Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Geoscience and Petroleum

Nghia Van Nguyen

Hierarchical Clustering Combination and an Algorithm to Solve the Tanglegram Layout Problem

Master's thesis in Petroleum Geosciences and Engineering
Supervisor: Carl Fredrik Berg
Co-supervisor: Damiano Varagnolo, Kurdistan Chawshin
June 2021

NTNU
Norwegian University of
Science and Technology

Nghia Van Nguyen

Hierarchical Clustering Combination and an Algorithm to Solve the Tanglegram Layout Problem

Master's thesis in Petroleum Geosciences and Engineering
Supervisor: Carl Fredrik Berg
Co-supervisor: Damiano Varagnolo, Kurdistan Chawshin
June 2021

Norwegian University of Science and Technology
Faculty of Engineering
Department of Geoscience and Petroleum

NTNU
Norwegian University of
Science and Technology

Abstract

In the field of petroleum geology, the implementation of machine learning approaches has been showing considerable potential for supplementing and replacing conventional and labor-intensive approaches. For example, the idea of using supervised learning algorithms has been proposed by some authors to predict and classify lithofacies. In general, though, there are some situations where merging the machine learning approaches with human inputs is valuable. In this master thesis, we focus on one specific instance of this need for fusing interpretations from machines with interpretations from humans.

More specifically, we consider that from the processes of classifying lithofacies in a supervised learning based way, the learning algorithms return a confusion matrix (i.e., what has been misclassified with what, and how often) which can be used as the input for performing hierarchical clustering (i.e., considering two lithofacies that are often mixed up by the machine as similar). This strategy of doing hierarchical clustering returns a tree-like diagram called a dendrogram. At the same time, humans can also produce an analogous dendrogram: geologists may indeed say that two lithofacies are similar by taking other types of data from rock properties into consideration. This human-based dendrogram might of course be topologically different from the former machine-based.

This thesis focuses on the problem of how to utilize both of these dendrograms so to improve the understanding of the geologists, and in this way considering both the numerical methods together with the conventional one as sources for understanding how to classify the lithology of some wells. To the best of our knowledge, combining multiple hierarchical clusterings (or dendrograms) is a practical way to address this problem, but it has been overlooked in petroleum geosciences.

The major aim of this study is thus to investigate how to combine hierarchical clusterings for a specific case. Specifically, the Min-trAnsiTive Combination of Hierarchical clusterings (MATCH) method is used to combine a dendrogram from computational method with one from a geologist's perspective. After the final dendrogram is formed, it is visually compared with the primary dendrograms by placing them in a diagram called tanglegram. It is well known that this tanglegram shows a propensity to be significantly tangly, which is known as a tanglegram layout problem. Solving this problem has been received much attention in the past decade, and several methods have been proposed, but they seem to be oversimplistic and inefficient in terms of producing the least tangly tanglegram. Therefore, in this work we develop a new untangle method that seeks to find an optimal solution for the tanglegram layout problem. The results show that the MATCH method works well to produce the combined dendrogram, and the new untangle method succeeds in finding better layouts of tanglegrams compared to one of the other methods.

Preface

This master thesis is written as a part of my Master of Science degree in Petroleum Geosciences and Engineering at the Norwegian University of Science and Technology (NTNU). It is a continuation of the specialization project submitted during the fall semester of 2020 where an attempt to develop an algorithm for improving entanglement values of tanglegrams was made.

I would like to express my deep sense of gratitude to Professor Carl Fredrik Berg of the department of Geosciences and Petroleum at NTNU for his motivation and expertise. He was always willing to offer constructive advice and positive encouragement, put hypothetical questions to me so that I can comprehend what I could do better.

Further, I wish to thank Damiano Varagnolo of the department of Engineering Cybernetics at NTNU for his supervision. He was the one that contributed the inspirational idea for my study in the beginning. Without his enthusiasm and insights, it would have been more challenging to start my master thesis.

I would like to thank my co-supervisor Kurdistan Chawshin of the department of Geosciences and Petroleum at NTNU for her crucial assistance. She did a great job with her PhD program which provided me with essential data for the genesis of my master thesis.

In addition, the technical help provided by Philipp Schlegel of the Drosophila Connectomics Group, Department of Zoology, University of Cambridge was highly appreciated. His contribution in `https://github.com/schlegelp/tanglegram` gave me the direct inspiration for my codes in Python.

Finally, I must express my heartfelt gratitude to my supportive family and friends. I am surrounded with their love and support in both times of joy and distress. I cannot thank you enough for all you have done for me.

Nghia Nguyen Van

Trondheim, June 10, 2021

Table of Contents

List of Tables

List of Figures

Abbreviations

TL	=	tanglegram layout
HCC	=	hierarchical clustering combination
CD	=	cophenetic difference
PD	=	path difference
CMD	=	cluster membership divergence
PMD	=	partition membership divergence
SMD	=	sub-tree membership divergence
MNED	=	maximum number of edge distance
AL	=	abstract level
MATCH	=	min-transitive combination of hierarchical clusterings
S&U	=	shuffle and untangle
CT	=	computerized tomography
ND	=	numerical dendrogram
GD	=	geologist dendrogram
CD	=	combined dendrogram

Chapter 1

Introduction

1.1 Background

Machine learning is a part of Artificial Intelligence, which builds a mathematical model based on sample data in order to perform various tasks such as predictions or decisions (Mitchell, 1997). The term ”machine learning” was coined by an American pioneer in the field of computer gaming and artificial intelligence named Arthur Samuel in 1959 when he wrote a computer program to play checkers. His idea was to develop a program which could learn to play checkers better than the programmer. It was achieved by having the program play thousands of games against itself. By 1970, the performance of the program had been comparable to the performance of a respectable amateur player (Schaeffer, 2008). Since then, machine learning has been developed and used in a wide variety of different scientific fields and real-world businesses, such as text or document classification, computer vision applications, computational biology applications, and many other problems (Mohri et al., 2018). A striking feature of machine learning is that it is advantageous for solving problems that are challenging to be managed manually due to the huge amount of data. Although problems in machine learning can be of various types, they can be typically divided into three categories:

1. Supervised learning;
2. Unsupervised learning;
3. Reinforcement learning.

In this thesis, only unsupervised learning has been considered. Unsupervised learning is a type of algorithm that learns underlying structures from unlabeled data. The users have access to input features only and do not have an associated target variable. It is radically different from supervised learning in which the users take a set of labeled examples as training data and have access to the target variable. Since labeled examples may not exist in unsupervised learning, it is not a trivial task to quantitatively evaluate the performance of unsupervised algorithms (Mohri et al., 2018). This means that data interpretation is

more important than the evaluation of the method. Two main classes in unsupervised learning are dimensionality reduction and cluster analysis. The former helps to determine how the data is distributed in the space, and the later use the input features to divide a set of unlabeled data into natural groupings.

Cluster analysis, or clustering, is utilized in a variety of engineering and scientific disciplines, including image analysis, bioinformatics, computer graphics, and machine learning (Chernoff, 1975). In fact, the terminology differs in various scientific fields. For example, in biology and ecology, the term ”numerical taxonomy” is used as a substitute for cluster analysis, while some social scientists refer it as ”typology” or ”classification analysis” (Chernoff, 1975). Although being called in different terms, cluster analysis' goal is unique, which is to partition, or segment a set of objects into groups (clusters) such that objects belonging to the same cluster are more similar to each other than to those in other clusters. Cluster analysis has been employed as an effective tool to reveal structure and relations in the data. For example, clustering is often used in phylogenetic analysis and comparative genomics to determine ancestral species, design vaccines, and relate the evolution of species by forming phylogenetic trees based on the similarity in biological macromolecules such as DNA, RNA, and protein. For example, since the COVID-19 pandemic caused by Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2), biologists have been using cluster analysis to examine the origin and continuing evolution of SARS-CoV-2 (Figure 1.1). In reality, evolutionary study of multiple SARS-CoV-2 variants helps scientists to better understand how transmissible the viruses might be and the effectiveness of currently authorized vaccines against them. In petroleum geology, cluster analysis is mainly used in stratigraphic zonation of logging data and lithology classification. The next decade is likely to witness a considerable rise in applications of cluster analysis in all kinds of scientific fields.

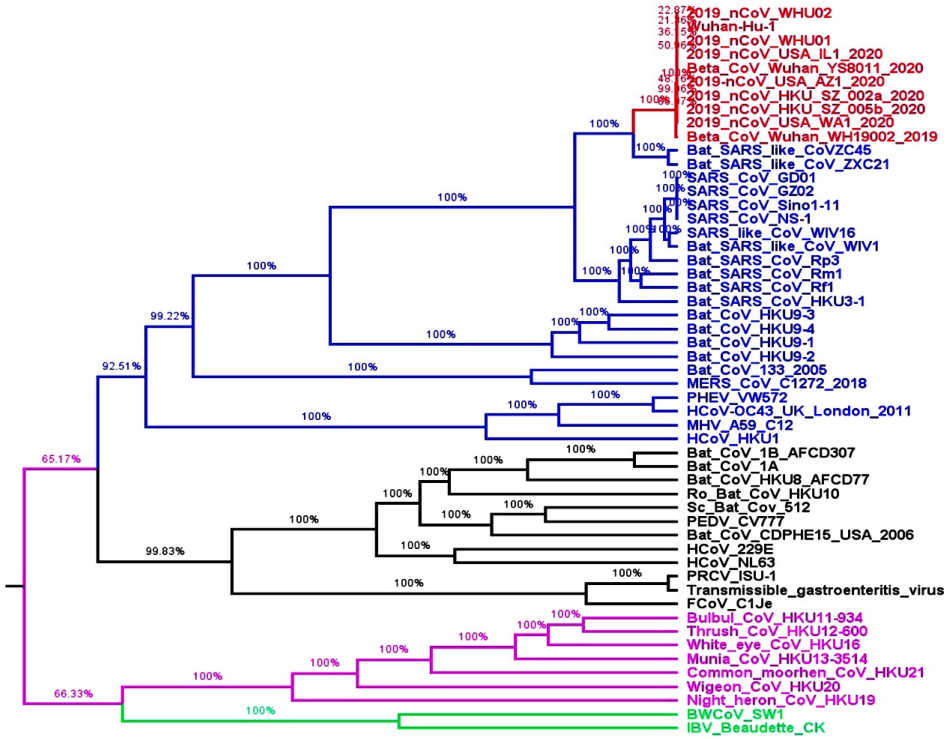


Figure 1.1: Evolutionary phylogenetic tree analysis of Coronaviruses (Rehman et al., 2020): Colours represent different genera of Coronaviruses; black, alpha coronavirus; blue, beta coronavirus; red, SARS-CoV-2; green, delta coronavirus; purple, gamma coronavirus. The percentages are the indicators of amino acid similarities between coronaviruses. The phylogenetic tree shows that SARS-CoV-2 is a beta-coronavirus and related to the bat SARS-like coronavirus (Rehman et al., 2020).

Cluster analysis itself is not a specific algorithm, but a general task to be tackled. It can be accomplished by a number of algorithms which differ fundamentally in their definitions of clusters and how to efficiently find them. Clustering algorithms may be classified as listed below:

1. Exclusive clustering;
2. Overlapping clustering;
3. Hierarchical clustering;
4. Probabilistic clustering.

In this master thesis, we will focus on hierarchical clustering algorithm. This algorithm accounts for combining similar objects to form new clusters based on their distances so called hierarchical levels. At different hierarchical levels, different clusters are formed by merging two other clusters, which is demonstrated by using a tree-like diagram known

as a dendrogram. The dendrogram describes the order in which objects are merged from the bottom-up view or clusters are split from the top-down view. More definitions and information of this approach are provided in Chapter 2.

1.2 Objective

The first aim for this study is to seek for a proper way to combine two dendrograms, one from a hierarchical clustering algorithm, and one from a geologist description. The hierarchical clustering combination (HCC), naturally, is not as easy as combining multiple classifiers. The main reason is that there are relationships or degrees in similarity among objects, which is demonstrated by the hierarchical levels in the dendrogram. In addition, the true number of clusters in hierarchical clustering is rarely known beforehand. These problems make it impossible to just combine two dendrogram like conjoining two pieces of a picture. It leads to a demand of extracting information from the dendrogram such that this information can be used as the representative of the dendrogram as well as the input for a HCC algorithm. In fact, it is common knowledge that any dendrogram can be presented by an intermediate matrix presentation called a description matrix, but it is not always true that a dendrogram could be recovered by an arbitrary matrix. Thus, we need to identify a proper description matrix that is suitable for the HCC algorithm.

The second aim of the research is to improve the visualization of dendrogram comparison. Specifically, the combined dendrogram will be compared to each of the based dendrograms by putting them face-to-face in the same diagram named a tanglegram. In a tanglegram, identical objects in two dendrograms are connected by lines. Therefore, a tanglegram is potentially tangly, which is partly revealed by the name of this special diagram. This problem is called the tanglegram layout (TL) problem. The degree of being tangly significantly depends on the number of objects in dendrograms and how different the hierarchies are. Several untangle methods are proposed and implemented in the R package $dendextend$ to solve the TL problem (Galili, 2015). Nevertheless, they are prone to not have the ability to find out the least tangled layout (drawing, embedding) if the number of objects is large. Hence a new untangle method is introduced in this thesis with an expectation that it can return a tanglegram layout that is less tangly than the result from a untangle method in R.

The main objectives of this master's thesis can be summarized as follows:

1. to examine and find out a type of description matrix that can be used for HCC;
2. to apply a HCC method to merge two dendrograms into one;
3. to develop a new untangle method which can return a tanglegram drawing that is less tangled than the optimal layout from a untangle method in R language.

1.3 Approach

With regards to the lithology classification problem, one can use the confusion matrix obtained from a machine learning lithology prediction as the input for hierarchical clustering

algorithm to create a dendrogram. This dendrogram is considered as a result of the numerical method. A geologist can also manually create a dendrogram (a tree) showing how a specific rock type is similar to the others in terms of the fundamental rock properties. These properties might include porosity, permeability, and grain size. In doing so the numerical method and the manual method of geologists (geologist method) are more likely to bring out different dendrograms, and each method has it own advantages and disadvantages. Particularly, the geologist method obtains a general view of different rock types, but it seems to be experiential, subjective and difficult to employ with a large data. Compared to it, the numerical method is more objective and can deal with complex problems. However, the shortcoming of the computational method is that it is hard to adjust the automated inner process. It poses a new problem of how to merge dendrograms from the numerical and geologist methods into one single dendrogram (combined dendrogram) which can be used as a representative of the two different methods. This problem is known as the hierarchical clustering combination (HCC) problem. In this master thesis, the procedure of HCC and its results will be shown. Furthermore, an approach in which the final dendrogram is compared to the two starting (base) dendrograms and an improvement in visualization of this approach will be presented at the end of the thesis.

1.4 Outlines

This master thesis is divided into seven chapters. *Chapter 2* includes the theoretical background of the present work, and it comprises six sections. The second section of this chapter overviews the basic concept of fuzzy set theory to help the reader to get accustomed to the definitions of crisp set, fuzzy set, fuzzy relation, and fuzzy equivalence relation. General aggregation operations on fuzzy sets and fuzzy relations are also covered in this section. In the next three sections, a deep introduction of hierarchical clustering, HCC, and dendrogram description matrices are presented. The last section in *Chapter 2* covers the concept of tanglegram and tanglegram layout problem with an example provided. This chapter might be heavy-reading due to the mathematical sophistication, but it is prerequisite for comprehending the work in the remainder of the thesis.

The methodology is outlined in *Chapter 3*. The definition of transitive closure is examined in this chapter prior to introducing the concepts and algorithms of the MATCH and square algorithms. These methods are of significance in combining hierarchical clusterings. The last section in this chapter represents a new method to solve the TL problem.

Chapter 4 deals with the results obtained from applying the methods introduced in *Chapter 3*. The first section in this chapter is devoted to the introduction of the case study in the thesis. The next section focuses on the application of the MATCH and square algorithms to combine dendrograms. The results from applying untangle methods are shown in the final section.

Chapter 5 covers the discussions of the results in *Chapter 4*. It includes a thorough observation and comparison between different dendrograms, various optimized tanglegram layouts, and distinct methods used to generate these results.

Our conclusions are drawn in *Chapter 6*.

The last chapter, *Chapter 7*, is devoted to the further work.

Chapter 2

Literature Review

2.1 Introduction

According to Meyer, Naessens, and Baets there is a one-to-one correspondence between a dendrogram and a fuzzy equivalence relation (Meyer et al., 2004). Thus, in the first subchapter, we will give brief account of some definitions about binary fuzzy relations, fuzzy equivalence relations, and crisp relations. It also covers general fuzzy complements, fuzzy intersections, and fuzzy unions, which play a central role in the calculation and implementation of the hierarchical clustering combination method in this thesis.

Hierarchical clustering is an unsupervised learning method whose goal is to group objects or patterns into groups called clusters such that the objects within each cluster are broadly similar to each other. The interpattern similarity is based on the proximity matrix which including distances between pairs of objects. To calculate these distances, a pattern matrix is required, and a distance function, which in most cases is the Euclidean distance, needs to be well defined between two patterns. Therefore, the second subchapter will cover definitions of a pattern matrix and a proximity matrix, which are used as input data for hierarchical clustering algorithms. Plus, the procedure behind hierarchical clustering algorithms and a number of terms such as linkages, ultrametric, and linkage matrices used throughout the thesis will be discussed in this subchapter. An example of Iris flower samples is provided in this chapter to illustrate definitions and generations of all kind of matrices and dendrograms.

It has been suggested that by combining the results of several hierarchical clustering algorithms, the resulting data clustering is improved (Dietterich, 2000). We will therefore introduce the definition of hierarchical clustering combination in this chapter. As we work with the dendrogram description matrices instead of figures of dendrograms to combine hierarchical clusterings, basic types of descriptors and their properties will be also introduced. Ultrametric is recognized as being one of the most important property of a dendrogram description matrix, and we need to take it into account to opt for proper description matrices to work with.

After a tanglegram is constructed from dendrograms, a need for improving the layout

of the tanglegram arises naturally. Essential definitions and equations related to tanglegram and tanglegram layout problem will be provided in the last section of this chapter. One untangle method from R language is also introduced to illustrate one solution for dealing with the tanglegram layout problem.

2.2 Fuzzy relation and Fuzzy Equivalence

2.2.1 Crisp Set

Let X be a nonempty set of all the possible elements of concern in a particular context. Each of these elements is called a member, or an element, of X, and X is called a universe set (Klir et al., 1997). To show that an object x is a member of X, we write

$$x \in X \tag{2.1}$$

If x is not an element of X, we write

$$x \notin X \tag{2.2}$$

A crisp (classical) set A of X is defined as a union of several members of the universe X. Given two crisp sets A and B on X, A is called a subset of B if every member of set A is also a member of set B (Klir et al., 1997), which is denoted by

$$A \subseteq B \tag{2.3}$$

Alternatively, one can say A is contained in B or B includes A. If A is a subset of B ($A \subseteq B$), but A is not equal to B ($A \neq B$), then we say A is a proper subset of B (Klir et al., 1997), written as

$$A \subset B \tag{2.4}$$

It means that there is at least one member of B that is not a member of A. Two sets A and B are equal, denoted by

$$A = B \tag{2.5}$$

if

$$A \subseteq B \text{ and } B \subseteq A \tag{2.6}$$

The empty set is denoted by $\emptyset$.

The Cartesian product of two sets A and B, denoted $A \times B$, is the set of all possible ordered pairs where the elements of A are first and the elements of B are second (Klir et al., 1997). That is,

$$A \times B = \{(a, b) \mid a \in A, b \in B\} \tag{2.7}$$

For example, let $A = \{a_1, a_2, a_3\}$ and $B = \{b_1, b_2\}$, the Cartesian product of A and B is

$$A \times B = \{(a_1, b_1), (a_1, b_2), (a_2, b_1), (a_2, b_2), (a_3, b_1), (a_3, b_2)\} \tag{2.8}$$

Operations on crisp sets consist of union, intersection, complement, difference, and multiplication. Let A and B be two subsets of the universe X, Klir et al. (1997) defined these operations as below:

- The union of two sets A and B, denoted by $A \cup B$, is a set containing all elements that belong to set A alone, to set B alone, or to both sets A and B. This is formally expressed by
$$A \cup B = \{x \mid x \in A \text{ or } x \in B\} \tag{2.9}$$

- The intersection of two sets A and B, denoted by $A \cap B$, is a set consisting of all elements that are both in A and B. Thus,
$$A \cap B = \{x \mid x \in A \text{ and } x \in B\} \tag{2.10}$$

- The complement of a set A, denoted by A^c, is a set of all elements that are in the universal set X but are not in A
$$A^c = X - A = \{x \mid x \notin A \text{ and } x \in X\} \tag{2.11}$$

- The difference of two sets A and B, denoted by $A \setminus B$, is a set containing elements that are in A but not in B
$$A \setminus B = \{x \mid x \in A \text{ and } x \notin B\} \tag{2.12}$$

- The multiplication of a real number r and a subset A is
$$rA = \{rx \mid x \in A\} \tag{2.13}$$

Note that the classical set operations follow the associative, distribution, and commutative laws (Klir et al., 1997).

One of the most important concepts in the set theory is the membership. Considering a universe X, membership in a crisp subset A of X is often viewed as a characteristic (membership) function that indicates which elements of X are members of set A and which are not (Klir et al., 1997). Set A is defined by its membership function, μ_A, as follows:
$$\mu_A(x) = \begin{cases} 1 \text{ for } x \in A \\ 0 \text{ for } x \notin A \end{cases} \tag{2.14}$$

It is noticed that the membership function in a crisp set A of X outputs only 0 or 1, exclusively, with 0 according as x does not belong to A and 1 according as x does belong to A. That is, the membership function μ_A maps elements of X to elements of the set $\{0, 1\}$. Formally,
$$\mu_A : X \longrightarrow \{0, 1\} \tag{2.15}$$

If μ_A is the membership function of set A, μ_B is the membership function of set B, the union of two sets A and B will have the membership function as
$$\mu_{A \cup B}(x) = \max\{\mu_A(x), \mu_B(x)\} \quad \forall x \in X \tag{2.16}$$

or, in abbreviated form
$$\mu_{A \cup B} = \mu_A \vee \mu_B \tag{2.17}$$

where "$\vee$" is the max operator. Similarly, the intersection of two sets A and B will have the membership function as

$$\mu_{A\cap B}(x) = \min\{\mu_A(x), \mu_B(x)\} \quad \forall x \in X \tag{2.18}$$

or, in abbreviated form

$$\mu_{A\cup B} = \mu_A \wedge \mu_B \tag{2.19}$$

where "$\wedge$" is the min operator.

