

**ANKARA YILDIRIM BEYAZIT UNIVERSITY**  
**GRADUATE SCHOOL OF NATURAL AND APPLIED**  
**SCIENCES**



**MST-based Cluster Analysis: A New Algorithm for  
Determining Inconsistent Edges**  
**PhD Thesis by**  
**Fadi Şaar**

**Department of Computer Engineering**

**March, 2021**

**ANKARA**

# **MST-based Cluster Analysis: A New Algorithm for Determining Inconsistent Edges**

**A Thesis Submitted to**

**The Graduate School of Natural and Applied Sciences of**

**Ankara Yıldırım Beyazıt University**

**In Partial Fulfilment of the Requirements for the Degree of Doctor of  
Philosophy in Department of Computer Engineering**

**by**

**Fadi Şaar**

**March, 2021**

**ANKARA**

## PhD THESIS EXAMINATION RESULT FORM

We have read the thesis entitled “**MST-based Cluster Analysis: A New Algorithm for Determining Inconsistent Edges**”, completed by **Fadi Şaar** under the supervision of **ASST. PROF. AHMET ERCAN TOPCU**, and we certify that in our opinion, it is fully adequate, in scope and in quality, as a thesis for the degree of PhD.

Asst. Prof. Ahmet Ercan TOPCU

---

Supervisor

Prof. Fatih Vehbi ÇELEBİ

---

Jury Member

Prof. Fatih KOYUNCU

---

Jury Member

Asst. Prof. Beytullah YILDIZ

---

Jury Member

Assoc. Prof. Hasan BULUT

---

Jury Member

Prof. Dr. Ergün ERASLAN

---

Director

Graduate School of Natural and Applied Science

## **ETHICAL DECLARATION**

I hereby declare that, in this thesis, which has been prepared in accordance with the Thesis Writing Manual of Graduate School of Natural and Applied Sciences,

- All data, information, and documents were obtained in the framework of academic and ethical rules,
- All of information, documents, and assessments are presented in accordance with scientific ethics and morals,
- All of the materials that have been utilized are fully cited and referenced,
- No changes have been made to the utilized materials,
- All of the works presented are original,

and in any contrary case of above statements, I accept to renounce all my legal rights.

**Date:03 March, 2021    Signature                    :.....**

**Name & Surname: Fadi Saar**

## ACKNOWLEDGMENT

Praises and thanks be to Allah Almighty for giving me the strength and ability to complete this thesis. It is also a great pleasure to acknowledge my deepest thanks and gratitude to **Asst. Prof. Ahmet Ercan TOPCU**, Professor of the Graduate School of Natural and Applied Sciences, of Ankara Yıldırım Beyazıt University, for his kind supervision, guidance, and encouragement during my study.

In addition to my advisor, I would like to thank the rest of my **thesis committee**, for their valuable suggestions, and insightful comments, which contributed greatly to this work.

Last but not least, the greatest thanks go to my beloved **family members** for their infinite support. I find no words to acknowledge the sacrifice, help, and inspiration rendered by **my parents, my wife, and my children** to take up this research.

**03 March, 2021**

**Fadi Şaar**

# **MST-BASED CLUSTER ANALYSIS: A NEW ALGORITHM FOR DETERMINING INCONSISTENT EDGES**

## **ABSTRACT**

In recent years, graph-based data clustering algorithms have become popular as they perform connectivity-based rather than centroid-based partitioning. Methods related to minimum spanning tree (MST)-based data clustering are types of graph-based algorithms that can recognize arbitrary shapes of clusters by eliminating inconsistent edges from MST graphs. In all MST-based data clustering algorithms, definition of inconsistent edges is the main problem that needs to be addressed. The longest edges in MST graphs are considered as inconsistent edges under ideal conditions. Nevertheless, outliers often exist in real-world tasks, which makes the longest edges inaccurate cluster separation indicators. In this study, we propose a new data clustering algorithm using MST and a critical distance method. The proposed algorithm solves the main issue of MST-based data clustering, namely identifying and removing inconsistent edges to obtain clusters even in the event that the dataset contains some outliers. It begins by constructing the MST over a given weighted graph based on Euclidean distance and then splits up the graph into clusters by eliminating inconsistent edges using critical distance as a threshold. Integration of the advantages of both MST and critical distance methodology to obtain optimal clusters is the main contribution of this work. The conducted experimental analysis and results using different datasets prove that our proposed clustering algorithm yields better overall performance compared with the most common data clustering algorithms.

**Keywords:** MST; Graph based clustering algorithm; Euclidean distance; Inconsistent edges.

## **MST TABANLI KÜME ANALİZİ: TUTARSIZ KENARLARI BELİRLEMEK İÇİN YENİ BİR ALGORİTMA**

### **ÖZ**

Son yıllarda grafik tabanlı veri kümeleme algoritmaları, orta nokta tabanlı bölümlere yerine bağlantı tabanlı gerçekleştirdikleri için popüler hale gelmektedir. Minimum yayılma ağacı (MST) tabanlı veri kümelemeyle ilgili yöntemler, MST grafiklerinden tutarsız kenarları ortadan kaldırarak rastgele kümelerin şekillerini tanıyabilen grafik tabanlı algoritma türleridir. Tüm MST tabanlı veri kümeleme algoritmalarında, tutarsız kenarların tanımlanması, ele alınması gereken ana sorundur. MST grafiklerinde en uzun kenarlar, ideal koşullar altında tutarsız kenarlar olarak kabul edilmektedir. Bununla birlikte, aykırı değerler gerçek veri kümelerinde genellikle bulunmakta ve bu da en uzun kenarları hatalı küme ayırma göstergeleri yapmaktadır. Bu çalışmada, MST ve kritik mesafe yöntemi kullanılarak yeni bir veri kümeleme algoritması önerilmektedir. Önerilen algoritma, MST tabanlı veri kümelemesinin ana sorununu, yani veri kümesinin bazı aykırı değerler içermesi durumunda bile kümeleri elde etmek için tutarsız kenarları tanımlama ve kaldırma sorununu çözmektedir. MST'yi Öklid mesafesine dayalı olarak belirli bir ağırlıklı grafik üzerinde inşa ederek başlar ve ardından kritik mesafeyi bir eşik olarak kullanarak tutarsız kenarları ortadan kaldırarak grafiği kümelere ayırmaktadır. Optimal kümeleri elde etmek için hem MST hem de kritik mesafe metodolojisinin avantajlarının entegrasyonu, bu çalışmanın ana katkısıdır. Farklı veri kümeleri kullanılarak gerçekleştirilen deneysel analiz ve sonuçlar, önerilen kümeleme algoritmamızın en yaygın veri kümeleme algoritmalarına kıyasla daha iyi genel performans sağladığını kanıtlamaktadır.

**Anahtar Kelimeler:** MST; Grafik tabanlı kümeleme algoritması; Öklid mesafesi; Tutarlı olmayan kenarlar.

## CONTENTS

PhD THESIS EXAMINATION RESULT FORM .....	iii
ETHICAL DECLARATION .....	iii
ACKNOWLEDGMENT .....	iv
ABSTRACT .....	v
ÖZ .....	vi
NOMENCLATURE .....	ix
LIST OF FIGURES .....	x
LIST OF TABLES .....	xii
<b>CHAPTER 1 .....</b>	<b>1</b>
<b>INTRODUCTION .....</b>	<b>1</b>
1.1 Motivation and Research Problem .....	2
1.2 Research Objectives and Contributions .....	3
1.3 Dissertation Organization .....	4
<b>CHAPTER 2 .....</b>	<b>6</b>
<b>LITERATURE REVIEW .....</b>	<b>6</b>
2.1 The Concept of Clustering .....	6
2.2 Applications of Clustering .....	7
2.3 Categories of Clustering Algorithms .....	9
2.4 Typical Distance Measures .....	12
2.5 Related Works .....	13
<b>CHAPTER 3 .....</b>	<b>19</b>
<b>THE PROPOSED MST-BASED DATA CLUSTERING ALGORITHM .....</b>	<b>19</b>
3.1 The Proposed Clustering Algorithm .....	20
3.2 Experimental Study .....	29
3.2.1 Experimental Setup .....	29
3.2.2 Experimental Results on Synthetic Datasets .....	30
3.2.3 Experimental Results on Real Datasets .....	34
<b>CHAPTER 4 .....</b>	<b>36</b>
<b>EXPERIMENTAL ANALYSIS AND DISCUSSION .....</b>	<b>36</b>
4.1 Cases Related to datasets that are free from outliers .....	37
4.2 Cases Related to Merging the Closest Clusters .....	38
4.3 Cases Related to Existence of Some Outlier Distances .....	39

<b>CHAPTER 5 .....</b>	<b>46</b>
<b>CONCLUSION AND FUTURE WORK .....</b>	<b>46</b>
5.1 Conclusion .....	46
5.2 Future Work .....	47
<b>REFERENCES.....</b>	<b>49</b>
<b>Appendix A – Distance Metrics .....</b>	<b>62</b>
<b>Appendix B – Common MST Algorithms.....</b>	<b>64</b>
<b>Appendix C – Data Preparation .....</b>	<b>69</b>
<b>Appendix D – Outliers .....</b>	<b>71</b>
<b>CURRICULUM VITAE.....</b>	<b>80</b>

## **NOMENCLATURE**

### **Acronyms**

AF	Apply Factor
AMST	Adaptive Minimum Spanning Tree
ANOVA	Analysis of Variance
DBSCAN	Density-based Spatial Clustering of Applications with Noise
DMST	Dynamic Minimum Spanning Tree
ESF	External Strength Factor
ES	Euclidian Strategy
EMST	Euclidian-based Minimum Spanning Tree
GDD	Gaussian Density Distance
HEMST	Hierarchical Minimum Spanning Tree
ISF	Internal Strength Factor
IQR	Interquartile Range
KNN	K-Nearest Neighbor
MST	Minimum Spanning Tree
MF	Merge Factor
NNN	Nearest Neighbor Network
RF	Receptivity Factor
SST	Shortest Spanning Tree
SAM	Split and Merge
SMST	Static Minimum Spanning Tree
ULO	Upper Limit Outlier

## LIST OF FIGURES

<b>Figure 2.1</b> Some of the most popular measures used in data clustering .....	12
<b>Figure 3.1</b> Sample data set that consists of 15 data points.....	22
<b>Figure 3.2</b> Fully-connected graph showing the distances between all of the data points .....	22
<b>Figure 3.3</b> Nearest neighbor graph showing the nearest neighbor distance for each data point .....	23
<b>Figure 3.4</b> Determining the critical distance $\lambda$ , which is the maximum distance of all of the nearest neighbor distances .....	23
<b>Figure 3.5</b> Minimum spanning tree graph of the dataset .....	24
<b>Figure 3.6</b> Receptivity factor (RF) value using a minimum spanning tree graph.....	24
<b>Figure 3.7</b> Initial result obtained after cutting the inconsistent edges from the MST graph using the $\lambda_{(data)}$ .....	25
<b>Figure 3.8</b> Merge factor (MF) using the minimum spanning tree graph.....	25
<b>Figure 3.9</b> Flowchart of the proposed algorithm.....	27
<b>Figure 3.10</b> Proposed algorithm as a Pseudo code.....	28
<b>Figure 3.11</b> Clustering results for DS1 and DS2.....	31
<b>Figure 3.12</b> Clustering results for DS3 and DS4.....	31
<b>Figure 3.13</b> Clustering results for DS5 and DS6.....	32
<b>Figure 3.14</b> Clustering results for DS7 and DS8.....	32
<b>Figure 3. 15</b> Clustering results for DS9 and DS10.....	33
<b>Figure 4.1</b> Clustering result and factor table of the 3D synthetic dataset .....	38
<b>Figure 4.2</b> Clustering result and factor table of the 2D synthetic dataset .....	38
<b>Figure 4.3</b> Clustering result and factor table of the experiment related to merging the closest clusters.....	39
<b>Figure 4.4</b> Clustering result and ANOVA graph before eliminating the outliers from the dataset.....	40
<b>Figure 4.5</b> Clustering result and ANOVA graph after eliminating the outliers from the dataset.....	41
<b>Figure 4.6</b> Clustering result and ANOVA graph before eliminating the outliers from the dataset.....	42
<b>Figure 4.7</b> Clustering result and ANOVA graph after eliminating the outliers from the dataset.....	43
<b>Figure 4.8</b> Clustering result and ANOVA graph after eliminating the outliers from the dataset.....	44

<b>Figure 4.9</b> Clustering result and ANOVA graph after eliminating the outliers from the dataset.....	44
<b>Figure B.1</b> Two possible spanning tree on the right and a graph on the left.....	64
<b>Figure B.2</b> Weighted graph on the left and its spanning trees on the right.....	65
<b>Figure B.3</b> Kruskal algorithm.....	66
<b>Figure B.4</b> Prim algorithm .....	67
<b>Figure D.1</b> Example of two-dimensional outliers .....	73
<b>Figure D.2</b> Example of contextual outlier .....	74
<b>Figure D.3</b> Human ECG output showing a collective outlier corresponding to an atrial premature contraction.....	75
<b>Figure D.4</b> Standard graph showing the median, quartiles, and outliers .....	78
<b>Figure D.5</b> Example of boxplot for a given dataset .....	78
<b>Figure D.6</b> Comparing the distributions of two samples using boxplots .....	79

## LIST OF TABLES

<b>Table 1.1</b> Popular clustering algorithms: (A) be sensitive to initial values, (B) time complexity, (C) define the number of clusters automatically, and (D) identify arbitrary shaped clusters. ....	3
<b>Table 2.1</b> Some of the most popular measures used in data clustering.....	12
<b>Table 3.1</b> Mathematical notations used in the algorithm .....	30
<b>Table 3.2</b> Description of the synthetic datasets .....	21
<b>Table 3.3</b> Description of the real datasets .....	30
<b>Table 3.4</b> Results of the purity metric for each algorithm using the synthetic dataset.	33
<b>Table 3.5</b> Results of the purity metric for each algorithm using the real dataset.....	34

# CHAPTER 1

## INTRODUCTION

There are many areas today in which huge amounts of data are produced every second to be processed and stored in sensor networks, cloud storage, social networks, etc. This has improved the fields of statistical data analysis, pattern recognition, machine learning, and data science in general [1]. Although such a volume of data offers enormous opportunities to both industry and academia, it poses challenges for effective retrieval and analysis as well [2]. Data can be compressed into meaningful summaries to alleviate the exponential space and time required for such operations, eliminating the need for data storage. Such summaries are equivalent to “clusters” in the literature on unsupervised learning, promoting better understanding of the data and helping to provide better data visualization. This approach uses methods that combine important aspects of information retrieval, pattern recognition, and machine learning. The main aim achieved in this process is cluster analysis. To group objects, cluster analysis uses only information that defines the objects and their relationships in the data. The goal is to have objects inside a cluster be correlated with each other and be dissimilar from the objects in any other cluster [1]. The bigger the difference between groups, and the larger the similarity inside a group, the better or more distinct the clustering will be. However, there is ambiguity concerning an appropriate clustering similarity metric.

Multiple methods have been proposed in the literature for measuring similarity such as Minkowski measures, Euclidean distance, proximity measures, and data space density that make data clustering become a multi-objective optimization problem. For all similarity measures, for clustering objects in a group, a threshold value should be established and objects that surpass that threshold should be considered as dissimilar and isolated in a different cluster [3]. For grouping data into clusters, various data clustering algorithms have been suggested in the literature. However, there is no general solution for solving all clustering issues as each algorithm has its own biases and is proposed with certain assumptions. In terms of complexity, data clustering is considered as an NP-hard clustering problem and therefore current algorithms use

heuristics or some approximation methods for reducing the search space to obtain a good clustering solution. Moreover, there are not any commonly accepted objective metrics for clustering validity or accuracy; all algorithms have their own advantages and disadvantages in solving difficult problems of data clustering [3, 4]. These clustering algorithms may be classified into different categories of methodologies including density-based, partitioning-based, hierarchically-based, and MST-based algorithms.

## 1.1 Motivation and Research Problem

A good clustering algorithm defined as an algorithm that does not need too many parameters, should be robust, efficient, able to identify arbitrarily shaped clusters, and insensitive to initial values [5]. Table 1.1 provides a summary of some of the most common clustering algorithms. As we mentioned previously, there is no algorithm in the literature that can solve all data clustering problems. This means that a user must choose an appropriate algorithm and the relevant parameters for particular datasets. Normally, though, users do not have a priori knowledge about data sets. This is what is known as clustering dilemma. Multi-objective clustering [6] and clustering ensembles [7–9] are two methods used to mitigate this dilemma to some extent. However, the results related to clustering ensemble methods are generally not unique, and methods related to multi-objective clustering are generally very complicated.

Clustering methods based on minimum spanning tree (MST) can recognize arbitrarily shaped clusters by eliminating inconsistent edges from the MST graph. A major problem posed for all MST-based clustering algorithms by inconsistent edges. These are only the longest edges in the MST graph under ideal conditions. However, outliers often do exist in real-world datasets, which make the longest edges inaccurate cluster separation indicators. Considerable work has recently been carried out to formulate a criterion method to detect inconsistent edges [10–12]. While, there are a variety of clustering algorithms based on the MST, there is no specific method for defining and eliminating inconsistent edges from MST to obtain good clustering results. In addition, some algorithms need some hidden parameters to be specified a priori in order to achieve the optimal result for clustering. To list a few, 2MSTClus [13] is one of the

MST-based data clustering algorithms that utilizes some hidden parameters including number of inconsistent vertices ( $\beta$ ) and validity of the graph-cut ( $\lambda$ ). Another clustering algorithm is called CLICK, which also makes use of one hidden parameter during the adoption phase, which is a similarity threshold for merging sub graphs [14]. Similarly, the nearest neighbor network (NNN) algorithm recognizes clusters using one hidden parameter that is called neighborhood size [15]. Changing these parameters will result in changing the clustering structures and, therefore, the selection of appropriate values for these parameters requires expertise in the domain [10].

**Table 1.1** Popular clustering algorithms: (A) be sensitive to initial values, (B) time complexity, (C) define the number of clusters automatically, and (D) identify arbitrary shaped clusters.

Methods	Parameters	(A)	(B)	(C)	(D)
K-means	K	Yes	$O(nkd)$	No	No
Kernel K-means	K, $\sigma$	Yes	$O(nkd)$	No	Yes
K-Means++	K	Yes	$O(nkd)$	No	No
DBSCAN	Eps, Min Pts	No	$O(n^2)$	Yes	Yes
OPTICS	Eps, Min Pts	Yes	$O(n^2)$	Yes	Yes
NCut	K, $\sigma$	Yes	$O(n^3)$	No	Yes
Fast-MST	K	Yes	$O(n^{3/2} \log(n))$	No	Yes
SEMST	K	No	$O(n \log n)$	No	Yes

In this research, a new MST-based data clustering algorithm is proposed, considering the drawbacks and limitations of the most common clustering algorithms to date. The proposed algorithm solves the main issue of MST-based data clustering which is identifying and removing the inconsistent edges to obtain ideal distribution of the data points within clusters, regardless of the shape of the data distribution. It uses the critical distance method for the identification of inconsistent edges in MST graph. In addition, to obtain clusters, no parameters need to be determined in advance. The algorithm is capable of determining the existence of outliers and adopts the criterion proposed by our previous work [16] for assessing the quality of the clusters. It can be used in a variety of fields, such as e-commerce, health research, image segmentation, communication networks, market research, and many other applications.

## 1.2 Research Objectives and Contributions

The objective of this research was to develop a robust and parameter-free MST-based data clustering algorithm using Euclidean distance as a criterion to represent the similarity between the data points. The proposed algorithm will solve the main issue of MST-based clustering, which is identifying and removing the inconsistent edges in

order to obtain a set of clusters. It begins by constructing MST over the given weighted graph based on Euclidean distance, and then partitions the graph into clusters by removing inconsistent edges using the inconsistent distance methodology. The contribution herein lies in integrating the advantages of both MST and inconsistent distance methodology to identify clusters. The inconsistent distance represents the maximum distance within the clusters in a dataset. The essential contributions of this research are:

1. This study proposes a new MST-based clustering algorithm using critical distance methodology to solve the main issue of MST-based clustering related to identifying inconsistent edges in MST graphs.
2. Irrespective of the type data distribution, the proposed algorithm is able to define the number of clusters for the given dataset.
3. The proposed algorithm can deal with the existence of some outliers and it uses several metrics and analysis of variance (ANOVA) to evaluate and analyze the obtained results.
4. No hyper-parameters need to be specified in advance. The algorithm is flexible, simple, and easy to understand.
5. The algorithm is able to detect clusters with irregular boundaries with different size and different densities.
6. The order of the dataset points has no influence on the final results.

### **1.3 Dissertation Organization**

This dissertation is structured into six chapters:

Chapter 1: An overview of the main motivations, and the research problem statement, as well as the research objectives and contributions are summarized.

Chapter 2: This chapter presents the work in connection with a literature review of MST-based data clustering algorithms and other associated related works.

Chapter 3: This chapter explains the proposed algorithm in detail, describing the steps taken to implement the algorithm including the conducted experimental study. Pseudo code and other supported graphs and figures are also presented.

Chapter 4: The experimental analysis and discussion are described in this chapter, which contains a complete description of the synthetic and real-world databases that have been used to perform the experiments. The evaluation of the obtained results and analysis of the experiments using several scenarios are also presented.

Chapter 5: In the final chapter, the presented research is summarized and concluded with the findings and contributions of the dissertation. A review of the entire research objectives and results are reviewed. This chapter also emphasizes the scientific contributions of this research work. At the end, some prospective points for the future work of this research are presented.

# CHAPTER 2

## LITERATURE REVIEW

This chapter presents recent related research, including theoretical and methodological contributions, as well as substantive findings. Various related works are discussed and examined in this section. Indeed, the research methodology herein is at the crossroads of various research fields. To begin, the concept of clustering is introduced, as is referred to in this dissertation. Then, some terminologies in connection with the algorithm proposed are define. Various research studies relevant to the current discussion were examined. By analyzing various methods, the study explored the advantages and disadvantages of a particular method in order to give a more comprehensive overview of the relevant literature.

### 2.1 The Concept of Clustering

No standard definition of a cluster has been established to date. Even though a cluster was defined by Jain and Dubes [17] as a group of similar objects, they pointed out that different users have different motivations, even for an identical data set, and therefore, an operational definition is difficult to provide. Everitt [18] defined a cluster from the following three viewpoints:

- A cluster is a set of objects that are alike, and objects from different clusters are not the same.
- One cluster is a combination of points in the test space, is such a way that the distance between any point within the cluster and any point outside of the cluster is greater than the distance between any two points within the cluster.
- Clusters may be defined as linked multi-dimensional space regions with the relatively high density of objects, separated by a region that contains a relatively low density of points from other such regions.

