

After the dataset has been re-partitioned, the conquer and combine steps are similar to those used for producing the primary approximate MST. The algorithm is summarized as follows:

Secondary Approximate MST (SAM)
Input: MST of the subset centers MST_{cen}, dataset X;
Output: Approximate MST of X, MST_2;

Step 1. Compute the midpoint m_i of an edge $e_i \in MST_{cen}$, where $1 \le i \le K-1$.

Step 2. Partition dataset X into $K-1$ subsets, $S'_1, \cdots, S'_{K-1}$, by assigning each point to its nearest point from $m_1, \cdots, m_{K-1}$.

Step 3. Build MSTs, $MST(S'_1), \cdots, MST(S'_{K-1})$, with an exact MST algorithm.

Step 4. Combine the $K-1$ MSTs with **CA** to produce an approximate MST MST_2.

2.5. Combine two rounds of approximate MSTs

So far we have two approximate MSTs on dataset X, MST_1 and MST_2. To produce the final approximate MST, we first merge the two approximate MSTs to produce a graph, which has no more than $2(N-1)$ edges, and then apply an exact MST algorithm on this graph to achieve the final approximate MST of X.

Finally, the overall algorithm of proposed method is summarized as follows:

Fast MST (FMST)
Input: Dataset X;
Output: Approximate MST of X;

Step 1. Apply **DAC** on X to produce the K MSTs.

Step 2. Apply **CA** on the K MSTs to produce the first approximate MST, MST_1, and the MST of subset centers, MST_{cen}.

Step 3. Apply **SAM** on MST_{cen} and X to generate the secondary approximate MST, MST_2.

Step 4. Merge MST_1 and MST_2 into a graph G.

Step 5. Apply an exact MST algorithm on G, and the final approximate MST is achieved.

3. Complexity and accuracy analysis

3.1. Complexity analysis

The overall time complexity of the proposed algorithm **FMST**, T_{FMST} , can be evaluated as:

$$T_{FMST} = T_{DAC} + T_{CA} + T_{SAM} + T_{COM} \tag{3}$$

where T_{DAC}, T_{CA} and T_{SAM} are the time complexities of the algorithms **DAC**, **CA** and **SAM** respectively, T_{COM} is the running time of an exact MST algorithm on the combination of MST_1 and MST_2.

DAC consists of two operations: partitioning the dataset X with K-means and constructing the MSTs of the subsets with an exact MST algorithm. Now we consider the time complexity of DAC by the following theorems.

Theorem 1. *Suppose a dataset with N points is equally partitioned into K subsets by K-means, and an MST of each subset is produced by an exact algorithm. If the total running time of partitioning the dataset and constructing MSTs of the K subsets is T, then* $\arg\min_K T = \sqrt{N}$.

Proof. Suppose the dataset is partitioned into K clusters equally so that the number of data points in each cluster equals to N/K. The time complexity of partitioning the dataset and constructing the MSTs of K subsets are $T_1 = NKId$ and $T_2 = K(N/K)^2$, respectively, where I is the number of iterations of K-means and d is the dimension of the dataset. The total complexity is $T = T_1 + T_2 = NKId + N^2/K$. To find the optimal K corresponding to the minimum T, we solve $\partial T/\partial K = NId - N^2/K^2 = 0$ which results in $K = \sqrt{N/Id}$. Therefore, $K = \sqrt{N}$ and $T = O(N^{1.5})$ under the assumption that $I \ll N$ and $d \ll N$. Because convergence of K-means is not necessary in our method, we set I to 20 in all of our experiments. For very high dimensional datasets, $d \ll N$ may not hold, but for modern large high dimensional datasets may not. The situation for high dimensional datasets has been discussed in Section 4.5. □

Although the above theorem holds under the ideal condition of average partition, it can be supported by more evidences when the condition is not satisfied, for example, linear partition and multinomial partition.

Theorem 2. *Suppose a dataset is linearly partitioned into K subsets. If* $K = \sqrt{N}$*, then the time complexity is* $O(N^{1.5})$.

Proof. Let $n_1, n_2, \cdots, n_K$ be the numbers of data points of the K clusters. The K numbers form an arithmetic series, namely, $n_i - n_{i-1} = c$, where $n_1 = 0$ and c is a constant. The arithmetic series sums up to $sum = K * n_K/2 = N$, and thus, we have $n_K = 2N/K$ and $c = 2N/[K(K-1)]$. The time complexity of constructing MSTs of the subsets is then:

$$\begin{aligned} T_2 &= {n_1}^2 + {n_2}^2 + \cdots + {n_{K-1}}^2 \\ &= c^2 + (2c)^2 + \cdots + [(K-1)c]^2 \\ &= c^2 \times \frac{(K-1)K(2K-1)}{6} \\ &= [\frac{2N}{(K-1)K}]^2 \times \frac{(K-1)K(2K-1)}{6} \\ &= \frac{2}{3} \times \frac{(2K-1)N^2}{K(K-1)} \end{aligned} \quad (4)$$

If $K = \sqrt{N}$, then $T_2 = \frac{4}{3}N^{1.5} + \frac{2}{3}\frac{N^{1.5}}{N^{0.5}-1} = O(N^{1.5})$. Therefore, $T = T_1 + T_2 = O(N^{1.5})$ holds. □

Theorem 3. *Suppose a dataset is partitioned into K subsets, and the sizes of the K subsets follow a multinomial distribution. If* $K = \sqrt{N}$*, then the time complexity is* $O(N^{1.5})$.