2.2.2 Fuzzy set

The first systematic study on fuzzy set was carried out by L. Zadeh beginning with Zadeh (1965) and subsequently Zadeh (1971). According to Zadeh's definition (Zadeh, 1965), if X is a universe set of objects denoted generically by x, then a fuzzy set A in X is a set of ordered pairs

$$A = \{(x, \mu_A(x)) : x \in X\} \tag{2.20}$$

where,

$$\mu_A : X \longrightarrow [0, 1] \tag{2.21}$$

is the membership function and the value of $\mu_A(x)$ at x indicating the "grade of membership" of x in A. Alternatively, a fuzzy set A in X characterized by a membership function, $\mu_A(x)$, which associates with each point in X a real number on the close interval of $[0, 1]$. If $\mu_A(x) = 0$, then x is called not include in the fuzzy set. If $\mu_A(x) = 1$, then x is called full include in the fuzzy set. If $\mu_A(x)$ gets any value in between 0 and 1, then x is called partial include in the fuzzy set. The membership function in fuzzy sets is different from the one in crisp sets. The former gets the function values ranging from 0 to 1, while the later outputs either 0 or 1. To differentiate between fuzzy sets and crisp sets, Zadeh denominated the crisp set as "ordinary set" or only "set" (Zadeh, 1965).

Considering two fuzzy sets A and B in X, the following definitions involving these two fuzzy sets were defined by Zadeh (1965). Firstly, the definition of the empty fuzzy set is reviewed. A fuzzy set is empty if and only if its membership function is identically zero on X. Secondly, we shall introduce the definition of the fuzzy set equality. Specifically, two fuzzy sets A and B are equal, written as $A = B$, if and only if

$$\mu_A(x) = \mu_B(x) \quad \forall x \in X \tag{2.22}$$

Furthermore, we will go through the notion of containment between two fuzzy sets. The fuzzy set A is contained in the fuzzy set B, or B includes A, if and only if

$$\mu_A(x) \leq \mu_B(x) \quad \forall x \in X \tag{2.23}$$

The containment between these two fuzzy sets is denoted as $A \subseteq B$ (Zadeh, 1965). Remark that the subset notation here is generally understood to mean the containment property of fuzzy sets, not to refer to the concept that an element x in A belongs to the fuzzy set B as mentioned in Section 2.2.1 (Zadeh, 1965). From the definition of containment, it can be seen that for two fuzzy sets A and B, $A = B$ if and only if $A \subseteq B$ and $B \subseteq A$. An example of the containment between two fuzzy sets A and B in $\mathbb{R}$ is presented in

Figure 2.1. In this plot, the x axes signifies the values of all members in $\mathbb{R}$, the y axes implies the values of membership functions. With each x in $\mathbb{R}$ we get one value of grade of membership for each membership function, and the curves represent the value of the membership function at the respective $x \in \mathbb{R}$. As $\mu_A(x) \leq \mu_B(x)$ for all $x \in X$, we say A is a contained in B ($A \subseteq B$).

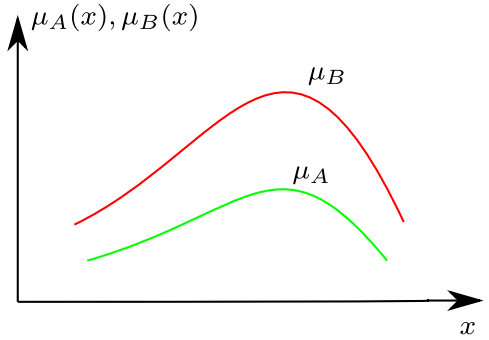


Figure 2.1: Illustration of a containment in $\mathbb{R}$.

The following definitions for complement, unions, and intersection of fuzzy sets are introduced by Zadeh (1965). The complement of a fuzzy set A, denoted A', has a membership function $\mu_{A'}$ is defined by

$$\mu_{A'}(x) = 1 - \mu_A(x) \tag{2.24}$$

The union of two fuzzy sets A and B with respective membership function μ_A and μ_B is a fuzzy set C, written as $C = A \cup B$, whose membership function μ_C is defined as

$$\mu_C(x) = \max\{\mu_A(x), \mu_B(x)\} \tag{2.25}$$

or, in abbreviated form

$$\mu_C = \mu_{A \cup B} = \mu_A \vee \mu_B \tag{2.26}$$

An alternative way of defining the union is that the union C of A and B is the smallest fuzzy set containing both A and B. It means that if D is any fuzzy set contains both A and B, it also contains the union of A and B (Zadeh, 1965). Notice that the union operator has the associative property (Zadeh, 1965), meaning that

$$A \cup (B \cup C) = (A \cup B) \cup C \tag{2.27}$$

Similarly, the intersection of fuzzy sets might be defined in a same manner. Specifically, the intersection of two fuzzy sets A and B with membership function μ_A and μ_B, respectively, is a fuzzy set C whose membership function μ_C is related to those of A and B by

$$\mu_C(x) = \min\{\mu_A(x), \mu_B(x)\} \tag{2.28}$$

or, in abbreviated form

$$\mu_C = \mu_{A \cap B} = \mu_A \wedge \mu_B \tag{2.29}$$

Alternatively, the intersection of two fuzzy sets A and B can be defined as the largest fuzzy set which is contained in both A and B. Similar to the union, the intersection of A and B also has the associative property. Furthermore, the operations of union and intersection obey the distributive laws (Zadeh, 1965). As an example, we have

$$C \cup (A \cap B) = (C \cup A) \cap (C \cup B) \tag{2.30}$$

and

$$C \cap (A \cup B) = (C \cap A) \cup (C \cap B) \tag{2.31}$$

The union and intersection of two fuzzy sets in a universe $\mathbb{R}$ are demonstrated in Figure 2.2. We assume that there are two fuzzy sets A and B in $\mathbb{R}$ with respective membership functions μ_A and μ_B, and these functions are shown in Figure 2.2 as the green and red curves, respectively. The membership function of the union of A and B is presented by the curve consisting two curve segments 1 and 2 while the two curve segments 3 and 4 compromises the membership function of the intersection.

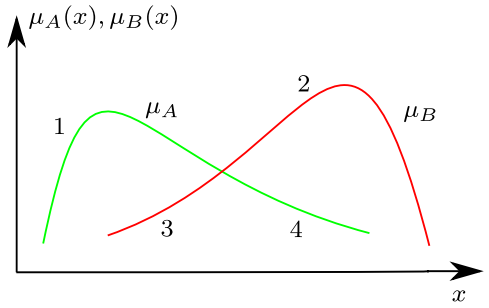


Figure 2.2: Illustration of the union and intersection of fuzzy sets in $\mathbb{R}$.

With regards to the notation of fuzzy sets, when X is a universe set, $X = \{x_1, x_2, ..., x_n\}$, a fuzzy set A on X can be expressed as

$$A = \mu_A(x_1)/x_1 + \mu_A(x_2)/x_2 + ... + \mu_A(x_n)/x_n \tag{2.32}$$

where $\mu_A(x_i)$ with $i \in [1, n]$ is the grade of membership of x_i in A, the plus sign represents the union, and the slash is not the division slash. For the sake of illustration, an example of fuzzy relation is given. An architect designing a house cares about the comfort of the house. One indicator of comfort of the house is the number of bedrooms in it. Given $X = \{1, 2, 3, 4, ..., 10\}$ be the universe set of possible types of houses described by x which is the number of bedrooms in a house. Then the fuzzy set A "comfortable house for a four-person family" may be described as

$$A = 0.2/1 + 0.3/2 + 0.7/3 + 1/4 + 0.6/5 + 0.1/6 \tag{2.33}$$

2.2.3 Fuzzy Relation

The concept of a "relation" is a generalization of "function" in fuzzy set theory (Zadeh, 1965). In fuzzy sets, we examine the membership function which associates with each

point in the universe set a real number between 0 and 1. However, in fuzzy relations, we investigate the membership function that associates each ordered pairs of two or more than two sets with a specific number in the interval $[0, 1]$. A fuzzy relation between two sets is called a binary fuzzy relation. Since this study is concerned with only binary fuzzy relations, hereafter, the adjective "binary" might be omitted.

Let X and Y be two crisp sets denoted generically by x and y, respectively. Thus, $X = \{x\}$ and $Y = \{y\}$. A binary fuzzy relation R on X and Y, denoted by $R(X, Y)$, is a fuzzy set characterized by a membership function μ_R that associates with each pair (x, y) its "grade of membership", $\mu_R(x, y)$, in R (Zadeh, 1971). This membership function μ_R and the binary fuzzy relation R can be expressed as:

$$\mu_R : X \times Y \to [0, 1] \tag{2.34}$$

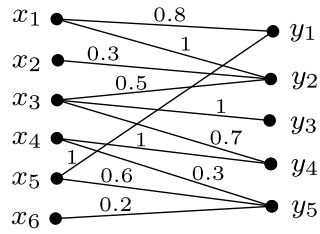
$$R = \{((x, y), \mu_R(x, y)) \mid x \in X, y \in Y\} \tag{2.35}$$

The degree of membership $\mu_R(x, y)$ lies in the interval $[0, 1]$ and is considered as the strength of the relation between x and y. It means that when $\mu_R(x, y) \geq \mu_R(x', y')$, x and y are more strongly related than x' and y'. If the degree of membership of a fuzzy relation takes only two values 0 and 1, this fuzzy relation is called a crisp relation. A crisp relation represents the presence or absence of association, interaction or interconnectedness between the elements of two or more sets.

There are two common ways to present a binary fuzzy relation R on X and Y. Firstly, the fuzzy relation $R(X, Y)$ can be given in the form of fuzzy matrix (membership matrix) whose elements represent the membership values of this relation (Zadeh, 1971). That is, if the membership matrix is denoted by $R = [r_{xy}]$, then $r_{xy} = \mu_R(x, y)$. For simplicity, we will use the same notation R for the relation and the associated membership matrix. Secondly, a useful representation of a binary relation is a sagittal diagram. In the sagittal diagram, each element of the sets X and Y is represented as a single node. Nodes corresponding to one set is clearly distinguished from nodes representing the other set. Then each pair of elements of X and Y is connected by lines (directed connection) if its grade of membership in R is non-zero (Kandasamy and Smarandache, 2003). Each connection in the sagittal diagram is labeled by the actual membership grade of the corresponding pair in R. To exemplify the definition and the representation of a binary fuzzy relation, consider the following example. Assuming that there are two sets, $X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$ and $Y = \{y_1, y_2, y_3, y_4, y_5\}$, a binary fuzzy relation R on X and Y has a corresponding membership matrix R and a sagittal diagram shown in Figure 2.3.

$$R = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{ccccc} y_1 & y_2 & y_3 & y_4 & y_5 \end{array} \\ \begin{bmatrix} 0.8 & 1 & 0 & 0 & 0 \\ 0 & 0.3 & 0 & 0 & 0 \\ 0 & 0.5 & 1 & 0.7 & 0 \\ 0 & 0 & 0 & 1 & 0.3 \\ 1 & 0 & 0 & 0 & 0.6 \\ 0 & 0.9 & 0 & 0 & 0.2 \end{bmatrix} \end{array}$$

(a) Membership matrix



(b) Sagittal diagram

Figure 2.3: Two ways to present a binary fuzzy relation R on X and Y.

As fuzzy relations are the extensional concept of fuzzy sets, we can apply operations of fuzzy sets to fuzzy relations. Let P and Q be two fuzzy relations from X to Y. The union of P and Q is defined by Zadeh (1965) as follows:

$$\mu_{P\cup Q}(x,y) = \max\{\mu_P(x,y), \mu_Q(x,y)\} \tag{2.36}$$

or, in abbreviated form

$$\mu_{P\cup Q} = \mu_P \vee \mu_Q \tag{2.37}$$

Analogously, the intersection of R and Q is defined by

$$\mu_{P\cap Q}(x,y) = \min\{\mu_P(x,y), \mu_Q(x,y)\} \tag{2.38}$$

or, in abbreviated form

$$\mu_{P\cap Q} = \mu_P \wedge \mu_Q \tag{2.39}$$

If P is a fuzzy relation from X to Y and Q is a fuzzy relation from Y to Z, the max-min composition of P and Q, which is denoted by $P \circ Q$ (Zadeh, 1965), is defined by

$$\mu_{P\circ Q}(x,z) = \max\{\min_{y\in Y}\{\mu_P(x,y), \mu_Q(y,z)\}\} \tag{2.40}$$

Let $P = [p_{ik}]$ and $Q = [q_{kj}]$, then the membership matrix of max-min composition of P and Q is given by

$$[r_{ij}] = [p_{ik}] \circ [q_{kj}] \tag{2.41}$$

where

$$r_{ij} = \max\{\min_{k}\{p_{ik}, q_{kj}\}\} \tag{2.42}$$

Observe that the calculation of $P \circ Q$ is almost similar to an ordinary matrix (dot) product. In particular, the elements of P and Q that are used in the calculation of R are the same as in the matrix product. However, the multiplication and the summation in dot products are replaced by the max operation and the min operation in the max-min composition, respectively. The max-min composition, which is associative and distributive, can be interpreted as indicating the strength of the existence of relation between the elements of X and Z. To illustrate the union and max-min composition operators, an example is given in Table 2.1. The table summarizes the data on three sets X, Y, and Z and two binary fuzzy relations, namely $P(X,Y)$ and $Q(Y,Z)$.

Table 2.1: Example of union and max-min composition operators.

P	y_1	y_2	y_3
x_1	0.3	0.2	1.0
x_2	0.8	1.0	1.0
x_3	0.0	1.0	0.0

Q	z_1	z_2	z_3
y_1	0.3	0.0	0.1
y_2	0.1	0.8	1.0
y_3	0.6	0.9	0.3

$P \cup Q$	z_1	z_2	z_3
x_1	0.3	0.2	1.0
x_2	0.8	1.0	1.0
x_3	0.6	1.0	0.3

P	y_1	y_2	y_3
x_1	0.1	0.2	0.0
x_2	0.3	0.3	0.0
x_3	0.8	0.9	1.0

Q	z_1	z_2	z_3
y_1	0.9	0.0	0.3
y_2	0.2	1.0	0.8
y_3	0.8	0.0	0.7

$P \circ Q$	z_1	z_2	z_3
x_1	0.2	0.2	0.2
x_2	0.3	0.3	0.3
x_3	0.8	0.9	0.8

2.2.4 Fuzzy Equivalence Relation

Given a binary fuzzy relation R on X and Y, in this thesis, we are only interested in a special case in which $X = Y$. Accordingly, we have the definition of a binary fuzzy relation on X, where $X = \{x_1, x_2, ..., x_n\}$ is a set of n elements. A fuzzy relation R on X is characterized by a membership function μ_R which associates each elements of X to all elements of X itself with a grade of membership. The fuzzy relation R on X can be illustrated by a (membership) matrix $R = \{r_{x_i x_j}\}$ of dimension $n \times n$, where $r_{x_i x_j} = \mu_R(x_i, x_j)$ is the membership grade between x_i and x_j. The membership grade is in the closed interval $[0, 1]$.

The fuzzy relation R on X is said to be reflexive if

$$\mu_R(x_i, x_i) = 1 \quad \forall i \in [1, n] \tag{2.43}$$

In contrast, the fuzzy relation R on X is said to be anti-reflexive if

$$\mu_R(x_i, x_i) = 0 \quad \forall i \in [1, n] \tag{2.44}$$

In addition, it is said to be symmetric if

$$\mu_R(x_i, x_j) = \mu_R(x_j, x_i) \quad \forall i, j \in [1, n] \tag{2.45}$$

Finally, it is min-transitive (transitive) if

$$R^2 = R \circ R \subseteq R \tag{2.46}$$

or, more explicitly,

$$\max\{\min_{k \in [1,n]} \{\mu_R(x_i, x_k), \mu_R(x_k, x_j)\}\} \leq \mu_R(x_i, x_j) \quad \forall i, j, k \in [1, n] \tag{2.47}$$

From such properties, a fuzzy relation which is reflexive and symmetric is called a similarity relation (Meyer et al., 2004). The matrix representation of a similarity relation is called a similarity matrix (Meyer et al., 2004). Additionally, a fuzzy relation which is anti-reflexive and symmetric is called a dissimilarity relation, and its corresponding matrix is called a dissimilarity matrix (de Oliveira and Pedrycz, 2007). If a similarity relation is min-transitive then it is called a fuzzy equivalence relation (Zadeh, 1971; Klir et al., 1997).

2.3 Hierarchical Clustering

2.3.1 Pattern Matrix and Proximity Matrix

We start with a set of n objects (samples, patterns) to be clustered, denoted by $X = \{x_i\}$ with $1 \leq i \leq n$, and each object is described by a set of m measurements (attributes, variables, features). Let x_{ij} be the jth measurement of the ith object, $1 \leq j \leq m$. Each object is now represented by a $m-$vector whose elements are the measurements characterizing the object (Jain and Dubes, 1988). For instance, the ith object is defined by a vector

$$x_i = (x_{i1} x_{i2} ... x_{im})^T, i \in [1, n] \tag{2.48}$$

where T denotes vector transpose. It is customary to display this set of objects by using a $n \times m$ matrix called a pattern matrix denoted by $R = [r_{ij}]$. In this pattern matrix, each row depicts a specific object and each column designates a measurement or feature. For example, the famous Iris flower data set which was introduced by the British statistician and biologist Ronald Fisher in 1936 has four features including the length and the width of the sepals and petals of Iris flowers, in centimeters (Fisher, 1936). The initial data set was composed of 50 samples from each of three species of Iris, namely Iris setosa, Iris virginica and Iris versicolor. The pattern matrix of the Iris data set is therefore a 150×4 matrix and one of the most popular data sets for machine learning in general and statistical classification in particular. For reasons of space, we take only 16 samples out of 150 samples in the Iris flower data set to study the procedure behind hierarchical clustering (Table 2.2).

Table 2.2: 16 samples in Iris data set.

Samples	Sepal length	Sepal width	Petal length	Petal width
I. setosa 9	4.4	2.9	1.4	0.2
I. setosa 18	5.1	3.5	1.4	0.3
I. setosa 27	5.0	3.4	1.6	0.4
I. setosa 36	5.0	3.2	1.2	0.2
I. setosa 45	5.1	3.8	1.9	0.4
I. versicolor 54	5.5	2.3	4.0	1.3
I. versicolor 63	6.0	2.2	4.0	1.0
I. versicolor 72	6.1	2.8	4.0	1.3
I. versicolor 81	5.5	2.4	3.8	1.1
I. versicolor 90	5.5	2.5	4.0	1.3
I. versicolor 99	5.1	2.5	3.0	1.1
I. virginica 108	7.3	2.9	6.3	1.8
I. virginica 117	6.5	3.0	5.5	1.8
I. virginica 126	7.2	3.2	6.0	1.8
I. virginica 135	6.1	2.6	5.6	1.4
I. virginica 144	6.8	3.2	5.9	2.3

In Table 2.2, regardless of the first row, the rest of this table is important for preparing a list of labels and a pattern matrix needed for a hierarchical clustering algorithm. The first column in this table gives information of names or labels of 16 samples. A list of labels is then created in such a way that the order of labels in the list is the same as in the first column in Table 2.2. The four last columns containing the measurements constitute the pattern matrix R of 16 Iris flower samples:

$$R = \begin{bmatrix} 4.4 & 2.9 & 1.4 & 0.2 \\ 5.1 & 3.5 & 1.4 & 0.3 \\ 5.0 & 3.4 & 1.6 & 0.4 \\ 5.0 & 3.2 & 1.2 & 0.2 \\ 5.1 & 3.8 & 1.9 & 0.4 \\ 5.5 & 2.3 & 4.0 & 1.3 \\ 6.0 & 2.2 & 4.0 & 1.0 \\ 6.1 & 2.8 & 4.0 & 1.3 \\ 5.5 & 2.4 & 3.8 & 1.1 \\ 5.5 & 2.5 & 4.0 & 1.3 \\ 5.1 & 2.5 & 3.0 & 1.1 \\ 7.3 & 2.9 & 6.3 & 1.8 \\ 6.5 & 3.0 & 5.5 & 1.8 \\ 7.2 & 3.2 & 6.0 & 1.8 \\ 6.1 & 2.6 & 5.6 & 1.4 \\ 6.8 & 3.2 & 5.9 & 2.3 \end{bmatrix}$$

Another way to display a set of n objects with m measurements is using a m-dimensional space whose orthogonal coordinate is a set of m coordinates. This space is called a pattern space where each orthogonal axis corresponds to one measurement, and each object of the set is pictured as a point. The task of hierarchical clustering can be considered as identifying points in spaces of many dimensions such that these points are close to one another or meet some spatial requirements (Jain and Dubes, 1988). One should not be misled that the hierarchical clustering only deals with two- or three-dimensional data.

To identify which objects are similar and which are not, it requires that an index of proximity, or association be generated between pairs of objects (Jain and Dubes, 1988). In a problem with n objects, there are $\frac{1}{2}n(n+1)$ different pairs of objects regardless of the order objects are written in the pairs. The measure of proximity for every pairwise combination of objects can be calculated based on the pattern matrix and stored in a $n \times n$ matrix called a proximity matrix. Let $[d(i,j)]$ denote the $n \times n$ proximity matrix where $d(i,j)$ denotes the proximity index between the ith and jth patterns. We can notice that the proximity index of a pair of objects is independent of the order in which the objects are written. As a result, all proximity matrices are symmetric, meaning that each row or column in the proximity matrix represents an object, which is different from pattern matrices in which each column defines an attribute.