The last two definitions of a cluster are more functional than the first. Since a cluster cannot be defined universally and operationally, there are a variety of clustering algorithms in the literature. The terminology of the clustering issue can be defined as follows. For a given data set,  $D = \{d_1, \dots, d_i, \dots, d_N\}$ , where  $d_{ij}$  is the  $j$ th attribute of the  $d_i$  data point, clustering is utilized to find a partition of  $D$ ,  $C = \{C_1, \dots, C_i, \dots, C_K\}$ , where  $K < N$ , such that:

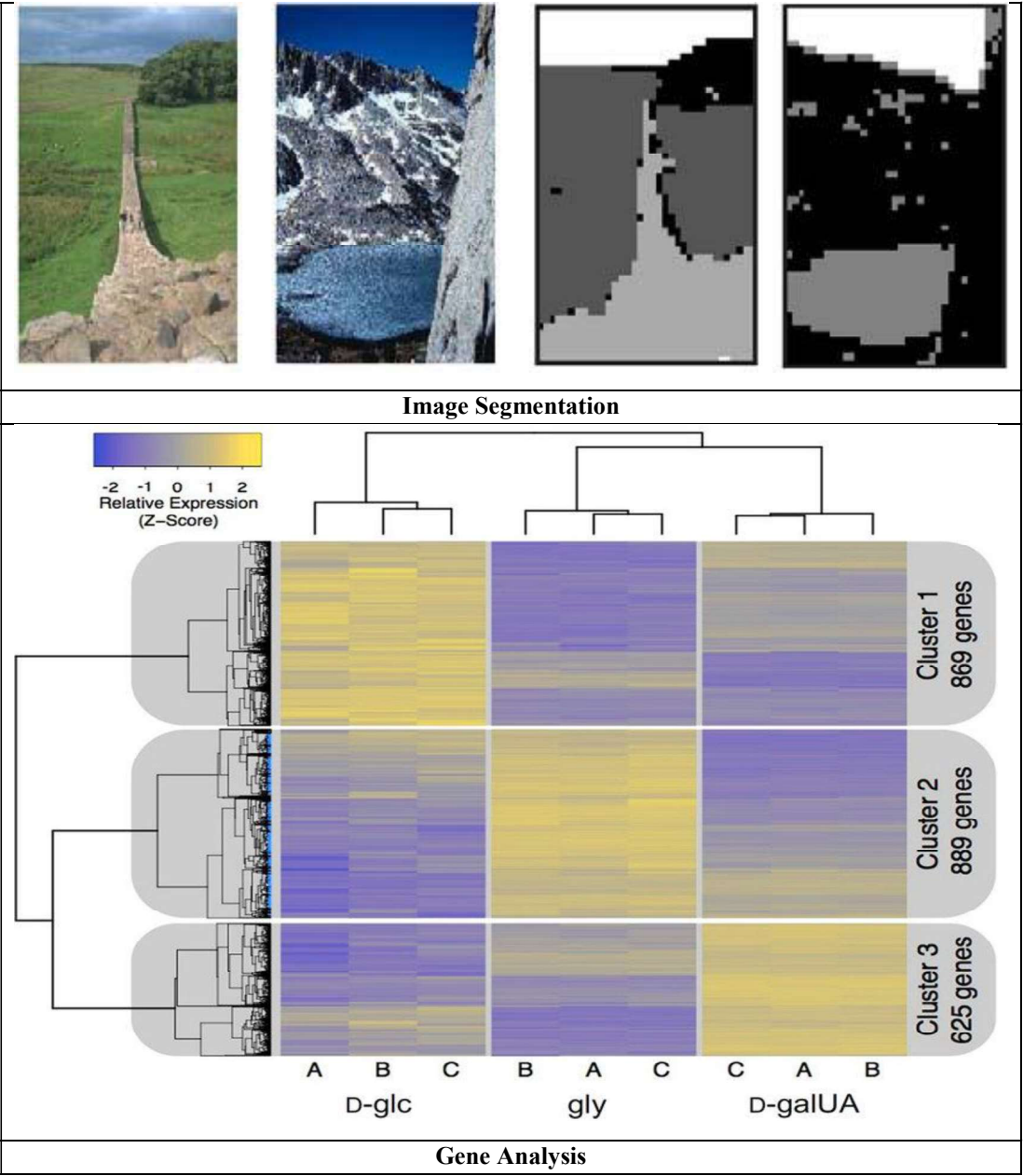
1.  $C_i \neq \emptyset, i = 1, \dots, K$
2.  $\bigcup_{i=1}^K C_i = D$
3.  $C_i \cap C_j = \emptyset, i, j = 1, \dots, K$  and  $i \neq j$

A partition that complies with the above three criteria is called a hard partition, and each data point belongs to a single cluster. Typically, a hard partition is created by a conventional approach of clustering, while a fuzzy clustering approach expands the idea of a hard partition by adding a membership matrix  $U$ , where  $u_{ij}$  element indicates to what extent  $x_i$  belongs to  $C_K$  [19]. The research in this dissertation concentrated solely on hard partitions.

## 2.2 Applications of Clustering

Some data analysis tasks in computer sciences include clustering, such as text analysis [20, 21], speaker diarization [22, 23], and image segmentation [24, 25]. Text analysis is used to divide documents into several groups corresponding to their content. For example, the results of a search engine can be clustered based on a given keyword so that users can see an overview of the results. In the diarization of speakers, the goal is to group speech segments coming from one speaker. Image segmentation is considered as a basic problem in the field of computer vision, where an image is separated into a number of uniform regions. If a region is seen as a cluster, then the segmentation process is considered as a clustering. Clustering is also widely applied in the field of medicine and biology, such as in the diagnosis and treatment of diseases [26, 27], protein function prediction [28], protein structure analysis [29, 30], gene function prediction [31, 32], and gene expression data analysis [33-35]. Moreover, clustering

in astronomy has long been used. The first such associated works were published more than a hundred years ago by the Royal Astronomical Society. Morphology of galaxy clustering [36], separation of stars and galaxies [37], and young stellar clusters [38] are the main applications of clustering in the field of astronomy. Additional applications involve analysis of commercial data, such as financial time series analysis [39, 40], market structure analysis [41, 42], and complex networks [43, 44]. Figure 2.1 presents two applications of data clustering, which are image segmentation and gene analysis.



**Figure 2.1** Applications of Clustering.

## 2.3 Categories of Clustering Algorithms

For organizing data into clusters, a wide variety of clustering algorithms have been proposed. However, there is no common solution to all of the problems related to clustering. There is no consensus on the ‘best’ algorithm, since each one is implemented with certain assumptions and has its own biases. These algorithms can be categorized into methodologies such as partitioning-based, hierarchical-based, density-based, and MST-based.

- **Partition-based clustering:** This group of algorithms is divided into two types: crisp and clustering. In case of crisp clustering, each data point in the dataset fits into only a single cluster [6, 8]. On the other hand, in the fuzzy clustering type, to some extent, each data set point can fit into more than one cluster [5]. In the case of overlapping clusters, fuzzy clustering can deal with boundaries to tackle such issues [6, 7]. Partitional-based clustering approaches are considered as dynamic, and data points may be moved from group to group. In addition, by utilizing the distance measures with suitable prototypes, they can be correlated with the knowledge of the cluster size/shape. The bulk of partition-based clustering algorithms utilize methods of alternating optimization. The iterative design of these techniques makes them highly vulnerable to local minima and sensitive to initialization. The sensitivity to both outliers and noise is also the key downside of partition-based algorithms, and it is difficult to determine the number of clusters [9]. In the community of partition-based clustering algorithms, the K-means algorithm is considered as one of the most popular algorithms [10, 11]. This algorithm has two issues, despite its simplicity and efficiency: first, since it is based on the initialization of the cluster, it cannot find the best result. It could therefore be stuck in local optimal solutions and strongly depend on the initial centroids, which affect its efficiency. The second problem relates to the number of clusters that needs to be identified before [11, 12].
- **Density-based clustering:** These algorithms make use of a multi-resolution grid as a data structure and form clusters using dense grids. The good side of this algorithm is that it does not require any specification criteria in a priori and it is capable of

dealing with datasets that contain large amounts of noise. Moreover, in the case of high-dimensional data, it does not work well [13, 14]. The common and classical density-based clustering algorithm is density-based spatial clustering of applications with noise (DBSCAN), which can discover arbitrary shaped clusters. However, four parameters need to be specified, which are not easy to specify [15]. The clustering in quest (CLIQUE) algorithm puts together density-based and grid-based clustering algorithms, and it performs well on small data sets. However, due to the nature of the rectangular grid, the shapes of the clusters produced by this algorithm are either vertical or horizontal [10]. Grid density-based algorithms are not bothered with data points, but with the value space that contains the data points. In this kind of algorithm, users are required to determine the grid size or the density threshold. The challenge here is how to determine the grid size and density thresholds. However, adaptive grids have been suggested to solve this problem. Thus, depending on the distribution of data, such techniques can automatically determine the size of the grids. Therefore, it is not necessary for the user to specify the density threshold or grid size. This processing time of this algorithm is fast, irrespective of the total number of objects in the dataset. The disadvantage, however, is that the number of cells inside the quantized dimension depends on each dimension [14].

- Hierarchical-based clustering: A hierarchy of clusters is obtained by these algorithms without having to specify the number of clusters beforehand. Therefore, in this case, problems associated with initialization and local minima do not appear [6, 14]. Hierarchical-based clustering methods can also be split up into agglomerative [45–48] and divisive [19, 49–52] methods. A divisive algorithm starts by considering the entire data set as one cluster, and then repeatedly splits up the large clusters into smaller ones, until each data point in the dataset fits into a single cluster. The divisive analysis (DIANA) clustering algorithm [21] is an example of divisive type of hierarchical clustering algorithm. On the other hand, the clustering is performed opposite that of agglomerative of clustering. It takes a data point from the dataset and considers it as a cluster, and then merges similar pairs of clusters iteratively until the specified number of clusters is reached. Balanced iterative reducing and clustering using hierarchies (BIRCH) [53],

Chameleon [20], complete linkage clustering [54] robust clustering using links (ROCK) [55], and single clustering using representatives (CURE) [56] are all examples of this type of hierarchical algorithm.

- **MST-based clustering:** The MST-based clustering algorithm is considered as a graph-based algorithm (named an unrooted tree) that incorporates a closely connected set of nodes, which can be formed as a cluster. In particular, clustering algorithms that use MST have attracted interest over the past few years, as they have the ability to discover the arbitrary shapes of clusters [57]. MST is a famous combinatorial optimization problem that is effectively utilized in different tasks, such as cluster analysis and image segmentation [58, 59]. A spanning tree can be defined as undirected, weighted, and acyclic subgraph. When a graph contains only one path (i.e. there are no cycles or loops in the graph) among all of the pairs of vertices, then the graph is considered as acyclic. If any edge in the graph has oriented to another edge, then the graph is considered as a directed graph; otherwise, it is considered as undirected graph. A MST, or as it is sometimes called, a shortest spanning tree, is a spanning tree that has the lowest total weight of all of the edges when compared with all of the other possible spanning trees [60–62]. The Boruvka [63], Kruskal [64], and Prim [65] algorithms are the most common MST extracting algorithms. Thus, in order to obtain a set of clusters in MST-based clustering, the main concept is to define and eliminate the inconsistent edges from a MST graph. It starts by creating a MST for a given weighted graph and then divides it into clusters by eliminating inconsistent edges [10]. Hence, one of the key problems to be solved in this research, was identifying the inconsistent edges for a given MST graph. In cluster construction, it is not easy to attain clusters using MST [66, 67], since all the nodes in an MST are connected with each other to form a connected graph, which is considered as a single cluster. Several clustering methods using MST were employed to resolve this issue, by restricting the growth of a MST using a certain threshold weight. This enables strongly connected clusters to be created. Moreover, by reorganizing their connections inside of the tree, these MST-based constructed clusters can be better adjusted [66]. Reorganization requests, however, are conducted in line according to the actual contacts in the graph. This should be done gradually and tested to enhance the

quality of the cluster before acceptance. MST-based clustering algorithms are elegant and appropriate for high-dimensional and modern large datasets where only the data point,  $N$ , is provided. However, a quadratic execution time is required to construct a MST. Furthermore, outliers often exist in real-world case studies, and the clusters may have different densities, which makes the longest edges an unreliable measure for separating the clusters. There is one advantage of MST-based clustering, which is considering the distances between data points when clustering. However, cluster separations and thus, the clustering quality, may be reduced by the existence of outliers [10].

## 2.4 Typical Distance Measures

A distance measure, or alternatively a similarity/dissimilarity measure, is used in machine learning and pattern recognition to measure how similar a pair of data points is. In semi-supervised and supervised learning, the distance metric can be learned from the labeled samples. Weinberger and Saul, for example, demonstrated how to learn a Mahalanobis distance metric for K-nearest neighbor (KNN) classification using labeled examples [56]. However, no labeled example is available in data clustering applications, and it is crucial for a clustering to select a suitable distance metric. The most typical measures are shown in Table 2.1. More details about the various measures used in data clustering are provided in Appendix A.

**Table 2.1** Some of the most popular measures used in data clustering.

Distance Metric	Formula	Description
<b>Euclidean</b>	$D(x_i, x_j) = \left( \sum_{l=1}^d  x_{il} - x_{jl} ^2 \right)^{1/2}$	$d$ represents the dimensionality. This metric is the most used measure in data clustering.
<b>Minkowski</b>	$D(x_i, x_j) = \left( \sum_{l=1}^d  x_{il} - x_{jl} ^p \right)^{1/p}$	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].
<b>Manhattan</b>	$D(x_i, x_j) = \sum_{l=0}^d  x_{il} - x_{jl} $	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].

## 2.5 Related Work

Recently, graph-based clustering algorithms have become common because they look for connectivity-based partitions rather than centroids. MST has an important role to play in analyzing the topological and dynamical features of complex networks [68]. For a given weighted graph, it is considered as the main transport backbone as recognized in many studies [57]. In 1971, Zahn [69] firstly proposed the MST-based clustering method by constructing the MST over a weighted graph, which formed the connected components, and then proceeded to eliminate inconsistent edges from the MST graph. One drawback of this algorithm is that it is vulnerable to outliers. Thus, an object that is isolated from all other objects forms a single group. However, Chowdbury and Murthy [70] suggested an inconsistency measure dependent on finding valley regions to fix this issue. Another solution for dealing with the existence of outliers was proposed by Laszlo and Mukherjee [71], where they specified a restriction on the minimum size of a cluster. Clustering algorithms based on MST are usually divided into three phases: 1) building the MST, 2) eliminating inconsistent edges from the MST graph to create a collection of connected components, and 3) repeating phase 2 until satisfying the terminating condition. Since the MST-based clustering approach was initially proposed by Zahn, more recent works have concentrated on identifying inconsistent edges of MST graphs. The inconsistent edges are the longest edges in the ideal condition where no outliers are present and clusters are well separated. However, in the case of the existence of outliers in the dataset, the longest edge does not necessarily correspond to the inconsistent edge [5]. Moreover, MST can be used for representing multidimensional gene expression data, such as in the study presented by Xu et al. [72]. Three objective functions were defined in their study. The first objective function was used to eliminate the  $k-1$  longest edges, thereby minimizing the overall weight of the  $K$  subtrees. The second algorithm was proposed in order to reduce the total distance between each point in a group with respect to the center. The third objective function was utilized to reduce the total distance between each point in the cluster with respect to the cluster's "representative". When inconsistent edges are eliminated according to lengths of edges, clustering results will be vulnerable to outliers. In the study of Grygorash et al. [73], a hierarchical MST-based clustering approach (HEMST) was proposed by Grygorash et al., that repeatedly

cuts the edges, fuses points within the resulting connected components, and then recreates the MST again. Besides the inconsistent edges, the concept of points density is another main factor influencing the efficiency of clustering results. In order to partition a given dataset, the conventional MST-based clustering algorithms use merely the information related to the edges within the tree, which makes these algorithms easily affected by the existence of outliers. More recently, MST-based techniques aim to define inconsistent edges depending on local densities around objects. Some approaches describe the density of points using the degrees of the vertexes. Luo et al., for example, proposed a clustering algorithm using neighborhood density difference estimation based on MST [74]. Wang et al. proposed to find a local density factor for each data point during the construction of an MST and discarding outliers [10].

In fact, clustering algorithms based on MST have been used successfully to detect clusters of varying sizes and shapes. For example, Zhong et al. [13] presented a graph-theoretical clustering method, which is robust to the difference of cluster sizes, densities and shapes. Based on the graph composed of two rounds of MST, the proposed method (2-MSTClus) classified cluster problems into two classes, as touching cluster problems and separated cluster problems, and identified the two classes of cluster problems automatically. Later, they proposed a new method based on split and merge clustering [75]. The algorithm used an MST graph for the initial prototypes and used the K-means algorithm to split the graph into sub-graphs [83]. Next, by considering the neighboring edges, the sub-graphs were filtered and merged. Some MST-based algorithms have been combined with other approaches, such as multivariate Gaussians [76], information theory [77], and k-means [25].

Zhou et al. [78] proposed another type of MST-based clustering algorithm, which they referred to as the adaptive MST-based clustering algorithm (AMST). They focused on the extraction of irregular shapes of clusters. The algorithm first uses a validity indicator for determining the ideal number of clusters. The candidate clusters are then assessed for their significance. Both data isolation and compactness are considered in the validity indicator. This will lead to specifying the best part of the MST for clustering. The proposed method was compared with the dynamic MST (DMST) and static MST (SMST) clustering methods. The DMST and SMST methods extract a

single cluster from the data when compared to the multiple clusters formed by the AMST. The AMST algorithm shows better accuracy, which has been supported by simulations when compared to the DMST and SMST. In addition to algorithms that use inconsistent edges to split up datasets into clusters, there are also other algorithms that utilize the MST to calculate pairwise dissimilarity, such as the minimax distance suggested by Fischer and Buhmann [79, 80]. Another MST-based clustering method was proposed by Halim and Uzma [66]. Their method uses evolution strategy (ES) to refine the obtained clusters. This technique relies on (1+1)-ES for obtaining the optimal MST-based clustering result. The Davies-Bouldin index serves as a fitness function in determining the efficiency of clusters obtained by ES populations. The proposed algorithm was evaluated using 11 benchmark datasets.

cciMST is a new MST-based clustering algorithm proposed by Lv et al. [58] using an algorithm for cluster center initialization. On the basis of dual densities of points and geodesic distance, cluster center initialization is used as algorithm in order to capture the intrinsic structure of the data sets. Moreover, a method using inconsistent edges located on shortest paths between cluster centers is used in order to specify the inconsistent edges using the densities of endpoints of the edges together with the lengths of the edges on the shortest paths. Third, by measuring the distance between points at the intersections of clusters, the authors suggested a novel inter-cluster separation method. In addition, to select the best clustering outcome, a novel internal clustering validation metric was suggested. By constructing Euclidean-based MST (EMST), Lopresti et.al. [81] suggested a red, green, and blue (RGB) color clustering algorithm. Each distinct color in the given data is considered as a point in the three-dimensional RGB color space. Thus, in the EMST, each color is considered as a node. The Euclidean distance between two color nodes in the tree is considered as the weight of an edge. They compute the average distance of the edges in the EMST once it is built. Subsequently, to form a set of disjoint subtrees, the edges that are longer than the average weight by a predetermined amount are removed from the tree. The members of a color cluster are the colors of each sub-tree. They indicated that when dealing with textures and when there are numerous colors in the data, the color clustering algorithm based on the EMST may fail. In the context of specifying the number of clusters, several approaches have been introduced in the literature to

determine the right number of clusters. Some of these methods include the elbow (Thorndike, 1953) method [82], core matrix (Honarkhah & Caers, 2010) [83], and criterion of the information (Sugar & James, 2003) [84]. As mentioned previously, because of the difficulties in selecting the input parameters or because models may not be appropriate for detecting the structures of clusters, the traditional approaches of data clustering algorithms will not be appropriate for clustering a heterogeneous forms of clusters. The majority of these algorithms often need quadratic time to do clustering of  $N$  (number of data points). Thus, for large sizes of datasets, the high computational cost of these methods limits their applications [20, 75]. To address these problems, a variety of hybrid clustering algorithms were suggested incorporating the advantages of these approaches. Datasets are analyzed in two phases using a hybrid clustering algorithm. In the first stage, the data set is divided into numerous sub clusters using various partitioning methods. The formed sub clusters are combined to actual clusters in the second stage on the basis of some merging criteria. Different hybrid clustering algorithms have been proposed in the literature based on the various partition and merging methods [20, 31, 55, 75, 85]. A hybrid CURE algorithm to handle large datasets was proposed to take advantage of both the partition-based and single link techniques [55]. The CURE method represents a cluster by a constant number of representative points and the similarity of two sub-clusters are determined using the closest distance between the representative points in different sub-clusters. The selection of the representative points of CURE depends upon the existence of outliers in the dataset, in addition to the shapes and sizes of the clusters; therefore, the results of the clustering also depend on the selection of the representative points [20, 86]. This algorithm has a computational complexity of  $O(N^2 \log N)$ . An efficient and robust data clustering with cohesion self-merging (CSM) [86] is a hybrid clustering approach that first uses k-means to partition the dataset into  $k$  sub-clusters and then iteratively merge them into actual clusters based on cohesion. The final results of the clustering may be unstable because the algorithm depends on k-means as a partitioning approach, which produces different clusters in different runs [75]. Split and merge using multiple prototypes with small separations was proposed by Liu et al. [31] to discover clusters of arbitrary shape and size. However, the clustering results of this approach depends on many experimental parameters [75].

Another example of hybrid clustering algorithm is CHAMELEON [20], which can be suitable for clustering heterogeneous datasets. It constructs a graph using the  $k$ -nearest neighborhood and then applies a graph partition technique to partition the dataset into several sub-clusters. The resulting sub-clusters are then merged into actual clusters depending on the relative inter-connectivity and closeness of the sub-clusters.  $O(N^2)$  is the computational complexity of this hybrid algorithm, because it constructs the  $k$ -nearest neighborhood graph on a complete graph, which limits its application in large datasets. Preprocessing is also done to choose the parameters needed to construct the graph of the nearest neighborhood [75]. For the construction of an approximate MST, an MST-based clustering method was proposed using the divide-and-conquer approach (DMST) [87]. It has the capability of identifying the longest edges in the early stage of clustering using the cut-and-cycle properties of a graph, which may not participate in the construction of the MST in order to prevent additional computations. However, the worst case complexity of the algorithm still remains  $O(N^2)$ , despite its sub-quadratic complexity in the average case. The clustering quality is also affected because during construction of the approximate MST, many of the edges are not considered [57, 88].