Proof. Let $n_1, n_2, \cdots, n_K$ be the numbers of data points of the K clusters. Suppose the data points are randomly assigned into the K clusters, and $n_1, n_2, \cdots, n_K \sim Multinomial(N, \frac{1}{K}, \cdots, \frac{1}{K})$. We have $E(n_i) = N/K$ and $Var(n_i) = (N/K) * (1 - 1/K)$. Since $E(n_i^2) = [E(n_i)]^2 + Var(n_i) = N^2/K^2 + N * (K-1)/K^2$, the expected complexity of constructing MSTs is $T_2 = \sum_{i=1}^{K} n_i^2 = K * E(n_i^2) = N^2/K + N * (K-1)/K$, if $K = \sqrt{N}$, then $T_2 = O(N^{1.5})$. Therefore $T = T_1 + T_2 = O(N^{1.5})$ holds. □

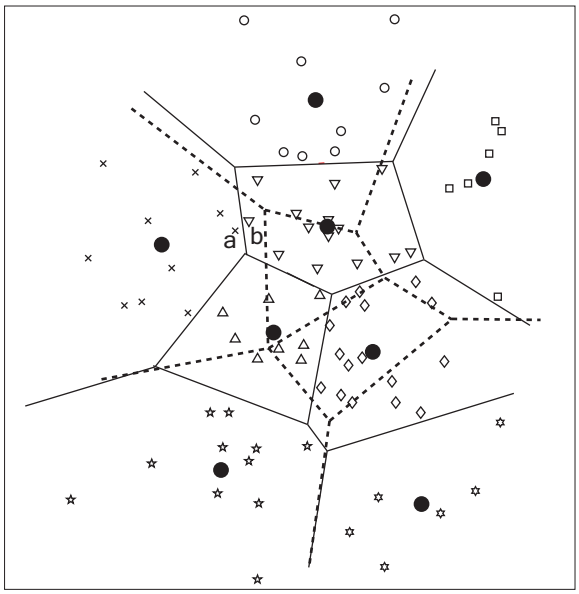


Figure 6: Merge of two Voronoi graphs. Voronoi graph in solid line is corresponding to the first partition, and that in dashed line corresponding to the secondary partition. Only the first partition is illustrated.

According to the above theorems, we have $T_{DAC} = O(N^{1.5})$. Proofs are given in Appendix.

In **CA**, the time complexity of computing the mean points of the subsets is $O(N)$, as one scan of the dataset is enough. Constructing MST of the K mean points by an exact MST algorithm takes only $O(N)$ time. In Step 3, the number of subset pairs is $K-1$, and for each pair, determining the connecting edge by **DCE** requires one scan on the two subsets, respectively. Thus, the time complexity of Step 3 is $O(2N \times (K-1)/K)$, which equals to $O(N)$. The total computational cost of **CA** is therefore $O(N)$.

In **SAM**, Step 1 computes $K-1$ midpoints, which takes $O(N^{0.5})$ time. Step 2 takes $O(N \times (K-1))$ to partition the dataset. The running time of Step 3 is $O((K-1) \times N^2/(K-1)^2) = O(N^2/(K-1))$. Step 4 is to call **CA** and has the time complexity of $O(N)$. Therefore, the time complexity of **SAM** is $O(N^{1.5})$.

The number of edges in the graph that is formed by combining MST_1 and MST_2 is at most $2(N-1)$. The time complexity of applying an exact MST algorithm to this graph is only $O(2(N-1)\log N)$. Thus, $T_{COM} = O(N \log N)$.

To sum up, the time complexity of the proposed fast algorithm is $O(N^{1.5})$.

3.2. Accuracy analysis

Most inaccuracies originate from points that are on the boundary regions of the partitions of K-means. The secondary partition is generated in order to capture these problematic points into the same clusters. Inaccuracies after the refinement

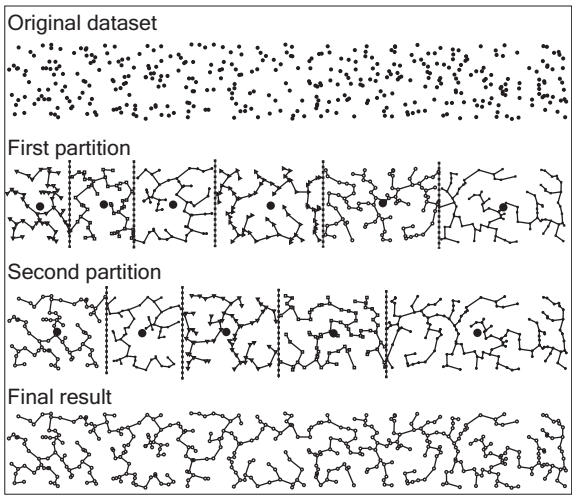


Figure 7: The collinear Voronoi graph case.

stage can therefore originate only if two points that should be connected by the exact MST, but are partitioned into different clusters both in the primary and in the secondary partition, and neither of the two conquer stages will be able to connect these points. In Fig. 6, few such pair of points are shown that belong to different clusters in both partitions. For example, point a and b belong to different clusters of the first partition, but are in the same cluster of the second one.

Since partitions generated by K-means form a Voronoi graph [16], the analysis of the inaccuracy can be related to degree of which the secondary Voronoi edges overlap that of the Voronoi edges of the primary partition. Let $|E|$ denote the number of edges of a Voronoi graph, in two-dimensional space, $|E|$ is bounded by $K - 1 \leq |E| \leq 3K - 6$, where K is the number of clusters (the Voronoi regions). In higher dimensional case it is more difficult to analysis.

A favorable case is demonstrated in Fig. 7. The first row is a dataset which consists of 400 points and is randomly distributed. In the second row, the dataset is partitioned into 6 clusters by K-means, and a collinear Voronoi graph is achieved. In the third row, the secondary partition has 5 clusters, each of which completely cover one boundary region in the second row. An exact MST is produced in the last row.