Depending on how an index of proximity is measured, it is either a similarity index or a dissimilarity index (Jain and Dubes, 1988). A similarity index between two objects could be, for example, a correlation coefficient between them. Whereas, a distance between two patterns in a pattern space is a dissimilarity index. Large distances mean that objects are not close to each other and then they are different. Roughly, the more two objects resemble one another, the larger the similarity index, and the smaller the dissimilarity index. One can determine whether a proximity index is a similarity index or a dissimilarity index by observing the elements on the main diagonal of the proximity matrix. If they are zero everywhere on the main diagonal, then the proximity matrix contains dissimilarity indices as objects have degrees of dissimilarity of zero with themselves. If, in each row, element on the main diagonal is larger than or equal to the maximum of all other elements in such row, then the proximity matrix contains similarity indices. These observations are part of

three properties of the proximity index (Jain and Dubes, 1988). Recall $d(i,j)$ defining the proximity index between the ith and jth patterns, the proximity index obeys the three properties as follows:

1. (a) if proximity index is a dissimilarity index: $d(i,i) = 0 \; \forall i$
   (b) if proximity index is a similarity index: $d(i,i) \geq \max_k d(i,k) \; \forall i$
2. it is symmetric, meaning that $d(i,j) = d(j,i) \; \forall (i,j)$
3. it is positive, meaning that $d(i,j) \geq 0 \; \forall (i,j)$

One can get dissimilarity-based proximity indices from the pattern matrix by using a distance measure metric. The most common metric for proximity indices in engineering is the Euclidean metric (distance). Suppose that we have a pattern matrix of n objects, and m measurements in the pattern matrix are on an ordinal scale. The Euclidean metric between two patterns ith and jth is defined as:

$$d(i,j) = \left(\sum_{k=1}^{m} (r_{ik} - r_{jk})^2 \right)^{\frac{1}{2}} \tag{2.49}$$

The Euclidean metric measures dissimilarity. That is, $d(i,j) \geq d(i,k))$ means that objects i and k resemble one another more than objects i and j. In addition to fulfilling three properties of a proximity index, the Euclidean distance satisfies two more criteria:

4. $d(i,j) = 0$ only if $x_i = x_j$
5. it obeys the triangle inequality: $d(i,j) \leq d(i,h) + d(h,j) \; \forall (i,j,h)$

With the use of Euclidean metric, the proximity matrix P containing the distances between each of the 16 Iris samples will be

$$P = \begin{bmatrix}
0 & 0.93 & 0.83 & 0.70 & 1.26 & 3.09 & 3.23 & 3.30 & 2.83 & 3.06 & 2.00 & 5.91 & 4.88 & 5.63 & 4.70 & 5.52 \\
0.93 & 0 & 0.26 & 0.39 & 0.59 & 3.06 & 3.12 & 3.04 & 2.79 & 2.99 & 2.05 & 5.61 & 4.61 & 5.28 & 4.55 & 5.22 \\
0.83 & 0.26 & 0 & 0.49 & 0.51 & 2.83 & 2.93 & 2.85 & 2.57 & 2.76 & 1.81 & 5.44 & 4.42 & 5.12 & 4.34 & 5.04 \\
0.70 & 0.39 & 0.49 & 0 & 0.95 & 3.18 & 3.24 & 3.23 & 2.91 & 3.13 & 2.13 & 5.83 & 4.83 & 5.52 & 4.73 & 5.45 \\
1.26 & 0.59 & 0.51 & 0.95 & 0 & 2.76 & 2.85 & 2.69 & 2.49 & 2.66 & 1.84 & 5.19 & 4.19 & 4.85 & 4.14 & 4.78 \\
3.09 & 3.06 & 2.83 & 3.18 & 2.76 & 0 & 0.59 & 0.78 & 0.30 & 0.20 & 1.11 & 3.02 & 2.00 & 2.82 & 1.74 & 2.67 \\
3.23 & 3.12 & 2.93 & 3.24 & 2.85 & 0.59 & 0 & 0.68 & 0.58 & 0.66 & 1.38 & 2.85 & 1.94 & 2.66 & 1.70 & 2.63 \\
3.30 & 3.04 & 2.85 & 3.23 & 2.69 & 0.78 & 0.68 & 0 & 0.77 & 0.67 & 1.46 & 2.64 & 1.64 & 2.37 & 1.62 & 2.29 \\
2.83 & 2.79 & 2.57 & 2.91 & 2.49 & 0.30 & 0.58 & 0.77 & 0 & 0.30 & 0.90 & 3.20 & 2.18 & 2.98 & 1.93 & 2.86 \\
3.06 & 2.99 & 2.76 & 3.13 & 2.66 & 0.20 & 0.66 & 0.67 & 0.30 & 0 & 1.10 & 2.99 & 1.94 & 2.76 & 1.71 & 2.61 \\
2.00 & 2.05 & 1.81 & 2.13 & 1.84 & 1.11 & 1.38 & 1.46 & 0.90 & 1.10 & 0 & 4.05 & 2.99 & 3.79 & 2.80 & 3.64 \\
5.91 & 5.61 & 5.44 & 5.83 & 5.19 & 3.02 & 2.85 & 2.64 & 3.20 & 2.99 & 4.05 & 0 & 1.14 & 0.44 & 1.48 & 0.87 \\
4.88 & 4.61 & 4.42 & 4.83 & 4.19 & 2.00 & 1.94 & 1.64 & 2.18 & 1.94 & 2.99 & 1.14 & 0 & 0.88 & 0.70 & 0.73 \\
5.63 & 5.28 & 5.12 & 5.52 & 4.85 & 2.82 & 2.66 & 2.37 & 2.98 & 2.76 & 3.79 & 0.44 & 0.88 & 0 & 1.37 & 0.65 \\
4.70 & 4.55 & 4.34 & 4.73 & 4.14 & 1.74 & 1.70 & 1.62 & 1.93 & 1.71 & 2.80 & 1.48 & 0.70 & 1.37 & 0 & 1.32 \\
5.52 & 5.22 & 5.04 & 5.45 & 4.78 & 2.67 & 2.63 & 2.29 & 2.86 & 2.61 & 3.64 & 0.87 & 0.73 & 0.65 & 1.32 & 0
\end{bmatrix}$$

Without losing generality and property the entries of a proximity matrix can be mapped into the $[0, 1]$ interval. This mapping turns a dissimilarity-based proximity matrix into a dissimilarity relation or dissimilarity matrix whose all diagonal entries are equal to zero. Whereas, if the proximity matrix containing similarity indices, the mapping turns it into a similarity relation or similarity matrix having all diagonal entries of 1. As being symmetric, the dissimilarity (similarity) matrix is usually displayed by its upper triangular part. For example, with the proximity matrix P of 16 Iris flower samples above, we can obtain a dissimilarity matrix D by mapping elements of P into the $[0, 1]$ interval as follows:

$$D = \begin{bmatrix} 0 & 0.16 & 0.14 & 0.12 & 0.21 & 0.52 & 0.55 & 0.56 & 0.48 & 0.52 & 0.34 & 1.00 & 0.83 & 0.95 & 0.79 & 0.93 \\ & 0 & 0.04 & 0.07 & 0.10 & 0.52 & 0.53 & 0.51 & 0.47 & 0.51 & 0.35 & 0.95 & 0.78 & 0.89 & 0.77 & 0.88 \\ & & 0 & 0.08 & 0.09 & 0.48 & 0.50 & 0.48 & 0.43 & 0.47 & 0.31 & 0.92 & 0.75 & 0.87 & 0.73 & 0.85 \\ & & & 0 & 0.16 & 0.54 & 0.55 & 0.55 & 0.49 & 0.53 & 0.36 & 0.99 & 0.82 & 0.93 & 0.80 & 0.92 \\ & & & & 0 & 0.47 & 0.48 & 0.45 & 0.42 & 0.45 & 0.31 & 0.88 & 0.71 & 0.82 & 0.70 & 0.81 \\ & & & & & 0 & 0.10 & 0.13 & 0.05 & 0.03 & 0.19 & 0.51 & 0.34 & 0.48 & 0.29 & 0.45 \\ & & & & & & 0 & 0.11 & 0.10 & 0.11 & 0.23 & 0.48 & 0.33 & 0.45 & 0.29 & 0.45 \\ & & & & & & & 0 & 0.13 & 0.11 & 0.25 & 0.45 & 0.28 & 0.40 & 0.27 & 0.39 \\ & & & & & & & & 0 & 0.05 & 0.15 & 0.54 & 0.37 & 0.50 & 0.33 & 0.48 \\ & & & & & & & & & 0 & 0.19 & 0.51 & 0.33 & 0.47 & 0.29 & 0.44 \\ & & & & & & & & & & 0 & 0.68 & 0.51 & 0.64 & 0.47 & 0.62 \\ & & & & & & & & & & & 0 & 0.19 & 0.07 & 0.25 & 0.15 \\ & & & & & & & & & & & & 0 & 0.15 & 0.12 & 0.12 \\ & & & & & & & & & & & & & 0 & 0.23 & 0.11 \\ & & & & & & & & & & & & & & 0 & 0.22 \\ & & & & & & & & & & & & & & & 0 \end{bmatrix}$$

We can convert dissimilarity matrices into similarity matrices and vice versa. If $D = d(i,j)$ is a dissimilarity matrix of n objects, the similarity matrix $S = s(i,j)$ is derived from D by

$$S = [1]_{n\times n} - D \tag{2.50}$$

where $[1]_{n\times n}$ is a $n \times n$ matrix whose all elements equal to one. This equation can be rewritten as

$$s(i,j) = 1 - d(i,j) \quad \forall i,j \in [1,n] \tag{2.51}$$

Similarly, a dissimilarity matrix $D = d(i,j)$ can be derived if a similarity matrix $S = s(i,j)$ is given by

$$D = [1]_{n\times n} - S \tag{2.52}$$

or

$$d(i,j) = 1 - s(i,j) \quad \forall i,j \in [1,n] \tag{2.53}$$

2.3.2 Linkage methods

In the previous section, we know that a distance metric such as Euclidean metric is used to measure distances between objects and create the dissimilarity matrix. In the agglomerative hierarchical clustering, one may wish to interpret the dissimilarity matrix to have a comprehension of which objects are the most similar in order to merge them. Such merging naturally leads to a need of comparing distances between clusters containing more than one object to know which clusters are the most similar in order to merge them. To determine the distance among clusters, we use a linkage or linkage method. There are several commonly used linkage methods such as single linkage, complete linkage, average linkage, weighted linkage, median linkage, centroid linkage, and Ward linkage. In the

scope of this study, we will focus on the two most common linkages: single linkage and complete linkage.

In single linkage, the distance between two clusters $U = \{u_i\}$ and $V = \{v_j\}$ is the distance of the two closest objects between clusters (Sneath and Sokal, 1973). In symbols

$$d(U, V) = \min(d(u_i, v_j)) \tag{2.54}$$

In other words, single linkage returns the minimum distance between objects belonging to two different clusters. The opposite linkage method is complete linkage. King (1967) defined the distance between two clusters $U = \{u_i\}$ and $V = \{v_j\}$ in the complete linkage is the distance of the two furthest objects between clusters

$$d(U, V) = \max(d(u_i, v_j)) \tag{2.55}$$

It means that the distance between clusters in complete linkage equals the maximum distance between each pair of elements across clusters. A conceptually similar method to single linkage is average linkage in which the distance between clusters is calculated as the average of all pairwise distances of two clusters. Weighted linkage performs the same calculation but weights distances based on the number of objects in clusters. Whereas, median linkage calculates distances based on the medians of each cluster. Somewhat similarly, centroid linkage calculates distance between centroids (means) of two clusters. The most conceptually complex is the Ward's linkage method which is designed to minimize information loss (Jr., 1963). More specifically, it pretends to merge two clusters into one, estimates the centroid of the resulting cluster and computes the sum of squared deviations between the centroid and the observations in the resulting cluster. Then, the difference between this result and the sum of squared deviations between the observations in each cluster before merging and the corresponding centroid is the distance between two clusters in the Ward's linkage method.

The difference between available hierarchical clustering methods rests upon which linkage method is employed. A hierarchical clustering using the single linkage method is called the single linkage clustering, and it is called the complete linkage clustering if the complete linkage is used. Keeping in mind the difference in measuring the distance between clusters, one can relate to the fact that the dendrogram from the single linkage clustering might be dissimilar with the one from the complete linkage clustering.

2.3.3 Hierarchical Clustering, Dendrogram, and Linkage Matrix

The input for a hierarchical clustering algorithm can be either a pattern matrix or a proximity matrix. It is worth to recall that the ith row in the pattern matrix or the proximity matrix represents the ith object which might have a name or label. Hence along with the pattern matrix or the proximity matrix, a label vector, whose elements are labels of the objects appearing in these matrices should be also inputted. Throughout this thesis, we use the label and the index interchangeably to denominate an object.

Hierarchical clustering is a method of cluster analysis, which transforms the pattern matrix or the proximity matrix into a sequence of nested partitions. We start introducing the notion of a sequence of nested partitions by recalling the set of n objects to be clustered

$$X = \{x_i\} = \{x_1, x_2, ..., x_n\}, 1 \leq i \leq n \tag{2.56}$$

where x_i is the ith object of the set. A collection of subsets of X, $C = \{C_1, C_2, ..., C_m\}$, is a partition of X if and only if the union of all components of C is the set X, and the intersection of any two different components is an empty set (Jain and Dubes, 1988); mathematically

$$\bigcup_{i=1}^{m} C_i = X \tag{2.57}$$

and

$$C_i \cap C_j = \emptyset \quad \forall i \neq j \tag{2.58}$$

A partition is alternatively called a clustering, and the components of the partition are called clusters. Partition B is nested in partition C if C is formed by merging clusters of B. In other words, if the partition B is nested in the partition C, there always exists a cluster of C which contains a cluster of B. For example, there is a set of objects $\{x_1, x_2, x_3, x_4, x_5, x_6\}$ and two clusterings, $C = \{(x_1, x_2, x_3), x_4, (x_5, x_6)\}$ and $B = \{(x_1, x_2), x_3, x_4, (x_5, x_6)\}$. Since two clusters of B, (x_1, x_2) and x_3, are merged into one single cluster, (x_1, x_2, x_3), to form the clustering C, we see that B is nested into C. A clustering of a given set of n objects, whose each cluster contains only one object, is called a disjoint clustering. The cluster in the disjoint clustering is called the singleton cluster. To distinguish between a singleton cluster with a cluster containing more than one object, we call the later as a non-singleton cluster. A clustering of n objects is called a conjoint clustering if it has only one cluster which contains all of n objects.

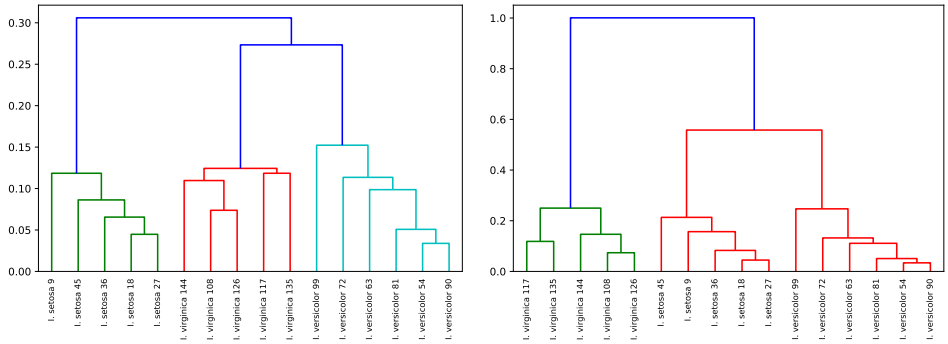
Two algorithms for hierarchical clustering can be found, namely agglomerative (bottom-up) and divisive (top-down). Initially, the n objects are considered to be clustered. The agglomerative algorithm starts the hierarchical clustering with a disjoint clustering at which each object is a singleton cluster. The index of a singleton cluster is the index of the object that constitutes the cluster. Pairs of these clusters are successively merged based on the basis of closeness of clusters to create new clusterings until the conjoint clustering is formed. This process will generate a sequence of nested partitions in which the number of clusters of the partition decreases from n to 1 as the sequence progresses. Reversely, the divisive algorithm starts with the conjoint clustering. Then this conjoint clustering is recursively subdivided into smaller clusters to form new partitions until the disjoint clustering is generated. Regardless of which approach is used, each time a new partition is formed, the number of clusters changes and the distances between clusters need to be calculated. Following this, it requires to update the proximity matrix after we have merged or split clusters.

In this thesis, we only use the agglomerative hierarchical clustering. Its algorithm can be summarized as follows:

1. Create the disjoint clustering of n clusters for the n objects under consideration.
2. Compute the proximity matrix D.
3. Merge the two closest clusters (clusters that have the smallest distance in the proximity matrix), say p and q, into a new cluster, denoted $U = \{u_i\} = \{p, q\}$, to form a new clustering.

4. Update the proximity matrix by deleting rows and columns corresponding to cluster p and q and adding a row and column corresponding to the newly formed cluster U. The distance between the new cluster U and the old cluster, say $V = \{v_j\}$, is calculated based on the given linkage method,
   - With the single linkage: $d(U, V) = \min(d(u_i, v_j)) = \min(d(p, v_j), d(q, v_j))$
   - With the complete linkage: $d(U, V) = \max(d(u_i, v_j)) = \max(d(p, v_j), d(q, v_j))$
5. Terminate if only single cluster remains. Else, go to step 3.

The objective of hierarchical clustering is to identify the natural clusters among objects characterized by many attributes. A sequence of nested partitions is the result of a hierarchical clustering algorithm. However, this sequence, which is a list of abstract symbols, is challenging for a human being to easily interpret. In order to clearly visualize the result of a hierarchical clustering, a dendrogram is constructed representing the sequence of nested partitions. A tree size of a dendrogram is the number of original observations to be clustered. In a dendrogram of n objects, we have all n observations on the x-axis and the distance (or hierarchical level) on the y-axis representing the scale of the proximity matrix. Each object on the x-axis constitutes a singleton cluster whose index is the index of the object of the singleton cluster. In the dendrogram, each singleton cluster is represented by a node called a terminal vertex or a leaf node or simply a leaf. These terminal vertices are connected by lines (edges) through interior vertices representing non-singleton clusters that are formed by merging two other clusters (or children clusters). The lines connecting a non-singleton cluster and its children clusters create a U-shaped link, and the interior vertex indicating a cluster merge is on top of the U-link. The ith interior vertex represents the $n+i$th cluster ($0 \leq i \leq n-2$), and each interior vertex is placed at a hierarchical level which is the distance between the children clusters. The $n-2$th interior vertex, which is called the root of the dendrogram, represents the $2n-2$th cluster or the conjoint clustering. A dendrogram with tree size of n has $n-1$ interior vertices and $2n-1$ different clusters. A cluster with an index less than n corresponds to one of the n original observations. Dendrograms of 16 Iris flower samples are presented in Figure 2.4. Noticeably, agglomerative hierarchical clustering with different linkages brings out different dendrograms.



(a) Single linkage

(b) Complete linkage

Figure 2.4: Hierarchical clustering for 16 Iris flower samples.

Hierarchical clustering is a helpful tool for segmenting observations as it allows users to identify the number of clusters in a dataset. Cutting a dendrogram at any hierarchical level returns a clustering and corresponding clusters. However, we need to decide at which hierarchical level we ought to cut the dendrogram to get a number of clusters that can best depict different groups. In a dendrogram, vertical lines connecting clusters represents the distance between them. Longer vertical lines indicate a larger the distance between the corresponding clusters. Therefore, we generally set a threshold and draw a horizontal line that cuts the tallest vertical line in the dendrogram. The number of clusters will be the number of intersections caused by vertical lines and the line drawn using the threshold. For example, in Figure 2.6b, if we cut the dendrogram at a distance of 0.2, we will have 3 clusters which corresponds to 3 types of Iris flowers.

In Python, the sequence of nested partitions of a dendrogram is summarized in a linkage matrix which is a $(n-1)$ by 4 matrix, where n is the number of objects to be clustered. To put it another way, the linkage matrix is the description of a sequence of nested partitions of the corresponding dendrogram. Each row in a linkage matrix tells us how a new non-singleton cluster is formed in order to create a new partition in the sequence. Specifically, in each row of the linkage matrix, the first two values are the indices of two children clusters which are combined to create a new cluster, the third value shows the hierarchical level at which these two clusters are merged, and the forth value denotes the number of singleton clusters (or objects) included in the new cluster. Since the dendrogram is the visualization of the corresponding sequence of nested partitions, and such sequence of nested partitions is summarized in a linkage matrix, each dendrogram corresponds to a linkage matrix. Addition to this, in Python, it is a prerequisite to have the linkage matrix prior to plotting the corresponding dendrogram as well as extracting the cophenetic matrix which will be dealt with in more detail in the next subsection. Therefore, there is a one-to-one relation between a linkage matrix and its dendrogram. Throughout the thesis we will use the linkage matrix to refer to the corresponding dendrogram. The dendrograms of 16 Iris flower samples generated by the single hierarchical clustering algorithm and the complete hierarchical clustering algorithm have the linkage matrices, Z_S and Z_C, respectively (Figure 2.5).

$$Z_S = \begin{bmatrix} 5 & 9 & 0.03384095 & 2 \\ 1 & 2 & 0.04476737 & 2 \\ 8 & 16 & 0.05076142 & 3 \\ 3 & 17 & 0.06553271 & 3 \\ 11 & 13 & 0.07375463 & 2 \\ 4 & 19 & 0.08627783 & 4 \\ 6 & 18 & 0.09866247 & 4 \\ 15 & 20 & 0.1096572 & 3 \\ 7 & 22 & 0.11350599 & 5 \\ 0 & 21 & 0.11844332 & 5 \\ 12 & 14 & 0.11844332 & 2 \\ 23 & 26 & 0.12433958 & 5 \\ 10 & 24 & 0.15228426 & 6 \\ 27 & 28 & 0.27335862 & 11 \\ 25 & 29 & 0.30597532 & 16 \end{bmatrix}$$

(a) Single linkage

$$Z_C = \begin{bmatrix} 5 & 9 & 0.03384095 & 2 \\ 1 & 2 & 0.04476737 & 2 \\ 8 & 16 & 0.05076142 & 3 \\ 11 & 13 & 0.07375463 & 2 \\ 3 & 17 & 0.08289305 & 3 \\ 6 & 18 & 0.11095497 & 4 \\ 12 & 14 & 0.11844332 & 2 \\ 7 & 21 & 0.13215313 & 5 \\ 15 & 19 & 0.1465356 & 3 \\ 0 & 20 & 0.15691402 & 4 \\ 4 & 25 & 0.21335906 & 5 \\ 10 & 23 & 0.24694619 & 6 \\ 22 & 24 & 0.2498278 & 5 \\ 26 & 27 & 0.55786266 & 11 \\ 28 & 29 & 1.00074268 & 16 \end{bmatrix}$$

(b) Complete linkage

Figure 2.5: Linkage matrices of two dendrograms of 16 Iris flower samples.