Another example of the hybrid clustering algorithm is split-and-merge (SAM). It builds a neighborhood graph of the MST to represent the dataset [73, 75]. It applies the  $k$ -means algorithm after selecting the  $\sqrt{N}$  highest degree nodes as the cluster center to split the dataset into large number of sub-clusters. In order to merge the sub-clusters into the actual clusters, the intra-similarity and inter-connectivity between the sub-clusters are designed. The computational complexity of the algorithm is  $O(N^2)$ , which is dominated by the construction of the graph. The datasets that contain clusters with a different dispersion level and diverse shape cannot be clustered using the SAM algorithm [89]. A parameter-free clustering technique called Gaussian density distance (GDD) was proposed using Gaussian kernel and distance, by considering both the shape and data density to obtain the clusters [90]. The algorithm works with a computational complexity of  $O(N^2)$ . The technique is not sufficient to be used for data sets that contain touching clusters. MST is represented as a useful graph structure that can be utilized to discover neighborhood information for data points. Thus, MST can be used to represent the underlying structures of heterogeneous types of datasets. Since

each pair of data points will be attached to an edge in the MST graph,  $O(N)$  is the cost of locating the nearest neighbor of any given data point. Thus, for  $O(N)$  data points,  $O(N^2)$  is the overall cost for building the MST graph. If the structure of the dataset could be represented by the local neighborhood graph rather than the complete graph, time complexity could be reasonably minimized. With this inspiration, Mishra and Mohanty [91] suggested a hybrid and fast MST-based clustering method using the local nearest neighbor to reduce the computational overhead of using complete MST graphs. As the initial step, to capture the geometry of clusters and based on the dispersion of data points, the datasets are to be divided into large numbers of sub-clusters. After that partitioning, the MST graph is built to recognize the adjacent pairs depending on the centroids of all sub-clusters. Furthermore, in order to find genuine clusters, a novel merging technique was presented by combining the adjacent sub-clusters repeatedly. With respect to average edge weights of sub-clusters and centers of adjacent pairs, two methods were offered to calculate the level of dispersion of each data point, namely intra-similarity and cohesion.

# CHAPTER 3

## THE PROPOSED MST-BASED DATA CLUSTERING ALGORITHM

The most problem in data clustering field is that we do not have prior knowledge about the given dataset. Moreover, the choice of input parameters such as the number of clusters, number of nearest neighbors and other factors for each clustering algorithm make the clustering more challengeable topic. Thus, any incorrect choice of these parameters yields bad clustering results. Furthermore, these algorithms suffer from unsatisfactory accuracy when the dataset contains clusters with different complex shapes, densities, sizes, noise and outliers. MST-based clustering algorithms are elegant and appropriate for high-dimensional and modern large datasets where only the data points  $N$  is provided. However, a quadratic execution time is required to construct MST. Furthermore, outliers often exist in real world case studies, and clusters may have different densities, which makes the longest edges an unreliable measure for separating the clusters. There is one advantage of MST-based clustering which is considering the distances between data points when clustering. However, cluster separations and thus the clustering quality may be reduced by the existence of outliers [10]. This chapter presents the proposed algorithm that is used to overcome the problems and challenges presented in the first chapter. The proposed algorithm tries to identify the inconsistent edges in MST graph by using the critical distance methodology. The Euclidian distance is used as the distance measure between any two vertices in the MST graph. The critical distance is the maximum distance of the nearest neighbors' distances in MST graph. It is used as a threshold to identify inconsistent edges that will be cut from MST graph to obtain the final clustering result. The proposed algorithm is simple, easy to understand, and free from any hyper-parameters that need to be specified by the user. The algorithm is discussed in detail in the following sections.

### 3.1 The Proposed Clustering Algorithm

On the basis of the principle of data clustering, to obtain ideal clusters, we must ensure that distances between any two objects within a cluster are smaller than distances between any two objects within separate clusters. Based on this definition, and using the MST graph, the proposed algorithm can efficiently define the inconsistent edges by utilizing critical distance as a threshold to get optimal clusters. The critical distance is defined as the maximum distance of nearest neighbors in the MST graph. We call this “critical distance” because it is the maximum distance inside clusters and any distance greater than the critical distance is considered as a distance between two clusters. Table 3.1 defines all mathematical notations used in the proposed algorithm.

**Definition 1.** (Concept of Clustering) In general, for a given dataset  $X$ , a clustering algorithm seeks to group  $X$  within several different clusters,  $C_1, C_2, \dots, C_K, C_i \neq \emptyset, C_i \cap C_j = \emptyset, X = C_1 \cup C_2 \dots \cup C_K, i, j = 1:K, i \neq j$ , so that the distance between any object in the cluster with respect to any object outside the cluster is greater than the distance between any closest pair of objects within a cluster.

**Definition 2.** (Fully connected graph) Given a dataset  $X = \{x_1, x_2, \dots, x_n\}$ ,  $G(X) = (V, E)$  represents a weighted, fully connected graph that consists of an edge set  $E = \{e_{ij} = (x_i, x_j) \mid x_i, x_j \in X, i \neq j\}$  and a vertex set  $V = X$  (i.e., a set containing  $T$  data points). The weight of each edge  $e_{ij}$  is denoted as  $W(x_i, x_j)$ .

**Definition 3.** (Nearest neighbor graph) In a metric space, let  $X$  be a set of objects.  $G_S$  is denoted as the nearest neighbor graph of  $X$ . It consists of  $|X|$  vertices, each corresponding to an object in  $X$ , and if  $a \in X$  is the nearest object to  $b \in X$  in  $X$ ,  $G_S$  contains an edge that connects the corresponding vertices of  $a$  and  $b$ .

**Definition 4.** (MST) Assuming that  $G(X) = (V, E)$ , which represents a weighted connected graph, with an edge set  $E = \{e_{ij} = (x_i, x_j) \mid x_i, x_j \in X, i \neq j\}$  and a vertex set  $V = X$  (i.e., a set containing  $T$  data points), the weight of each edge  $e_{ij}$  in graph  $G(X)$  is denoted as  $w(x_i, x_j)$ .  $W(\text{MST}) = \min \{\sum_{x_i, x_j \in V, e_{ij} \in E} W(x_i, x_j)\}$  represents the MST of graph  $G(X)$ , defined as a subset of  $E$  that links all vertices in  $V$  with a minimum total sum of weights and does not have a cycle. It fulfills following two conditions:

(1) Cycle property: In any cycle in a graph, the edge with the largest weight does not belong to the MST.

(2) Cut property: an edge must belong to an MST if it has the smallest weight and cross any two partitions belonging to the vertex set.

If each edge is given a weight (i.e.,  $w$ ) that denotes a distance (i.e.,  $\rho$ ) between two vertices, each edge in the MST will be the shortest distance between the two subtrees that are connected by that edge. Thus, theoretically, eliminating the longest (i.e., inconsistent) edges would allow the obtaining of some clusters.

**Definition 5.** (MST-Based clustering) Let MSTT denote the MST of a given dataset  $T$ . Let  $S_m, S_n$  be two subsets of  $T$  such that  $S_m, S_n \in T$ ,  $S_m \cap S_n = \emptyset$  and  $S_m \neq \emptyset, S_n \neq \emptyset$ .  $S_m$  and  $S_n$  are considered MST-based clusters in the event that they were created by eliminating the link that has an edge weight greater than critical distance  $\lambda_{(data)}$ .

**Definition 6.** (Critical distance  $\lambda_{(data)}$ ) For a given nearest neighbor graph  $G_S$ , the critical distance  $\lambda_{(data)}$  is defined as the maximum distance (weight) compared with all distances (weights) of nearest neighbor edges in  $G_S$ .

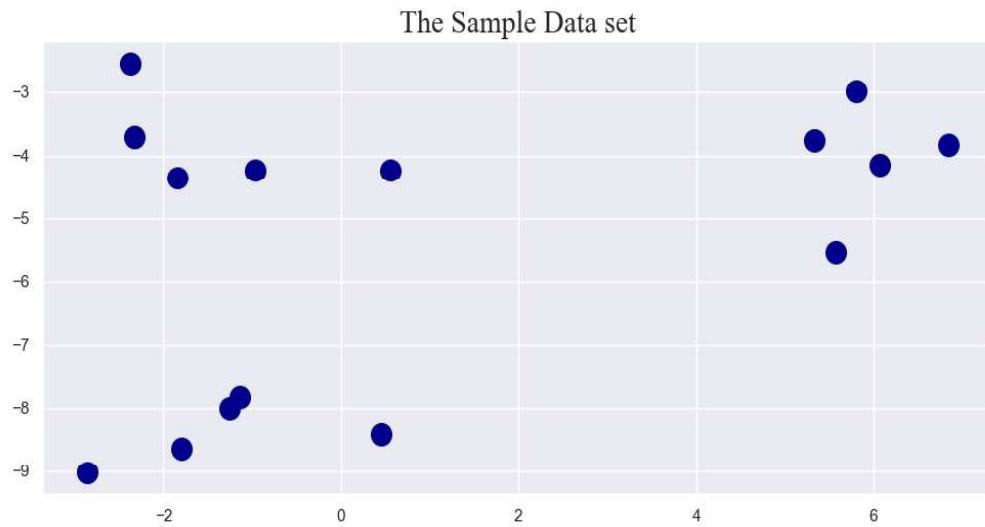
**Definition 7.** (Inconsistent edge) In a given MST where  $T=(V, E_T)$  for data set  $X$ , given  $e_{ab} \in E_T$  linking an end point  $a \in A$  to an end point  $b \in B$ ,  $e_{ab}$  is considered as an inconsistent edge and needs to be removed from the MST if the ratio (the weight of edge  $e_{ab}$ ) is larger than a given threshold which we call it ( $\lambda_{(data)}$ ).

**Table 3.1** Mathematical notations used in the algorithm

Notation	Description
$X$	$X \in R^p$ denotes the dataset list
FC_Graph	Denotes the fully connected graph that contains pairwise distances for each data point
NN_Graph	Denotes the nearest neighbor graph that contains all the nearest neighbor distances of data points
$\lambda_{(data)}$	Critical distance with respect to data that represents the maximum distance in NN_M
$\max\_d_{MST}$	Denotes the maximum weight in MST_Graph
RF	Denotes the receptivity factor, which is utilized to check the receptivity of the dataset for clustering
ULO	Upper Limit Outlier is considered as a threshold that specifies the starting point of extremism
Q1	Q1 represents the median for the lower half of the data
Q3	Q3 represents the median for the upper half of the data
IQR	Denotes Interquartile Range which is calculated by the formula $(Q3-Q1)$
$\min\_d_{(2c,all)}$	Denotes the minimum distance between two clusters with respect to others
MF	MF is the Merge Factor which is used to determine the necessity of merging the closest clusters
$\lambda_{(c,k)}$	Denotes the critical distance inside cluster $k$
$\text{near\_d}_{(c,k-n_k)}$	Denotes the nearest distance for cluster $k$ with respect to nearest cluster $n_k$
$ISF_{(c,k)}$	Internal Strength Factor for cluster $k$ , which is utilized to identify the cluster to be merged

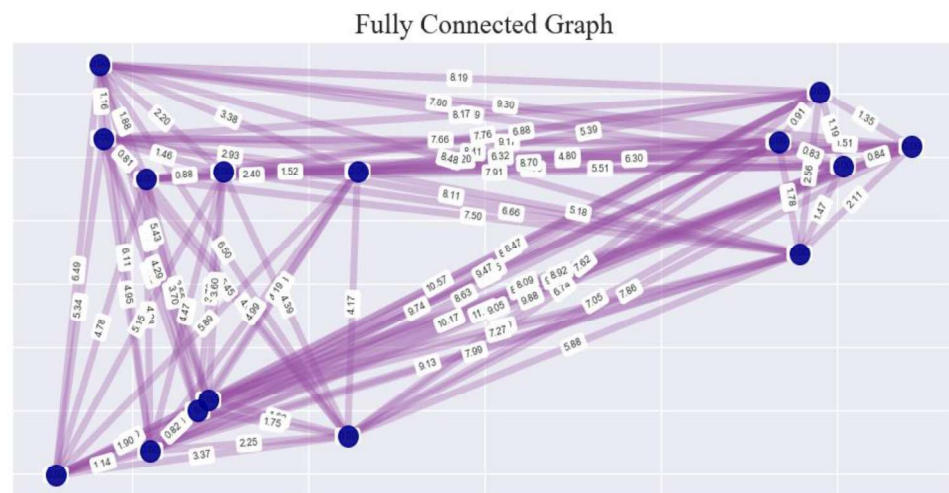
The procedure for clustering a sample dataset using the proposed algorithm is defined in several steps as follows:

- A sample of 15 data points from a synthetic dataset has been created to simplify explaining the proposed algorithm as shown in Figure 3.1. The synthetic dataset is free of outliers and it consists of three well-separated clusters.



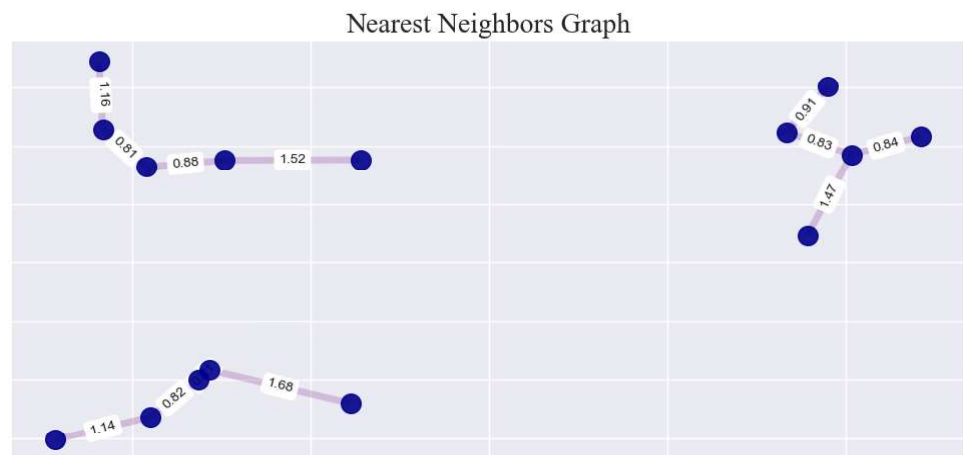
**Figure 3.1** Sample data set that consists of 15 data points

- The first step is to find the fully connected graph FC\_Graph which contains the distances for each object in the dataset with respect to the others, using the Euclidian distance formula as shown in Figure 3.2.



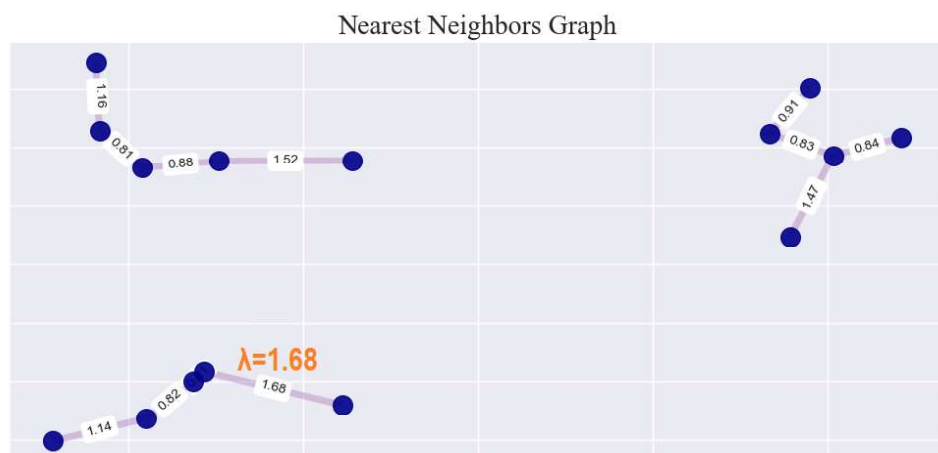
**Figure 3.2** Fully connected graph showing the distances between all of the data points

- The second step is to find the NN\_Graph which contains the nearest neighbor for each object in the dataset. Figure 3.3 represents the nearest neighbor graph.



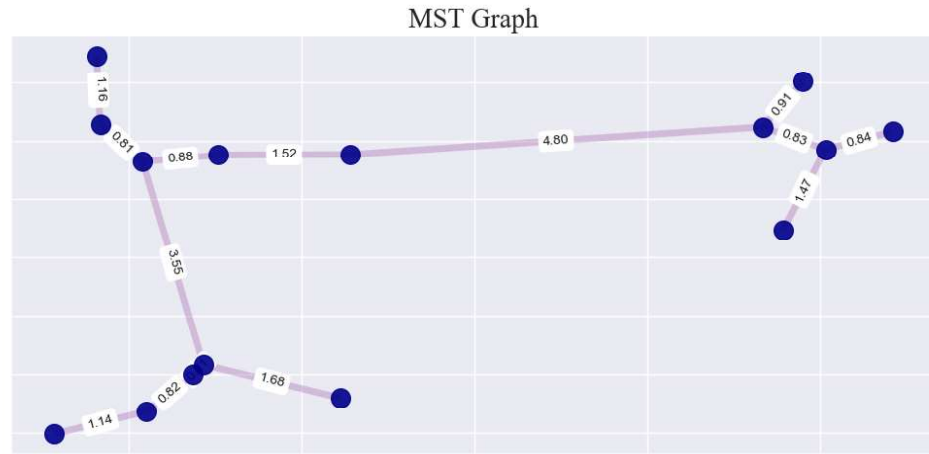
**Figure 3.3** Nearest neighbor graph showing the nearest neighbor distance for every data point

- The next step is to identify the critical distance from NN\_Graph which denotes the maximum distance taken from NN\_Graph as shown in Figure 3.4.



**Figure 3.4** Determining the critical distance  $\lambda$ , which is the maximum distance of all of the nearest neighbor distances

- In order to separate the clusters using the MST, we need to build the graph related to MST (MST\_Graph) using the nearest neighbor distances between the objects in the dataset as weights. Figure 3.5 shows the MST graph of the sample data set.

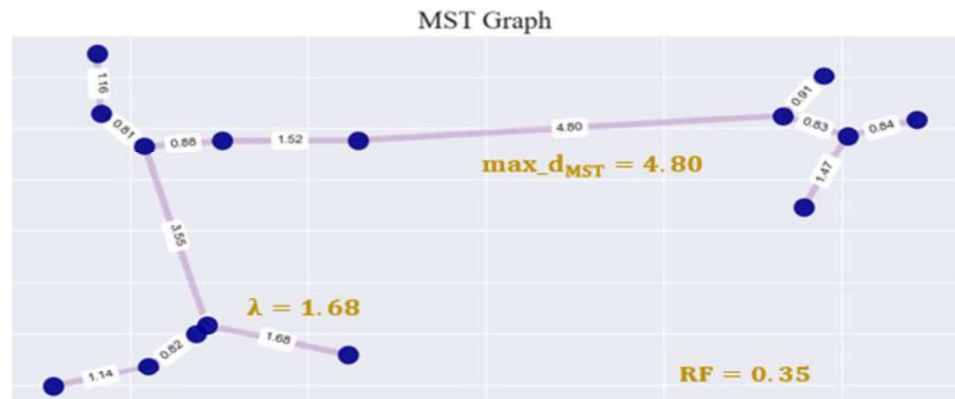


**Figure 3.5** Minimum spanning tree graph of the dataset

- After building MST\_Graph, we need to identify the maximum distance in MST\_Graph in order to calculate the receptivity factor (RF) which is utilized to check the receptivity of the dataset for clustering. The formula for RF is as the follows:

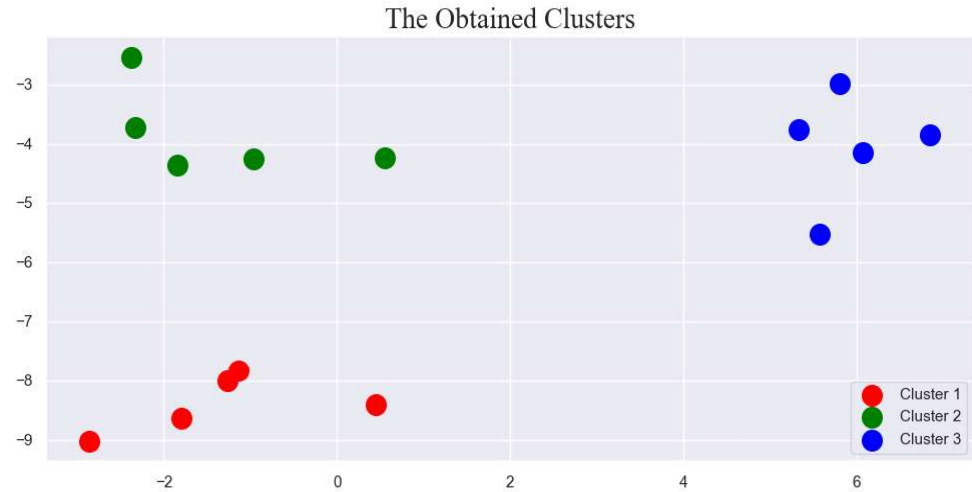
$$\mathbf{RF} = \frac{\lambda_{(\text{data})}}{\max\_d_{\text{MST}}} \quad (3.1)$$

- Thus, by comparing both the maximum distance in the MST graph which is  $\max\_d_{\text{MST}}$  with the maximum distance in the nearest neighbors graph which is  $\lambda_{(\text{data})}$ , using the RF, if  $\text{RF}=1$ , the distances are equal, and there are some outlier distances that leads to make the dataset to not have the receptivity for clustering. This case will be discussed in detail in chapter 4. However, based on the sample dataset presented in Figure 3.6 and after building MST\_Graph, the value of RF is  $\text{RF}=0.35$ . It means the dataset has receptivity for clustering in general and it can be clustered.



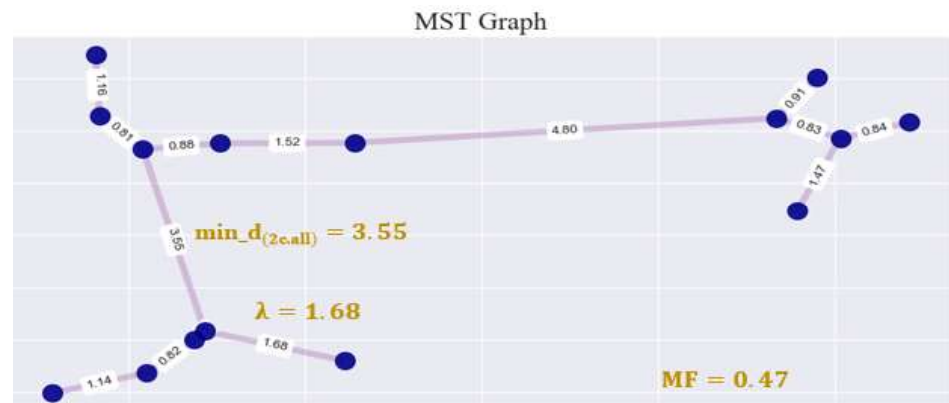
**Figure 3.6** Receptivity factor (RF) value using the MST graph

- Then, using the MST\_Graph, we need to filter out all the distances that are greater than critical distance and assign a zero value to them.
- The next step is to find the connected components from the MST\_Graph based on the inconsistent edges in MST\_Graph where the distance is equal to zero by assigning a label for each series of connected objects. The initial obtained result is shown in Figure 3.7.