4. Experiments

In this section, experimental results are presented to illustrate the efficiency and the accuracy of the proposed fast approximate MST algorithm. The accuracy of FMST is tested with both synthetic datasets and real applications. As a framework, the proposed algorithm can be incorporated with any exact or even approximate MST algorithm, of which the running time is definitely reduced. Here we only take into account Kruskal's and Prim's algorithms because of their popular. As in Kruskal's algorithm, all the edges need to be sorted into nondecreasing order, it is difficult to apply the algorithm to large datasets. Furthermore, Prim's algorithm may employ Fibonacci heap to reduce the running time, we therefore use it rather than Kruskal's algorithm in our experiments as the exact MST algorithm.

Experiments were conducted on a PC with an Intel Core2 2.4GHz CPU and 4GB memory running Windows 7. The algorithm for testing the running time is implemented in C++, while the other tests are performed in Matlab (R2009b).

4.1. Running time

4.1.1. Running time on different datasets

We first perform experiments on four typical datasets with different size and dimensions for testing the running time. The four datasets are described as in Table 1.

Table 1: The description of four datasets

	t4.8k	MNIST	ConfLongDemo	MiniBooNE
Data size	8,000	10,000	164,860	130,065
Dimension	2	784	3	50

Dataset t4.8k is designed for testing CHAMELEON clustering algorithm [28] and available from [54]. MNIST is a dataset of 10 handwriting digits and contains 60,000 training patterns and 10,000 test patterns of 784 dimensions. We just use the test set, which can be found in [55]. The last two sets are from UCI machine learning repository [56]. ConfLongDemo has 8 attributes, of which only 3 numerical attributes are used here.

From each dataset, subsets with different size are randomly selected to test the running time as a function of data size. The subset sizes of the first two datasets gradually increase with step 20, the third with step 100 and the last with step 1000.

In general, the running time of constructing an MST of a dataset depends on the size of the dataset but not on the underlying structure in the dataset. In our FMST method, K-means is employed to partition a dataset, and the size of subsets depends on initialization of K-means and distributions of the datasets, which leads to different time costs. We therefore perform FMST ten times on each dataset for alleviating the effects of the random initialization of K-means.

The running time of FMST and Prim's algorithm on the four datasets is illustrated in the first row of Fig. 8. From the results, we can see that FMST is computationally more efficient than Prim's algorithm, especially for the large datasets ConfLongDemo and MiniBooNE. The efficiency for MiniBooNE shown in the rightmost of the second and third row in Fig. 8, however, deteriorates because of the high dimensionality.

Although the complexity analysis indicates that the time complexity of proposed FMST is $O(N^{1.5})$, the actual running time can be different. We analyze the actual processing time by fitting an exponential function $T = aN^b$, where T is the running time and N is the number of data points. The the results are shown in Table 2.

4.1.2. Running time with different Ks

We have discussed the number of clusters K and set it to $\sqrt{N}$ in Section 2.2.1, and have also presented some supporting

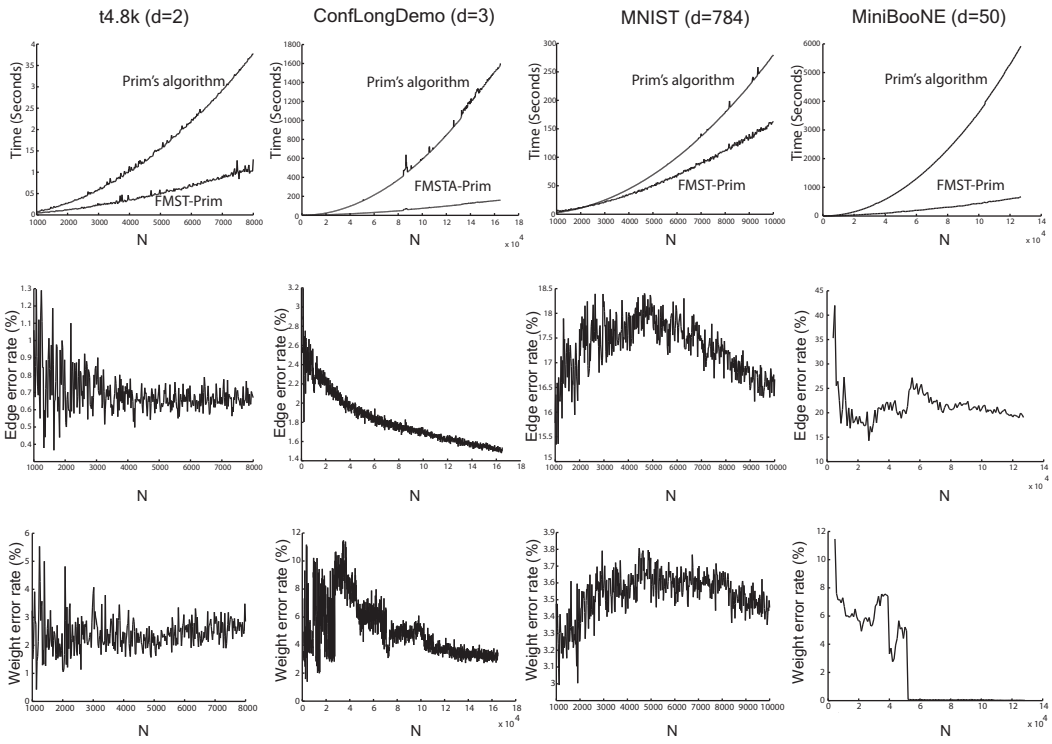


Figure 8: The results of the test on the four datasets. The first row shows the running time of t4.8k, ConfLongDemo, MNIST and MiniBooNE, respectively. The second row shows corresponding edge error rates. The third show shows corresponding weight error rates.