We should remark that the order of clusters in the two first values in each row of the linkage matrix determines the order of the corresponding clusters appearing in the dendrogram from left to right. In fact, within a row of the linkage matrix, if we swap the two first elements defining the two clusters to be merged, the positions of these two clusters will be interchanged in the corresponding dendrogram. Accordingly, the altered linkage matrix defines a new arrangement of objects in the dendrogram with the same hierarchy. The term "tanglegram layout" has come to be used to refer to this new arrangement of the dendrogram. Then the starting dendrogram has an original dendrogram layout with respect to the original linkage matrix, and many other layouts created by altering the original linkage matrix. As each altered linkage matrix circumscribes a unique dendrogram layout, we can now use the linkage matrix to mean the layout of the dendrogram. In this thesis we will exploit the linkage matrix as a key tool to alter the dendrogram layout while still preserving the topology. For example, suppose we have a dendrogram identified by a $(n-1) \times 4$ linkage matrix Z, and we want to swap two clusters that are joined to form the $n+i$th cluster represented by the ith interior vertex, $0 \leq i \leq n-2$. It can be done by interchanging the two first elements in the ith row of the linkage matrix Z, $Z[i] = \big[Z[i,0], Z[i,1], Z[i,2], Z[i,3]\big]$. To interchange positions of $Z[i,0]$ and $Z[i,1]$ we do a simply matrix multiplication

$$Z[i]J \tag{2.59}$$

where,

$$J = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.60}$$

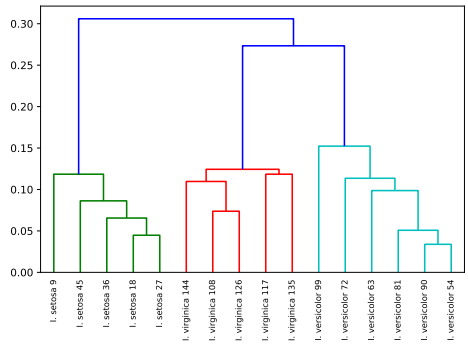
Then we have

$$\begin{aligned} Z[i]J &= \big[Z[i,0], Z[i,1], Z[i,2], Z[i,3]\big] \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \big[Z[i,1], Z[i,0], Z[i,2], Z[i,3]\big] \end{aligned} \tag{2.61}$$

The matrix product, $\big[Z[i,1], Z[i,0], Z[i,2], Z[i,3]\big]$, is then assigned to $Z[i]$ in the linkage matrix Z to define a new dendrogram layout. Compared to the original dendrogram layout, this new dendrogram layout has two clusters that are merged at the ith interior vertex swapped. Take the linkage matrix Z_S of the dendrogram of 16 Iris flower samples as an example. If we want to interchange positions of two clusters, I. versicolor 54 (the 5th cluster) and I. versicolor 90 (the 9th cluster), we just need to multiply the first row of the linkage matrix, $Z_S[0]$, with the matrix J above. The new linkage matrix and the corresponding dendrogram layout of this example are reported in Figure 2.6.

$$Z_S = \begin{bmatrix} 9 & 5 & 0.03384095 & 2 \\ 1 & 2 & 0.04476737 & 2 \\ 8 & 16 & 0.05076142 & 3 \\ 3 & 17 & 0.06553271 & 3 \\ 11 & 13 & 0.07375463 & 2 \\ 4 & 19 & 0.08627783 & 4 \\ 6 & 18 & 0.09866247 & 4 \\ 15 & 20 & 0.1096572 & 3 \\ 7 & 22 & 0.11350599 & 5 \\ 0 & 21 & 0.11844332 & 5 \\ 12 & 14 & 0.11844332 & 2 \\ 23 & 26 & 0.12433958 & 5 \\ 10 & 24 & 0.15228426 & 6 \\ 27 & 28 & 0.27335862 & 11 \\ 25 & 29 & 0.30597532 & 16 \end{bmatrix}$$

(a) New linkage matrix



(b) New dendrogram layout

Figure 2.6: New layout of a dendrogram of 16 Iris flower samples after the positions of two clusters merged at the first interior vertex are swapped.

Since interchanging positions of clusters in a dendrogram is crucial for untangle methods which will be introduced in the next chapters, we will define a swap function $\Omega_i(Z)$ that takes the linkage matrix Z as the input and swaps the two first elements in the ith row of the linkage matrix Z

$$\Omega_i(Z) = Z' \mid Z'[i] = Z[i]J \;\&\; Z'[k] = Z[k] \quad \forall k \neq i \tag{2.62}$$

where,

$$J = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.63}$$

The function $\Omega_i(Z)$ interchanges positions of the two first elements in the ith row of the linkage Z, meaning that it swaps two clusters that are merged to create the ith interior vertex in the dendrogram encoded as the linkage matrix Z. Hereafter, the usage of the swap function $\Omega_i(Z)$ is considered as to rotate the ith interior vertex in the dendrogram layout Z in order to form a new dendrogram embedding.

Each dendrogram layout imposes an order among the objects appearing as the leaves of the dendrogram. Swapping clusters at an interior vertex of the dendrogram causes a change in this order of objects. Therefore, we need a vector recording the objects in a dendrogram from right to left. This vector is called a leaf node vector and can be extracted from the linkage matrix. Suppose that we are working with a $(n-1) \times 4$ linkage matrix Z, the corresponding leaf node vector, denoted by an array V, firstly has two elements

$$V = [Z[n-1, 0], Z[n-1, 1]] \tag{2.64}$$

If an element in this vector, say $V[i]$, is greater than or equal to n, it will be substituted by two values, $Z[V[i]-n, 0]$ and $Z[V[i]-n, 1]$. It is essential to not change the order of these two values. We keep doing such substitution until all elements of the vector V are smaller than n, meaning that all elements in V are indices of original observations. Then the leaf node vector V is eventually formed showing the order of objects in the dendrogram layout from left to right. The leaf node vector plays an important roll in the calculation of the entanglement of a tanglegram, which will be revealed in section 2.6.

2.3.4 Cophenetic Matrix and Ultrametric Inequality

Besides the ordinary distances, there exists another type of distance between observations, which is the cophenetic distance. The cophenetic distance between two observations is described as the distance between the largest two clusters that contain the two objects individually. Within the context of a dendrogram, cophenetic distance between two objects is the hierarchical level at which they first occur in the same cluster (Jain and Dubes, 1988). Suppose the ith and jth patterns are original observations in clusters p and q, respectively, and p and q are joined to generate another cluster u. This is, p and q are the largest two clusters that contain the two objects individually before they are merged into cluster u which contains both the ith and jth patterns. Hence the cophenetic distance between the ith and jth observations is simply the distance between clusters p and q, which can be found in the third column in the linkage matrix. It is worthwhile to note that since different linkage methods define the distances between clusters differently, the cophenetic distances of the same pair of objects generated by distinct linkages might be unequal.

A cophenetic matrix of n objects is a $n \times n$ matrix whose elements are the cophenetic distances between each pair of n objects. Let $d_C(i, j)$ denote the cophenetic distance between the ith and jth patterns, then the cophenetic matrix is the matrix of values $D_C = [d_C(i, j)]$, which can be constructed from the corresponding linkage matrix Z. With each pair of objects, say ith and ith objects, we find the kth row in the linkage matrix Z such that $Z[k, 0]$ and $Z[k, 1]$ are the indices of the largest two clusters that contain the two objects individually. Then $Z[k, 2]$ is the cophenetic distance between ith and ith objects, or $d_C(i, j) = Z[k, 2]$. The cophenetic matrix stemming from single linkage hierarchical clustering of 16 Iris samples is displayed as below:

$$D_C = \begin{bmatrix}
0 & 0.12 & 0.12 & 0.12 & 0.12 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 \\
 & 0 & 0.04 & 0.07 & 0.09 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 \\
 & & 0 & 0.07 & 0.09 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 \\
 & & & 0 & 0.09 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 \\
 & & & & 0 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 & 0.31 \\
 & & & & & 0 & 0.10 & 0.11 & 0.05 & 0.03 & 0.15 & 0.27 & 0.27 & 0.27 & 0.27 & 0.27 \\
 & & & & & & 0 & 0.11 & 0.10 & 0.10 & 0.15 & 0.27 & 0.27 & 0.27 & 0.27 & 0.27 \\
 & & & & & & & 0 & 0.11 & 0.11 & 0.15 & 0.27 & 0.27 & 0.27 & 0.27 & 0.27 \\
 & & & & & & & & 0 & 0.05 & 0.15 & 0.27 & 0.27 & 0.27 & 0.27 & 0.27 \\
 & & & & & & & & & 0 & 0.15 & 0.27 & 0.27 & 0.27 & 0.27 & 0.27 \\
 & & & & & & & & & & 0 & 0.27 & 0.27 & 0.27 & 0.27 & 0.27 \\
 & & & & & & & & & & & 0 & 0.12 & 0.07 & 0.12 & 0.11 \\
 & & & & & & & & & & & & 0 & 0.12 & 0.12 & 0.12 \\
 & & & & & & & & & & & & & 0 & 0.12 & 0.11 \\
 & & & & & & & & & & & & & & 0 & 0.12 \\
 & & & & & & & & & & & & & & & 0
\end{bmatrix}$$

Since the cophenetic matrix contains the cophenetic distances between objects, it satisfies the properties of a dissimilarity matrix as discussed in Section 2.2.4. In fact, a cophenetic matrix $[d_C(i,j)]$ of n objects defines a fuzzy relation that has the following properties. First, it is symmetric

$$d_C(i,j) = d_C(j,i) \quad \forall i,j \in [1,n] \tag{2.65}$$

Secondly, it is irreflexive

$$d_C(i,i) = 0 \quad \forall i \in [1,n] \tag{2.66}$$

Besides satisfying these two criteria, Jain and Dubes (1988) pointed out that a cophenetic matrix obeys a ultrametric inequality

$$\max\{d_C(i,k), d_C(k,j)\} \geq d_C(i,j) \quad \forall i,j,k \in [1,n] \tag{2.67}$$

A matrix satisfying the ultrametric inequality is called a ultrametric matrix. Then a cophenetic matrix is a ultrametric dissimilarity matrix.

Jain and Dubes (1988) mentioned that a cophenetic matrix captures the structure that a given hierarchical clustering algorithm is imposing on the data. To put it another way, the cophenetic matrix represents the output of a hierarchical clustering algorithm, or the resulting dendrogram. Each dendrogram has a unique cophenetic matrix recording distances at which objects are merged to create new partitions in the dendrogram, which a proximity matrix might not achieve. It can be argued that it is the sequence of nested partitions forming the hierarchy assures the ultrametric property of the cophenetic matrix. By using the cophenetic matrix as the input for a given hierarchical clustering algorithm instead of the

dissimilarity based proximity matrix, one can reconstruct a dendrogram that is exactly the same as the one obtained by using the corresponding proximity matrix. Moreover, if we use a cophenetic matrix as the input, dendrograms from both single linkage clustering and complete linkage clustering are exactly the same (Jain and Dubes, 1988). The same result is obtained by using the average linkage and weighted linkage methods as well. All suggest that the correspondence between a cophenetic matrix and a hierarchical structure is one-to-one. Alternatively, the correspondence between a ultrametric dissimilarity matrix and a dendrogram is one-to-one because a cophenetic matrix is a ultrametric dissimilarity matrix. This statement leads to the one-to-one relationship between a min-transitive similarity matrix (or a fuzzy equivalence relation) and a dendrogram. The justification for this relation comes from the fact that a dissimilarity matrix $D = \{d(i,j)\}$ of n objects is ultrametric if and only if its corresponding similarity matrix $S = \{s(i,j)\} = \{1\} - D$ is min-transitive. Indeed, if S is min-transitive we have

$$\max\{\min\{s(i,k), s(k,j)\}\} \leq s(i,j) \quad \forall i,j,k \in [1,n] \tag{2.68}$$

which can be rewritten

$$\min\{s(i,k), s(k,j)\} \leq s(i,j) \quad \forall i,j,k \in [1,n] \tag{2.69}$$

From equations 2.51 and 2.69 we get

$$\min\{1 - d(i,k), 1 - d(k,j)\} \leq 1 - d(i,j) \quad \forall i,j,k \in [1,n] \tag{2.70}$$

which yields

$$1 - \max\{d(i,k), d(k,j)\} \leq 1 - d(i,j) \quad \forall i,j,k \in [1,n] \tag{2.71}$$

and hence

$$\max\{d(i,k), d(k,j)\} \geq d(i,j) \quad \forall i,j,k \in [1,n] \tag{2.72}$$

which implies that the dissimilarity matrix D is ultrametric.

2.4 Hierarchical Clustering Combination (HCC)

In hierarchical clustering, the use of different linkage method to a given dataset usually produces different dendrograms with unalike qualities. It means that the samples are categorized into clusters in different ways. The exist of multiple dendrograms really confounds the users when they want to choose an appropriate clustering for further usage. Hence many researchers have tried to combine the output of several hierarchical clusterings to improve the quality and robustness of clustering (Mirzaei and Rahmati, 2010). This problem is known as the hierarchical clustering combination (HCC) problem.

Since a hierarchical clustering is illustrated as a unique dendrogram, HCC can be considered as a dendrogram combination problem. Mirzaei and Rahmati (2010) put forward a definition of HCC:

> "Given a set of dendrograms find a new dendrogram which is a proper representative of the whole dendrograms set."

A proper representative of a set of dendrograms here is a combined dendrogram that is as close as possible to all dendrograms of the set. It implies that the combined dendrogram must maintain the structure of the input dendrograms in some extend. However, combining dendrograms with different hierarchy structures is not as easy as combining multiple classifiers. The reason for this is that in hierarchical clustering, objects are tied in a fixed hierarchical relationship which is shown by edges, interior indices and hierarchical levels in a dendrogram. Additionally, there is usually the existence of mismatch between clusters in different dendrograms (Mirzaei and Rahmati, 2010). Therefore, in order for the combination method to be possible it is necessary to work with an intermediate matrix presentation of dendrograms instead of working directly with dendrograms. In fact, any dendrogram can be presented by a matrix, but it is not always true that a dendrogram could be recovered by an arbitrary matrix. Based on that, the definition of hierarchical clustering combination is rewritten by Mirzaei and Rahmati (2010) as follows:

> "Given input dendrograms, each could be represented by its matrix representation, our goal is to construct a consensus matrix by aggregating the base dendrograms matrices. The consensus matrix must have an associated dendrogram, which is the result of the combination algorithm."

Summarizing, a hierarchical clustering combination method starts with description matrices of the input hierarchical clusterings, and the method should have an aggregator to combine these description matrices in order to extract an aggregated matrix (consensus matrix). This final matrix must satisfy some requirements to ensure that it has an associated dendrogram. In case we use similarity based description matrices of primary dendrograms, the consensus matrix must be transitive according to the one-to-one correspondence between a fuzzy equivalence relation and a dendrogram. In the same way, if dissimilarity based description matrices are used, the consensus matrix must be ultrametric to have an associated dendrogram based on knowledge in section 2.3.4.

2.5 Dendrogram Description Matrices

As reported in Section 2.4, the direct combination of hierarchical clustering is difficult. The most common approach to dendrogram comparison and combination is to introduce an intermediate matrix presentation of dendrograms, which is known as a description matrix or descriptor. Each dendrogram having n input objects can be represented by a symmetric description matrix of size $n \times n$. This matrix expresses the relative position of each pair of objects in a dendrogram (Podani and Dickinson, 1984). Let $T = \{t_{ij}\}$ denote the description matrix of a dendrogram H, then the element t_{ij} might be a measure of (dis)similarity between two objects i and j within this dendrogram.

Description matrices have an attribute of being ultrametric, which is introduced in section 2.3.4. Ironically, not every descriptor is ultrametric. It depends on how the descriptor is constructed. In fact, several descriptors have been defined to present different structural aspects of a dendrogram. Their definitions are given as follows:

- **Cophenetic Difference (CD)**: The broad accepted use of the term cophenetic difference of a given pair of objects in a dendrogram refers to the lowest hierarchical level at which the objects are joined together. This definition is similar to the definition of cophenetic distances between objects. Therefore, the cophenetic difference matrix is the cophenetic matrix introduced in section 2.3.4. Accordingly, the cophenetic difference matrix satisfies the requirement for being an ultrametric (Rohlf and Sokal, 1981). Since only hierarchical levels are taken into account, dendrograms with the same hierarchy but different scales on the y-axis might have very different CD matrices (Podani and Dickinson, 1984).

- **Path Difference (PD)**: For each pair of objects, for instance i, j, instead of considering the hierarchical levels, now we count a number of interior vertices along the path between i and j in a dendrogram. The matrix contains such numbers is called a Path (or cladistic or topological) Difference. This descriptor reflects the genealogical relations between pairs of objects rather than the (dis)similarities between them. Podani and Dickinson (1984) pointed out that the fundamental problem with PD matrix is that interior vertices in a dendrogram receive equal weight. This is not reasonable since the root representing the highest hierarchical level should be the most weighted vertex and the vertices right below the root should have the less weight, and so on (Podani and Dickinson, 1984). PD is not an ultrametric, therefore, it is not possible to recover a unique dendrogram from a PD descriptor Rohlf and Sokal (1981).

- **Cluster Membership Divergence (CMD)**: For each pair of objects i and j, instead of counting all vertices along the path between objects, now we count the number of objects in the smallest cluster containing both i and j. This number is called the CMD of i and j. Unlike PD, each interior vertex is now weighted according to the number of objects that are merged to create a cluster represented by the interior vertex. With this descriptor, the hierarchical levels are not preserved.

- **Partition Membership Divergence (PMD)**: This descriptor utilizes the property that a dendrogram is the illustration of a sequence of nested partitions. PMD of a given pair of objects, say i, j, is the number of partitions implied by the dendrogram in which this pair of objects does not belong to the same cluster. In other words, PMD of a pair of objects indicates the index of the partition at which objects are in the same cluster for the first time. One can see that PMD preserves the ordering of hierarchical levels in the dendrogram, and it is ultrametric.

- **Sub-tree Membership Divergence (SMD)**: In a dendrogram of n objects, each interior vertex represents a cluster whose objects can be considered as a subset of n objects. Therefore, each interior vertex usually refers to a hierarchical clustering of a subset of objects. The dendrogram of this hierarchical clustering is called a sub-tree. A dendrogram of n objects, then, has $n-1$ sub-trees corresponding to the $n-1$ interior vertices. SMD for each pair i, j is defined as the number of sub-trees in which objects i and j are not assigned together in a same cluster. One can notice that the elements of the main diagonal in SMD are not necessarily equal. As the result, SMD is not ultrametric (Podani and Dickinson, 1984).

- **Maximum Number of Edge Distance (MNED)**: In this descriptor, the original levels of the hierarchy in the dendrogram are discarded, and an abstract level (AL) for each cluster is created (Mirzaei et al., 2008). The cluster including only one object has an abstract level of zero. The abstract level of other clusters is computed as follows. Let C_i and C_j be two clusters, and let $AL(C_i)$ and $AL(C_j)$ denote the abstract levels associated to C_i and C_j, respectively. Assume that these two clusters are merged into a new cluster, C_{ij}. The abstract level of the cluster C_{ij} is defined by:

$$AL(C_{ij}) = \max\{AL(C_i), AL(C_j)\} + 1 \tag{2.73}$$

  MNED of a pair of objects, say i and j, is the abstract level of the smallest cluster that contains both i and j. Generally, MNED is not ultrametric.

For illustration, we have 6 descriptor matrices of a dendrogram in Figure 2.7 shown in Table 2.3. It is essential to notice that all descriptors are symmetric, so, it is sufficient to show just the upper triangle matrices.

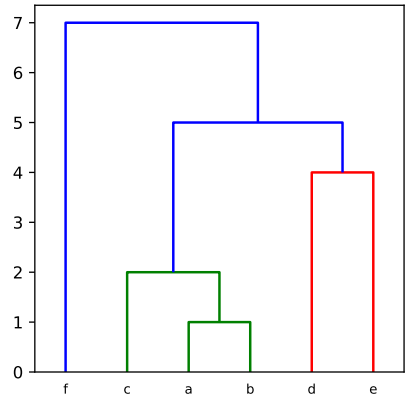


Figure 2.7: Example dendrogram.

Table 2.3: Different descriptors of the example dendrogram in Figure 2.7.

Object	CD						PMD						CMD					
a	0	1	2	5	5	7	0	1	2	4	4	5	1	2	3	5	5	6
b		0	2	5	5	7		0	2	4	4	5		1	3	5	5	6
c			0	5	5	7			0	4	4	5			1	5	5	6
d				0	4	7				0	3	5				1	2	6
e					0	7					0	5					1	6
f						0						0						1

Object	SMD						PD						MNED					
a	1	1	2	3	3	4	0	1	2	4	4	4	0	1	2	3	3	4
b		1	2	3	3	4		0	2	4	4	4		0	2	3	3	4
c			2	3	3	4			0	3	3	3			0	3	3	4
d				2	2	4				0	1	3				0	1	4
e					2	4					0	3					0	4
f						4						0						0

2.6 Tanglegram and Tanglegram Layout Problem

A tanglegram is a comparative drawing (embedding) of a pair of dendrograms, one facing the other, with matching objects connected by straight-line segments called inter-tree edges or tangle edges (Gezlaw et al., 2012). It is usually used for visually comparing two dendrograms from different hierarchical clustering methods. For instance, two dendrograms generated by the single linkage hierarchical clustering and the complete linkage hierarchical clustering of 16 Iris flower samples are placed in a tanglegram which can be seen in Figure 2.8. It is necessary that two dendrograms in the tanglegram should have the same set of objects so that two sets of objects between two dendrograms are in one-to-one correspondence. According to the natural structure of the tanglegram, the dendrogram on the left hand side of the tanglegram is called the left dendrogram. Analogously, the one on the right hand side of the tanglegram is called the right dendrogram. Since a dendrogram has various layouts, the tanglegram has also a number of layouts depending on the dendrogram embeddings. Assume that the layouts of the left dendrogram and the right dendrogram of a tanglegram are identified by two linkage matrices, Z_l and Z_r, respectively. Then the corresponding tanglegram layout can be represented by a three dimensional array $L = [Z_l, Z_r]$. Hereafter, we shall use this array L to denominate the corresponding tanglegram and its tanglegram layout.

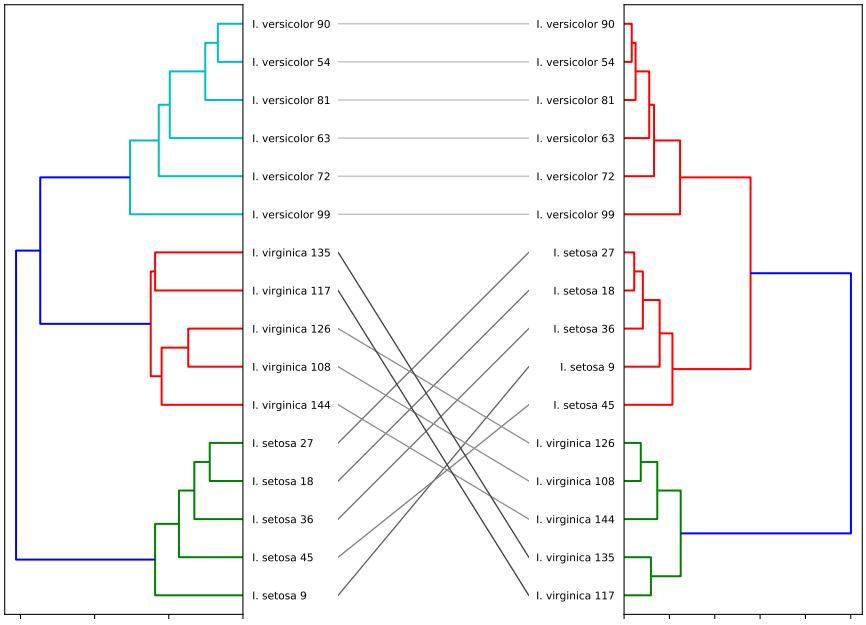


Figure 2.8: Original tanglegram layout of 16 Iris flower samples with the entanglement of 0.46.