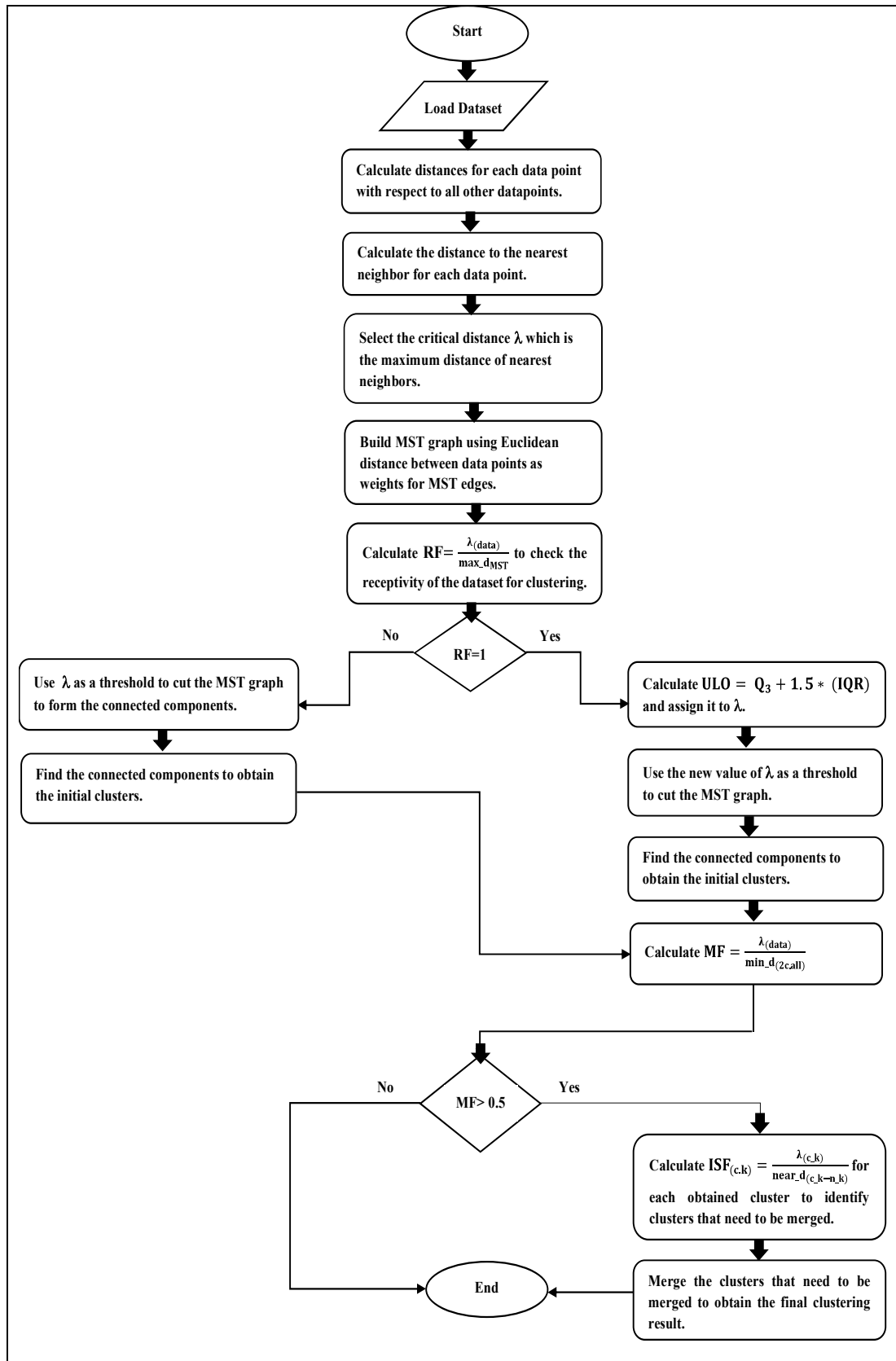
**Figure 3.7** Initial result obtained after cutting the inconsistent edges from the MST graph using the  $\lambda_{(data)}$

- After obtaining the initial clustering result, we need to test the necessity of merging the closest clusters by calculating the merge factor (MF). As shown in Figure 3.8, the value of this factor is  $MF=0.47$ , which means that there is no need for merging clusters in the obtained result. Thus, the initial obtained result is considered the final clustering result. The formula of MF is the following: 
$$MF = \frac{\lambda_{(data)}}{\min_{d(2c.all)}} \quad (3.2)$$



**Figure 3.8** Merge factor (MF) using the minimum spanning tree graph

According to the flowchart and pseudo code presented in figure 3.7 and 3.8 respectively, the proposed algorithm started by loading the dataset and calculating the distances for each data point to form the fully connected graph. Next step is to calculate the distance of the nearest neighbor for each data point using the fully connected graph. From the nearest neighbor graph the critical distances is determined which represents the maximum distance among the nearest neighbor distances. The next step is to build the MST using the nearest neighbor graph. Then, the receptivity factor (RF) needs to be calculated to check the receptivity of the dataset for clustering. If the value of RF is equal to one, this means that the dataset does not have receptivity for clustering due to the existence of some outliers. In this case, the critical distance needs to be replaced by upper limit outlier (ULO) which represents the extremism threshold for a given dataset. The next step is to use ULO as a threshold to cut the MST graph and obtain the initial clustering result. Then, we need to check the necessity of merging closest clusters using merge factor (MF). If the value of MF is greater than 0.5, this gives a general indication that there is a need for merging some closest clusters. Else, there is no need to merge clusters and the initial clustering result can be considered as the final clustering result that can be obtained using the proposed algorithm. In the case where MF is greater than 0.5, to specify the clusters that need to be merged, Internal strength factor ( $ISF_{(c,k)}$ ) needs to be calculated for each cluster to determine clusters need to be merged. Cluster that has the maximum value of  $ISF_{(c,k)}$  indicates that, this cluster needs to be merged with the nearest cluster. After merging the closest clusters, the obtained clusters considered as the final clustering result that can be obtained using the proposed algorithm. On the other hand, when the value of RF is not equal to one, this means the dataset does have a receptivity for clustering. It means that, the critical distance can be used as a threshold to cut the MST graph and obtain the initial clustering result. Then, MF needs to be calculated to check the necessity of merging closest clusters as it is discussed previously in the case where RF equals to one.



**Figure 3.9** Flowchart of the proposed algorithm.

---

**Input:** Data set  $X = \{x_i, x_i \in R^p, i = 1, 2, \dots, n\}$ .

**Output:** Connected components that form the obtained clusters.

---

```

1- Load the input dataset X
2- Calculate the pairwise distances in the form of fully connected graph:
   FC_Graph=Find_FC_Graph(X)
3- Find the nearest neighbor graph using FC_Graph: NN_Graph= Find_NN_Graph(FC_Graph)
4- Select the critical distance which is  $\lambda_{(data)} = \text{Max\_weight}(\text{NN\_Graph})$ 
5- Build the MST_Graph using FC_Graph: Build_MST(FC_Graph)
6- Calculate  $\text{RF} = \frac{\lambda_{(data)}}{\text{max\_d}_{\text{MST}}}$ 
7- if RF==1 then
8-     ULO =  $Q_3 + 1.5 * (\text{IQR})$ 
9-      $\lambda_{(data)} = \text{ULO}$ 
10- end if
11-   for each distance in MST_Graph do the following:
12-       if MST_Graph.distance[i] >  $\lambda_{(data)}$  then
13-           MST_Graph.distance[i]=0
14-       end if
15-   end for
16-   Find n_components, labels= connected_components(MST_Graph)
17-   Calculate  $\text{MF} = \frac{\lambda_{(data)}}{\text{min\_d}_{(2c,all)}}$ 
18-   if MF > 0.5 then
19-       for each connected component obtained from Step 16:
20-           Calculate  $\text{ISF}_{(c,k)} = \frac{\lambda_{(c,k)}}{\text{near\_d}_{(c,k-n_k)}}$ 
21-       end for
22-       Call MergeNearestClusters( $\text{ISF}_{(c,k)}$ )
23-   end if
24-   else:
25-       Terminate
26-   else:
27-       for each distance in MST_Graph do the following:
28-           if MST_Graph.distance[i] >  $\lambda_{(data)}$  then
29-               MST_Graph.distance[i]=0
30-           end if
31-       n_components, labels= connected_components(MST_Graph)
32-       Goto step 18

```

---

**Figure 3.10** Proposed algorithm as a Pseudo code.

## 3.2 Experimental Study

### 3.2.1 Experimental Setup

We evaluate the proposed algorithm using 10 synthetic data sets (DS1-DS10) and 9 real-world datasets (DSR1-DSR9). The datasets used for evaluation have different characteristic and shapes. DS1-DS10 are created synthetically and taken from the literature [6, 92–95]. These datasets are carefully chosen to cover several cluster characteristics in terms of dispersion, geometry, and density. The nine real datasets are adopted from the UCI machine learning repository [96]. Descriptions of the synthetic and real datasets are presented in Table 3.2 and Table 3.3 respectively. The experiments are implemented using Python on a personal computer that has 8 GB of RAM with an Intel i5 3.0 GHz processor. The operating system used is 64bit windows. The proposed algorithm is compared with DBSCAN [46], K-means [32], Single linkage [97], Birch [53], and Spectral clustering [36].

In the above five algorithms, k-means was one of the partition-based clustering algorithms, and single linkage and Birch were the most popular algorithms of the hierarchical clustering algorithms, both of which are traditional clustering algorithms. Spectral clustering is one of the graph-based clustering algorithms. DBSCAN is one of the most common density-based clustering algorithms. For evaluation of clustering results, we use cluster accuracy (AC) as an external clustering validation metric. It is the ratio of overall number of data points that are correctly categorized in each cluster over the size of sample [98]. AC takes value between 0 and 1. The larger the value of AC, the better the result of clustering. For a given dataset that includes K number of clusters indicated as  $C_1, C_2, \dots, C_k$ . Suppose that,  $P_i$  represents the number of objects that are properly classified to cluster  $C_i$ . AC is defined as follows:

$$AC = \frac{\sum_{i=1}^k P_i}{|D|} \quad (3.3)$$

**Table 3.2** Description of the synthetic datasets

Data set	No. of Instances	No. of Attributes	No. of Classes
DS1-Donut	1000	2	4
DS2-DPC	831	2	5
DS3-Twenty	1000	2	20
DS4-Long1	1000	2	2
DS5-Lsun	1000	2	3
DS6-Shapes	1000	2	4
DS7-Smile1	1000	2	4
DS8-Spiral2	1000	2	2
DS9-Wingnut	1016	2	2
DS10-Zilnik5	512	2	4

**Table 3.3** Description of the real datasets

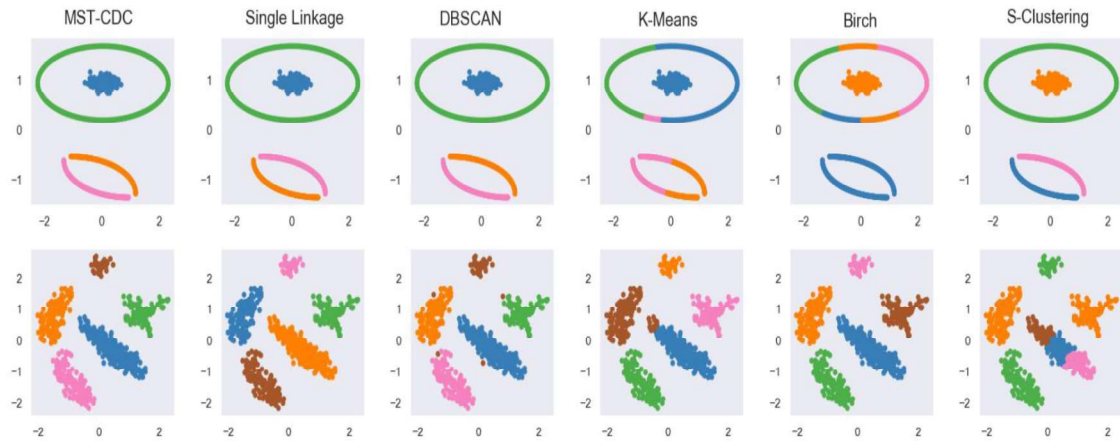
Data set	No. of Instances	No. of Attributes	No. of Classes
DSR1-Haberman	306	3	2
DSR2-Soybean	47	35	4
DSR3-Liver	345	6	2
DSR4-Appendicitis	106	7	2
DSR5-Hypatitis	80	20	2
DSR6-Sonar	208	60	2
DSR7-Spectfheart	267	44	2
DSR8-Bands	365	19	2
DSR9-Tumor	569	30	2

### 3.2.2 Experimental Results using Synthetic Datasets

The results of the purity metric for each algorithm using synthetic datasets are presented in Table 3.4. The description of each experimental result conducted using the synthetic datasets are as follows:

**DS1:** This dataset included four clusters that were different in terms of shapes and densities. Figure 3.11 presents the clustering results using DS1. Single linkage, DBSCAN, spectral clustering, and the proposed algorithm were able to detect the clusters properly. K-means, on the other hand, performed poorly for the clusters that had non-spherical shapes. Note that it was able to properly detect the clusters that had spherical shapes. Similarly, the Birch algorithm favored clusters with spherical shapes and similar sizes, because it used the notion of the diameter to control the boundary of a cluster [53]. Thus, its clustering output was close to that produced by K-means.

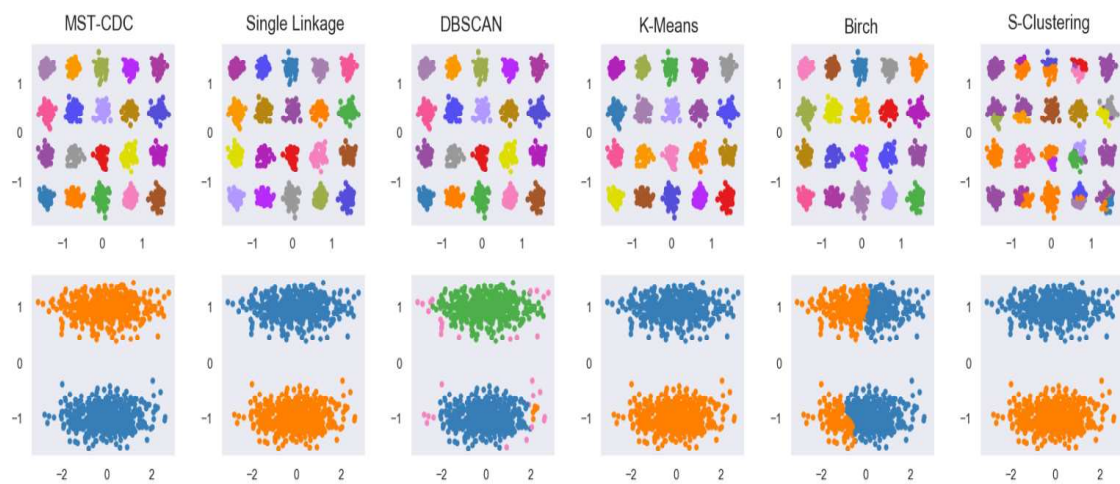
**DS2:** As shown in Figure 3.11, this dataset consisted of four clusters that differed according to density. Single linkage, DBSCAN, and MST-CDC were able to recognize the four clusters properly. However, the actual clusters were not identified properly by K-means, Birch, or spectral clustering. Because the dataset included objects of different densities, it was also difficult to choose the correct parameter combination for each algorithm.



**Figure 3.11** Clustering results for DS1 and DS2

**DS3:** The data set consisted of 20 clusters and they were close to each other, as shown in Figure 3.12. All of the presented clustering algorithms were able to identify the clusters properly, except for spectral clustering, since spectral clustering will always try to give two roughly equal clusters. For example, if the data set is a Gaussian blob, spectral clustering will find a relatively efficient way to cut it in half.

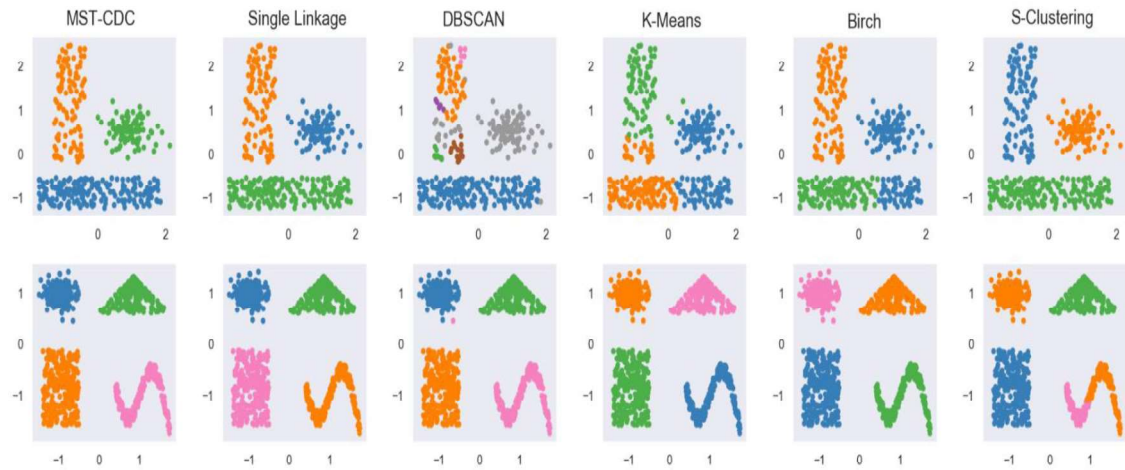
**DS4:** This dataset had two clusters of diverse densities, as shown in Figure 3.12. Except for Birch and DBSCAN, all of the algorithms were able to produce the clusters properly. The reason why Birch failed to recognize the clusters properly was that the dataset consisted of objects with varying densities and Birch usually does not perform well if the cluster is not spherical in shape.



**Figure 3.12** Clustering results for DS3 and DS4

**DS5:** This data set consisted of three clusters with different shapes and densities that were close to each other. Only the proposed algorithm and the single linkage algorithm were able to identify the clusters properly, as shown in Figure 3.13.

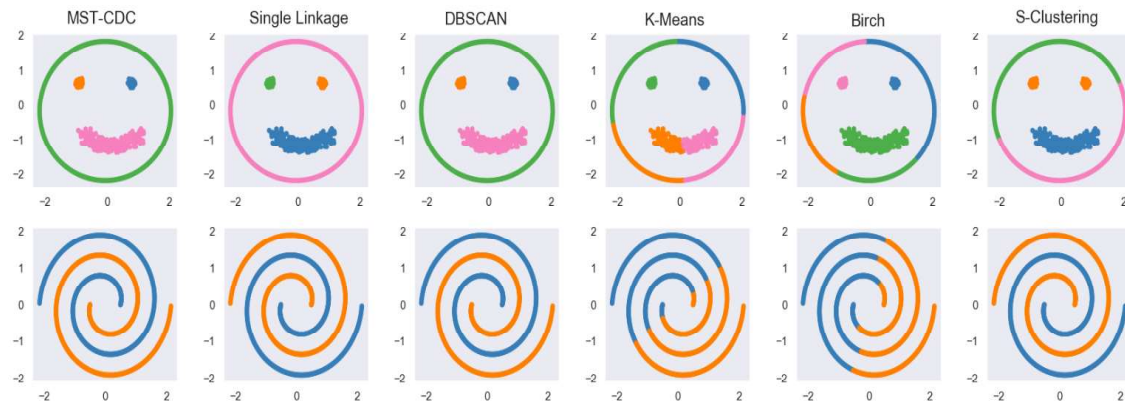
**DS6:** In this dataset, 4 synthetic clusters were created with different shapes and densities, and the clusters were well separated from each other. All of the algorithms were able to produce the four clusters properly, except for spectral clustering, which was only able to identify the top right cluster, as shown in Figure 3.13.



**Figure 3.13** Clustering results for DS5 and DS6

**DS7:** This dataset was composed of four clusters that formed smile shapes. The clusters had different shapes and densities. The proposed algorithm, single linkage, and DBSCAN were able to detect the clusters correctly, as shown in Figure 3.14. However, K-means, Birch, and spectral clustering failed to cluster this dataset since, the shape of the clusters was not spherical and they had different densities and diversities.

**DS8:** This dataset contained two clusters with spiral shapes. Similar to DS7, the proposed algorithm together with single linkage and DBSCAN were able to produce the clusters properly, while the rest were not able to work properly with the spiral shape of the clusters. Figure 3.14 illustrates the results of all of the algorithms.

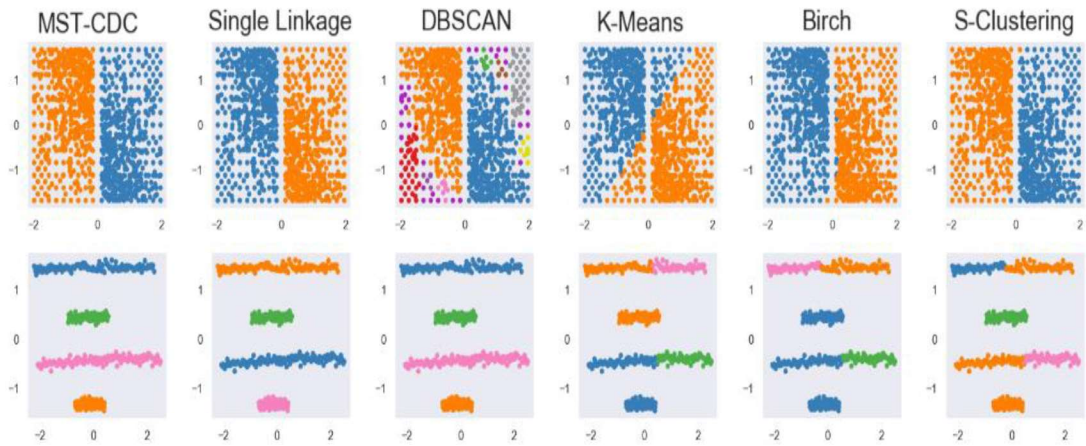


**Figure 3.14** Clustering results for DS7 and DS8

**DS9:** This data set contained two clusters with high density that were very close to each other. Only the proposed algorithm and single linkage were able to identify these two clusters properly, while the rest failed to obtain the clusters correctly, as shown in Figure 3.15.