Table 2: The exponent bs obtained by fitting $T = aN^b$

	b			
	t4.8k	MNIST	ConfLongDemo	MiniBooNE
FMST	1.57	1.62	1.54	1.44
Prim's Alg.	1.88	2.01	1.99	2.00

theorems in Section 3. In practical applications, however, the value is slightly different. Some experiments are performed on dataset t4.8k and ConfLongDemo to study the effect of different Ks on running time. The experimental results are illustrated in Fig. 9, from which we find that if K is set to 38 for t4.8k and 120 for ConfLongDemo, the running time will be minimum. But according to the previous analysis, K would be set to $\sqrt{N}$, namely 89 and 406 for the two datasets, respectively. Therefore, K is practically set to $\frac{\sqrt{N}}{C}$, where $C > 1$. For dataset t4.8k and ConfLongDemo, C is approximate 3.

4.2. Accuracy on synthetic datasets

4.2.1. Measures by edge error rate and weight error rate

The accuracy is another important performance of FMST. Two accuracy rates are defined: edge error rate ER_{edge} and weight error rate ER_{weight}. Before ER_{edge} is defined, we present a notation of equivalent edge of an MST, because the MST may not be unique. The equivalence property is described as:

Equivalence Property. Let T and T' be the two different MSTs of a dataset. For any edge $e \in (T \setminus T')$, there must exist another edge $e' \in (T' \setminus T)$ such that $(T' \setminus \{e'\}) \cup \{e\}$ is also an MST. We call e and e' a pair of equivalent edges.

Proof. The equivalency property can be operationally restated as: Let T and T' be the two different MSTs of a dataset, for any edge $e \in (T \setminus T')$, there must exist another edge $e' \in (T' \setminus T)$ such that $w(e) = w(e')$ and e connects T'_1 and T'_2, where T'_1 and T'_2 are the two subtrees generated by removing e' from T', $w(e)$ is the weight of e.

Let G be the cycle formed by $\{e\} \cup T'$, we have:

$$\forall e' \in (G \setminus \{e\} \setminus (T \cap T')), w(e) \geq w(e') \tag{5}$$

Otherwise, an edge in $G \setminus \{e\} \setminus (T \cap T')$ should be replaced by e when constructing T'.

Furthermore, the following claim holds: there must exist at least one edge $e' \in (G \setminus \{e\} \setminus (T \cap T'))$, such that the cycle formed by $\{e'\} \cup T$ contains e. We prove this claim by contradiction.

Assume that all the cycles G'_j formed by $\{e'_j\} \cup T$ do not contain e, where $e'_j \in (G \setminus \{e\} \setminus (T \cap T'))$, $1 \leq j \leq |G \setminus \{e\} \setminus (T \cap T')|$. Let $G_{union} = (G'_1 \setminus \{e'_1\}) \cup \cdots \cup (G'_l \setminus \{e'_l\})$, where $l = |G \setminus \{e\} \setminus (T \cap T')|$. G can be expressed as $\{e\} \cup \{e'_1\} \cup \cdots \cup \{e'_l\} \cup G_{delta}$, where $G_{delta} \subset (T \cap T')$. As G is a cycle, $G_{union} \cup \{e\} \cup G_{delta}$ must be also a cycle, this is contradict because $G_{union} \subset T$, $G_{delta} \subset T$ and $e \in T$. Therefore the claim is correct.

As a result, there must exist at least one edge $e' \in (G \setminus \{e\} \setminus (T \cap T'))$ such that $w(e') \geq w(e)$.

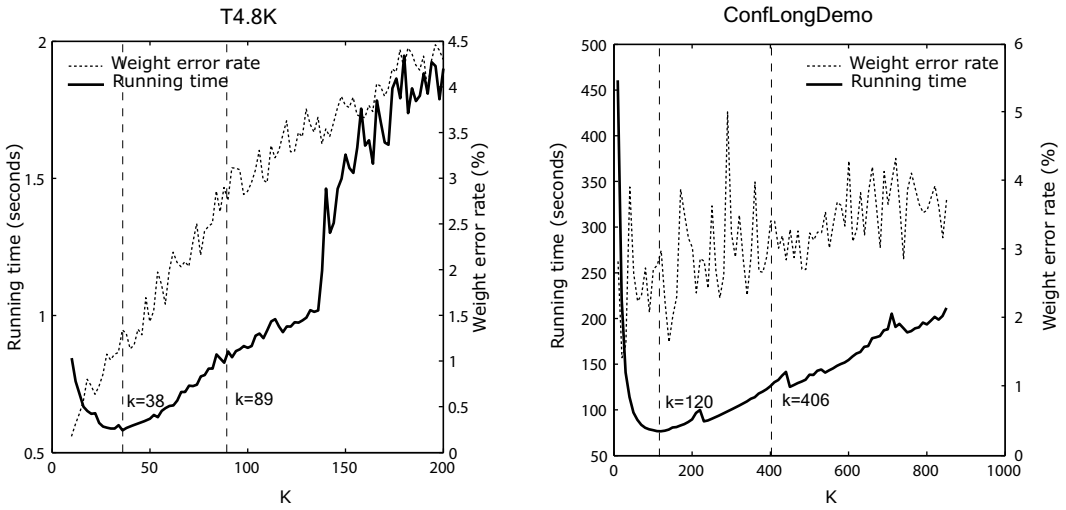


Figure 9: Performances (running time and weight error rate) as a function of K. The left shows the running time and weight error rate of FMST on t4.8k, and the right on ConfLongDemo.