In a tanglegram, a crossing occurs when two inter-tree edges intersect, and the number of crossings is dependent upon the layout of the two dendrograms in the tanglegram. In general terms, the "worst" tanglegram layout is defined as a tanglegram layout when the leaf node vector of one dendrogram in the tanglegram is a complete reverse of the leaf node vector of the other dendrogram. From a practical standpoint, a tanglegram with many crossing inter-tree edges can be intricately tangled and hard to analyze (Gezlaw et al., 2012). It leads to a need for finding a drawing of the tanglegram with zero crossings. This problem is known as the planar embedding problem (Gezlaw et al., 2012). If that embedding does not exist, we may want to find an embedding with as few crossings as possible. The problem of finding a graphical layout of two trees that produces the minimum crossings is known as the crossing minimization or the tanglegram layout (TL) problem (Bansal et al., 2009). Note that the minimum of crossings is not a tree-distance measure since two dendrograms with very different topologies might have no crossing at all in the tanglegram (Buchin et al., 2008; Vienne, 2018). Thus, the number of crossings between two dendrograms has a tendency to provide an approximation of their level of topological similarity (or congruence) (Vienne, 2018). In this thesis, we advocate the idea of minimizing the number of crossings for visualization purposes, and we use the optimized tanglegram to interpret the similarity and dissimilarity between dendrograms.

Counting the number of crossings is a difficult measurement as one inter-tree edge can intersect many other inter-tree edges causing the complexity of crossings. It is even more challenging with big tree sizes. Therefore, an entanglement value was formed to substitute the number of crossings in tanglegrams. Assume that we are working on n objects and having an arbitrary tanglegram layout, $L = [Z_l, Z_r]$. The entanglement measurement starts by recalling the leaf node vector V of a dendrogram layout Z. Let V_l and V_r are the leaf node vectors of the left dendrogram and the right dendrogram, respectively. We can alternatively call the leaf node vector of the left dendrogram, V_l, as the left leaf node vector. Similarly, V_r is called the right leaf node vector. Noticeably, the indices of objects in these leaf node vectors are different from the indices of objects in the pattern matrix or proximity matrix. Let differentiate between these two types of indices by calling the former as the temporary indices and the later as the original indices of objects. Entanglement is measured by firstly giving the objects in the left leaf node vector the values of 0 til $n-1$, and then matching these numbers with the right leaf node vector. In particular, we define a matching vector M_l whose elements are the temporary indices of objects in the left leaf node vector

$$M_l = [V_l(V_l[i])] \tag{2.74}$$

In other words, the ith element in the vector M_l is simply the value of i, meaning that

$$M_l[i] = i \tag{2.75}$$

or

$$M_l = [0, 1, 2, ..., n-1] \tag{2.76}$$

The temporary indices of objects in the left leaf node vector is matched with objects in the right leaf node vector V_r by defining a matching vector M_r whose the jth element is the temporary index of the object $V_r[j]$ in the left leaf node vector

$$M_r = [V_l(V_r[j])] \tag{2.77}$$

An illustration of how to match the temporary indices of objects in the left leaf node vector with objects in the right leaf node vectors in the original tanglegram layout of 16 Iris flower samples (Figure 2.8) is given in Figure 2.9.

V_l	0	4	3	1	2	15	11	13	12	14	10	7	6	8	5	9
M_l	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

V_r	12	14	15	11	13	4	0	3	1	2	10	7	6	8	5	9
M_r	8	9	5	6	7	1	0	2	3	4	10	11	12	13	14	15

Figure 2.9: Example of the left and right leaf node vectors and the corresponding matching vectors M from the original tanglegram layout of 16 Iris flower samples (Figure 2.8).

Now, we calculate the Euclidean distance between two vectors M_l and M_r, which is called the displacement between the left and right dendrograms of the arbitrary tanglegram layout $L = [Z_l, Z_r]$

$$d(M_l, M_r) = \sqrt{\sum_{i=0}^{n-1} \Big(M_l[i] - M_r[i] \Big)^2} \tag{2.78}$$

The same displacement between dendrograms constituting the "worst" tanglegram layout $L^w = [Z_l^w, Z_r^w]$ should be also computed. Let V_l^w and V_r^w be the leaf node vectors from the left and right dendrograms in the "worst" tanglegram layout, respectively. Then the matching vector M_l^w containing the temporary indices of objects in the left leaf node vector V_l^w is

$$M_l^w = [0, 1, 2, ..., n-1] \tag{2.79}$$

According to the definition of the "worst" tanglegram layout, one leaf node vector is the completely reversed vector of the other. Hence matching the temporary indices of objects in the left leaf node vector with objects in the right leaf node vector in the "worst" tanglegram layout gives the matching vector M_r^w as follows

$$M_r^w = [n-1, n-2, n-3, ..., 0] \tag{2.80}$$

Then applying Equation 2.78 for the "worst" tanglegram layout yields the displacement between dendrograms in the "worst" tanglegram layout

$$\begin{aligned} d(M_l^w, M_r^w) &= \sqrt{(n-1)^2 + (n-3)^2 + ... + (1-n)^2} \\ &= \sqrt{(n-1)^2 + (n-2-1)^2 + ... + (n-2(n-1)-1)^2} \\ &= \sqrt{\sum_{j=0}^{n-1} \Big(n - 2j - 1 \Big)^2} \end{aligned} \tag{2.81}$$

Entanglement value of the arbitrary tanglegram layout $L = [Z_l, Z_r]$ is obtained by a function $\epsilon(L)$ which divides the displacement between two dendrograms in this arbitrary tan-

glegram layout by that number in the "worst" tanglegram layout

$$\epsilon(L) = \frac{\sqrt{\sum_{i=0}^{n-1}\left(M_l[i] - M_r[i]\right)^2}}{\sqrt{\sum_{j=0}^{n-1}\left(n - 2j - 1\right)^2}} \tag{2.82}$$

where n is the tree size. The entanglement value indicates the quality of the alignment of the two dendrogram in the tanglegram layout. Notice that entanglement is a measure between 0 and 1. Entanglement of zero means that the tanglegram layout has no crossing and entanglement of 1 indicates that the tanglegram layout is the "worst" tanglegram layout.

Since this thesis is technically based on the programming language Python, we looked for available solutions to solve the TL problem in this programming language. For many years, the statistical library $SciPy$ of Python has been considered as an effective tool for dendrogram modelling. Nevertheless, addressing the TL problem is not covered in this library. To the best of our knowledge, few researchers have addressed the TL problem, but no comprehensive framework has been available in Python for improving tanglegram layout. In the programming language R, Galili introduced a package named $dendextend$ which provides not only functions for manipulating dendrogram's structure but also several advanced methods for resolving the TL problem (Galili, 2015). Among 6 untangle methods implemented in the R package $dendextend$, the "step2side" method is one of the most commonly used methods for finding a better layout of a tanglegram. Based on the fact that a dendrogram is an object which can be rotated on its hinges (interior vertices) without changing its topology, the method "step2side" rotates dendrograms to best fit one another.

The "step2side" algorithm for untangling the tanglegram $L = [Z_l, Z_r]$ of n objects can be described in three main steps:

1. Create new tanglegram layouts by keeping the left dendrogram Z_l unchanged and rotating the right dendrogram Z_r. The right dendrogram is rotated by using the swap function $\Omega_i(Z_r)$, $0 \leq i \leq n-2$, to interchange the clusters merged at each interior vertex starting from the first interior vertex to the root. Each time we have a new tanglegram layout, if it has a smaller entanglement than the previous tanglegram layout, $\epsilon([Z_l, \Omega_i(Z_r)]) < \epsilon([Z_l, Z_r])$, then Z_r is replaced by $\Omega_i(Z_r)$.

2. Create new tanglegram layouts by keeping the right dendrogram Z_r unchanged and rotating the left dendrogram Z_l. The left dendrogram is rotated by using the swap function $\Omega_i(Z_l)$, $0 \leq i \leq n-2$, to interchange the clusters merged at each interior vertex starting from the first interior vertex to the root. Each time we have a new tanglegram layout, if it has a smaller entanglement than the previous tanglegram layout, $\epsilon([\Omega_i(Z_l), Z_r]) < \epsilon([Z_l, Z_r])$, then Z_l is replaced by $\Omega_i(Z_l)$.

3. Repeat steps 1 and 2 until a local optimal solution is reached.

The algorithm of the "step2side" method is summarized in a flowchart detailed in Figure 2.10. Variables used in this flowchart are described in Table 2.4.

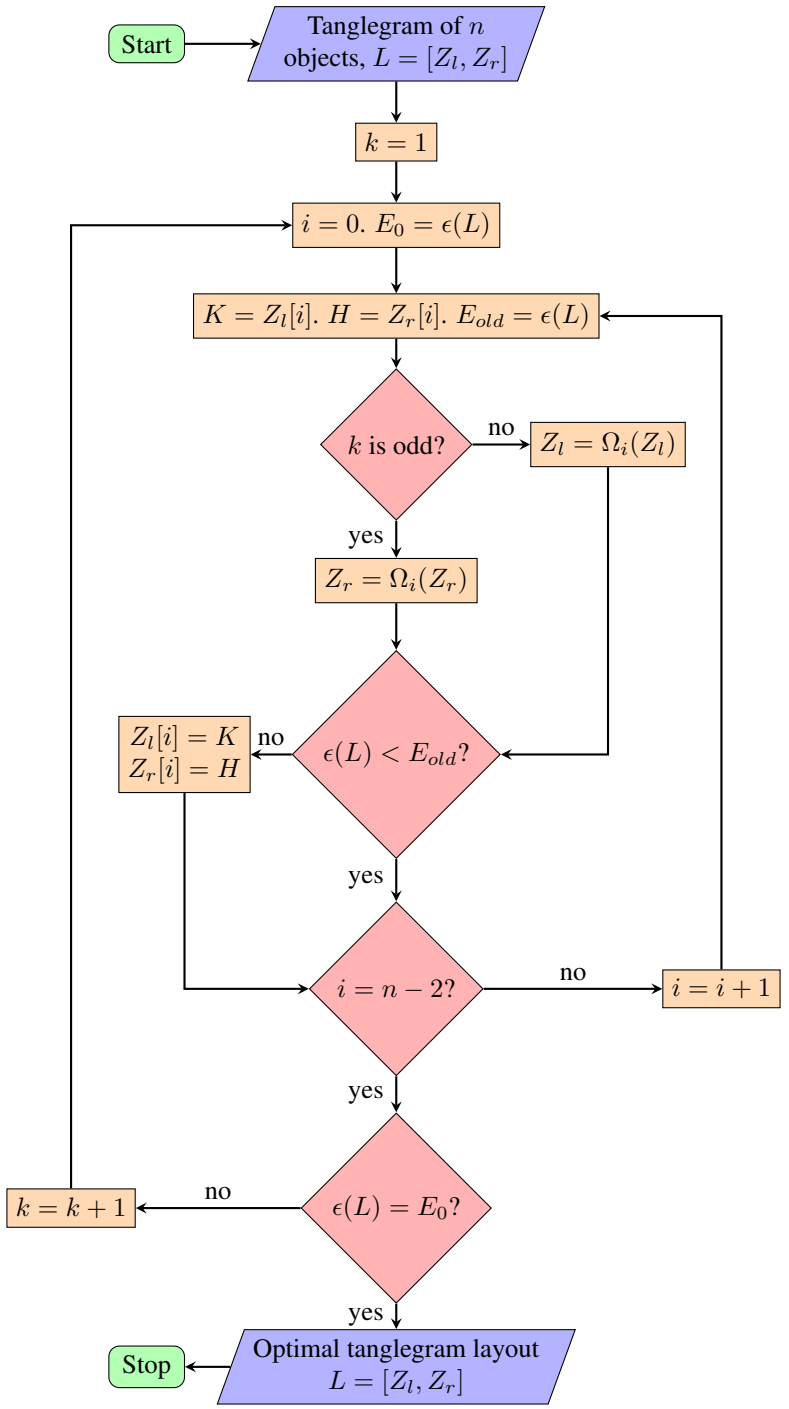


Figure 2.10: Algorithm of "step2side" method in R.

Table 2.4: Summary of variables used in ”step2side” algorithm.

Symbol	Description	Variable type
n	Number of objects in a dendrogram	integer
Z_l	The left dendrogram encoded as a linkage matrix	ndarray
Z_r	The right dendrogram encoded as a linkage matrix	ndarray
L	The tanglegram consitituted by the left and right dendrograms	ndarray
E_0	The entanglement of L before rotating a dendrogram	float
E_{old}	The entanglement of L before interchanging clusters merged at an interior vertex of a dendrogram	float
H, K	Duplicated rows before using the swap function	ndarray
i, k	Indices	integer

An attempt to untangle the original tanglegram of 16 Iris samples is made by employing the ”step2side” algorithm. It gives a new tanglegram layout with an entanglement of 0.43 (Figure 2.11). This tanglegram layout has a smaller entanglement value than the original one, however, it bears a close resemblance to the original tanglegram layout and still looks tangled with a lot of crossings.

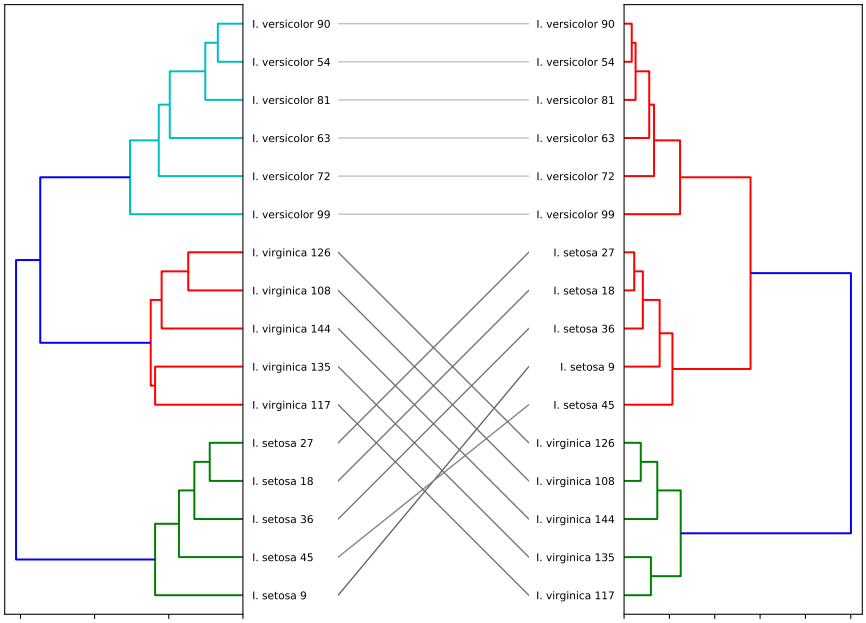


Figure 2.11: Optimized tanglegram layout having the entanglement of 0.43 after using the ”step2side” method.

Chapter 3

Methodology

3.1 Introduction

As mention in section 2.3.4, there is one-to-one correspondence between ultrametric dissimilarity matrices and dendrograms. Some of the dendrogram description matrices can be converted into dissimilarity matrices which meet the requirement of the ultrametric property. These descriptors are considered as the representation of dendrograms and can be used as the input for reconstructing and combining dendrograms. The correspondence between a transitive similarity matrix and a dendrogram is also one-to-one. In other words, a dendrogram could be constructed corresponding to a similarity matrix if it is transitive. Additionally, the definition of HHC in section 2.4 mentions that the consensus matrix formed by aggregating the dendrogram description matrices must have an associated dendrogram. Therefore, it can be reasoned out that this consensus matrix must be either ultrametric or min-transitive, depending on whether the consensus matrix is a dissimilarity matrix or similarity matrix.

We assume that we use description matrices of base dendrograms in the form of similarity matrices as the input for a HCC algorithm. The definition of HCC can be redefined as the aim to aggregate the descriptors of base dendrograms into a consensus matrix which must be min-transitive to generate an associated dendrogram. This problem was studied by Mirzaei and Rahmati. In 2010, they proposed an algorithm called Min-trAnsiTive Combination of Hierarchical clusterings (MATCH), aiming to construct a min-transitive consensus matrix from multiple dendrogram description matrices (Mirzaei and Rahmati, 2010). MATCH algorithm starts with similarity matrices of base hierarchical clusterings, and makes use of min-transitive closure to aggregate these input matrices into a transitive consensus matrix. In the two first sections below the formula for calculating the min-transitive closure and the main steps in MATCH algorithm will be described. We also propose a new algorithm that can provide the same results as the MATCH algorithm, but it has less time complexity.

Several untangle methods in R language are implemented to solve the TL problem. In section 2.6 we investigate one of the most widely used untangle methods in R lan-

guage: the "step2side" method. In the example case of 16 Iris flower samples, we must remark that there is no big difference between the optimized tanglegram stemming from the "step2side" method and the original tanglegram. The optimized tanglegram layout still has a large entanglement (0.43). This can be attributed to the big tree size or the complexity of the crossings. The "step2side" method might be good enough to solve the TL problem of other cases with small number of objects, but in this case, it is just able to reduce the entanglement of the original tanglegram by 0.03. If we assume that there exists a tanglegram layout with no crossings at all between inter-tree edges for this 16 Iris flower samples case, then the result from the "step2side" method is a local minimum in this optimization problem. Can we find a tanglegram layout that is superior to the result from "step2side" method? The work in this master thesis was motivated by this question. In the last subchapter, we propose a new untangle method called a shuffle and untangle ($S\&U$) method aiming to find a tanglegram layout that has a smaller entanglement than the optimized tanglegram from the "step2side" method. The rest of this chapter will present concepts needed for our proposed $S\&U$ method.

3.2 Transitive Closure

According to the definition of HCC in Section 2.4, the consensus matrix must have an associated dendrogram. Whereas, as reported in Section 2.3.4, the transitivity of the consensus matrix implies the existence of a dendrogram. With that in mind, we need to study an algorithm to turn a relation (or the corresponding membership matrix) to a transitive one. Available algorithms to perform this task can be divided into 3 categories, (1) the algorithms which calculate the min-transitive closure, (2) the algorithms which compute the min-transitive opening, and (3) the algorithms that calculates the min-transitive approximation of the input matrix (Naessens et al., 2002). In the sequel, we shall merely introduce the min-transitive closure algorithm as it is used to develop the MATCH and square algorithms in the next section.

The closure of a crisp relation R with respect to a specific property P is a relation that has the property P, contains R, and has the least possible members (Klir et al., 1997). For fuzzy relations, the closure of a fuzzy relation R with respect to a specific property P is defined as the smallest relation R' containing R and satisfying the property P (Klir et al., 1997). Bandler and Kohout (1988) claim that R' is P-closure of R, if and only if R' satisfies three properties as follows,

1. R' has property P
2. $R \subseteq R'$
3. If $R \subseteq S$ and S has property P, then $R' \subseteq S$

If there exists two P-closure of R, say R' and S, according to the third property, we have $R' \subseteq S$ and $S \subseteq R'$. Consequently, we have $R' = S$, meaning that a P-closure, if it exists, must be unique.

If we substitute the min-transitive for the property P in the definition of closure above, we obtain the definition of the min-transitive closure of a fuzzy relation which is simply

called the transitive closure. Specifically, the transitive closure of an arbitrary fuzzy relation S is defined as the unique min-transitive fuzzy relation $\overline{S}$ on X that contains S and is itself contained in any transitive fuzzy relation containing S. In other words, transitive closure $\overline{S}$ of a fuzzy relation S is a fuzzy relation that satisfies the min-transitive property, and its elements are the smallest possible values greater than or equal to elements of S (Mirzaei and Rahmati, 2010). Let S be a fuzzy relation on X with n is the order of the corresponding membership matrix, the formula to compute the transitive closure $\overline{S}$ of S is constructed by Zadeh (1971) as follows:

$$\overline{S} = S \cup S^2 \cup ... \cup S^n = \bigcup_{k=1}^{n} S^k \tag{3.1}$$

where

$$S^k = S^{k-1} \circ S \quad \forall k \in [2, n] \tag{3.2}$$

and

$$S^1 = S \tag{3.3}$$

Equation 3.1 helps us to transform any fuzzy relation into a min-transitive fuzzy relation by calculating the transitive closure. In particular, we can transform a similarity relation into an equivalence relation by calculating its transitive closure.

An algorithm for computing a min-transitive closure $\overline{S}$ of an arbitrary relation S with the corresponding membership matrix of order n was introduced by Klir et al. (1997) (Figure 3.1). This algorithm is applicable to both crisp and fuzzy relations. In Figure 3.1, the min-transitive closure algorithm can terminate prior to computing the relation S^n when no new relation is produced, meaning that the min-transitive closure of the relation S is found. Indeed, having the same results in two consecutive iterations means that we have

$$\overline{S} \cup (\overline{S} \circ \overline{S}) = \overline{S} \tag{3.4}$$

Since the union is the max-based operator, Equation 3.4 yields

$$\overline{S} \circ \overline{S} \subseteq \overline{S} \tag{3.5}$$

inferring that $\overline{S}$ is min-transitive.

Input: Fuzzy relation S
Output: Transitive relation $\overline{S}$
Step 1: $\overline{S} = S$
Step 2: Calculate $\overline{S} = \overline{S} \cup (\overline{S} \circ \overline{S})$
Step 3: Terminate if $\overline{S}$ is unchanged in two consecutive iterations or the number of iterations approaches n.
Else, go to step 2.

Figure 3.1: General min-transitive closure algorithm.

Consider the following fuzzy relation

$$R = \begin{bmatrix} 1 & 0.1 & 1 & 0.6 \\ 0.1 & 1 & 0.2 & 0.2 \\ 1 & 0.2 & 1 & 0.6 \\ 0.6 & 0.2 & 0.6 & 1 \end{bmatrix}$$

This fuzzy relation R on X is a similarity relation as it has the properties of reflexivity and symmetry. Yet, it is not transitive since it does not satisfy Equation 2.47. As an example, $\mu_R(x_1, x_3) = 1$ and $\mu_R(x_3, x_2) = 0.2$, but $\mu_R(x_1, x_2) = 0.1 < \min(1, 0.2)$. Indeed, the max-min composition of R is not a subset of R, or alternatively

$$R^2 = R \circ R = \begin{bmatrix} 1 & 0.2 & 1 & 0.6 \\ 0.2 & 1 & 0.2 & 0.2 \\ 1 & 0.2 & 1 & 0.6 \\ 0.6 & 0.2 & 0.6 & 1 \end{bmatrix} \not\subseteq R$$

In this instance, the relation R^2 on X is transitive after one max-min composition. Further compositions give the same result, $R^4 = R^3 = R^2$.

3.3 MATCH Algorithm

MATCH algorithm, which was proposed by Mirzaei and Rahmati (2010), is an algorithmic framework combining multiple hierarchical clusterings into a combined hierarchical clustering.