**DS10:** This was the last data set in the synthetic part of the data sets. It consisted of four parallel clusters with different densities. The cluster results are illustrated in Figure 3.15. The proposed algorithm, single linkage, and DBSCAN were able to identify the proper clusters. K-means was able to discover the sphere-shaped clusters properly, whereas it produced unsatisfactory partitions for the non-sphere-shaped clusters. For the spectral algorithm, the similarity matrix was constructed by a Gaussian kernel function with Euclidean distance. However, its clustering result was similar to that of K-means.

Table 3.4 presents a comparison of the various algorithms with the proposed algorithm according to purity metric using synthetic datasets. It is clear that, both the proposed algorithm and the single linkage algorithm had the best purity score when compared with the others.



**Figure 3. 15** Clustering results for DS9 and DS10.

**Table 3.4** Results of the purity metric for each algorithm using the synthetic datasets

Dataset/Algorithm	MST_CDC	Single-Linkage	DBSCAN	K-Means	Birch	S-Clustering
DS1-Donut	1	1	1	0.606	0.537	1
DS2-DPC	1	1	0.995	0.667	1	0.631
DS3-Twenty	1	1	1	1	1	0.615
DS4-Long1	1	1	0.963	1	0.528	1
DS5-Lsun	1	1	0.845	0.425	0.822	1
DS6-Shapes	1	1	0.999	1	1	0.702
DS7-Smile1	1	1	1	0.683	0.478	0.685
DS8-Spiral2	1	1	1	0.198	0.439	1
DS9:Wingnut	1	1	0.987	0.909	0.990	0.998
DS10:Zilnik5	1	1	1	0.656	0.410	0.687

### 3.2.3 Experimental Results using Real Datasets

The proposed algorithm was also used to evaluate nine real datasets that were adopted from the UCI Repository [96]. The details of these datasets were provided above in Table 3. Experiments were conducted using several well-known algorithms and the best obtained results are indicated in bold in Table 5. For the Haberman dataset, spectral clustering has the best performance compared with the rest. Note that the purity score of the proposed algorithm was also very close to the score of spectral clustering. In the case of the Soybean dataset, the proposed algorithm, DBSCAN, and single linkage outperformed the other algorithms, as shown in Table 3.5. For the Liver dataset, the proposed algorithm performed the best. However, for the Appendicitis dataset, spectral clustering again performed better than the other algorithms. For Hepatitis, Sonar, Spectfheart, and Bands, both the proposed algorithm and DBSCAN had the best purity scores when compared with the others. Finally, for the last dataset, Tumor, as can be seen in Table 4.5, the proposed algorithm outperformed the others.

**Table 3. 5** Results of the purity metric for each algorithm using the real datasets

Dataset/Algorithm	MST_CDC	K-Means	DBSCAN	Single-Linkage	Birch	S-Clustering
Haberman	0.735	0.510	0.722	0.738	0.732	<b>0.741</b>
Soybean	<b>0.765</b>	0.446	<b>0.765</b>	<b>0.765</b>	0.404	0.404
Liver	<b>0.579</b>	0.547	0.536	0.546	0.533	0.571
Appendicitis	0.801	0.811	0.707	0.792	0.660	<b>0.839</b>
Hepatitis	<b>0.837</b>	0.775	<b>0.837</b>	0.825	0.750	0.800
Sonar	<b>0.533</b>	0.528	<b>0.533</b>	0.528	0.528	0.528
Spectfheart	<b>0.794</b>	0.662	<b>0.794</b>	0.790	0.617	0.790
Bands	<b>0.630</b>	0.575	<b>0.630</b>	0.627	0.531	0.627
Tumor	<b>0.660</b>	0.634	0.522	0.657	0.657	0.657

In conclusion, in this chapter, the proposed algorithm is explained in detail; describing the conducted experimental study and the steps required to implement the algorithm. The proposed algorithm attempts to identify the inconsistent edges in MST graph by using the critical distance methodology. The Euclidian distance is used as the distance measure between any two vertices in the MST graph. The critical distance is the maximum distance of the nearest neighbors' distances in MST graph. It is used as a threshold to identify inconsistent edges that will be cut from MST graph to obtain the clustering result. The proposed algorithm is free from any hyper-parameters that need

to be specified by the user. Two different experimental cases are presented in this chapter to evaluate the proposed algorithm using synthetic and real datasets. In both cases, the obtained clustering results using the proposed algorithm are satisfied with the concept of clustering saying that the distance between any two points within a cluster should be less than the distance between any two points in different clusters. Moreover, the proposed algorithm has overall better performance compared with the most common algorithms (single linkage, DBSCAN, K-means, Birch, and spectral clustering).

# CHAPTER 4

## EXPERIMENTAL ANALYSIS AND DISCUSSION

To further evaluate the clustering performance of the proposed algorithm, we performed experimental clustering analysis including 3 different cases and using 6 different datasets carefully chosen for the discussion of cases where the observer needs to decrease or increase the number of the clusters obtained in accordance with the purpose of the study or depending on the distribution nature of the dataset. We adopted the factors proposed in our previous work [16] to prove the flexibility and efficiency of the proposed algorithm for dealing with such cases. The cases are divided into three groups as follows:

1. **Cases related to datasets that are free of outliers:** In these cases, we evaluate the performance of our MST-based clustering algorithm on both two and three dimensional synthetic datasets. Merge factor (MF),  $ESF_{(c,k-oth)}$ ,  $ESF_{(c,k-n)}$ , and  $ISF_{(c,k)}$  are used to validate the obtained clusters.
2. **Cases related to merging the closest clusters:** In the second set of cases, we address the requirement of merging some nearest clusters using MF and the internal strength factor ( $ISF_{(c,k)}$ ).
3. **Cases associated with the presence of some outlier distances:** The receptivity factor (RF) is used in this case to check whether a dataset has receptivity to being clustered into several groups or not. Thus, several clusters can be obtained by removing the outliers from the dataset. For this set of cases, we also discuss cases that include datasets where the critical distance ( $\lambda_{(data)}$ ) is large because of the existence of some outliers, leading to the inclusion of all objects in the dataset inside one cluster. Lambda ( $\lambda_{(data)}$ ) in this case will be replaced with the Upper Limit Outlier (ULO) or its multiples, known in statistics as an extremism threshold.

Moreover, One-way ANOVA is also used to analyze the obtained clusters statistically to investigate the correlation between the distances within each obtained cluster. The One-way ANOVA function is also used to plot box-and-whisker plots which include

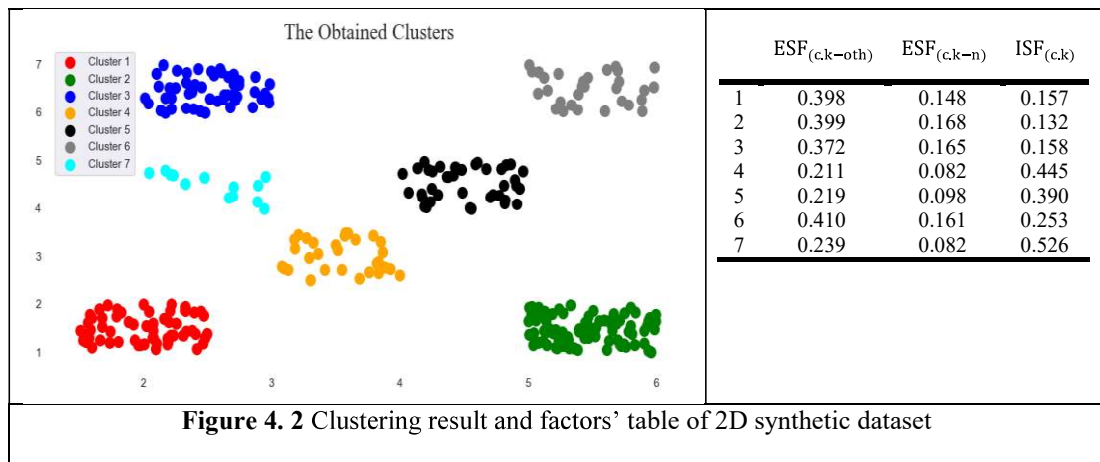
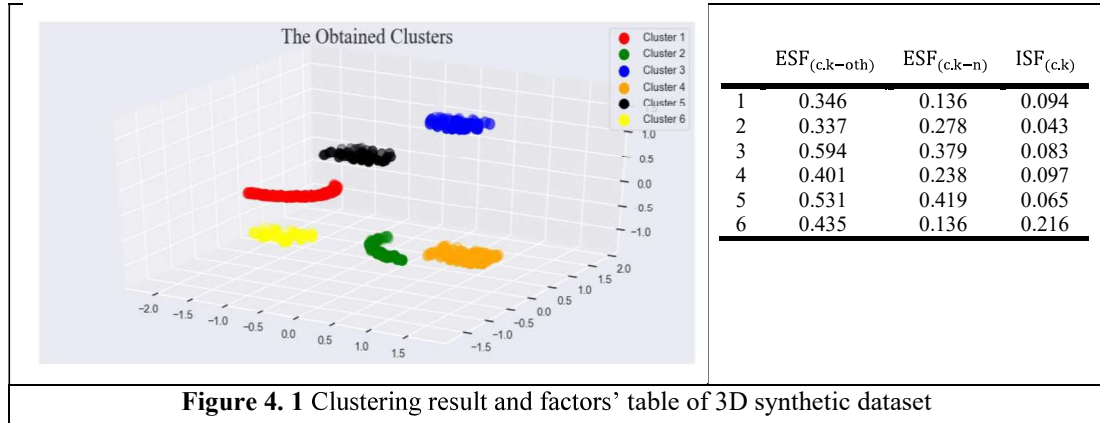
interquartile ranges, medians, the ranges of the data, and the 95% confidence intervals for the medians. All of these cases are discussed with figures and statistics in detail in the next sections.

#### 4.1 Cases Related to datasets that are free of outliers

The results presented in Figures 4.1 and 4.2 show that the experiments for this set satisfy the clustering concept and the merging of clusters is not necessary. For the experiment shown in Figure 4.1,  $MF=0.232$  which means that the obtained clusters are well separated as the obtained MF is less than one and close to zero. As a result, the necessity for merging is weak in this case.  $ESF_{(c,k-oth)}$ ,  $ESF_{(c,k-n)}$ , and  $ISF_{(c,k)}$  are factors proposed in our previous work that can also be used to analyze and evaluate the clustering result. In the factors' table in Figure 4.1, cluster number 6 is the most separated cluster with respect to all others because it has a value of  $ESF_{(c,k-oth)} = 0.410$ . Cluster number 2 has a value of  $ESF_{(c,k-n)} = 0.168$  which means that this cluster is considered the most separated cluster with respect to the nearest cluster. However, cluster number 4 can be said to be the least separated cluster with respect to all others because of its value of  $ESF_{(c,k-oth)} = 0.211$ . It also has the lowest  $ESF_{(c,k-n)}$  value, which is  $ESF_{(c,k-n)} = 0.082$ , and this indicates that this cluster is considered the least separated cluster with respect to the nearest cluster. Another factor to evaluate the internal strength of any cluster is  $ISF_{(c,k)}$ .  $ISF_{(c,k)}$  evaluates data coherence inside a cluster with respect to the nearest cluster as presented in Figure 4.2, where cluster number 2 has a value of  $ISF_{(c,k)} = 0.043$ , indicating that this cluster has the strongest data coherence of all clusters. However, cluster number 6 has a value of  $ISF_{(c,k)} = 0.216$ , which indicates the weakness of this cluster regarding data coherence. The formulas of all the factors used in these cases are as follows:

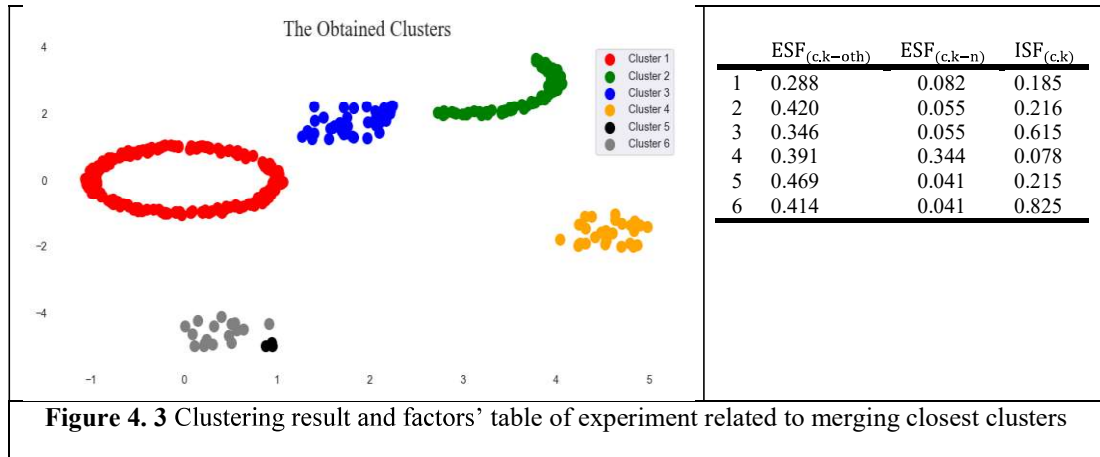
$$ISF_{(c,k)} = \frac{\lambda_{(c,k)}}{near\_d_{(c,k-n,k)}} \quad (4.1) ,$$

$$ESF_{(c,k-n)} = \frac{near\_d_{(c,k-n,k)}}{max\_d_{(all)}} \quad (4.2), \quad ESF_{(c,k-oth)} = \frac{avg\_d_{(c,k-oth)}}{max\_d_{(all)}} \quad (4.3)$$



## 4.2 Cases Related to Merging the Closest Clusters

The obtained results in these cases entail some clusters that are close to each other that need to be merged in one cluster. The results of the conducted experiment for these cases are illustrated in Figure 4.3, which highlights the need for merging nearest clusters. The merge factor (MF) is utilized as a general indication for the requirement of merging certain clusters located close to each other for a given dataset. Furthermore, in order to identify the cluster that has to be merged with its nearest cluster, another indicator is used to play this rule which is  $ISF_{(c,k)}$ . For the experiment shown in Figure 4.3,  $MF=0.825$ , which is close to one. Thus, this gives a general indication that there is a need to merge some nearest clusters. At this point, it is necessary to specify the closest clusters needing to be merged by using  $ISF_{(c,k)}$ . As shown in Figure 4.3,  $ISF_{(c,k)}=0.825$ , which belongs to cluster number 6, and this reveals that cluster number 6 has to be merged with the closest cluster. Note that the minimum distance between candidate clusters needs to be specified for the merging of the two clusters.



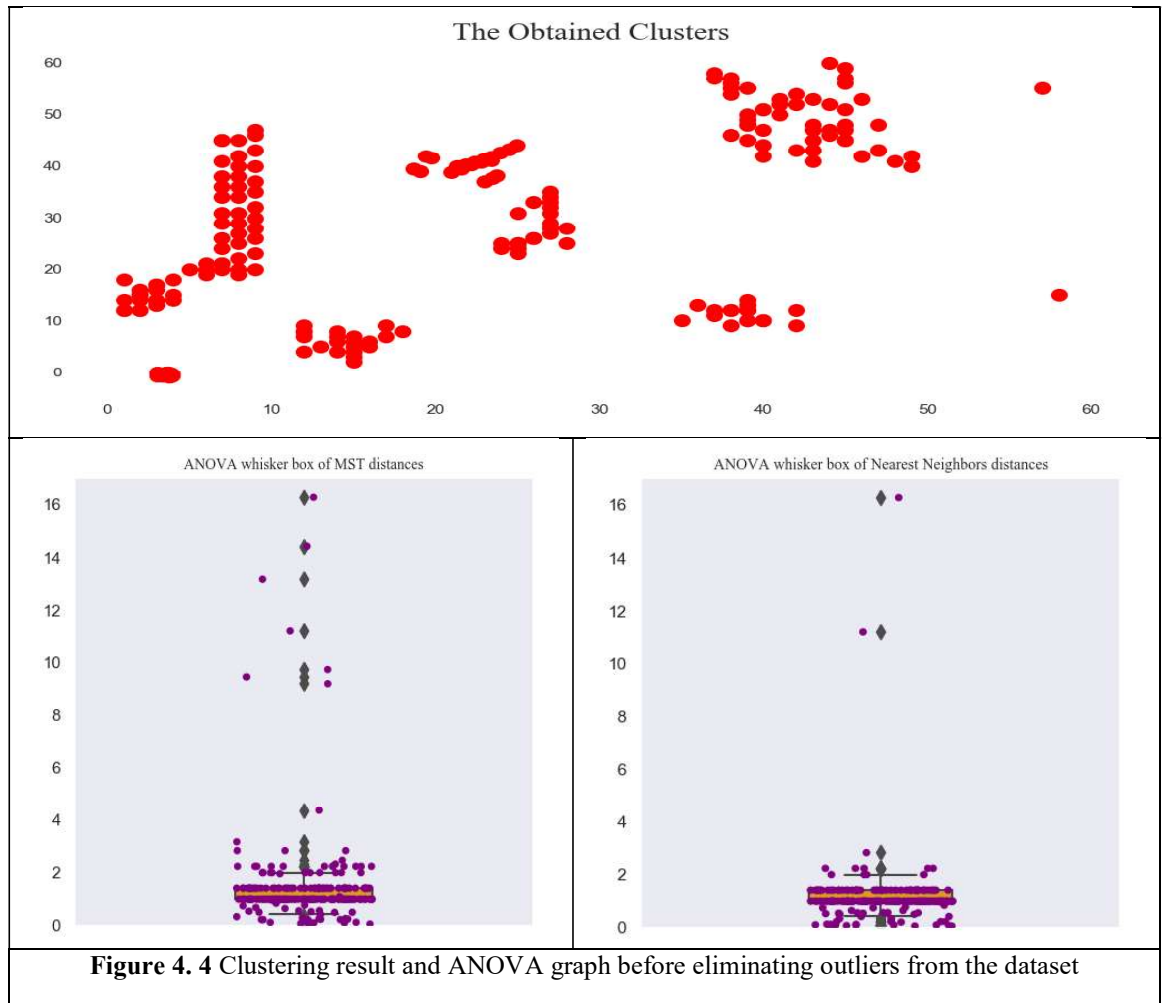
### 4.3 Cases Associated with the Presence of Some Outlier Distances

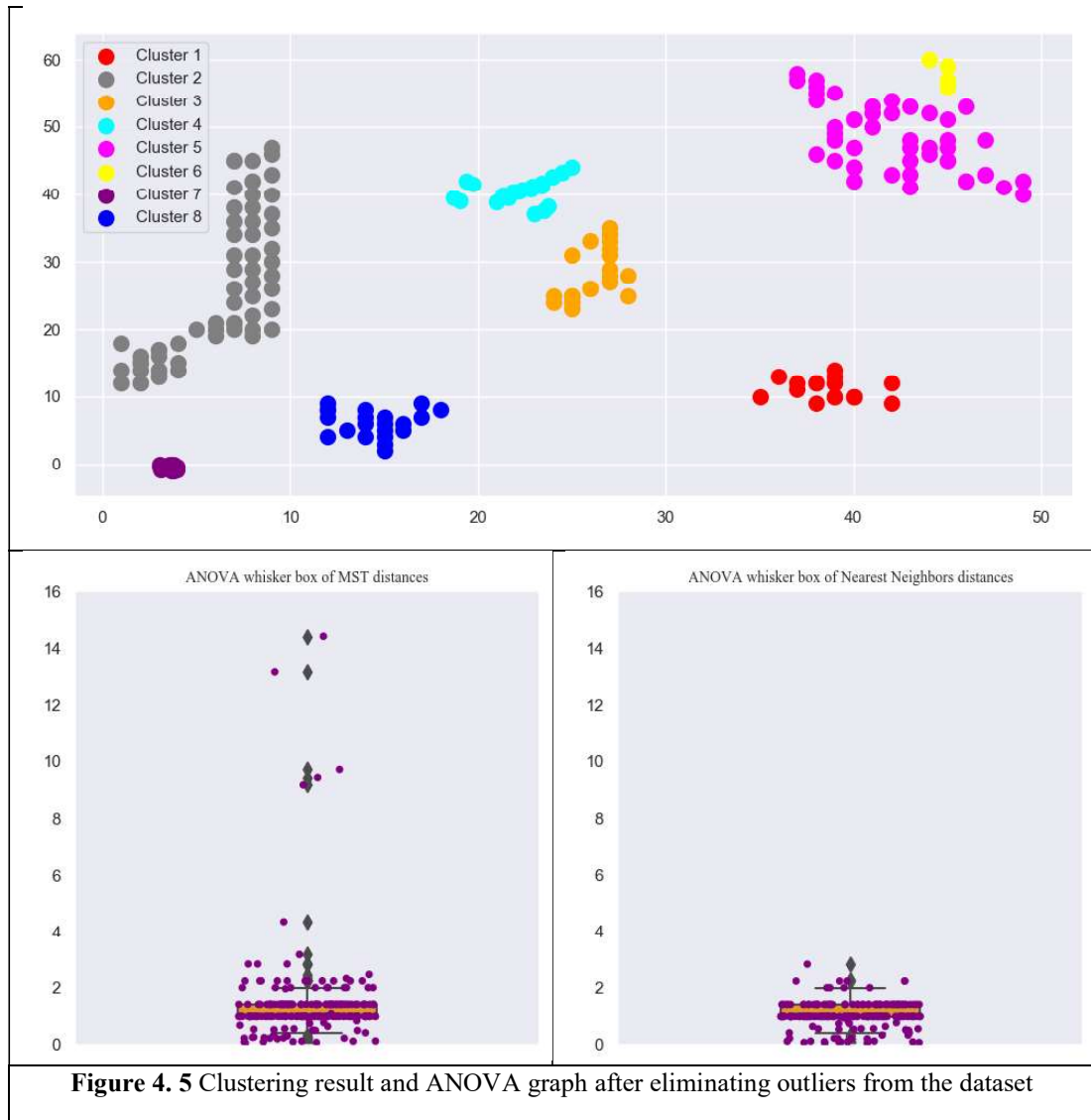
On the basis of the MST cut property, clustering algorithms based on MST could be very beneficial in discovering small clusters that are linked with typical clusters using long edges, and these small clusters could be treated as outliers. However, cutting small sets of isolated nodes located within a graph based on MST-based clustering criteria may leads to a bad cut when there are no outliers existing. Therefore, there must be a method to check for the existence of outliers. RF together with ANOVA graphs can be used to indicate whether the dataset contains some outlier distances or not. The presence of outlier distances either results in including the entire dataset within one cluster or leads to a result that contains some clusters that are not optimal. Thus, by comparing both the maximum distance in the nearest neighbor graph and the maximum distance in the MST graph using RF, if  $RF=1$  it means that the distances are equal and the dataset does not satisfy with the concept of clustering because of the presence of some outlier distances in the dataset. To overcome such a problem, two methods are proposed which are eliminating the outlier distances from dataset and substituting the lambda value with the upper limit outlier (ULO) as discussed in the following sections.