Combine this result with (5), we have the following: for $e \in (T \setminus T')$, there must exist an edge $e' \in (T' \setminus T)$ such that $w(e) = w(e')$. Furthermore, as e and e' are in the same cycle G, $(T' \setminus \{e'\}) \cup \{e\}$ is still an MST. □

According to the equivalency property, we define a criterion to determine whether an edge belongs to an MST:

Let T be an MST and e be an edge of a graph. If there exists an edge $e' \in T$ such that $|e| = |e'|$ and e connects T_1 and T_2, where T_1 and T_2 are the two subtrees achieved by removing e' from T, then e is a *correct edge*, i.e., belongs to an MST.

Suppose E_{appr} is the set of the correct edges in an approximate MST, the edge error rate ER_{edge} is defined as:

$$ER_{edge} = \frac{N - |E_{appr}| - 1}{N - 1} \tag{6}$$

The second measure is defined as the differ of the sum of the weights in FMST and the exact MST, which is called weight error rate ER_{weight}:

$$ER_{weight} = \frac{W_{appr} - W_{exact}}{W_{exact}} \tag{7}$$

where W_{exact} and W_{appr} are the sum of weights of the exact MST and FMST, respectively.

The edge error rates and weight error rates of the four datasets are shown in the third row of Fig. 8. We can see that both the edge error rate and the weight error rate decrease with the increase of the data size. For datasets with high dimension, the edge error rates are bigger, for example, the maximum edge error rates of MNIST are approximate to 18.5%, while those of t4.8k and ConfLongDemo less than 3.2%. In contrast, the weight error rates decrease when the dimensionality increases. For instance, the weight error rates of MNIST are less than 3.9%. This is the phenomenon of the curse of dimensionality. The high dimensional case will be discussed further in Section 4.5.

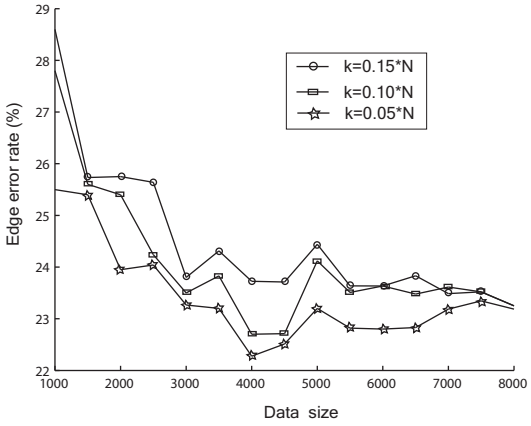


Figure 10: The edge error rate of Lai's method on t4.8k.

4.2.2. Accuracy with different Ks

Globally, the edge and weight error rates increase with K. This is because the bigger the K, the more the split boundaries, from which error edges come. But when K is small, the error rates increases slowly with K. In Fig. 9, we can see that the weight error rates are still low when K is set to approximate $\frac{\sqrt{N}}{3}$.

4.2.3. Comparison to Lai's method

The purpose of the method in [45] is to detect the clusters efficiently by removing the longer edges of the MST at an early stage, and no approximate MST is produced. The method in [34] is designed for the same purpose, an approximate MST is generated in the first stage. Therefore, we compare the proposed FMST with the method in [34].

The accuracy of the approximate MST produced in [34] is relevant to a parameter: the number of the nearest neighbors of a data point. This parameter is used to update the priority

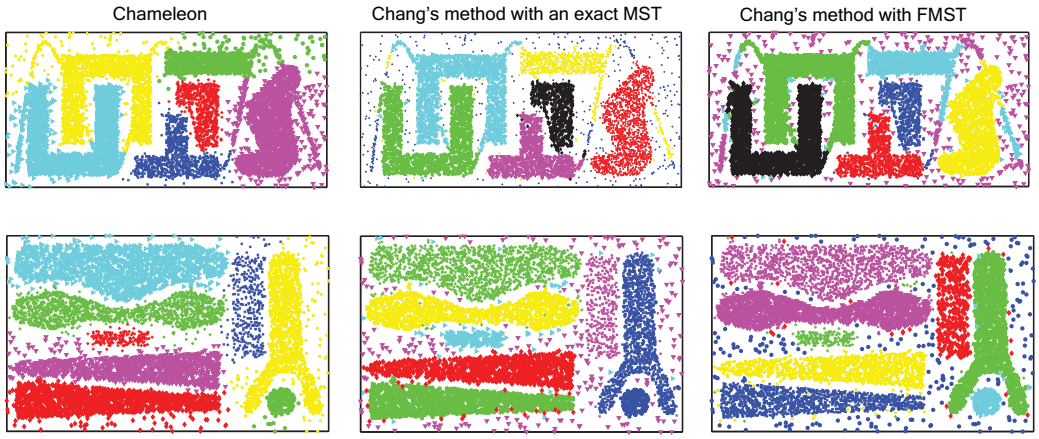


Figure 11: Clustering results for t4.8k (above) and t8.8k (below).

queue when an algorithm like Prim's is employed to construct an MST. In general, the larger the number is, the more accurate the approximate MST is. However, this parameter is also relevant to the computational cost of the approximate MST, which is $O(dN(b + k + k \log N))$, where k is the number of nearest neighbors and b is the number bits of a Hilbert number. Here we only focus on the accuracy of the the method, and the number of nearest neighbors is set to $N * 0.05$, $N * 0.10$, $N * 0.15$, respectively. The accuracy is tested on t4k.8k, and the result is shown in Fig.10. From the result, the edge error rates are more than 22%, and much higher than that of FMST, even if the number of nearest neighbors is set to $N * 0.15$, which leads to the lost of the computational efficiency of the method.