Before introducing the main steps of the MATCH algorithm, we first introduce the input data for this algorithm. Assume that we have a set of m initial dendrograms. As discussed in section 2.5, the ith dendrogram of the set is represented by a descriptor whose elements can be normalized in the $[0, 1]$ interval, resulting a dissimilarity based descriptor matrix D_i. This dissimilarity matrix D_i can be converted into a similarity matrix S_i by the simple transformation $S_i = [1] - D_i$. Therefore, from the set of the hierarchical clusterings, we have a set of similarity matrices $\{S_i\}$, $1 \leq i \leq m$, which is the input for the MATCH algorithm.

The MATCH algorithm aggregates the m similarity matrices of the base dendrograms into a transitive consensus matrix by using the general min-transitive closure approach as highlighted in Figure 3.1. Specifically, MATCH algorithm starts with a matrix $\overline{S} = I$, where I is an identity matrix, and gradually composes $\overline{S}$ with all similarity matrices. With each similarity matrix, the algorithm does the following calculations. Firstly, it computes a max-min composition of $\overline{S}$ and the similarity matrix. Secondly, it finds a union of this max-min composition with $\overline{S}$ and assigns this resulting union to $\overline{S}$. After doing these calculations for the last similarity matrix, the procedure is repeated for all similarity matrices in the set starting with the first similarity matrix. The whole calculation continues until $\overline{S}$ does not change in an epoch, meaning that the final transitive consensus matrix is found.

From the consensus matrix, which is the last $\overline{S}$, the associated final hierarchical clustering is generated. This consensus matrix is a similarity-based transitive matrix $\overline{S}$, so it can be converted into a dissimilarity-based matrix (dissimilarity matrix) $\overline{D} = [1] - \overline{S}$. Since

the consensus matrix $\overline{S}$ is transitive, the associated dissimilarity matrix $\overline{D}$ is ultrametric. Once this ultrametric dissimilarity matrix is known, the final hierarchical clustering is uniquely constructed by performing standard hierarchical clustering algorithms such as single linkage hierarchical clustering or complete linkage hierarchical clustering, and the MATCH algorithm subsequently terminates.

The flowchart of MATCH algorithm is illustrated in Figure 3.2. In this figure, the main step is the iteration helping to find the aggregated matrix. It is built based on steps of the min-transitive closure algorithm (Figure 3.1). The description of all variables used in the flowchart is listed in Table 3.1.

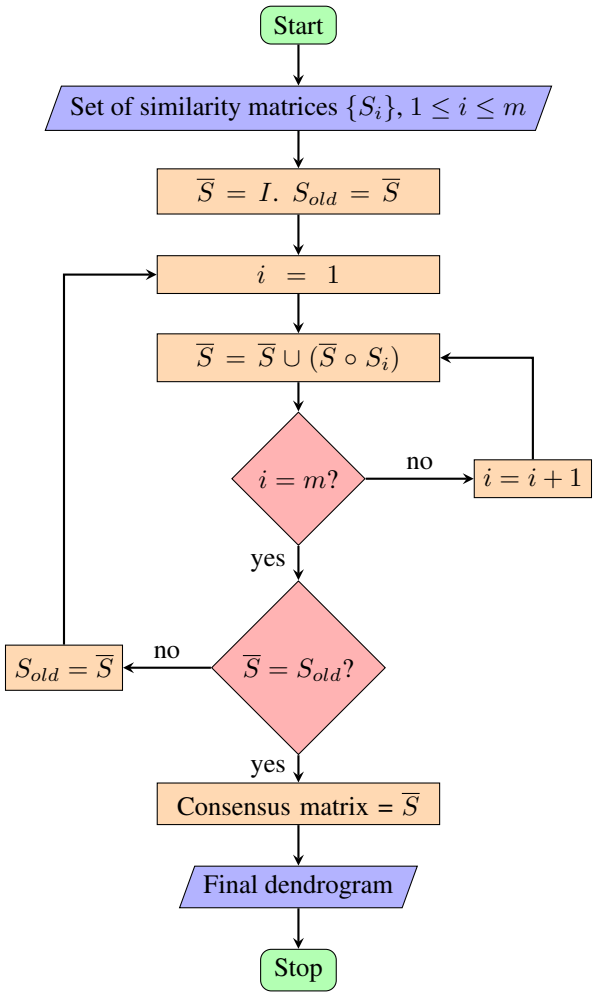


Figure 3.2: MATCH algorithm.

Table 3.1: Summary of variables used in MATCH and square algorithms.

Symbol	Description	Variable type
m	Number of similarity matrices in a set	integer
i	Index of a similarity matrix in the set	integer
S_i	A similarity matrix	ndarray
$\overline{S}$	Consensus matrix	ndarray
S_{old}	Consensus matrix before each iteration	ndarray

3.4 Square Algorithm

For a fuzzy reflexive relation R, Klir et al. (1997) suggested calculating its transitive closure by calculating the sequence of relations

$$R^2 = R \circ R \tag{3.6a}$$
$$R^4 = R^2 \circ R^2 \tag{3.6b}$$
$$\vdots$$
$$R^{2^k} = R^{2^{k-1}} \circ R^{2^{k-1}} \tag{3.6c}$$

until no new relation is produced or $2^k \geq n-1$, meaning that the transitive closure is generated. This is clearly a less time-consuming approach than the general algorithm in Figure 3.1 as it skips calculating the relations to the power of odd numbers. As in this thesis, we only work with the similarity matrices which are reflexive, we can rely on such calculation of the sequence of relations to calculate the transitive closure of a similarity matrix. By being inspired by this approach and the MATCH algorithm, we propose an algorithm called the square algorithm utilizing the calculation of the sequence of relations as in Equations 3.6a- 3.6c to combine the similarity matrices of base dendrograms into a transitive consensus matrix.

The input data for the square algorithm are similar to the MATCH algorithm. A set of m similarity matrices of the base dendrograms are gradually aggregated by using the max-min composition operator to form a matrix $\overline{S}$. In doing so we keep calculating the sequence of relations

$$\overline{S}^2 = \overline{S} \circ \overline{S} \tag{3.7a}$$
$$\overline{S}^4 = \overline{S}^2 \circ \overline{S}^2 \tag{3.7b}$$
$$\vdots$$
$$\overline{S}^{2^k} = \overline{S}^{2^{k-1}} \circ \overline{S}^{2^{k-1}} \tag{3.7c}$$

until the results in two consecutive calculations are the same or $2^k \geq m-1$, meaning that we gain the transitive consensus matrix from m similarity matrices. This transitive consensus matrix has an associated dendrogram which is the combined dendrogram of m base dendrograms. The square algorithm is summarized in a flowchart in Figure 3.3. All variables in this flowchart are the same as those used in the flowchart of the MATCH algorithm (Table 3.1).

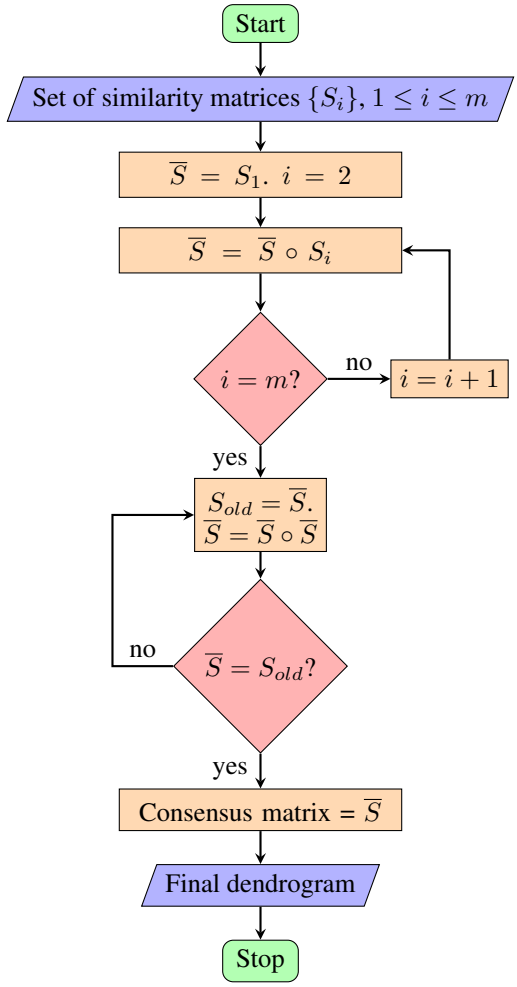


Figure 3.3: Square algorithm.

3.5 Shuffle and Untangle Method

Shuffle and untangle ($S\&U$) is a greedy algorithm aiming to solve the TL problem to get better results than the "step2side" method.

Prior to introducing the algorithm of our $S\&U$ method, some assumptions and definitions need to be introduced. Suppose we are working with a tanglegram constituted by two dendrograms of the same set of n objects. As reported in section 2.6, the tanglegram layout is characterized by the layout of its dendrograms. Firstly, let two linkage matrices Z_l and Z_r define the layouts of the left dendrogram and the right dendrogram, respectively. Then the corresponding tanglegram has a layout L that can be described by a three dimensional array $L = [Z_l, Z_r]$. More often than not, to find out a better drawing for a given

tanglegram, we have to deal with a set of n different tanglegram layouts simultaneously, denoted $\mathcal{L} = \{L_i\}$, $0 \leq i \leq n-1$. In this set, each element L_i is a three dimensional array defining a tanglegram layout. Secondly, we utilize the swap function $\omega_i(Z)$ in Section 2.3.3 to define a function $\mu_i([Z, Z'])$ which takes the tanglegram layout, $[Z, Z']$, as the input and returns a set of 4 different tanglegram layouts

$$\mu_i([Z, Z']) = \{[Z, Z'], [\omega_i(Z), Z'], [Z, \omega_i(Z')], [\omega_i(Z), \omega_i(Z')]\} \tag{3.8}$$

In other words, the function $\mu_i([Z, Z'])$ shuffles clusters merged at the ith interior vertex in both dendrograms constituting the tanglegram layout $[Z, Z']$ to get new tanglegram layouts. Therefore, we consider the function $\mu_i([Z, Z'])$ as a function that shuffles the tanglegram at the ith interior vertex. In case we want to shuffle the clusters joined at the ith interior vertex of the left dendrogram and clusters merged at the jth interior vertex of the right dendrogram, we use the function

$$\mu_{i,j}([Z, Z']) = \{[Z, Z'], [\omega_i(Z), Z'], [Z, \omega_j(Z')], [\omega_i(Z), \omega_j(Z')]\} \tag{3.9}$$

Additionally, we need to define another function $\tau_i(Z)$ which returns the index of an interior vertex whose one of two children clusters is the ith cluster

$$\tau_i(Z) = t \mid Z[t, 0] = i \text{ or } Z[t, 1] = i \tag{3.10}$$

Finally, we employ the entanglement function in Section 2.6 to define a function $\xi(\mathcal{L})$ which takes a set of n tanglegram layouts, $\mathcal{L} = \{L_i\}$, $0 \leq i \leq n-1$, as the input and outputs the tanglegram layout with the smallest entanglement amongst all layouts in the set

$$\xi(\mathcal{L}) = L_i \mid \epsilon(L_i) = \min_{j=0}^{n-1} \epsilon(L_j) \tag{3.11}$$

We construct the algorithm of the $S\&U$ method applied for the tanglegram layout $L_0 = [Z_l, Z_r]$ as follows:

1. Initializing: Create a set $\mathcal{L} = \{L_0\}$ consisting of the single object $L_0 = [Z_l, Z_r]$. Next, we calculate the entanglement of the tanglegram layout L_0, $\epsilon(L_0)$. Let m define the number of interior vertices we want to shuffle in the second step. Set $m = 1$.

2. Shuffling: We shuffle each tanglegram layout L_i in $\mathcal{L}$ at the $n - m - 1$th interior vertex to get a set of four different tanglegram layouts including the layout L_i. Afterwards, $\mathcal{L}$ is the union of all these sets

$$\mathcal{L} = \bigcup_{i=0}^{4^{m-1}-1} \mu_{n-m-1}(L_i) \tag{3.12}$$

3. Untangling: With each tanglegram layout L_i in $\mathcal{L}$ that has not been optimized before, we perform two optimization algorithms, a coarse optimization and a fine optimization:

(a) Coarse optimization: Shuffle the tanglegram layout L_i at each interior vertex from the first to the $n - m - 2$th interior vertex. After each time we do the shuffle, we obtain a set of 4 different tanglegram layouts. Following this, the tanglegram layout having the smallest entanglement among these 4 layouts will be saved and used for the next shuffle. The layout having the smallest entanglement after shuffling the $n - m - 2$th interior vertex is assigned to L_i, and its entanglement, $\epsilon(L_i)$, is considered as an optimal entanglement. We repeat the coarse optimization for L_i until the optimal entanglement cannot be decreased.

(b) Fine optimization: Further optimization is performed on the resulting tanglegram layout L_i from the coarse optimization. With each singleton cluster, we find the interior vertices in both dendrograms such that one of two children clusters of these interior vertices is the singleton cluster. Then, we shuffle the tanglegram at these interior vertices to get a set of 4 tanglegram layouts. The tanglegram layout with the smallest corresponding entanglement among these 4 layouts is used for the next rotation. The layout having the smallest entanglement after optimizing through all objects is subsequently assigned to L_i. Its entanglement value, $\epsilon(L_i)$, is assigned to the optimal entanglement. We repeat this fine optimization for L_i until the newly optimal entanglement is the same as the previous one.

With the completion of these steps, all tanglegram layouts L_i in $\mathcal{L}$ are optimized. The tanglegram layout having the smallest entanglement among all optimized tanglegram layouts in $\mathcal{L}$ is considered as the best optimal layout with the associated best optimal entanglement.

4. Stopping criteria: The algorithm terminates if a tanglegram embedding with a target entanglement is obtained or if the best optimal entanglement is the same in two consecutive values of m. If the stopping criteria are not fulfilled, we increase m by 1 and go back to step 2.

A flowchart of this algorithm is shown in Figure 3.4 with the hope that the reader can conceptualize the way the $S\&U$ method works and easily implement it. All variables used in this flowchart are described in Table 3.2.

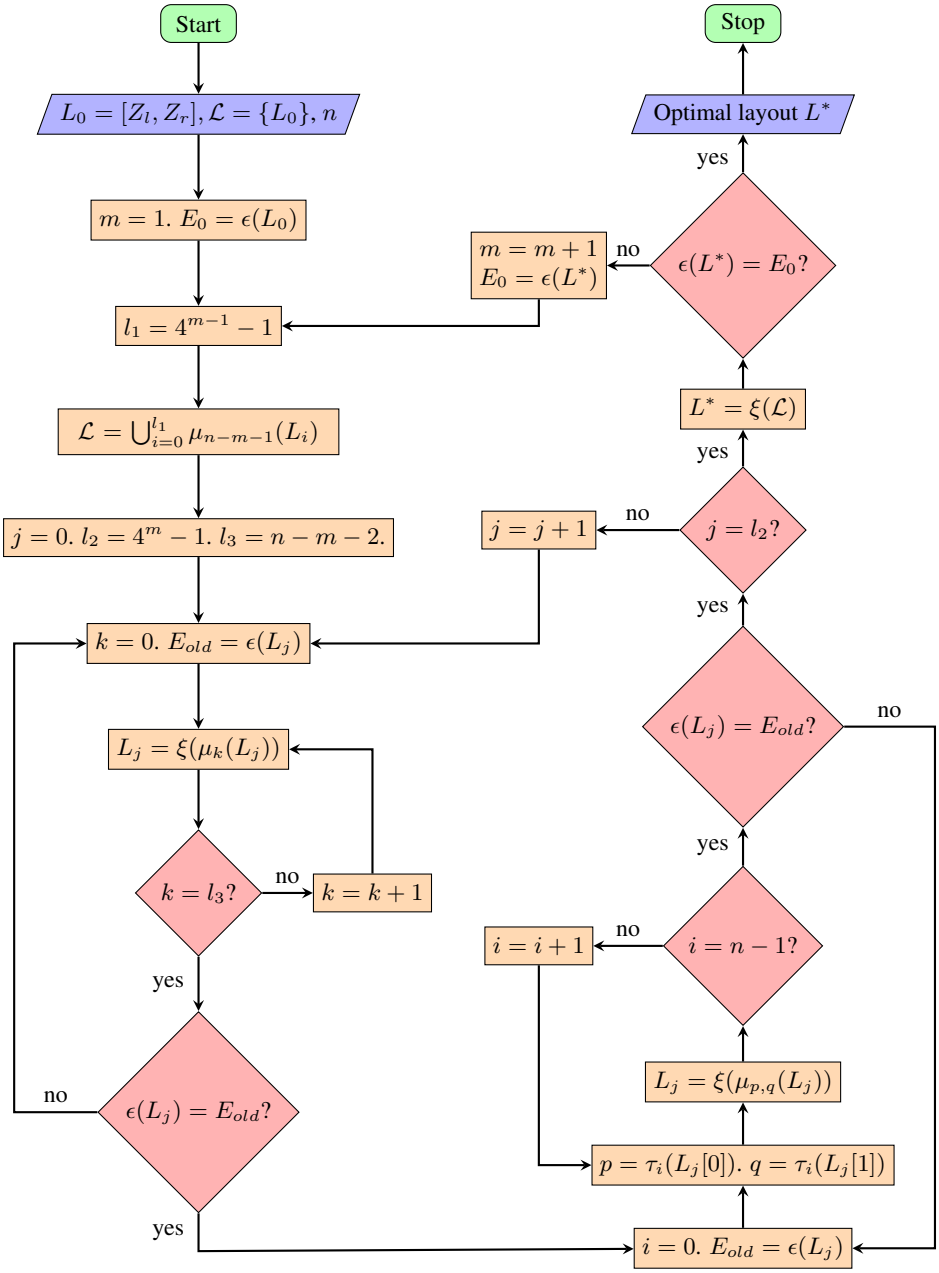


Figure 3.4: Algorithm of $S\&U$.

Table 3.2: Summary of variables used in $S\&U$ algorithm.

Symbol	Description	Variable type
$\mathcal{L}$	A set of tanglegram layouts, each layout L_i is defined by the left and right dendrograms encoded as linkage matrices	ndarray
L^*	The optimal tanglegram layout after optimization	ndarray
Z_l	The left dendrogram encoded as a linkage matrix	ndarray
Z_r	The right dendrogram encoded as a linkage matrix	ndarray
J	A matrix defined in Section 2.3.3 used in the matrix multiplication to swap two first elements in a row of a linkage matrix	ndarray
n	The number of objects in a dendrogram	integer
m	The number of interior vertices to be shuffled	integer
E_0	The entanglement of the optimal tanglegram layout before shuffling	float
E_{old}	The entanglement of a tanglegram layout before each time we do the coarse optimization or fine optimization	float
i, j, k, p, q	Variables to store indices	integer
l_1	The number of tanglegram layouts before shuffling	interger
l_2	The number of tanglegram layouts after shuffling	integer
l_3	The index of the last interior vertex we swap in the coarse optimization	integer

Applying the proposed untangle method for the tanglegram of the 16 Iris flower example gives us an optimal tanglegram as shown in Figure 3.5. From the graph we can see that the optimized tanglegram has no crossings at all. Compared to the one from the "step2side" method (Figure 2.11) with the entanglement of 0.43, the tanglegram optimized by the $S\&U$ method obviously has a smaller entanglement (entanglement of zero) which the "step2side" method has failed to provide.

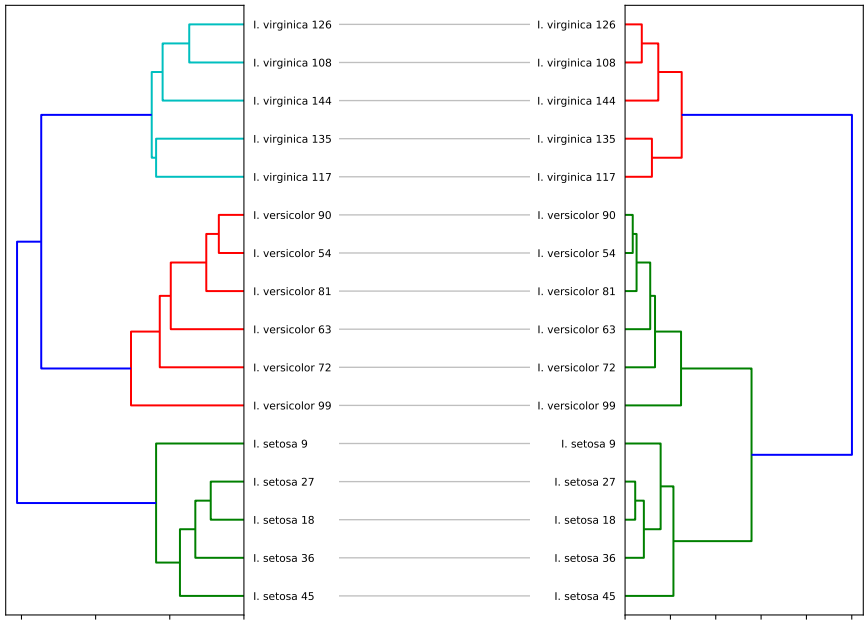


Figure 3.5: Optimized tanglegram layout from the $S\&U$ method.

Chapter 4

Applications and Results

A case study approach has been used in this thesis. We attempt to combine two base dendrograms into a final dendrogram using the MATCH algorithm and the square algorithm. The final dendrogram will be paired with two original dendrograms to form tanglegrams. Then the "step2side" untangle method and $S\&U$ method are used to untangle the tanglegrams. The input data and the results of all these experiments will be displayed in the following sections.

4.1 Materials

With regards to one of two original dendrograms, we should refer to the work of Chawshin et al. (2021). The authors evaluated the possibility of automated lithologies classification using 2D whole core X-ray computerized tomography (CT) images and convolutional neural network. In their study, a convolutional neural network model was trained to learn 20 lithofacies classes from 2D whole core CT scans. To evaluate the performance of the trained model, the authors used the model to predict the lithofacies classes, and a confusion matrix of dimensions 20×20 was formed by cross-classifying the predicted lithofacies and the actual lithofacies obtained from the core descriptions. The confusion matrix can be found in Figure 4.1. If the accuracy of the prediction is 100%, all elements in the main diagonal of the confusion matrix will be 1, and the other elements will have values of zero. In practice, Chawshin remarked that the resulted confusion matrix does not achieve the matrix form described above. More specifically, some off-diagonal elements of the confusion matrix have the values greater than zero, meaning that there are some misclassifications in the automated lithology prediction. These misclassifications can be attributed to the similar texture and gray-scale values, and the confusion matrix reflects these similarities in the confusion space (Chawshin et al., 2021).