#### 4.3.1 Eliminating Outlier distances from the dataset

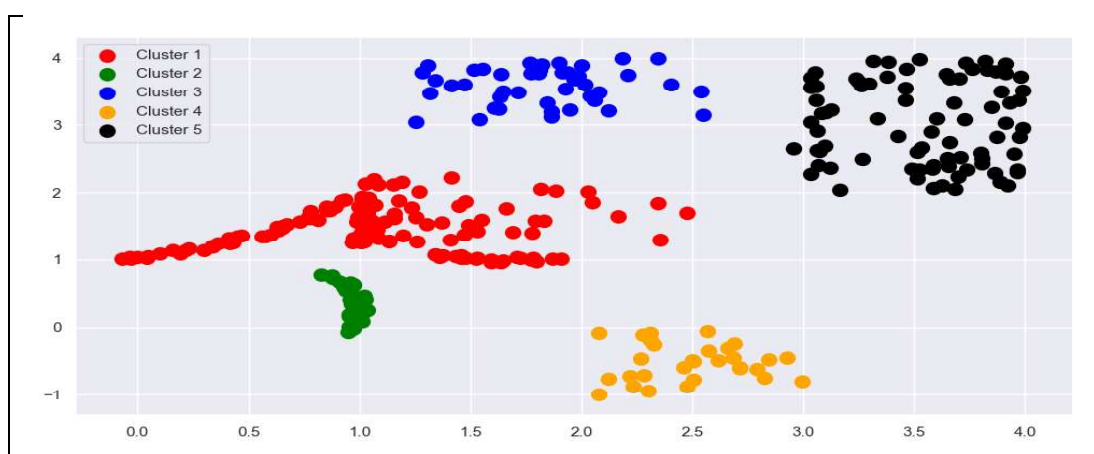
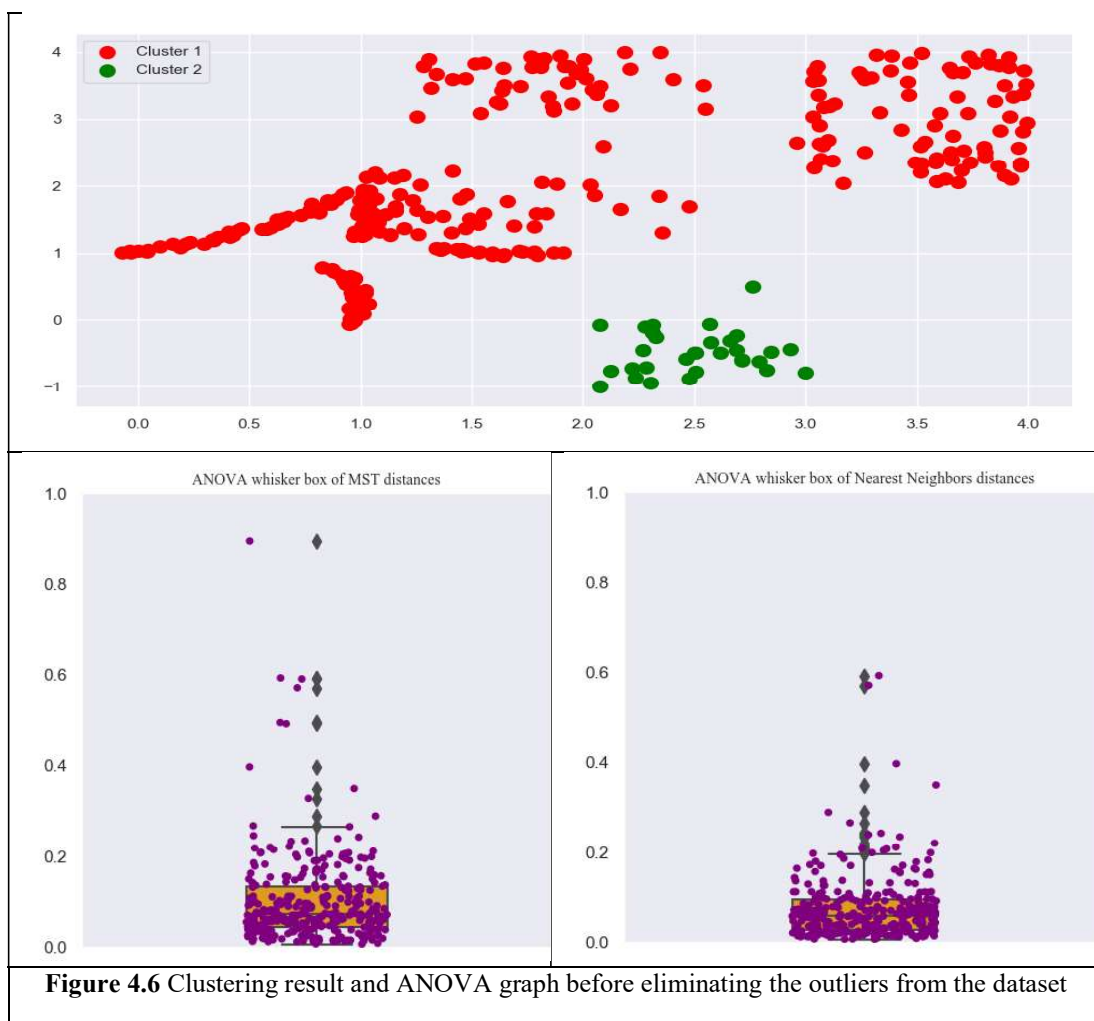
It is possible to acquire several clusters after removing the outliers from the dataset. The case shown in Figure 4.4 is an example of a case where the presence of some outliers results in a large value of critical distance and this leads to the grouping the all

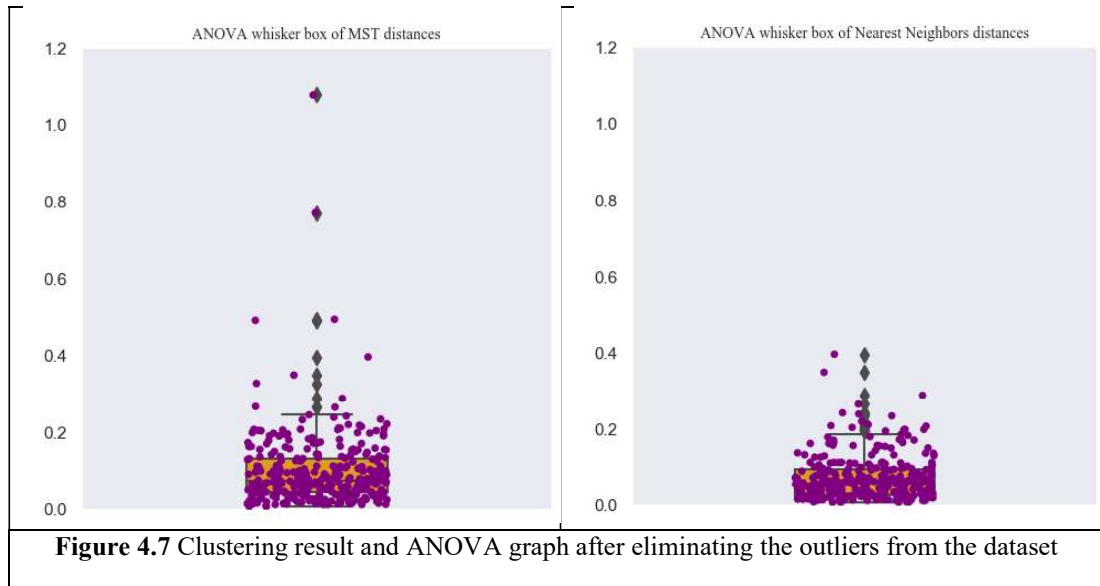
dataset points in a single cluster. It is clearly seen in Figure 4.4 and from the ANOVA graph of nearest neighbor distances that the clustering results involve two outliers' distances, which prevents the establishment of several clusters. However, seven clusters can be obtained after removing the outliers from the dataset as shown in Figure 4.5.





Similarly, in the experiment performed in Figure 4.6, the number of obtained clusters was two before eliminating the outlier points. The dataset described in Figure 4.6 contained two outlier data points that made the lambda value relatively large, thus, preventing the algorithm from obtaining more clusters. The coordinates of the outlier points were (2.091, 2.601) and (2.759, 0.500), respectively, and the number of obtained clusters became five after eliminating the two outliers. Figure 4.7 shows the obtained clusters and ANOVA graphs after eliminating the two outlier data points from the dataset.





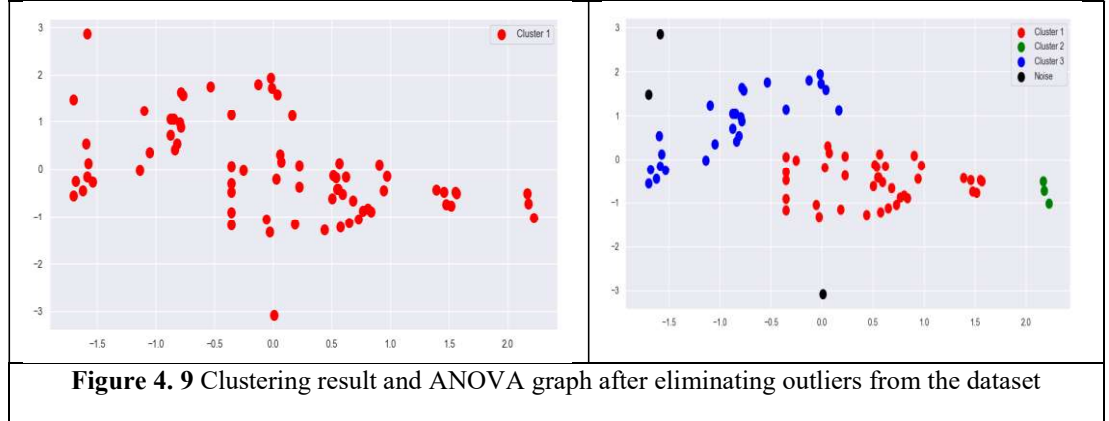
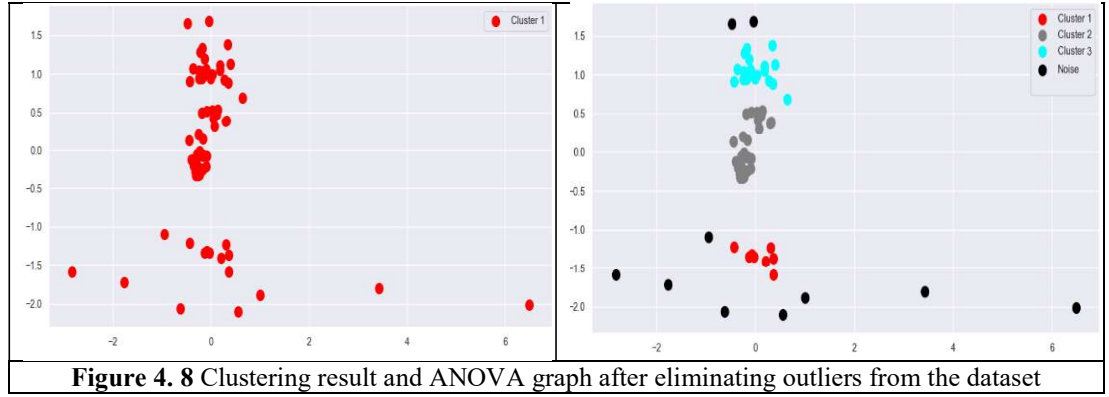
#### 4.3.2 Substituting the Critical Distance Value with ULO

Generally, the proposed algorithm produces a set of clusters that are consistent with the clustering concept. However, in the process of data clustering, it is seen that the majority of the real datasets have a random distribution of data points, which makes it difficult and complicated to get some optimal clusters. It is still possible, though, to utilize the proposed algorithm to get some clusters even when the value of the critical distance leads to the grouping of all dataset points into one cluster due to outliers. Consequently,  $\lambda_{(data)}$  (the critical distance ) needs to be replaced by an appropriate value that enables the obtaining of some clusters. The Upper Limit Outlier (ULO) or its multiples will be chosen to replace the critical distance value. Note that in statistics ULO is defined as a threshold that determines the starting point of extremism for a given dataset. The formula for the ULO is as follows:

$$ULO = Q_3 + 1.5 * (IQR) \quad (4.4)$$

Here, IQR is the interquartile range, computed using the equation:  $IQR = (Q_3 - Q_1)$  where  $Q_3$  represents the interquartile range of the 75<sup>th</sup> percentile and  $Q_1$  represents the interquartile range of the 25<sup>th</sup> percentile. As presented on the left side of Figure 4.8, all of the dataset points are included inside one cluster since critical distance  $\lambda_{(data)}$  is large due to outlier existence. To solve this problem, the critical distance has to be substituted with the ULO using the following formula:  $\lambda_{(data)} = 1.5 * ULO$ . Thus, the obtained clusters become 3 as shown on the right side of Figure 4.8. The black points

in the obtained results represent outlier points that are detected by the algorithm after cutting the MST graph using  $\lambda_{(\text{data})} = 1.5 * \text{ULO}$  as a threshold to cut the MST edges. Note that any connected components that are below the minimum cluster size, which is 3, are considered as outliers. Similarly, the last experiment shown in Figure 4.9 denotes a case in which there are some outlier distances existing and the clusters are homogeneous. We can handle this case either by using  $\lambda_{(\text{data})} = 1.5 * \text{ULO}$  as a threshold to cut the MST edges and obtaining some clusters or eliminating some outlier data points and obtaining clusters as described in Section 4.3.1.



In this chapter, an experimental analysis and discussion is performed to evaluate the clustering performance of the proposed algorithm. Three different cases are discussed using different datasets that are carefully chosen to test the performance of the proposed algorithm. The factors proposed in our previous work [16] are also adopted to prove the flexibility and efficiency of the proposed algorithm. In the first set of cases where datasets are free of outliers, we evaluate the performance of our MST-based clustering algorithm on both two and three dimensional synthetic datasets. Merge

factor (MF),  $ESF_{(c,k-oth)}$ ,  $ESF_{(c,k-n)}$ , and  $ISF_{(c,k)}$  are used to validate the obtained clusters. In the second set of cases where the obtained results contain clusters that are close to each other and need to be merged. In this case, we address the requirement of merging some nearest clusters using MF and the internal strength factor ( $ISF_{(c,k)}$ ). The last set of cases are associated with the presence of some outlier distances. In this case, the receptivity factor (RF) is used to check whether a dataset has receptivity to being clustered into several groups or not. Thus, several clusters can be obtained by removing the outliers from the dataset. For this set of cases, we also discuss cases that include datasets where the critical distance ( $\lambda_{(data)}$ ) is large because of the existence of some outliers, leading to the inclusion of all objects in the dataset inside one cluster. Lambda ( $\lambda_{(data)}$ ) in this case will be replaced with the Upper Limit Outlier (ULO) or its multiples, known in statistics as an extremism threshold.

# CHAPTER 5

## CONCLUSION AND FUTURE WORK

### 5.1 Conclusion

In all algorithms for MST-based clustering, defining the inconsistent edges is the main problem that needs to be addressed. Using the critical distance methodology, the algorithm proposed here attempts to determine the inconsistent edges in the field of MST-based clustering. The Euclidian distance between vertices of the MST graph is used as a distance metric. The proposed algorithm is free of any hyper parameters that would need to be specified by users. It can also deal with cases where a given dataset contains some outliers. To prove the efficiency of the algorithm, 3 groups of experiments were implemented and some factors were used to validate and analyze the obtained clusters. The experimental results conducted on synthetic and real data sets illustrated that the proposed clustering algorithm has overall better performance compared with the most common algorithms (single linkage, DBSCAN, K-means, Birch, and spectral clustering). For example, with the Liver and Tumor datasets, the proposed algorithm outperforms all other clustering algorithms with clustering accuracy equal to 0.579 and 0.660 respectively.

The following are the contributions made by this dissertation, as originally highlighted in Section 1.2.

1. **This study proposed a new MST-based clustering algorithm using inconsistent distance methodology to solve the main issue of MST-based clustering related to removing the inconsistent edges in the MST graph:** The proposed algorithm is presented in Section 3.1, in detail.
2. **The proposed algorithm is able to determine the number of clusters for a given dataset, regardless of the shape distribution of the data:** This contribution was achieved in Chapter 3, in Sections 3.2.1 and 3.2.2, when the proposed algorithm was evaluated on 10 synthetic data sets, DS1–DS11, and 9 real-world datasets, DSR1–DSR9. The data sets used for evaluation had

different characteristics and shapes. The conducted experiments clearly showed the efficiency of the algorithm in determining the proper number of clusters for a given dataset.

3. **It has the capability of determining the existence of outlier points and using several metrics and ANOVA to evaluate the quality of the obtained data clustering result:** The experimental analysis and discussion performed in Chapter 4 discussed several cases of how the proposed algorithm was able to deal with the existence of outliers.
4. **It is flexible, easy to understand and implement, simple, and does not need to determine any parameters in advance:** The Pseudo code presented in Chapter 3, Section 3.1, described the implementation simplicity of the algorithm.
5. **The algorithm is capable of detecting clusters with irregular boundaries that have different sizes and different densities:** The proposed algorithm was evaluated using 10 synthetic data sets, DS1–DS11, and 9 real-world datasets, DSR1–DSR9, as described in Chapter 3, in Sections 3.2.1 and 3.2.2. The data sets used for evaluation had different characteristics and shapes.
6. **The algorithm is stable, and the order of the dataset points has no impact on the final results:** The methodology explained in Chapter 3 proved the stability of the proposed algorithm.

## 5.2 Future Work

As future work, more experimental analysis will be conducted using different evaluation metrics to analyze the obtained results. Moreover, a comprehensive comparative analysis will be performed using other data clustering algorithms. Another future aim is to promote the computational efficiency of the proposed algorithm. Thus, time complexity is a restriction of the present work that needs to be considered in the future. The proposed algorithm needs  $O(N^2)$ , which is quite acceptable. The bottleneck lies in calculating minimax distances (constructing the MST graph). However, there is some research in the literature that has been conducted with the aim of developing a faster MST algorithm that we might consider adopting in the future to improve the time complexity of our algorithm. We will also try to expand

the current research to make use of the density of each object in the dataset and combine it with distances between each object to produce more robust clustering results. Thus, cutting the MST graph will depend on not only the distances between the objects, but also the density of each object in the dataset.

## REFERENCES

- [1] Nerurkar, P., Shirke, A., Chandane, M. & Bhirud, S., “Empirical Analysis of Data Clustering Algorithms”. *Procedia Computer Science*, 125, 770–779, 2018.
- [2] Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Khalil, I., Zomaya, A. Y., Foufou, S. & Bouras, A., “A survey of clustering algorithms for big data: Taxonomy and empirical analysis”. *IEEE Transactions on Emerging Topics in Computing*, 2(3), 267–279, 2014.
- [3] Chapelle, O., Scholkopf, B. & Zien, A., “Semi-supervised learning”. *IEEE Transactions on Neural Networks*, 20(3), 542–542, 2009.
- [4] Estivill-Castro, V., “Why so many clustering algorithms: a position paper”. *ACM SIGKDD Explorations Newsletter*, 4(1), 65–75, 2002.
- [5] Liu, Q., Zhang, R., Hu, R., Wang, G., Wang, Z., & Zhao, Z., “An improved path-based clustering algorithm”, *Knowledge-based Systems*, 163, 69–81, 2019.
- [6] Handl, J. & Knowles, J., “An evolutionary approach to multi-objective clustering”, *IEEE Transactions on Evolutionary Computation*, 11(1), 56–76, 2007.
- [7] Topchy, A., Jain, A. K. & Punch, W., “Clustering ensembles: models of consensus and weak partitions”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12), 1866–1881, 2005.
- [8] Ayad, H. G. & Kamel, M. S., “Cumulative voting consensus method for partitions with variable number of clusters”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1), 160–173, 2008.
- [9] Gionis, A., Mannila, H., & Tsaparas, P., “Clustering aggregation”, *ACM Transactions on Knowledge Discovery from Data*, 1(1), 1–30, 2007.

- [10] Wang, X., Wang, X. L., Chen, C. & Wilkes, D. M., “Enhancing minimum spanning tree-based clustering by removing density-based outliers”. *Digital Signal Processing: A Review Journal*, 23(5), 1523–1538, 2013.
- [11] Wu, J., Li, X., Jiao, L., Wang, X. & Sun, B., “Minimum spanning trees for community detection”, *Physica A*, 392(9), 2265–2277, 2013.
- [12] Pirim, H., Ekşioğlu, B. & Perkins, A. D., “Clustering high throughput biological data with b-mst, a minimum spanning tree based heuristic”, *Computers in Biology and Medicine*, 62(C), 94, 2015.
- [13] Zhong, C., Miao, D. & Wang, R., “A graph theoretical clustering method based on two rounds of minimum spanning trees”, *Pattern Recognition*, 43(3), 752–766, 2010.
- [14] Sharan, R. & Shamir, R., “Click: a clustering algorithm with applications to gene expression analysis”, In: *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*, 8, 307–316, 2000.
- [15] Huttenhower, C., Flamholz, A. I., Landis, J. N., Sahi, S., Myers, C. L., Olszewski, K. L., et al., “Nearest neighbor networks: clustering expression databased on gene neighborhoods”, *BMC Bioinformatics*, 8(250), 1–13, 2007.
- [16] Farag, H. K., Fadi, S., Ahmet, E. T. & Fionn, M., “A new data clustering algorithm based on inconsistent distance methodology”. *Journal of Expert Systems with Application*, 2019.
- [17] Bereta, M. & Burczynski, T., “Immune k-means and negative selection algorithms for data analysis”, *Information Sciences*, 179, 1407–1425, 2009.
- [18] Bezdek, J. C. & Pal, N. R., “Some new indexes of cluster validity”, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 28, 301–315, 1998.
- [19] Chang, H. & Yeung, D. Y., “Robust path-based spectral clustering”, *Pattern Recognition*, 41, 191–203, 2008.