4.3. Accuracy on clustering

In this subsection, the accuracy of FMST is tested in clustering application. Path-based clustering employs minimax distance metric to measure the dissimilarities of data points [17] [18]. For a pair of data points x_i, x_j, the minimax distance D_{ij} is defined as:

$$D_{ij} = \min_{\mathcal{P}_{ij}^k} \{ \max_{(x_p, x_{p+1}) \in \mathcal{P}_{ij}^k} d(x_p, x_{p+1}) \} \tag{8}$$

where $\mathcal{P}_{ij}^k$ denotes all possible paths between x_i and x_j and k is an index to enumerate the paths, and $d(x_p, x_{p+1})$ is the Euclidean distance between x_p and x_{p+1}.

The minimax distance can be computed by all-pair shortest path algorithm, such as Floyd-Warshall algorithm. However, this algorithm runs in time $O(N^3)$. An MST is be used to compute the minimax distance more efficiently in [31]. To make the path-based clustering robust to outliers, Chang and Yeung [8] improved minimax distance and incorporated it into spectral clustering. We test FMST within this method on two synthetic datasets (t4.8k and t8.8k) from [28] and one color image dataset from [8]. For segmenting the color images, the color and spatial features are used. Each image has 154,401 (321×481) pixels and is divided into 1855 (35×53) overlapping patches of size 13×13 pixels each. One patch has five features: average RGB colors and two coordinates of the patch in the image. The coordinate features are less important than color features, therefore, a weight of 0.5 is assigned to coordinate features and 1 to color features.

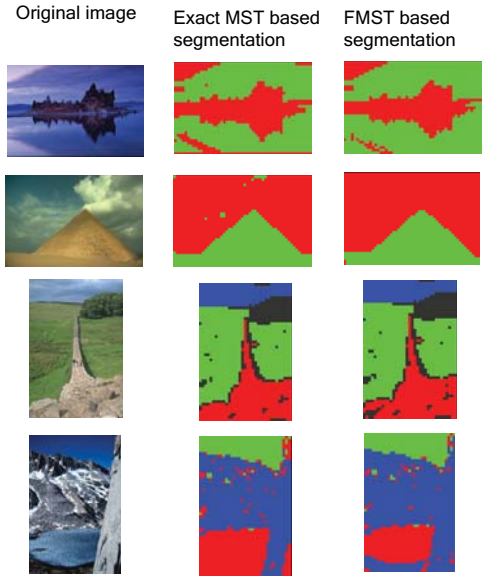
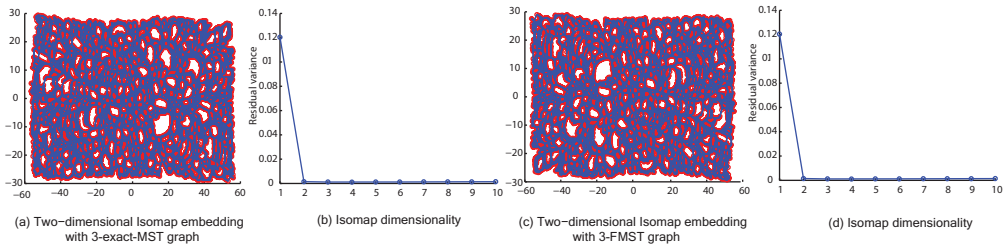


Figure 12: Robust path-based clustering of images using exact MST and FMST. In each row, the leftmost picture is original image, the rightmost is FMST based clustering, while the middle is exact MST based clustering.

For computing minimax distances, exact MST and FMST are used respectively. In Fig. 11, the clustering results of t4.4k and t8.8k are shown. Robust path-based method with FMST has similar performance to that with exact MST. Moreover, CHAMELEON performs well on both datasets, while robust path-based method also performs well except a strange phenomenon: a small number of data points (usually outliers) may be partitioned into a cluster far away.

In Fig. 12, robust path-based method with MST and FMST are applied to four images respectively. From the figure, we can observe that the segmentation results of FMST based clustering



(a) Two−dimensional Isomap embedding with 3-exact-MST graph (b) Isomap dimensionality (c) Two−dimensional Isomap embedding with 3-FMST graph (d) Isomap dimensionality

Figure 13: 3-MST graph based ISOMAP using exact MST and FMST respectively. In (a) and (c), the two dimensional embedding is illustrated. (b) and (d) are corresponding resolutions.

are similar to those of exact MST based clustering.

4.4. Accuracy on manifold learning

MST has been used for manifold learning [48, 49]. For a kNN based neighborhood graph, an improperly selected k may lead to a disconnected graph, and degrade the performance of manifold learning. To address this problem, Yang in [48] used MSTs to construct a k-edge connected neighborhood graph. We implement the method of [48], with exact MST and FMST respectively, to reduce the dimensionality of a manifold.

The FMST based and the exact MST based dimensionality reduction are performed on dataset Swiss-roll, which has 20,000 data points. In experiments, we select the first 10,000 data points because of the memory requirement, and set $k = 3$. The accuracy of FMST based dimensionality reduction is compared with that of a exact MST based in Fig. 13. The intrinsic dimensionality of Swiss-roll can be detected by the "elbow" of the curves in (b) and (d). Obviously, the MST graph based method and the FMST graph based method have almost identical residual variance, and both indicate the intrinsic dimensionality is 2. Furthermore, Fig. 13 (a) and (c) show that the two methods have similar two-dimensional embedding results.

4.5. Discussion on high dimensional datasets

As described in the experiments, the performances of both computation and accuracy of the proposed method are reduced when applied to high dimensional datasets. Since the time complexity of FMST is $O(N^{1.5})$ under the condition of $d \ll N$, when the number of dimensions d is getting big and even approximate to N, the computational cost will degrade to $O(N^{2.5})$. However, it is still more efficient than corresponding Kruskal's or Prim's algorithm.