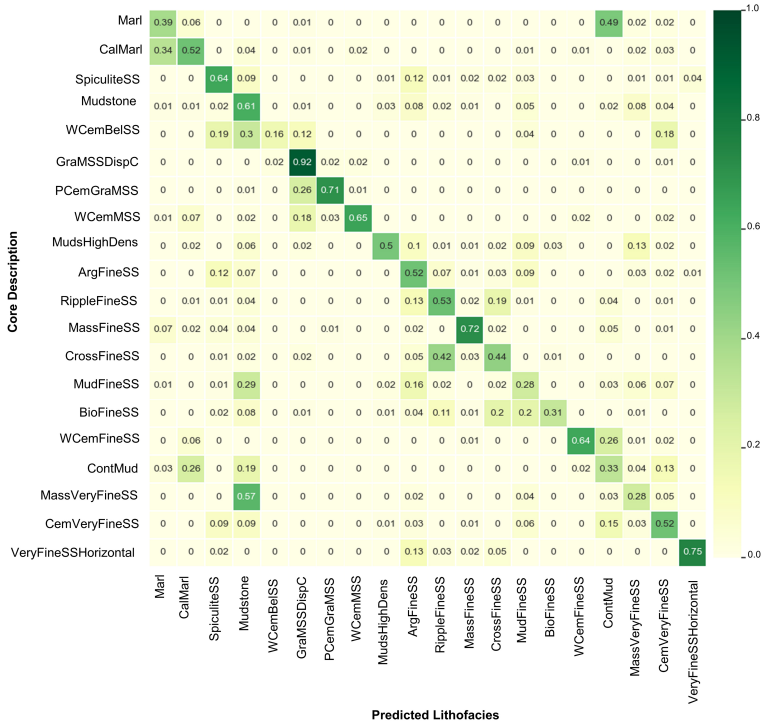
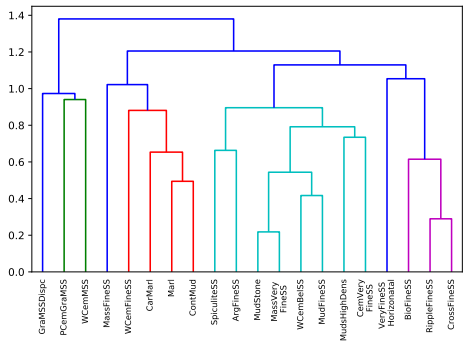


Figure 4.1: Confusion matrix (Chawshin et al., 2021).

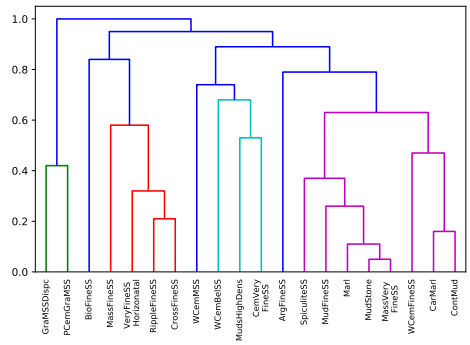
By considering each row in the confusion matrix as a vector representing an object and each column as a measurement, one can see that the confusion matrix has the form of a pattern matrix. The indices of rows in the confusion matrix are now assigned to the indices of the corresponding objects. For a better visualization of the similarities between studied lithofacies, Chawshin used the confusion matrix as the input pattern matrix for an agglomerative hierarchical clustering algorithm and generated a dendrogram which is referred as a numerical dendrogram in our study (Figure 4.2a). Let ND stand for the numerical dendrogram in this thesis. The index of a singleton cluster in the numerical dendrogram is the index of the object constituting the singleton cluster. From the numerical dendrogram, one can know which lithofacies is the most similar to the others in terms of the degree of confusion. According to Chawshin et al. (2021), the misclassified lithofacies classes share similar greyscale values, grain size, and texture properties. For example, Mudstone and MassVeryFineSS are grouped to create the first non-singleton cluster in the numerical dendrogram due to the high degree of confusion. In fact, these lithofacies classes are very finegrained and share very similar gray-scales and textural properties. Therefore, they have a tendency to be misclassified by the classifier (Chawshin et al., 2021). The numerical dendrogram has the corresponding linkage matrix, Z_{ND}, shown in Figure 4.2c. Hereafter, we shall use the linkage matrix Z_{ND} to refer to the original layout of the numerical dendrogram.

The similarity between lithofacies can also be indicated by the similar transport proper-

ties (porosity and permeability). Accordingly, apart from the similarities in the grain sizes and grayscale values, Chawshin utilized the porosity-permeability crossplot (Figure 4.3) from available core analysis measurements and geological domain knowledge to manually create another dendrogram of 20 studied lithofacies (objects). This dendrogram, which we shall call as the geologist dendrogram (GD), is presented in Figure 4.2b. The index of a singleton cluster in the geological dendrogram is the same as in the numerical one. We ought to remark that the geologist dendrogram was not generated by any hierarchical clustering algorithm, and thus neither pattern matrix nor proximity matrix was used. The geologist dendrogram is made based on the observations of the geologist over the similarities in grain size, grayscale, and transport properties of 20 lithofacies. For example, Mudstone, MassVeryFineSS, Marl, and MudFineSS are very fine-grained and fall into the low porosity and permeability group (marked by the red ellipsoid in Figure 4.3), therefore, they are combined to create the very first interior vertices in the geologist dendrogram. The distances between clusters are not calculated by a distance metric or linkage methods, but they are objectively defined by the geologist and converted into quantitative measurements. These distances, or the hierarchical levels at which clusters are formed, are defined in the $[0, 1]$ interval and displayed in the third column of the geologist dendrogram's linkage matrix Z_{GD} (Figure 4.2d). Similar to the numerical dendrogram, the linkage matrix Z_{GD} will be used to refer to the original layout of the geologist dendrogram.



(a) Numerical dendrogram (ND)



(b) Geologist dendrogram (GD)

$$Z_{ND} = \begin{bmatrix} 3 & 17 & 0.21817424 & 2 \\ 10 & 12 & 0.28965497 & 2 \\ 4 & 13 & 0.41665333 & 2 \\ 0 & 16 & 0.49426713 & 2 \\ 20 & 22 & 0.54378304 & 4 \\ 14 & 21 & 0.61478994 & 3 \\ 1 & 23 & 0.65347788 & 3 \\ 2 & 8 & 0.66309879 & 2 \\ 8 & 18 & 0.73457471 & 2 \\ 24 & 28 & 0.79181227 & 6 \\ 15 & 26 & 0.88114509 & 4 \\ 27 & 29 & 0.89580039 & 8 \\ 6 & 7 & 0.94037227 & 2 \\ 5 & 32 & 0.97329338 & 3 \\ 11 & 30 & 1.02188551 & 5 \\ 19 & 25 & 1.05388962 & 4 \\ 31 & 35 & 1.12939143 & 12 \\ 34 & 36 & 1.20515315 & 17 \\ 33 & 37 & 1.38021951 & 20 \end{bmatrix}$$

(c) Linkage matrix of ND

$$Z_{GD} = \begin{bmatrix} 3 & 17 & 0.05 & 2 \\ 0 & 20 & 0.11 & 3 \\ 1 & 16 & 0.16 & 2 \\ 10 & 12 & 0.21 & 2 \\ 13 & 21 & 0.26 & 4 \\ 19 & 23 & 0.32 & 3 \\ 2 & 24 & 0.37 & 5 \\ 5 & 6 & 0.42 & 2 \\ 15 & 22 & 0.47 & 3 \\ 8 & 18 & 0.53 & 2 \\ 11 & 25 & 0.58 & 4 \\ 26 & 28 & 0.63 & 8 \\ 4 & 29 & 0.68 & 3 \\ 7 & 32 & 0.74 & 4 \\ 9 & 31 & 0.79 & 9 \\ 14 & 30 & 0.84 & 5 \\ 33 & 34 & 0.89 & 13 \\ 35 & 36 & 0.95 & 18 \\ 27 & 37 & 1 & 20 \end{bmatrix}$$

(d) Linkage matrix of GD

Figure 4.2: Base dendrograms and linkage matrices.

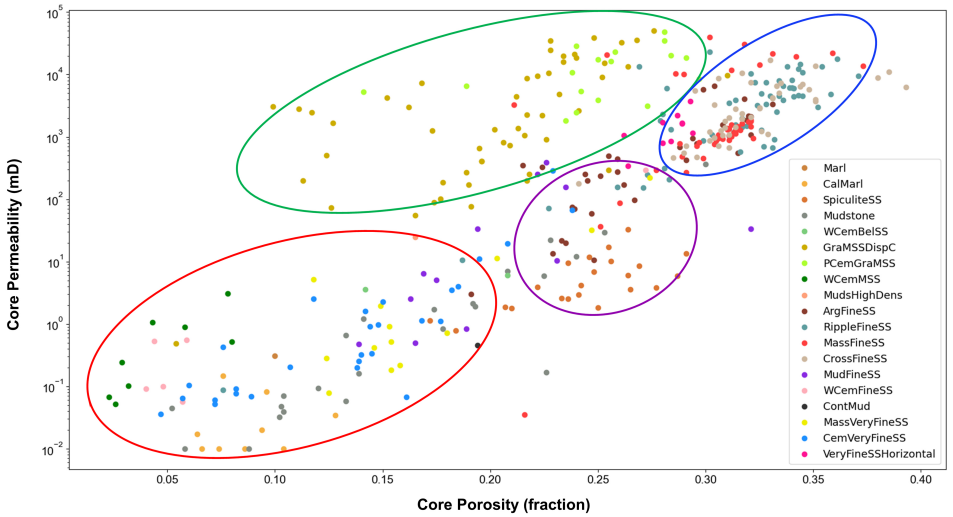


Figure 4.3: Porosity-permeability cross-plot with ellipsoids which mark groups of similar lithofacies classes (Chawshin et al., 2021).

The major difference that can be observed from two base dendrograms in Figure 4.2 is the hierarchy. To put it another way, 20 lithofacies are merged differently to create different clusters in two dendrograms. It can be attributed to the different input data used to create dendrograms. In addition, the scales of hierarchical levels in two dendrograms are also different. There are some clusters that contain the same objects in both dendrograms, their hierarchical levels are, however, not equal.

4.2 Hierarchical Clustering Combination

4.2.1 Input Similarity Matrices

As reported in section 3.3, the MATCH and square algorithms will be used to combine the numerical dendrogram Z_{ND} and the geologist dendrogram Z_{GD} in this thesis. The input data for these algorithms are the similarity based descriptor matrices of these base dendrograms. These descriptors need to be mapped into the $[0, 1]$ interval due to two reasons. Firstly, the mapping might turn these descriptors into the dissimilarity matrices from which we can obtain the similarity matrices. Secondly, without mapping, it is incapable to directly use the description matrices containing the information of hierarchical levels to perform the HCC since the scales of hierarchical levels in the two base dendrograms are different. The attempt to map elements in the descriptors in a range of $[0, 1]$ is considered as a normalization to minimize the differences in the scales of the hierarchical levels in different dendrograms. The similarity based descriptor matrices are computed from the dissimilarity matrices by the simple transformation in Equation 2.50.

It is improbable that any descriptor in section 2.5 can be used for the MATCH and square algorithms. Mapping the entries of a dissimilarity based description matrix into the

$[0, 1]$ interval turns this matrix to a similarity matrix if and only if the description matrix is symmetric and anti-reflexive. Ironically, not all 6 dendrogram description matrices are anti-reflexive. For instance, the cluster membership divergence and sub-tree membership divergence are not anti-reflexive on account of the fact that some diagonal elements in the these matrices are not zero. Furthermore, to maintain the one-to-one relationship between an input similarity matrix and the corresponding base dendrogram, the similarity matrix must be transitive. To this end, the corresponding dissimilarity based descriptor needs to be ultrametric. Among 4 anti-reflexive descriptors, there are only the cophenetic difference and the partition membership divergence that are ultrametric. Accordingly, either the cophenetic difference or the partition membership divergence can be use in the HCC. Between these two descriptors, we will use the cophenetic difference matrices to combine dendrograms in this research. The main reason is that the feature of the dendrogram that the corresponding cophenetic difference captures is the hierarchical levels. Therefore, using the cophenetic difference matrix to reconstruct or represent a base dendrogram helps to preserve the hierarchical levels of the base dendrogram. Such preserved hierarchical levels will contribute to generate the consensus matrix from which the final dendrogram is constructed. Let Co_{ND} and Co_{GD} denote the cophenetic difference matrices of the numerical dendrogram and the geologist dendrogram, respectively. They are calculated from the corresponding linkage matrices and shown in Figure 4.4a and Figure 4.4b. Note that for the sake of illustration, the entries of these cophenetic matrices are rounded to two decimal places. If we are not provided with the linkage matrices together with the base dendrograms, meaning that the hierarchical levels are not determined with exact numbers, then the partition membership divergence will be used to construct the similarity matrix.

$$\begin{bmatrix}
0.00 & 0.47 & 0.87 & 0.87 & 0.87 & 1.00 & 1.00 & 1.00 & 0.87 & 0.87 & 0.87 & 0.74 & 0.87 & 0.87 & 0.87 & 0.64 & 0.36 & 0.87 & 0.87 & 0.87 \\
 & 0.00 & 0.87 & 0.87 & 0.87 & 1.00 & 1.00 & 1.00 & 0.87 & 0.87 & 0.87 & 0.74 & 0.87 & 0.87 & 0.87 & 0.64 & 0.47 & 0.87 & 0.87 & 0.87 \\
 & & 0.00 & 0.65 & 0.65 & 1.00 & 1.00 & 1.00 & 0.65 & 0.48 & 0.82 & 0.87 & 0.82 & 0.65 & 0.82 & 0.87 & 0.87 & 0.65 & 0.65 & 0.82 \\
 & & & 0.00 & 0.39 & 1.00 & 1.00 & 1.00 & 0.57 & 0.65 & 0.82 & 0.87 & 0.82 & 0.39 & 0.82 & 0.87 & 0.87 & 0.16 & 0.57 & 0.82 \\
 & & & & 0.00 & 1.00 & 1.00 & 1.00 & 0.57 & 0.65 & 0.82 & 0.87 & 0.82 & 0.30 & 0.82 & 0.87 & 0.87 & 0.39 & 0.57 & 0.82 \\
 & & & & & 0.00 & 0.71 & 0.71 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 \\
 & & & & & & 0.00 & 0.68 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 \\
 & & & & & & & 0.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 \\
 & & & & & & & & 0.00 & 0.65 & 0.82 & 0.87 & 0.82 & 0.57 & 0.82 & 0.87 & 0.87 & 0.57 & 0.53 & 0.82 \\
 & & & & & & & & & 0.00 & 0.82 & 0.87 & 0.82 & 0.65 & 0.82 & 0.87 & 0.87 & 0.65 & 0.65 & 0.82 \\
 & & & & & & & & & & 0.00 & 0.87 & 0.21 & 0.82 & 0.45 & 0.87 & 0.87 & 0.82 & 0.82 & 0.76 \\
 & & & & & & & & & & & 0.00 & 0.87 & 0.87 & 0.87 & 0.74 & 0.74 & 0.87 & 0.87 & 0.87 \\
 & & & & & & & & & & & & 0.00 & 0.82 & 0.45 & 0.87 & 0.87 & 0.82 & 0.82 & 0.76 \\
 & & & & & & & & & & & & & 0.00 & 0.82 & 0.87 & 0.87 & 0.39 & 0.57 & 0.82 \\
 & & & & & & & & & & & & & & 0.00 & 0.87 & 0.87 & 0.82 & 0.82 & 0.76 \\
 & & & & & & & & & & & & & & & 0.00 & 0.64 & 0.87 & 0.87 & 0.87 \\
 & & & & & & & & & & & & & & & & 0.00 & 0.87 & 0.87 & 0.87 \\
 & & & & & & & & & & & & & & & & & 0.00 & 0.57 & 0.82 \\
 & & & & & & & & & & & & & & & & & & 0.00 & 0.82 \\
 & & & & & & & & & & & & & & & & & & & 0.00
\end{bmatrix}$$

(a) Cophenetic difference of the numerical dendrogram after mapping into $[0, 1]$, Co_{ND}.

$$\begin{bmatrix}
0.00 & 0.63 & 0.37 & 0.11 & 0.89 & 1.00 & 1.00 & 0.89 & 0.89 & 0.79 & 0.95 & 0.95 & 0.95 & 0.26 & 0.95 & 0.63 & 0.63 & 0.11 & 0.89 & 0.95 \\
 & 0.00 & 0.63 & 0.63 & 0.89 & 1.00 & 1.00 & 0.89 & 0.89 & 0.79 & 0.95 & 0.95 & 0.95 & 0.63 & 0.95 & 0.47 & 0.16 & 0.63 & 0.89 & 0.95 \\
 & & 0.00 & 0.37 & 0.89 & 1.00 & 1.00 & 0.89 & 0.89 & 0.79 & 0.95 & 0.95 & 0.95 & 0.37 & 0.95 & 0.63 & 0.63 & 0.37 & 0.89 & 0.95 \\
 & & & 0.00 & 0.89 & 1.00 & 1.00 & 0.89 & 0.89 & 0.79 & 0.95 & 0.95 & 0.95 & 0.26 & 0.95 & 0.63 & 0.63 & 0.05 & 0.89 & 0.95 \\
 & & & & 0.00 & 1.00 & 1.00 & 0.74 & 0.68 & 0.89 & 0.95 & 0.95 & 0.95 & 0.89 & 0.95 & 0.89 & 0.89 & 0.89 & 0.68 & 0.95 \\
 & & & & & 0.00 & 0.42 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 \\
 & & & & & & 0.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 \\
 & & & & & & & 0.00 & 0.74 & 0.89 & 0.95 & 0.95 & 0.95 & 0.89 & 0.95 & 0.89 & 0.89 & 0.89 & 0.74 & 0.95 \\
 & & & & & & & & 0.00 & 0.89 & 0.95 & 0.95 & 0.95 & 0.89 & 0.95 & 0.89 & 0.89 & 0.89 & 0.53 & 0.95 \\
 & & & & & & & & & 0.00 & 0.95 & 0.95 & 0.95 & 0.79 & 0.95 & 0.79 & 0.79 & 0.79 & 0.89 & 0.95 \\
 & & & & & & & & & & 0.00 & 0.58 & 0.21 & 0.95 & 0.84 & 0.95 & 0.95 & 0.95 & 0.95 & 0.32 \\
 & & & & & & & & & & & 0.00 & 0.58 & 0.95 & 0.84 & 0.95 & 0.95 & 0.95 & 0.95 & 0.58 \\
 & & & & & & & & & & & & 0.00 & 0.95 & 0.84 & 0.95 & 0.95 & 0.95 & 0.95 & 0.32 \\
 & & & & & & & & & & & & & 0.00 & 0.95 & 0.63 & 0.63 & 0.26 & 0.89 & 0.95 \\
 & & & & & & & & & & & & & & 0.00 & 0.95 & 0.95 & 0.95 & 0.95 & 0.84 \\
 & & & & & & & & & & & & & & & 0.00 & 0.47 & 0.63 & 0.89 & 0.95 \\
 & & & & & & & & & & & & & & & & 0.00 & 0.63 & 0.89 & 0.95 \\
 & & & & & & & & & & & & & & & & & 0.00 & 0.89 & 0.95 \\
 & & & & & & & & & & & & & & & & & & 0.00 & 0.95 \\
 & & & & & & & & & & & & & & & & & & & 0.00
\end{bmatrix}$$

(b) Cophenetic difference of the geologist dendrogram, Co_{GD}.

Figure 4.4: Cophenetic difference matrices of base dendrograms.

Let S_{ND} and S_{GD} denote the similarity matrices of the two base dendrograms. They are computed from the corresponding dissimilarity matrices (or the cophenetic difference descriptors after being mapped in the interval $[0, 1]$). We have

$$S_{ND} = [1] - Co_{ND} \tag{4.1}$$

and

$$S_{GD} = [1] - Co_{GD} \tag{4.2}$$

where $[1]$ is a 20×20 matrix whose all elements have the value of 1. Figure 4.5 reveals the input similarity matrices, S_{ND} and S_{GD}, for both MATCH algorithm and square algorithm.

$$
\begin{bmatrix}
1.00 & 0.53 & 0.13 & 0.13 & 0.13 & 0.00 & 0.00 & 0.00 & 0.13 & 0.13 & 0.13 & 0.26 & 0.13 & 0.13 & 0.13 & 0.36 & 0.64 & 0.13 & 0.13 & 0.13 \\
 & 1.00 & 0.13 & 0.13 & 0.13 & 0.00 & 0.00 & 0.00 & 0.13 & 0.13 & 0.13 & 0.26 & 0.13 & 0.13 & 0.13 & 0.36 & 0.53 & 0.13 & 0.13 & 0.13 \\
 & & 1.00 & 0.35 & 0.35 & 0.00 & 0.00 & 0.00 & 0.35 & 0.52 & 0.18 & 0.13 & 0.18 & 0.35 & 0.18 & 0.13 & 0.13 & 0.35 & 0.35 & 0.18 \\
 & & & 1.00 & 0.61 & 0.00 & 0.00 & 0.00 & 0.43 & 0.35 & 0.18 & 0.13 & 0.18 & 0.61 & 0.18 & 0.13 & 0.13 & 0.84 & 0.43 & 0.18 \\
 & & & & 1.00 & 0.00 & 0.00 & 0.00 & 0.43 & 0.35 & 0.18 & 0.13 & 0.18 & 0.70 & 0.18 & 0.13 & 0.13 & 0.61 & 0.43 & 0.18 \\
 & & & & & 1.00 & 0.29 & 0.29 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
 & & & & & & 1.00 & 0.32 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
 & & & & & & & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
 & & & & & & & & 1.00 & 0.35 & 0.18 & 0.13 & 0.18 & 0.43 & 0.18 & 0.13 & 0.13 & 0.43 & 0.47 & 0.18 \\
 & & & & & & & & & 1.00 & 0.18 & 0.13 & 0.18 & 0.35 & 0.18 & 0.13 & 0.13 & 0.35 & 0.35 & 0.18 \\
 & & & & & & & & & & 1.00 & 0.13 & 0.79 & 0.18 & 0.55 & 0.13 & 0.13 & 0.18 & 0.18 & 0.24 \\
 & & & & & & & & & & & 1.00 & 0.13 & 0.13 & 0.13 & 0.26 & 0.26 & 0.13 & 0.13 & 0.13 \\
 & & & & & & & & & & & & 1.00 & 0.18 & 0.55 & 0.13 & 0.13 & 0.18 & 0.18 & 0.24 \\
 & & & & & & & & & & & & & 1.00 & 0.18 & 0.13 & 0.13 & 0.61 & 0.43 & 0.18 \\
 & & & & & & & & & & & & & & 1.00 & 0.13 & 0.13 & 0.18 & 0.18 & 0.24 \\
 & & & & & & & & & & & & & & & 1.00 & 0.36 & 0.13 & 0.13 & 0.13 \\
 & & & & & & & & & & & & & & & & 1.00 & 0.13 & 0.13 & 0.13 \\
 & & & & & & & & & & & & & & & & & 1.00 & 0.43 & 0.18 \\
 & & & & & & & & & & & & & & & & & & 1.00 & 0.18 \\
 & & & & & & & & & & & & & & & & & & & 1.00
\end{bmatrix}
$$