- [20] Karypis, G., Han, E. H. & Kumar, V., “CHAMELEON: a hierarchical clustering algorithm using dynamic modeling”, *IEEE Transactions on Computers*, 32, 68–75, 1999.
- [21] Kaufman, L. & Rousseeuw, P. J., “Finding Groups in Data: An Introduction to Cluster Analysis”, Wiley, New York, 1990.
- [22] Jain, A. K. & Dubes, R.C., “Algorithms for Clustering Data”, Prentice-Hall, Englewood Cliffs, New Jersey, 1988.
- [23] Jain, A. K. & Law, M. C., “Data clustering: a user’s dilemma”, In: *Pattern Recognition and Machine Intelligence*, LNCS, Vol. 3776, Springer-Verlag, Berlin, Heidelberg, pp. 1–10, 2005.
- [24] Hsu, C. C., Chen, C. L. & Su, Y. W., “Hierarchical clustering of mixed data based on distance hierarchy”, *Information Sciences*, 177, 4474–4492, 2007
- [25] Jain, A. K., Murthy, M. & Flynn, P., “Data clustering: a review”, *ACM Computing Surveys*, 31, 264–323, 1999.
- [26] Milligan, G. W., Soon, S. C. & Sokol, L. M., “The effect of cluster size, dimensionality, and the number of clusters on recovery of true cluster structure”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5, 40–47, 1983.
- [27] Ng, R. T. & Han, J., “CLARANS: a method for clustering objects for spatial data mining”, *IEEE Transactions on Knowledge and Data Engineering*, 14, 1003–1016, 2002.
- [28] McLachlan, G. & Basford, K., “Mixture Models: Inference and Application to Clustering”, Marcel Dekker, New York, 1988.
- [29] Lee, C. H., Zaïane, O. R., Park, H. H., Huang, J. & Greiner, R., “Clustering high dimensional data: a graph-based relaxed optimization approach”, *Information Sciences*, 178, 4501–4511, 2008.

- [30] Lin, C. R. & Chen, M. S., “Combining partitional and hierarchical algorithms for robust and efficient data clustering with cohesion self-merging”, *IEEE Trans. Knowl. Data Eng.* 17, 145–159, 2005.
- [31] Liu, M., Jiang, X. & Kot, A. C., “A multi-prototype clustering algorithm”, *Pattern Recognition*, 42, 689–698, 2009.
- [32] MacQueen, J., “Some methods for classification and analysis of multivariate observations”, In: *The 5th Berkeley Symposium on Mathematics, Statistics and Probability*, 281–297, 1967.
- [33] King, B., “Step-wise clustering procedures”, *Journal of the American Statistical Association*, 69, 86–101, 1967.
- [34] Lai, J. Z. C. & Huang, T. J., “An agglomerative clustering algorithm using a dynamic k-nearest-neighbor list”, *Information Sciences*, 181, 1722–1734, 2011.
- [35] Lee, J. S. & Olafsson, S., “Data clustering by minimizing dysconnectivity”, *Information Sciences*, 181, 732–746, 2011.
- [36] Shi, J. & Malik, J., “Normalized cuts and image segmentation”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22, 888–905, 2000.
- [37] Schaeffer, S.E., “Graph clustering”, *Computer Science Review*, 1, 27–64, 2007.
- [38] Sneath, P. H. A. & Sokal, R. R., “Numerical Taxonomy”, Freeman, San Francisco, London, 1973.
- [39] Veenman, C. J., Reinders, M. J. T. & Backer, E., “A maximum variance cluster algorithm”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24, 1273–1280, 2002.
- [40] Wang, W., Yang, J. & Muntz, M., “STING: a statistical information grid approach to spatial datamining”, In: *Proceedings of the International Conference on Very Large Data Bases*, 186–195, 1997.

- [41] Toussaint, G. T., “The relative neighborhood graph of a finite planar set”, *Pattern Recognition*, 12, 261–268, 1980.
- [42] Theodoridis, S. & Kouttoubas, K., “Pattern Recognition”, 4th ed., Academic Press, Amsterdam, 2009.
- [43] Warrens, M. J., “On the equivalence of Cohen’s kappa and the Hubert-Arabie Adjusted Rand index”, *Journal of Classification*, 25, 177–183, 2008.
- [44] Wu, Z. & Leahy, R., “An optimal graph theoretic approach to data clustering: theory and its application to image segmentation”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15, 1101–1113, 1993.
- [45] Corman, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C., “Introduction to Algorithms”, 2nd ed., MIT Press, Cambridge, Massachusetts, 2001.
- [46] Ester, M., Kriegel, H. P., Sander, J. & Xu, X., “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, Oregon, pp. 226–231, 1996.
- [47] Figueiredo, M. & Jain, A. K., “Unsupervised learning of finite mixture models”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24, 381–396, 2002.
- [48] Fränti, P., Virtajoki, O. & Hautamäki, V., “Fast agglomerative clustering using a k-nearest neighbor graph”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28, 1875–1881, 2006.
- [49] Cheng, D., Kannan, R., Vempala, S. & Wang, G., “A divide-and-merge methodology for clustering”, *ACM Transactions on Database Systems*, 31, 1499–1525, 2006.
- [50] Chiang, M. C., Tsai, C. W. & Yang, C. S., “A time-efficient pattern reduction algorithm for k-means clustering”, *Information Sciences*, 181, 716–731, 2011.

- [51] Chu, Y., Huang, J., Chuang, K., Yang, D. & Chen, M., “Density conscious subspace clustering for high-dimensional data”, *IEEE Transactions on Knowledge and Data Engineering*, 22, 16–30, 2010.
- [52] Chu, Y., Chen, Y., Yang, D. & Chen, M., “Reducing redundancy in subspace clustering”, *IEEE Transactions on Knowledge and Data Engineering*, 21, 1432–1446, 2009.
- [53] Zhang, T., Ramakrishnan, R. & Livny, M., BIRCH: An efficient data clustering method for very large databases. *SIGMOD Record*, 25(2), 103–114, 1996.
- [54] Defays, D., “An efficient algorithm for the complete link method,” *The Computer Journal*, 20, 364-366, 1977.
- [55] Guha, S., Rastogi, R. & Shim, K., “CURE: an efficient clustering algorithm for large database”, *Information Systems*, 26(1), 35–58, 2001.
- [56] Weinberger, K. Q. & Saul, L. K., “Distance metric learning for large margin nearest neighbor classification”, *Journal of Machine Learning Research*, 10, 207–244, 2009.
- [57] Jothi, R., Mohanty, S. K. & Ojha, A., “Fast approximate minimum spanning tree based clustering algorithm”, *Neurocomputing*, 272, 542–557, 2018.
- [58] Lv, X., Ma, Y., He, X., Huang, H. & Yang, J., “CciMST: A clustering algorithm based on minimum spanning tree and cluster centers”, *Mathematical Problems in Engineering*, Vol. 2018, Article ID 8451796, 14 pages, 2018.
- [59] Pirim, H., Ekşioğlu, B. & Perkins, A. D., “Clustering high throughput biological data with B-MST, a minimum spanning tree based heuristic”, *Computers in Biology and Medicine*, 62, 94–102, 2015.
- [60] Morris, O. J., Lee, M. D. J. & Constantinides, A. G., “Graph theory for image analysis: an approach based on the shortest spanning tree”, *IEE Proceedings F - Communications, Radar and Signal Processing*, 133(2), 146–152, 1986.

- [61] Lloyd, E. K., Bondy, J. A. & Murty, U. S. R., “Graph theory with applications”, The Mathematical Gazette, 62, 1978.
- [62] Harju, T., “Lecture notes on graph theory”, Physical Review Letters, 107 (8), 085504, 2011.
- [63] Borůvka, O., O jistém problému minimálním, Práce Moravské přírodovědecké společnosti 3 (3), 37–58, 1926.
- [64] Kruskal, J. B., “On the shortest spanning subtree of a graph and the traveling salesman problem”, Proceedings of the American Mathematical Society, 7(1), 48–50, 1956.
- [65] Prim, R. C., “Shortest connection networks and some generalizations”, Bell System Technical Journal, 36(6), 1389–1401, 1957.
- [66] Halim, Z., & Uzma, “Optimizing the minimum spanning tree-based extracted clusters using evolution strategy”. Cluster Computing, 1–15, 2017.
- [67] Srinivasan, G., “A clustering algorithm for machine cell formation in group technology using minimum spanning trees”. International Journal of Production Research, 32(9), 2149–2158, 1994.
- [68] Wu, Z., Braunstein, L. A., Havlin, S. & Stanley, H. E., “Transport in weighted networks: Partition into superhighways and roads”. Physical Review Letters, 96, 148702, 2006.
- [69] Zahn, C. T., “Graph-theoretical methods for detecting and describing gestalt clusters”. IEEE Transactions on Computers, 100, 68–86, 1971.
- [70] Chowdhury, N. & Murthy, C. A., “Minimal spanning tree based clustering technique: Relationship with Bayes classifier”. Pattern Recognition, 30, 1919–1929, 1997.

- [71] Laszlo, M. & Mukherjee, S., “Minimum spanning tree partitioning algorithm for micro aggregation”. *IEEE Transactions on Knowledge and Data Engineering*, 17, 902–911, 2005.
- [72] Xu, Y., Olman, V. & Xu, D., “Clustering gene expression data using a graph-theoretic approach: An application of minimum spanning trees”, *Bioinformatics*, 18(4), 536–545, 2002.
- [73] Grygorash, O., Zhou, Y. & Jorgensen, Z., “Minimum spanning tree based clustering algorithms”. In: *Proceedings of the 18th International Conference on Tools with Artificial Intelligence*, 73–81, 2006
- [74] Luo, T. & Zhong, C., “A neighborhood density estimation clustering algorithm based on minimum spanning tree:”, In: *Lecture Notes in Computer Science*, 6401, 557–565, 2010.
- [75] Zhong, C., Miao, D. & Fränti, P., “Minimum spanning tree based split-and-merge: a hierarchical clustering method”, *Information Sciences*, 181(16), 3397–3410, 2011.
- [76] Vathy-Fogarassy, A., Kiss, A. & Abonyi, J., “Hybrid minimal spanning tree and mixture of gaussians based clustering algorithms”, In: *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence*, 313–330, 2006.
- [77] Müller, A. C., Nowozin, S. & Lampert, C. H., “Information theoretic clustering using minimum spanning trees”, In: *Proceedings of the 34th Symposium of the German Association for Pattern Recognition*, 205–215, 2012.
- [78] Zhou, R., Shu, L. & Su, Y., “An adaptive minimum spanning tree test for detecting irregularly-shaped spatial clusters”. *Computational Statistics and Data Analysis*, 89, 134–146, 2015.
- [79] Fischer, B. & Buhmann, J. M., “Path-based clustering for grouping of smooth curves and texture segmentation”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(4), 513–518, 2003.

- [80] Fischer, B. & Buhmann, J.M., “Bagging for path-based clustering”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11), 1411–1415, 2003.
- [81] Lopresti, D. & Zhou, J., “Locating and recognizing text in www Data”. *Information Retrieval*, 2, 177–206, 2000.
- [82] Thorndike, R. L., “Who belong in the family?” *Psychometrika*, 18(4), 267–276, 1953.
- [83] Honarkhah, M. & Caers, J., “Stochastic simulation of patterns using distance-based pattern modeling”. *Mathematical Geosciences*, 42(5), 487–517, 2010.
- [84] Sugar, C. A. & James, G. M., “Finding the number of clusters in a data set: An information theoretic approach”. *Journal of the American Statistical Association*, 98, 750–763, 2003.
- [85] Lin, C. R. & Chen, M. S., “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.
- [86] Huang, H., Gao, Y., Chiew, K., Chen, L. & He, Q., “Towards effective and efficient mining of arbitrary shaped clusters”. In: *Proceedings of the 30th International Conference on Data Engineering*, 28–39, 2014
- [87] Wang, X., Wang, X. & Wilkes, D. M., “A divide-and-conquer approach for minimum spanning tree-based clustering”, *IEEE Transactions on Knowledge and Data Engineering*, 21(7), 945–958, 2009.
- [88] Zhong, C., Malinen, M., Miao, D. & Fränti, P., “A fast minimum spanning tree algorithm based on k-means”. *Information Sciences*, 295, 1–17, 2015.
- [89] Cheng, Q., Lu, X., Liu, Z., Huang, J. & Cheng, G., “Spatial clustering with density-ordered tree”. *Physica A: Statistical Mechanics and its Applications*, 460, 188–200, 2016.

- [90] Güngör, E. & Özmen, A., “Distance and density based clustering algorithm using gaussian kernel”. *Expert Systems with Applications*, 69, 10–20, 2017.
- [91] Mishra, G. & Mohanty, S. K., “A fast hybrid clustering technique based on local nearest neighbor using minimum spanning tree”. *Expert Systems with Applications*, 132, 28–43, 2019.
- [92] Rodriguez, A. & Laio, A., “Clustering by fast search and find of density peaks”, *Science*, 344(6191), 1492–1496, 2014.
- [93] Handl, J. & Knowles, J., “Multi-objective clustering with automatic determination of the number of clusters”. UMIST, Manchester, Tech. Rep. TR-COMPSYSBIO-2004-02 (2004).
- [94] Ultsch, A., “Clustering with SOM: U\*C”, In” *Proceedings of the International Workshop on Self Organizing Feature Maps*, 31–37, 2005.
- [95] Zelnik-Manor, L. & Pietro Perona, P.. “Self-tuning spectral clustering”. *Advances in Neural Information Processing Systems*, 2004.
- [96] Blake, C. & Merz, C., “UCI Repository of Machine Learning databases, <http://www.ics.uci.edu/~mllearn/MLRepository.html>, Department of Information and Computer Science, University of California, Irvine, California, 55, 1998.
- [97] Sneath, P. H. A. & Sokal, R. R., “Numerical taxonomy. The principles and practice of numerical classification”, *Taxon*, 12(5), 190–199, 1963.
- [98] Huang, Y. J., Powers, R. & Montelione, G. T., “Protein NMR recall, precision, and F-measure scores (RPF scores): structure quality assessment measures based on information retrieval statistics”, *Journal of the American Chemical Society*, 127(6), 1665–1674, 2005.
- [99] Xu, R. & Wunsch, D., “Survey of clustering algorithms”. *IEEE Transactions on Neural Networks*, 16(3), 2005.

- [100] Irani, J., Pise, N. & Phatak, M., “Clustering techniques and similarity measure used in clustering: a survey”, *International Journal of Computer Applications*, 134(7), 9–14, 2016.
- [101] <http://www.cs.cmu.edu/afs/cs/academic/class/15210-s15/www/lectures/mst-notes.pdf>
- [102] <https://www.baeldung.com/cs/kruskals-vs-prim-s-algorithm>
- [103] Wu, B. Y. & Chao, K.M., “Spanning trees and optimization problems”. CRC Press, 2004.
- [104] Gan, G., Ma, C. H. & Wu, J., “Data clustering: theory, algorithms, and applications”, *ASA-SIAM Series on Statistics and Applied Probability*, SIAM, 2007.
- [105] Abubaker, M. B. M., “Efficient Data Clustering Algorithms”. The Islamic University. <http://hdl.handle.net/20.500.12358/18774>, 2011.
- [106] Moore, D. S. & McCabe, G. P., “Introduction to the Practice of Statistics”, 3rd Edition, W. H. Freeman, New York, 1999.
- [107] Anscombe, F. J., “Rejection of outliers”. *Technometrics*, 2, 123–147, 1960.
- [108] Barnett, V. & Lewis, T., “Outliers in statistical data. 3rd Edition, John Wiley & Sons, Kluwer Academic Publishers, Boston/Dordrecht/London, 1994.
- [109] Abdallah, J. & Astal, A.L., “Comparison of methods for detecting outliers in medical data, <http://www.alazhar.edu.ps/arabic/He/files/20154009.pdf>, 2018.
- [110] Singh, K. & Upadhyaya, S., “Outlier detection: applications and techniques”. *International Journal of Computer Science Issues*, 9(1), 307, 2012.
- [111] Chandola, V., Banerjee, A. & Kumar, V., “Outlier detection: a survey”. *ACM Computing Surveys*, 14, 15, 2007.

- [112] Kannan, K.S., Manoj, K. & Arumugam, S., “Labeling methods for identifying outliers”. *International Journal of Statistics and Systems*, 10(2), 231–238, 2015.
- [113] Kolbaşı, A. & Ünsal, A., “A comparison of the outlier detecting methods: an application on Turkish foreign trade data”. *Journal of Mathematics and Statistics*, 5, 213–234, 2019.

# APPENDICES

**Appendix A:** Distance Metrics

**Appendix B:** MST Algorithms

**Appendix C:** Data Preparation

**Appendix D:** Outliers

## Appendix A – Distance Metrics

In this appendix, some of the commonly used metrics in data clustering, are explained, which comprise the Mahalanobis distance, the Cosine distance, and the Minkowski distance.

**Mahalanobis distance:** Mahalanobis metric eliminates the distance distortion that is produced by the linear combinations of the data attributes. The formula of this metric is defined as shown below:

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

Here,  $\Sigma$  represents the covariance matrix of the dataset. If  $\Sigma = \text{Identity}$ , then the Mahalanobis distance will be the same as the Euclidean distance. The Mahalanobis distance is invariant under nonsingular transformation, which is an important property of this metric. Depending on the number of objects that are in the dataset, it may suffer from the high computation required to determine the covariance matrix [99].

**Cosine distance:** In the case of the Cosine distance, the distance between two vectors is calculated using the cosine of the angle between two vectors. The formula for this metric can be defined as shown below [99, 100]:

$$d_{cos}(X, Y) = \frac{X \cdot Y}{|X||Y|},$$

Here,  $|\cdot|$  is the (2-norm) and  $X \cdot Y$  is the dot product.

**Minkowski distance:** A Minkowski distance or metric is utilized to measure the similarity or dissimilarity between two objects,  $X$  and  $Y$ , in a dataset. This general metric is defined as shown below (p-norm):

$$d_p(X, Y) = \left( \sum_{i=1}^d |X_i - Y_i|^p \right)^{1/p}, p \geq 1,$$

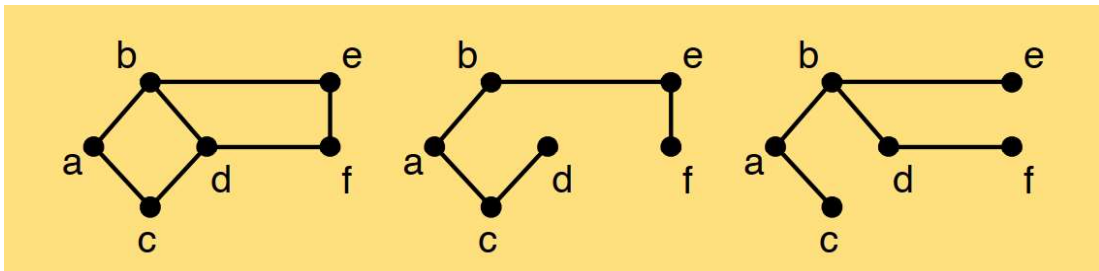
Here,  $d$  is the dimension of the  $X (x_1, x_2, \dots, x_d)$  and  $Y (y_1, y_2, \dots, y_d)$  objects. Hence, when  $p = 2$  (2-norm), the metric  $d_p$  is called the Euclidean distance. While, when  $p = 1$  (1-norm), then  $d_p$  is called the Manhattan distance or city the block distance, which

represents the shortest path between  $X$  and  $Y$ , where each segment of the path is parallel to a coordinate axis. It is called the maximum distance when  $p = \infty$  ( $\infty$ -norm), where the distance between  $X$  and  $Y$  corresponds to the maximum distance between the projections of  $X$  and  $Y$  onto each of the  $d$  coordinate axes [100].

## Appendix B – Common MST Algorithms

For a given graph that is both weighted and undirected, the MST represents a tree that extends across the graph and then minimizes the cumulative weight of the edges that are in the tree. In the appendix herein, first, the spanning tree and MSTs will be described precisely, and then the two sequential algorithms (Kruskal and Prim algorithms) and one parallel algorithm (Boruvka algorithm) that will be used to construct the MST will be explained. These algorithms all utilize a cutting property that is essential, which will also be defined.

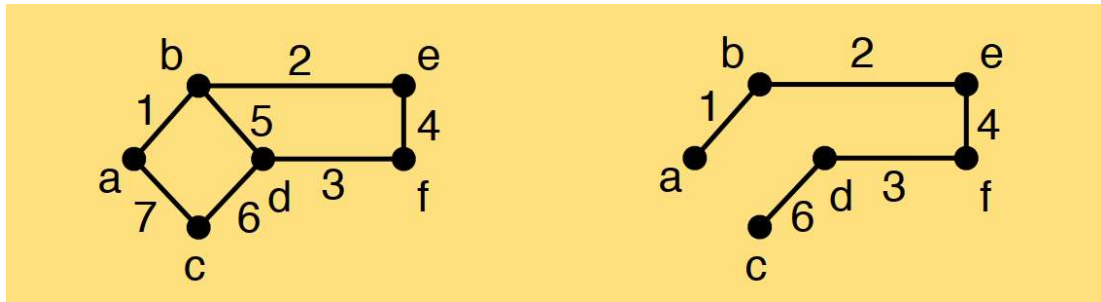
**MST definition:** If we want to find a subset of edges which form a tree using a given connected and undirected graph, while at the same time, ‘touching’ all of the vertices of the graph. This kind of tree is known as a spanning tree. Although a graph might comprise a number of spanning trees, they will all have  $|V| - 1$  edges and  $|V|$  vertices [101].



**Figure B.1** Two possible spanning trees on the right and a graph on the left [101]

The basic way to create a spanning tree is to conduct a graph search. In the DFS tree of a DFS, a spanning tree is formed by the path from a given source to all of the vertices. Similarly, by adding each of the edges that leads to an unvisited vertex to the tree being discovered, it is possible to build a spanning tree that is based on BFS. Thus, to construct spanning trees, BFS and DFS are considered as work-efficient methods. However, they are not good parallel algorithms, as was mentioned previously. Another method that can be used to create a spanning tree is the utilization of graph contraction, which can be performed in parallel. The aim throughout the algorithm is to make use of star contraction and then add all of the edges that were selected to define the stars on the spanning tree. It should be noted here that there will be numerous spanning trees

for each graph. For weighted graphs, the interest lies in obtaining a spanning tree that has the minimum total weight, that is, the summed weights of its edges [101].



**Figure B.2** Weighted graph on the left and its spanning trees on the right [101]

### Algorithms for Minimum Spanning Trees

There are several algorithms that are available for use in the creation of MSTs. However, all of them are based on a common underlying property, i.e. the light-edge, as cuts in a given graph. This property instinctively states that if the graph given is divided into two parts, then the minimum edge that is between these two divided parts must be in the MST. Therefore, the light-edge allows for the algorithmic identification of the edges of a MST. Furthermore, the light-edge possesses an important implication, where any edge that crosses a cut and has a minimum weight can be added up directly to the MST [101]. In fact, the Kruskal and Prim algorithms, which will be discussed in this appendix, will indeed take advantage of this. As an example, the Kruskal algorithm greedily builds the MST as a result of adding the overall minimum edge. On the other hand, through consideration of a cut between the current MST and the rest of the graph, the Prim algorithm can incrementally grow an MST. The Kruskal and Prim algorithms will be briefly reviewed in the next section.