The accuracy of FMST is reduced because of the curse of dimensionality, which includes distance concentration phenomenon and the hubness phenomenon [40]. The distance concentration phenomenon is that the distances between all pairs of data points from a high dimensional dataset are almost equal, in other words, the traditional distance measures become ineffective, and the distances computed with the measures become unstable [25]. For constructing an MST in terms of these distances, the results of Kruskal's or Prim's algorithm are meaningless, so is the accuracy of the proposed FMST. Furthermore, hubness phenomenon in a high dimensional dataset, which implies some data points may appear in many more KNN lists than other data points, shows the nearest neighbors become also meaningless. Obviously, the hubness affects the construction of an MST in the same way.

The intuitive way to address the above problems caused by the curse of dimensionality is to employ dimensionality reduction methods, such as ISOMAP, LLE, or subspace based methods for a concrete task in machine learning, such as subspace based clustering. Similarly, for constructing an MST of a high dimensional dataset, one may preprocess the dataset with dimensionality reduction or subspace based methods for the purpose of getting more meaningful MSTs.

5. Conclusion

In this paper, we have proposed a fast MST algorithm with a divide and conquer scheme. Under the assumption that the dataset is partitioned into equal sized subsets in divide step, the time complexity of the proposed algorithm is theoretically $O(N^{1.5})$, which holds reasonably well in practice. Accuracy of FMST is analyzed experimentally using edge error rate and weight error rate. Furthermore, two practical applications are considered, and the experiments indicated that the proposed FMST can be applied on large datasets.

Acknowledgment

The work of C. Zhong was supported by Natural Science Foundation of China: No. 61175054, 61075056, Zhejiang Provincial Natural Science Foundation of China: No. Y1090851, and the Center for International Mobility (CIMO). The work of D. Miao was supported by the National Natural Science Foundation of China: No. 60970061, 61075056, and the Research Fund for the Doctoral Program of Higher Education: No. 20060247039.

References

[1] L. An, Q.S. Xiang, & Chavez, S., A fast implementation of the minimum spanning tree method for phase unwrapping, IEEE Trans. Medical Imaging, 19(2000), 805-808.

[2] Abdel-Wahab, H., Stoica, I., Sultan, F., & Wilson, K. (1997). A simple algorithm for computing minimum spanning trees in the internet. *Information Sciences*, 101, 47-69.

[3] B. Awerbuch, Optimal distributed algorithms for minimum weight spanning tree, counting, leader election, and related problems, In Proceedings of the 19th ACM Symposium on Theory of Computing, 1987.

[4] D.A. Bader, Cong, G., Fast shared-memory algorithms for computing the minimum spanning forest of sparse graphs, Journal of Parallel and Distributed Computing, 66(2006), 1366-1378.

[5] J.C. Bezdek, N.R. Pal, Some new indexes of cluster validity, IEEE Trans. Systems, Man and Cybernetics, Part B, 28(1998), 301-315.

[6] O. Borůvka, O jistém problému minimálním (About a Certain Minimal Problem). *Práce moravské přírodovědecké společnosti v Brně III, 1926, 37-58 (in Czech with German summary)*.

[7] P.B. Callahan, S.R. Kosaraju, Faster algorithms for some geometric graph problems in higher dimensions, In Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms, 1993.

[8] H. Chang, D.Y. Yeung, Robust path-based spectral clustering, Pattern Recognition, 41(2008), 191-203.

[9] B. Chazelle, A minimum spanning tree algorithm with inverse-Ackermann type complexity, Journal of the ACM, 47(2000), 1028-1047.

[10] G. Chen, et al., The multi-criteria minimum spanning tree problem based genetic algorithm. Information Sciences, 177(2007), 5050-5063.

[11] D. Cheriton, R.E. Tarjan, Finding Minimum Spanning Trees, SIAM Journal on Computing, 5(1976), 24-742.

[12] K.W. Chong, Y. Han, T.W. Lam, Concurrent threads and optimal parallel minimum spanning trees algorithm, Journal of the ACM, 48(2001), 297-323.

[13] G. Choquet, Etude de certains réseaux de routes, Comptesrendus de l'Acadmie des Sciences, 206(1938), 310 (in French).

[14] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, second ed, 2001, The MIT Press.

[15] E.W. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik, 1(1959), 269-271.

[16] Q. Du, V. Faber, M. Gunzburger, Centroidal Vorono Tessellations: Applications and Algorithms, SIAM Review, 41(1999), 637-676.

[17] B. Fischer, J.M. Buhmann, Path-based clustering for grouping of smooth curves and texture segmentation, IEEE Trans. Pattern Anal. Mach. Intell., 25(2003), 513-518.

[18] B. Fischer, J.M. Buhmann, Bagging for path-based clustering, IEEE Trans. Pattern Anal. Mach. Intell., 25(2003), 1411-1415.

[19] K. Florek, J. Łkaszewicz, H. Perkal, H. Steinhaus, S. Zubrzycki, Sur la liaison et la division des points d'un ensemble fini, Colloquium Mathematicum, 2(1951), 282-285.

[20] M.L. Fredman, R.E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, Journal of the ACM, 34(1987), 596-615.

[21] H.N. Gabow, Z. Galil, T.H. Spencer, R.E. Tarjan, Efficient algorithms for finding minimum spanning trees in undirected and directed graphs, Combinatorica, 6(1986),109-22.

[22] H.N. Gabow, Z. Galil, T.H. Spencer, Efficient implementation of graph Algorithms using contraction, Journal of the ACM, 36(1989), 540-572.

[23] R.G. Gallager, P.A. Humblet, P.M. Spira, A distributed algorithm for minimum-weight spanning trees, ACM Transactions on Programming Languages and Systems, 5(1983), 66-77.

[24] J.C. Gower, G.J.S. Ross, Minimum Spanning Trees and Single Linkage Cluster Analysis, Journal of the Royal Statistical Society, Series C (Applied Statistics), 18(1969), 54-64.

[25] C.M. Hsu, M.S. Chen, On the Design and Applicability of Distance Functions in High-Dimensional Data Space, IEEE Trans. Knowledge and Data Engineering, 21(2009), 523-536.

[26] V. Jarník, O jistém problému minimálním (About a Certain Minimal Problem). Práce moravské přírodovědecké společnosti v Brně, VI(1930), 57-63 (in Czech).

[27] P. Juszczak, D.M.J. Tax, E. Pękalska, R.P.W. Duin, Minimum spanning tree based one-class classifier, Neurocomputing, 72(2009), 1859-1869.

[28] G. Karypis, E.H. Han, V. Kumar, CHAMELEON: A hierarchical clustering algorithm using dynamic modeling, IEEE Trans. Comput., 32(1999), 68-75.

[29] M. Khan, G. Pandurangan, A fast distributed approximation algorithm for minimum spanning trees, Distributed Computing, 20(2008), 391-402.

[30] K. Li, S. Kwong, J. Cao, M. Li, J. Zheng, R. Shen, Achieving balance between proximity and diversity in multi-objective evolutionary algorithm, Information Sciences, 182(2012), 220-242.

[31] K.H. Kim, S. Choi, Neighbor Search with Global Geometry: A Minimax Message Passing Algorithm, In Proceedings of the 24th International Conference on Machine Learning, 2007, pp. 401-408.

[32] J.B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, Proceedings of the American Mathematical Society, 7(1956), 48-50.

[33] B. Lacevic, E. Amaldi, Ectropy of diversity measures for populations in Euclidean space, Information Sciences, 181(2011), 2316-2339.

[34] C. Lai, T. Rafa, D.E. Nelson, Approximate minimum spanning tree clustering in high-dimensional space, Intelligent Data Analysis, 13(2009), 575-597.

[35] W.B. March, P. Ram, and A.G. Gray. Fast euclidean minimum spanning tree: algorithm, analysis, and applications. in Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. 2010. ACM.

[36] T. Öncan, Design of capacitated minimum spanning tree with uncertain cost and demand parameters, Information Sciences, 177(2007), 4354-4367.

[37] D. Peleg, V. Rubinovich, A near tight lower bound on the time complexity of distributed minimum spanning tree construction. SIAM Journal on Computing, 30(2000), 1427-1442.

[38] S. Pettie, V. Ramachandran, A randomized time-work optimal parallel algorithm for finding a minimum spanning forest. *SIAM Journal on Computing*, 31(2000), 1879-1895.

[39] R.C. Prim, Shortest connection networks and some generalizations, Bell System Technical Journal, 36(1957), 567-574.

[40] M. Radovanovic, A. Nanopoulos, M. Ivanovic, Hubs in Space: Popular Nearest Neighbors in High-Dimensional Data, Journal of Machine Learning Research, 11(2010), 2487-2531.

[41] M.R. Rezaee, B.P.F. Lelieveldt, J.H.C. Reiber, A new cluster validity index for the fuzzy c-mean, Pattern Recognition Letters, 19(1998), 237-246.

[42] Sollin M. (1965). Le trace de canalisation, in: C. Berge, *A. Ghouilla-Houri (Eds.), Programming, Games, and Transportation Networks*. Wiley, New York (in French).

[43] S. Sundar, A. Singh, A swarm intelligence approach to the quadratic minimum spanning tree problem, Information Sciences, 180(2010), 3182-3191.

[44] P.M. Vaidya, Minimum spanning trees in *k*-dimensional space, SIAM Journal on Computing, 17(1988), 572-582.

[45] X. Wang, X. Wang, D.M. Wilkes, A divide-and-conquer approach for minimum spanning tree-based clustering, IEEE Trans, Knowledge and data engineering, 21(2009), 945-958.

[46] Y. Xu, V. Olman, D. Xu, Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees, Bioinformatics, 18(2002), 536-545.

[47] Y. Xu, E.C. Uberbacher, 2D image segmentation using minimum spanning trees, Image and Vision Computing, 15(1997), 47-57.

[48] L. Yang, k-Edge Connected Neighborhood Graph for Geodesic Distance Estimation and Nonlinear Data Projection, In Proceedings of the 17th International Conference on Pattern Recognition, 2004, (ICPR'04).

[49] L. Yang. Building k edge-disjoint spanning trees of minimum total length for isometric data embedding, IEEE Trans. Pattern Analysis and Machine Intelligence, 27(2005), 1680-1683.

[50] A.C. Yao, An $O(|E| \log \log |V|)$ algorithm for finding minimum spanning trees, Information Processing Letters, 4(1975), 21-23.

[51] C.T. Zahn, Graph-theoretical methods for detecting and describing gestalt clusters, IEEE Trans. Computer, C20(1971), 68-86.

[52] C. Zhong, D. Miao, R. Wang, A graph-theoretical clustering method based on two rounds of minimum spanning trees, Pattern Recognition, 43(2010), 752-766.

[53] C. Zhong, D. Miao, and P. Fränti, Minimum spanning tree based split-and-merge: A hierarchical clustering method. Information Sciences, 181(2011), 3397-3410.

[54] HTTP://glaros.dtc.umn.edu/gkhome/cluto/cluto/download.

[55] HTTP://yann.lecun.com/exdb/mnist.

[56] HTTP://archive.ics.uci.edu/ml/.