#### Kruskal Algorithm:

The key idea behind the Kruskal algorithm is to sort the edges depending on their weight. Then, we begin to take edges on the basis of the lowest weight, one-by-one. In a situation where we take an edge, and if this edge forms a cycle, then this edge will not be included in the MST. The edge of this MST will otherwise be included in the tree. The challenge here is to easily identify the cycles. To achieve that, a different

type of data structure can be used, such as the disjoint-set data structure. This kind of data structure helps us to easily combine two nodes into a single component. It also enables us to quickly check whether the two nodes have previously been merged together. Consequently, before adding an edge, we need to first check if both ends of the edge have been combined previously. If so, the edge will not be included in the MST. Otherwise, the edge needs to be added to the MST, and then both of the nodes need to be merged together into the disjoint-set data structure [102].

---

```

Data: edges: List of edges of the graph
Result: Returns the cost and the edges of the MST
sort(edges);
totalCost  $\leftarrow$  0;
for edge  $\in$  edges do
    if  $\neg$  dsu.isMerged(edge.u, edge.v) then
        totalCost  $\leftarrow$  totalCost + edge.weight;
        mst.add(edge);
        dsu.merge(edge.u, edge.v);
    end
end
return totalCost, mst;

```

---

**Figure B.3** Kruskal algorithm [102]

First, we sort the list of edges according to their weight in ascending order. Second, we iterate over all of the edges. We check each edges to determine whether its ends had previously been merged. If so, this edge will be ignored. Otherwise, the total cost of the MST will be increased and this edge will be added to the resulting MST. In the disjoint-set data structure, we also combine both of the ends of this edge. Finally, we only return the taken edges and the total cost of the calculated MST.  $O(E \cdot \log(V))$  is the complexity of the Kruskal algorithm, where  $V$  is the number of vertices inside of the graph and  $E$  is the number of edges. The cause for this complexity is the result of the cost of sorting [102, 103].

### **Prim Algorithm:**

The Prim algorithm is essentially an updated version of the Dijkstra algorithm. We first choose to start with a node and then add all of the neighbors of this node to a priority queue. After that, several steps are performed. In each step, utilizing the edge with the lowest weight, we extract the node that we were able to reach. Thus, the

priority queue must include the node and the weight of the edge that enabled us to reach this node. Moreover, based on the passed weight, the nodes that are inside of it must also be sorted. For every extracted node, we apply it to the resulting MST and change the total cost of the MST. Furthermore, we add all of the neighbors for each of the extracted nodes to the queue as well. We should ensure that each of the nodes will only appear once in the queue, so as to achieve a better complexity. We may, for instance, use the `addOrUpdate` function, which takes the edge that led us to this node and the node with the weight. If the node is already inside of the queue, and the new weight is greater than the one that was saved, the function will delete the old node and only add the new one. Otherwise, it simply adds it along with the given weight when the node is not in the queue [103].

---

```

Data: G: The given graph
        source: The node to start from
Result: Returns the cost of the MST
totalCost  $\leftarrow$  0;
included  $\leftarrow$  {false };
Q.addOrUpdate(source, 0,  $\Phi$ );
while  $\neg$  Q.empty() do
    u  $\leftarrow$  Q.getNodeWithLowestWeight();
    totalCost  $\leftarrow$  totalCost + u.weight;
    if u.edge  $\neq$   $\Phi$  then
        | mst.add(u.edge);
    end
    included[u.node]  $\leftarrow$  true;
    for v  $\in$  G.neighbors(u.node) do
        if  $\neg$ included[v.node] then
            | Q.addOrUpdate(v.node, weight(u.node, v.node),
            | v.edge);
        end
    end
end
return totalCost, mst;

```

---

**Figure B.4** Prim algorithm [103]

At the beginning, we add the source node to the queue without an edge and with a zero weight. The  $\emptyset$  symbol is used here to refer that we stored an empty value. In addition, to zero, we initialize the overall cost and denote all of the nodes as not yet being involved in the MST. Then, several steps are performed. The node that has the lowest weight is deleted from the queue, in each step. For each of the nodes that is removed, the weight of the extracted edge raises the costs of the MST. Furthermore, in the case of the edge of the node extracted, we add it to the resulting MST. We iterate over the neighbors after we finish handling the node that was extracted. We use the function

addOrUpdate in the case where the neighbor has not yet been included in the resulting MST, in order to add this neighbor to the queue. Furthermore, we add the edge itself, in addition to the weight of the edge.  $O(E+V \cdot \log(V))$  is the complexity of the Prim algorithm, where  $V$  is the number of vertices inside of the graph and  $E$  is the number of edges [103, 104].

## Appendix C – Data Preparation

### Dealing with missing values

In a matrix notation, the collected data can be interpreted using two modes. Therefore, the missing values could be in columns or records. In columns, we delete the column if most of data are missing. In records, we delete the record if the majority of the data are missing. On the other hand, if the values that are missing in the data are not very high, then these missing values need to substituted, as shown below:

1. Substitute the missing values with the average value of all of the column data, which this is performed before running clustering algorithms.
2. Using a clustering algorithm before substituting with the average value:
  - a. Identify the column (variable) that is closest to the column which has the missing values. Then replace these missing values with the average value of the column that is most similar.
  - b. Construct clusters that are dependent on all of the variables, then replace the missing values by taking the average value of the objects belonging to the same cluster.

### Normalization

Dataset attributes can be obtained from various scales. Hence, the attributes with large scales may cause a bias. Therefore, we need to normalize the data, such that all of the dataset attributes of the dataset would be in the same scale. Furthermore, various types, such as ordinal, nominal, interval, categorical, or ratio, may also be used to describe the objects in the dataset. These objects can be clustered into different approaches. Converting the attributes of different types into attributes of the same type is one of these approaches [104].

### Sampling

Sampling can be utilized to select a portion of large datasets because it is too costly and time consuming to process the full dataset. The simplest method of sampling can

be done randomly. However, to do more effective sampling, it should approximately reflect the same characteristics as the original dataset [105].

### **Dimensionality reduction**

A significant amount of memory and time is needed to analyze a high-dimensional dataset. Dimensionality reduction methods can be classified into two major types: feature selection and feature transformation. In the feature selection method, a subset of meaningful dimensions is selected from the original high dimensions of the entire dataset. In the feature transformation method, the original high-dimensional space is either non-linearly or linearly projected into a low-dimensional space [105].

## Appendix D – Outliers

Any observation in a given dataset that is inconsistent with the rest of the observations is called an outlier. Outliers are sometimes caused by discordant values, the name of contaminants, rogue observations, or by spurious. The outlier is inconsistent in the context that the possible future behavior of datasets coming from the same source is not indicative. Outliers were also described by Moore and McCabe [106] as an observation that exists beyond the overall pattern that a distribution possesses.

**Causes of Outliers:** Several different causes can allow outliers to arise. In 1960, Anscombe classified outliers into two main categories, as outliers that were created by the inherent variability of the data, and outliers that were caused by errors in the data [107]. Barnett & Lewis [108] stated that not all of the illegitimate scores will show up as outliers, and not all of the outliers will be illegitimate contaminants. Therefore, for a given dataset, it is essential to consider the range of causes that may be responsible for the existence of the outliers. Whenever we encounter outliers in a given dataset, the ideal way to tackle them is to realize the reason for their existence. Thus, based on the reason of their occurrence, we can specify the most suitable method to deal with them. Moreover, other causes for the existence of outliers can be categorized as experimental error, data processing error, data entry errors, measurement error, intentional outlier or motivated miss-reporting, and sampling error [109].

**Data entry errors:** Outliers in data may be caused as the result of errors related to humans, including errors that are caused during data recording, entry, or collection. Commonly, correction of these types of errors can be performed through referring to the original subjects or, if necessary, even the documents, and then correcting the incorrect value. Recalculation or re-estimation of the correct answer is also another choice that is available.

**Measurement error:** This kind of errors is the most public source of outliers. This is produced when the utilized measurement mechanism turns out to be faulty. For instance: there are 5 weighing machines; one is faulty and the other four are correct. Thus, the weight calculated by those on the faulty machine will be lower/higher than

those of the rest in the group. The weights calculated by the faulty machine can produce outliers.

**Experimental error:** Experimental error is another reason for outliers. For instance: a runner missed out on focusing on the ‘Go’ call, which caused him to start late in a 100-m sprint comprising 7 runners. Therefore, the runner has a run time that is more than that of the other runners. He will be an outlier for his total run time.

**Intentional outlier or motivated miss-reporting:** This is usually seen in self-reported measures that contain sensitive data. For instance: teens usually under record the amount of alcohol that they drink. The actual value is only reported by a fraction of them. Here, since most of the teenagers are under reporting their alcohol consumption, the actual values may look like outliers.

**Data processing error:** Usually, we extract data from several sources each time we perform data mining. However, some extraction or manipulation errors may lead to the existence of outliers in the dataset.

**Sampling error:** For instance, the height of an athlete needs to be measured. Mistakenly, a few basketball players are included in the sample. This inclusion in the dataset is would potentially cause the existence of outliers.

## **Types of Outliers**

In the following three groups, the types of outliers are classified as: point, contextual, and collective outliers [110].

### **Point Outliers**

The instance is called a point outlier in a given dataset if it can be considered irregular in comparison to the rest of instances that are in the dataset. Outliers such as this have been the focus of the majority of the outlier detection research. It is considered the simplest type of outlier. If we consider the identification of credit card fraud as a real-life case study, if we use the credit card transaction dataset of an individual and assume that the data definition possesses only one feature, which is the amount that was spent. For a transaction where a person has spent a very high amount compared with the

normal range of expenditure, this instance is considered as a point outlier [110]. In Figure D.1 the point that is marked as O1 and the point that is marked as O2 deviate significantly from the labeled regions, G1 and G2.

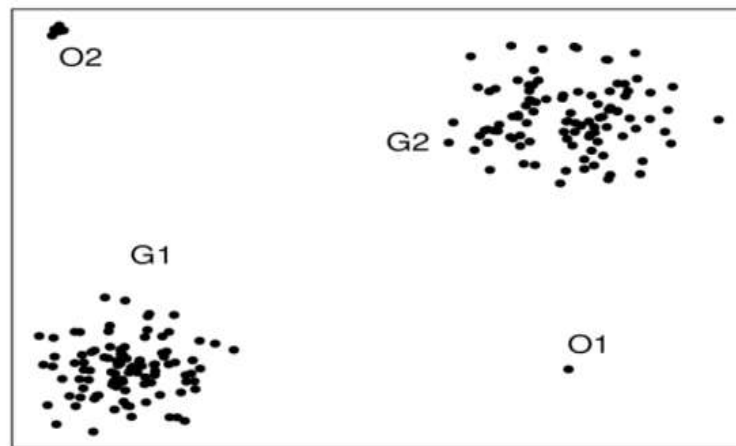


Figure D.1 Example of two-dimensional outliers [109]

### Contextual Outliers

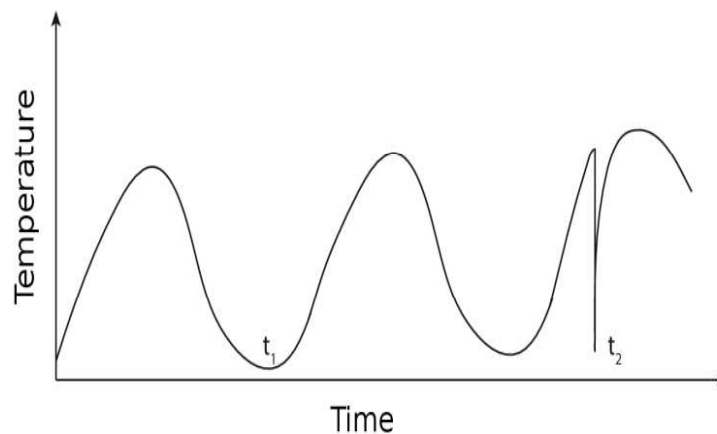
A contextual outlier exists in a given dataset if there is an instance that is described as anomalous in a particular context. The definition of such a context is determined depending on the structure of the data set and it has to be defined as being a part of the problem formulation. Two sets of attributes are used to define each data instance, comprising behavioral attributes and contextual attributes [110].

- Behavioral attributes:** The non-contextual characteristics of an instance in a given dataset is defined by the behavioral attributes. The amount of rainfall at any location is an example of a behavioral attribute, using a spatial data set, which describes the average global rainfall of the whole world. The anomalous behavior is determined in a specific context using the values of the behavioral attribute. In a given context, a data instance might be a contextual outlier, but in a different context, an identical data instance (in terms of behavioral attributes) could be considered normal. This property is important when it comes to identifying behavioral and

contextual attributes. For instance, in the detection of credit card fraud using the time of purchase as a contextual attribute. Suppose that a person that usually has a \$100 weekly shopping bill, excluding the week of Christmas holiday, when it might reach as much as \$1000. Hence, a new \$1000 purchase in one week in July is then considered as a contextual outlier, because in the context of time, it does not comply with the normal behavior of the person (while they are considered to spend the same amount during the week of Christmas holiday). The importance of contextual outliers in the target application domain determines the choice of applying the contextual outlier detection technique. If the contextual attributes are readily available and defining a context is straightforward, then applying a detection technique for contextual outliers makes sense. However, if defining a context is not easy, it becomes difficult to apply such techniques [110, 111].

- **Contextual attributes:** To define the neighborhood (or context) of an instance in a given dataset, the contextual attributes are utilized for this purpose. For instance, the contextual attributes in a spatial dataset are the latitude and longitude of the location. The contextual attribute in a time series type of dataset is the time, which defines the position of an instance on the entire sequence [109, 110].

In the figure below, the points labeled  $t_1$  and  $t_2$  represent the same temperature value. Here, point  $t_1$  is not considered as an outlier, while point  $t_2$  is considered as an outlier.

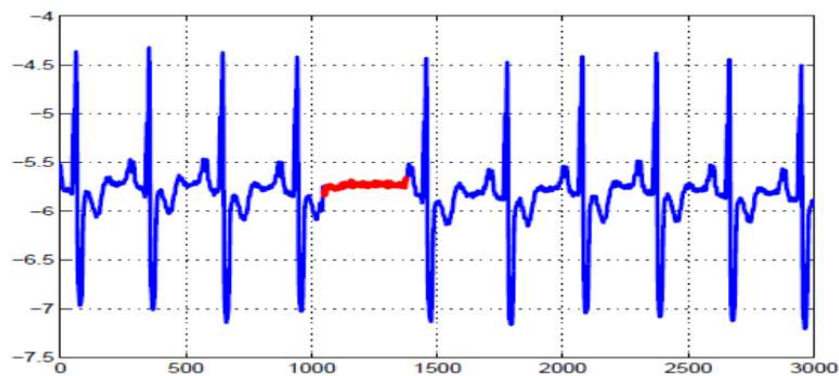


**Figure D.2** Example of contextual outlier [110]

## Collective Outliers

If there is an anomaly in a collection of related data instances with respect to the whole dataset, it is called a collective outlier. In a collective outlier, the individual data instances may not be outliers by themselves; however, the anomalous exists in the context of their occurrence together as a collection. It may be noted that the successive occurrence of a low value for a long time is also considered as an outlier, while a low value by itself is not considered as an outlier. It should be also noted that collective outliers can only occur in datasets in which the data instances are related, while point outliers may occur in any dataset. The occurrence of contextual outliers in a given dataset, however, depends on the availability of the context attributes. A collective outlier or a point outlier can also be a contextual outlier, if examined with respect to a context. Thus, a collective outlier detection problem or a point outlier detection problem can be converted to a detection problem of a contextual outlier by integrating the context information [109, 110].

Figure D.3 shows an example of a human electrocardiogram signal output. The highlighted area (in red) refers to the existence of an outlier, because for an abnormally long time, the signal has had the same low value (corresponding to an atrial premature contraction). It may be noted that the successive occurrence of low values for a long time is an outlier, while the low value by itself is not an outlier.



**Figure D.3** Human ECG output showing a collective outlier corresponding to an atrial premature contraction [110]

## Effects of Outliers

The results of the statistical and modeling data analysis can be drastically changed by the outliers. In a given data set, there are several negative impacts of outliers:

- It decreases the power of statistical tests and increases the error variance.
- Outliers can decrease normality if they are non-randomly distributed.
- Outliers can also influence the basic assumption of ANOVA, regression, and the assumptions of other statistical models.
- The standard deviation is also increased by the effects of outliers.

## Common Outlier Detection Methods

Most of the outlier detection methods existing in the literature solve a particular issue, which is affected by several aspects, such as the type of outlier to be detected, availability of the labeled data, and the nature of the data. Often, the application domain in which the outliers need to be identified define these aspects. The Tukey method was utilized in this research as a labeling outlier detection technique. Some of the most common outlier labeling approaches will be explained in the next sections.

## Z-Scores

The Z-score, or standard score, is a method of labelling a data point based on its relationship to the standard deviation and mean of a set of points. The Z-score of an observation is described as in the following formula:

$$Z_i = \frac{x_i - \bar{x}}{sd}$$

where  $sd$  is the standard deviation and  $X_i \sim N(\mu, \sigma^2)$ .

The aim of taking Z-scores is to remove the scale and location effects of the data, so that multiple datasets can be directly compared with each other. Moreover, the concept behind the Z-score as an outlier detection method is that, once the data is rescaled and centered, anything that is too far from zero would be marked as an outlier (typically the threshold is a Z-score of  $-3$  or  $3$ ) [112, 113].

### The Tukey Method (Boxplot)

The Boxplot or Tukey method was proposed by Tukey et al. (2004), and is one of the graphical techniques that is utilized most commonly for analyzing a univariate data set. This graphical tool is simple and easy to use for displaying information related continuous univariate data, such as upper quartile, lower quartile, the median, upper extreme, and lower extreme of a given data set. In order to find outliers using this method, the interquartile range is utilized to filter out very small or very large numbers [112]. The formulas are as follows:

High outliers =  $Q1 + 1.5 * Q3 - Q1$ , Low outliers =  $Q1 - 1.5 * Q3 - Q1$

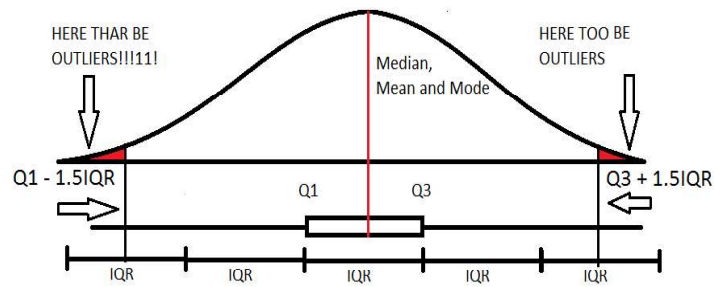
OR

High outliers =  $Q1 + 1.5 * IQR$ , Low outliers =  $Q1 - 1.5 * IQR$ ,

where IQR is the interquartile range, Q1 is the first quartile, and Q3 is the third quartile,

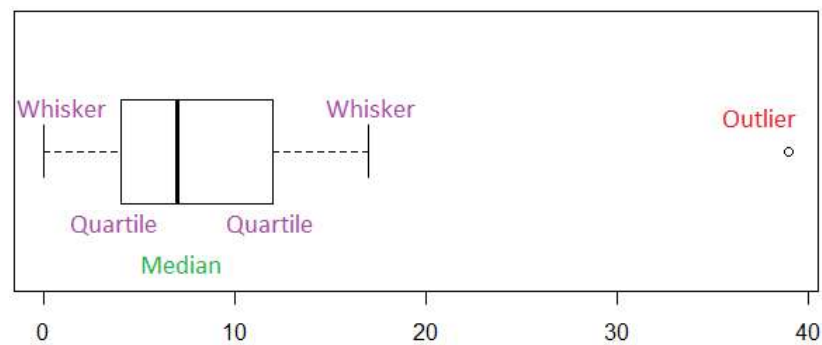
Two values or ‘fences’ are given using these equations. Thus, a fence is used to cordon off the outlier values from all of the values that are found in the bulk of the given data. The phases for detecting the outliers using the IQR are given as follows:

- Phase 1: Find the median and IQR.
- Phase 2: Find Q1 and Q3. Q1 represents the median in the lower half of the data. While, Q3 represents the median for the upper half of the data. In order to get the IOR, we need to subtract Q1 from Q3.
- Phase 3: In order to get the lower fence, we need to calculate  $1.5 * IQR$  and subtract it from Q1.
- Phase 4: Similarly, to get the upper fences, we need to calculate  $1.5 * IOR$  and add it to Q3.
- Phase 5: The last step is to add the fences to the data to identify the outliers.



**Figure D.4** Standard graph showing the median, quartiles, and outliers [112]

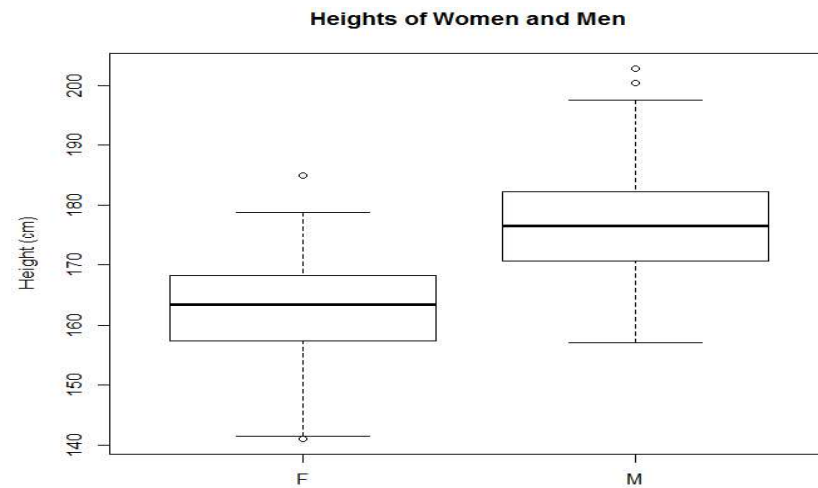
For a given dataset  $\{0, 1, 2, 4, 5, 5, 7, 10, 10, 12, 13, 17, 39\}$ , the boxplot looks like this:



**Figure D.5** Example of boxplot for a given dataset [112]

Here,

- The region between Q1 and Q3 is shown in the box from 3 to 12.5.
- The median is the line going through the middle of the box at 7.
- The whiskers are the lines going out beyond the ends of the box. They determine the range of values that are not considered as outliers.
- 17 is the biggest value before the upper limit of 26.75 is hit. Thus, 17 represents the upper whisker. While, the lower whisker goes to the lowest value.
- An outlier is shown as an individual dot at 39.
- The furthest outliers on both sides are the minimum and maximum.
- On a side in the box, if no outliers exist, the end of the whisker is that maximum or minimum.
- We can also compare the distributions of two samples using boxplots. The heights of adult women and men are shown in Figure D.6 below, as an example.



**Figure D.6** Comparing the distributions of two samples using boxplots [112]

As shown in Figure D.6, there is some overlap between the two distributions. It is also noted that men are taller than women in general. However, the variance of the men and women is about the same. Thus, both distributions appear to be symmetrical.