(a) Similarity matrix of the numerical dendrogram, S_{ND}.

$$
\begin{bmatrix}
1.00 & 0.37 & 0.63 & 0.89 & 0.11 & 0.00 & 0.00 & 0.11 & 0.11 & 0.21 & 0.05 & 0.05 & 0.05 & 0.74 & 0.05 & 0.37 & 0.37 & 0.89 & 0.11 & 0.05 \\
 & 1.00 & 0.37 & 0.37 & 0.11 & 0.00 & 0.00 & 0.11 & 0.11 & 0.21 & 0.05 & 0.05 & 0.05 & 0.37 & 0.05 & 0.53 & 0.84 & 0.37 & 0.11 & 0.05 \\
 & & 1.00 & 0.63 & 0.11 & 0.00 & 0.00 & 0.11 & 0.11 & 0.21 & 0.05 & 0.05 & 0.05 & 0.63 & 0.05 & 0.37 & 0.37 & 0.63 & 0.11 & 0.05 \\
 & & & 1.00 & 0.11 & 0.00 & 0.00 & 0.11 & 0.11 & 0.21 & 0.05 & 0.05 & 0.05 & 0.74 & 0.05 & 0.37 & 0.37 & 0.95 & 0.11 & 0.05 \\
 & & & & 1.00 & 0.00 & 0.00 & 0.26 & 0.32 & 0.11 & 0.05 & 0.05 & 0.05 & 0.11 & 0.05 & 0.11 & 0.11 & 0.11 & 0.32 & 0.05 \\
 & & & & & 1.00 & 0.58 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
 & & & & & & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
 & & & & & & & 1.00 & 0.26 & 0.11 & 0.05 & 0.05 & 0.05 & 0.11 & 0.05 & 0.11 & 0.11 & 0.11 & 0.26 & 0.05 \\
 & & & & & & & & 1.00 & 0.11 & 0.05 & 0.05 & 0.05 & 0.11 & 0.05 & 0.11 & 0.11 & 0.11 & 0.47 & 0.05 \\
 & & & & & & & & & 1.00 & 0.05 & 0.05 & 0.05 & 0.21 & 0.05 & 0.21 & 0.21 & 0.21 & 0.11 & 0.05 \\
 & & & & & & & & & & 1.00 & 0.42 & 0.79 & 0.05 & 0.16 & 0.05 & 0.05 & 0.05 & 0.05 & 0.68 \\
 & & & & & & & & & & & 1.00 & 0.42 & 0.05 & 0.16 & 0.05 & 0.05 & 0.05 & 0.05 & 0.42 \\
 & & & & & & & & & & & & 1.00 & 0.05 & 0.16 & 0.05 & 0.05 & 0.05 & 0.05 & 0.68 \\
 & & & & & & & & & & & & & 1.00 & 0.05 & 0.37 & 0.37 & 0.74 & 0.11 & 0.05 \\
 & & & & & & & & & & & & & & 1.00 & 0.05 & 0.05 & 0.05 & 0.05 & 0.16 \\
 & & & & & & & & & & & & & & & 1.00 & 0.53 & 0.37 & 0.11 & 0.05 \\
 & & & & & & & & & & & & & & & & 1.00 & 0.37 & 0.11 & 0.05 \\
 & & & & & & & & & & & & & & & & & 1.00 & 0.11 & 0.05 \\
 & & & & & & & & & & & & & & & & & & 1.00 & 0.05 \\
 & & & & & & & & & & & & & & & & & & & 1.00
\end{bmatrix}
$$

(b) Similarity matrix of the geologist dendrogram, S_{GD}.

Figure 4.5: Input similarity matrices.

4.2.2 Consensus Matrix and Final Dendrogram

The MATCH algorithm and the square algorithm aggregate the input similarity matrices into a transitive consensus matrix by employing the transitive closure approach. Despite having different calculations of the transitive closure, both MATCH algorithm and square algorithm return the same consensus matrix S_{CS} (Figure 4.6a). This similarity based consensus matrix is a transitive similarity matrix. Indeed, S_{CS} is transitive because we have

$$S_{CS} \circ S_{CS} = S_{CS} \tag{4.3}$$

The similarity based consensus matrix, S_{CS}, is converted to a dissimilarity based consensus matrix D_{CS} (Figure 4.6b) based on Equation 2.52

$$D_{CS} = [1] - S_{CS} \tag{4.4}$$

Since the similarity based consensus matrix S_{CS} is transitive, the corresponding dissimilarity based consensus matrix D_{CS} is ultrametric. The ultrametric property of this dissimilarity matrix assures the existence of an associated dendrogram. Therefore, the dissimilarity based consensus matrix D_{CS} is used as the distance matrix to construct the combined dendrogram (CD) by using a given hierarchical clustering algorithm (Figure 4.7). We should remark that both single hierarchical clustering algorithm and complete hierarchical clustering algorithm generate the same dendrogram because the input distance matrix is a ultrametric dissimilarity matrix similar to a cophenetic matrix. The fact remains that the dissimilarity matrix D_{CS} is the cophenetic matrix of the combined dendrogram. The linkage matrix of the combined dendrogram, Z_{CD}, is given in Figure 4.8.

$$
\begin{bmatrix}
1.00 & 0.64 & 0.63 & 0.89 & 0.70 & 0.26 & 0.26 & 0.26 & 0.43 & 0.52 & 0.26 & 0.26 & 0.26 & 0.74 & 0.26 & 0.53 & 0.64 & 0.89 & 0.43 & 0.26 \\
 & 1.00 & 0.63 & 0.64 & 0.64 & 0.26 & 0.26 & 0.26 & 0.43 & 0.52 & 0.26 & 0.26 & 0.26 & 0.64 & 0.26 & 0.53 & 0.84 & 0.64 & 0.43 & 0.26 \\
 & & 1.00 & 0.63 & 0.63 & 0.26 & 0.26 & 0.26 & 0.43 & 0.52 & 0.26 & 0.26 & 0.26 & 0.63 & 0.26 & 0.53 & 0.63 & 0.63 & 0.43 & 0.26 \\
 & & & 1.00 & 0.70 & 0.26 & 0.26 & 0.26 & 0.43 & 0.52 & 0.26 & 0.26 & 0.26 & 0.74 & 0.26 & 0.53 & 0.64 & 0.95 & 0.43 & 0.26 \\
 & & & & 1.00 & 0.26 & 0.26 & 0.26 & 0.43 & 0.52 & 0.26 & 0.26 & 0.26 & 0.70 & 0.26 & 0.53 & 0.64 & 0.70 & 0.43 & 0.26 \\
 & & & & & 1.00 & 0.58 & 0.32 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 \\
 & & & & & & 1.00 & 0.32 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 \\
 & & & & & & & 1.00 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 & 0.26 \\
 & & & & & & & & 1.00 & 0.43 & 0.26 & 0.26 & 0.26 & 0.43 & 0.26 & 0.43 & 0.43 & 0.43 & 0.47 & 0.26 \\
 & & & & & & & & & 1.00 & 0.26 & 0.26 & 0.26 & 0.52 & 0.26 & 0.52 & 0.52 & 0.52 & 0.43 & 0.26 \\
 & & & & & & & & & & 1.00 & 0.42 & 0.79 & 0.26 & 0.55 & 0.26 & 0.26 & 0.26 & 0.26 & 0.68 \\
 & & & & & & & & & & & 1.00 & 0.42 & 0.26 & 0.42 & 0.26 & 0.26 & 0.26 & 0.26 & 0.42 \\
 & & & & & & & & & & & & 1.00 & 0.26 & 0.55 & 0.26 & 0.26 & 0.26 & 0.26 & 0.68 \\
 & & & & & & & & & & & & & 1.00 & 0.26 & 0.53 & 0.64 & 0.74 & 0.43 & 0.26 \\
 & & & & & & & & & & & & & & 1.00 & 0.26 & 0.26 & 0.26 & 0.26 & 0.55 \\
 & & & & & & & & & & & & & & & 1.00 & 0.53 & 0.53 & 0.43 & 0.26 \\
 & & & & & & & & & & & & & & & & 1.00 & 0.64 & 0.43 & 0.26 \\
 & & & & & & & & & & & & & & & & & 1.00 & 0.43 & 0.26 \\
 & & & & & & & & & & & & & & & & & & 1.00 & 0.26 \\
 & & & & & & & & & & & & & & & & & & & 1.00
\end{bmatrix}
$$

(a) Similarity based consensus matrix, S_{CS}.

$$
\begin{bmatrix}
0.00 & 0.36 & 0.37 & 0.11 & 0.30 & 0.74 & 0.74 & 0.74 & 0.57 & 0.48 & 0.74 & 0.74 & 0.74 & 0.26 & 0.74 & 0.47 & 0.36 & 0.11 & 0.57 & 0.74 \\
 & 0.00 & 0.37 & 0.36 & 0.36 & 0.74 & 0.74 & 0.74 & 0.57 & 0.48 & 0.74 & 0.74 & 0.74 & 0.36 & 0.74 & 0.47 & 0.16 & 0.36 & 0.57 & 0.74 \\
 & & 0.00 & 0.37 & 0.37 & 0.74 & 0.74 & 0.74 & 0.57 & 0.48 & 0.74 & 0.74 & 0.74 & 0.37 & 0.74 & 0.47 & 0.37 & 0.37 & 0.57 & 0.74 \\
 & & & 0.00 & 0.30 & 0.74 & 0.74 & 0.74 & 0.57 & 0.48 & 0.74 & 0.74 & 0.74 & 0.26 & 0.74 & 0.47 & 0.36 & 0.05 & 0.57 & 0.74 \\
 & & & & 0.00 & 0.74 & 0.74 & 0.74 & 0.57 & 0.48 & 0.74 & 0.74 & 0.74 & 0.30 & 0.74 & 0.47 & 0.36 & 0.30 & 0.57 & 0.74 \\
 & & & & & 0.00 & 0.42 & 0.68 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 \\
 & & & & & & 0.00 & 0.68 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 \\
 & & & & & & & 0.00 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 & 0.74 \\
 & & & & & & & & 0.00 & 0.57 & 0.74 & 0.74 & 0.74 & 0.57 & 0.74 & 0.57 & 0.57 & 0.57 & 0.53 & 0.74 \\
 & & & & & & & & & 0.00 & 0.74 & 0.74 & 0.74 & 0.48 & 0.74 & 0.48 & 0.48 & 0.48 & 0.57 & 0.74 \\
 & & & & & & & & & & 0.00 & 0.58 & 0.21 & 0.74 & 0.45 & 0.74 & 0.74 & 0.74 & 0.74 & 0.32 \\
 & & & & & & & & & & & 0.00 & 0.58 & 0.74 & 0.58 & 0.74 & 0.74 & 0.74 & 0.74 & 0.58 \\
 & & & & & & & & & & & & 0.00 & 0.74 & 0.45 & 0.74 & 0.74 & 0.74 & 0.74 & 0.32 \\
 & & & & & & & & & & & & & 0.00 & 0.74 & 0.47 & 0.36 & 0.26 & 0.57 & 0.74 \\
 & & & & & & & & & & & & & & 0.00 & 0.74 & 0.74 & 0.74 & 0.74 & 0.45 \\
 & & & & & & & & & & & & & & & 0.00 & 0.47 & 0.47 & 0.57 & 0.74 \\
 & & & & & & & & & & & & & & & & 0.00 & 0.36 & 0.57 & 0.74 \\
 & & & & & & & & & & & & & & & & & 0.00 & 0.57 & 0.74 \\
 & & & & & & & & & & & & & & & & & & 0.00 & 0.74 \\
 & & & & & & & & & & & & & & & & & & & 0.00
\end{bmatrix}
$$

(b) Dissimilarity based consensus matrix, D_{CS}.

Figure 4.6: Consensus matrix.

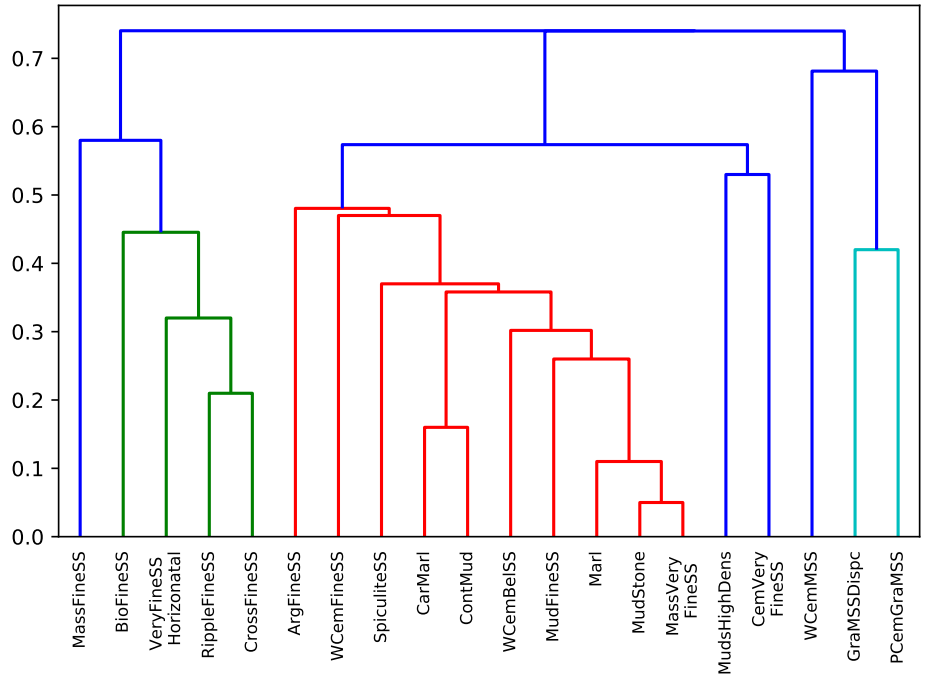


Figure 4.7: The combined dendrogram.

$$Z_{CD} = \begin{bmatrix} 3 & 17 & 0.05 & 2 \\ 0 & 20 & 0.11 & 3 \\ 1 & 16 & 0.16 & 2 \\ 10 & 12 & 0.21 & 2 \\ 13 & 21 & 0.26 & 4 \\ 4 & 24 & 0.30 & 5 \\ 19 & 23 & 0.32 & 3 \\ 22 & 25 & 0.36 & 7 \\ 2 & 27 & 0.37 & 8 \\ 5 & 6 & 0.42 & 2 \\ 14 & 26 & 0.45 & 4 \\ 15 & 28 & 0.47 & 9 \\ 9 & 31 & 0.48 & 10 \\ 8 & 18 & 0.53 & 2 \\ 32 & 33 & 0.57 & 12 \\ 11 & 30 & 0.58 & 5 \\ 7 & 29 & 0.68 & 3 \\ 34 & 36 & 0.74 & 15 \\ 35 & 37 & 0.7404 & 20 \end{bmatrix}$$

Figure 4.8: Linkage matrix of the combined dendrogram.

4.3 Tanglegrams

From the two base dendrograms and the combined dendrogram, we can create 3 different tanglegrams to draw an analogy between dendrograms. The first tanglegram encompassing the numerical dendrogram Z_{ND} and the geologist dendrogram Z_{GD} is denoted by $L_{NG} = [Z_{ND}, Z_{GD}]$. The second tanglegram constituted by the numerical dendrogram Z_{ND} and the combined dendrogram Z_{CD} is denoted by $L_{NC} = [Z_{ND}, Z_{CD}]$. The last tanglegram, $L_{GC} = [Z_{GD}, Z_{CD}]$, contains the geologist dendrogram Z_{GD} and the combined dendrogram Z_{CD}.

4.3.1 Original Tanglegram Layouts

To differentiate between tanglegram layouts before and after we employ the untangle methods, we define the original tanglegram layouts as tanglegram layouts formed by original dendrogram layouts. In other words, two dendrograms whose layouts are created by default and not altered yet constitute an original tanglegram drawing. The tanglegram L_{NG} has the original layout displayed in Figure 4.9, and the entanglement of this tanglegram layout is 0.729. The original tanglegram layouts of L_{NC} and L_{GC} can be seen in Figures 4.10 and 4.11, respectively.

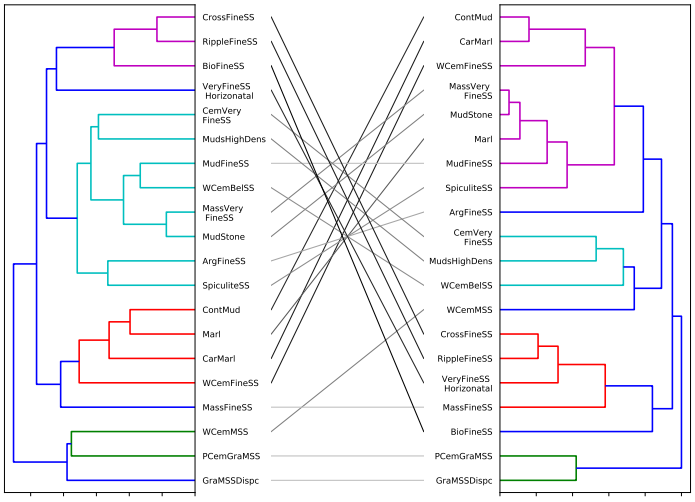


Figure 4.9: The original tanglegram layout of L_{NG} with entanglement of 0.729.

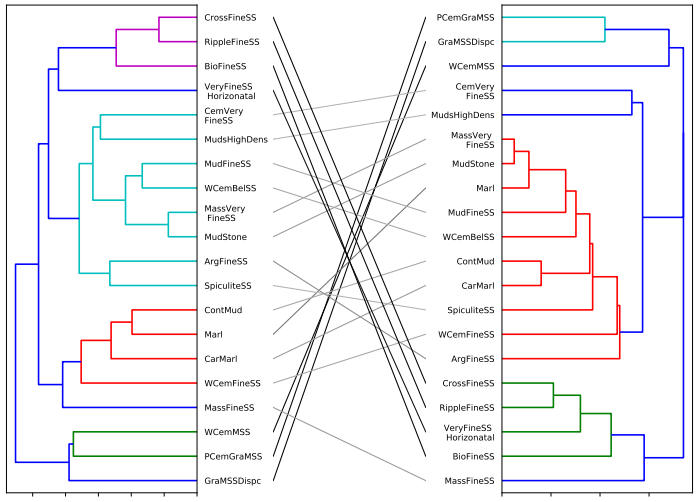


Figure 4.10: The original tanglegram layout of L_{NC} with entanglement of 0.841.

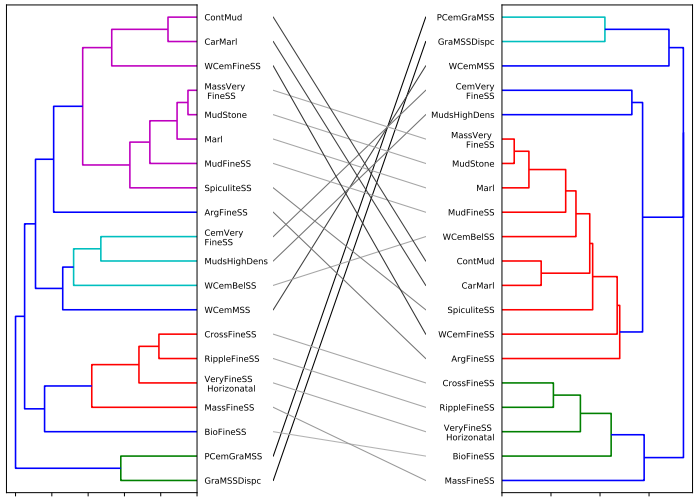


Figure 4.11: The original tanglegram layout of L_{GC} with entanglement of 0.684.

4.3.2 Optimized Tanglegram Layouts

As expected, the original tanglegram layouts are considerably tangly and hard to be interpreted. Therefore, two untangle methods will be used to untangle these layouts so that we can easily see the similarities or dissimilarities between dendrograms.

”Step2side” Method

The first untangle method used to diminish the entanglement values of the three tanglegrams is the ”step2side” method. The optimized tanglegram layout of L_{NG} by using the ”step2side” method is presented in Figure 4.12. Compared to the original tanglegram embedding of L_{NG}, the optimized layout of L_{NG} now has an entanglement decreased from 0.73 to 0.24 by the ”step2side” method. It is apparent from Figure 4.13 and Figure 4.14 that the ”step2side” method returns the optimized tanglegram layouts of L_{NC} and L_{GC} with entanglement values of 0.24 and 0.055, respectively.

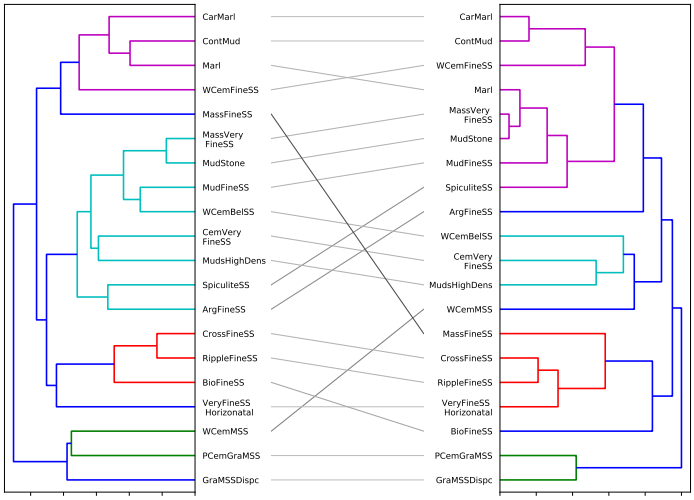


Figure 4.12: Optimized tanglegram layout of L_{NG} with entanglement of 0.24 after applying the ”step2side” method.

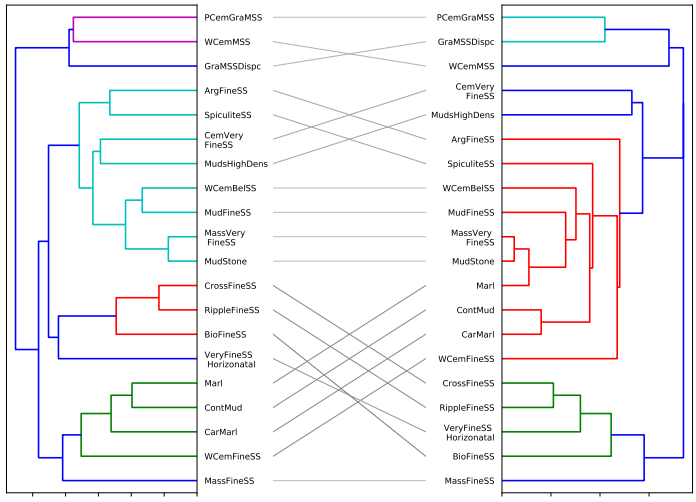


Figure 4.13: Optimized tanglegram layout of L_{NC} with entanglement of 0.236 after applying the "step2side" method.

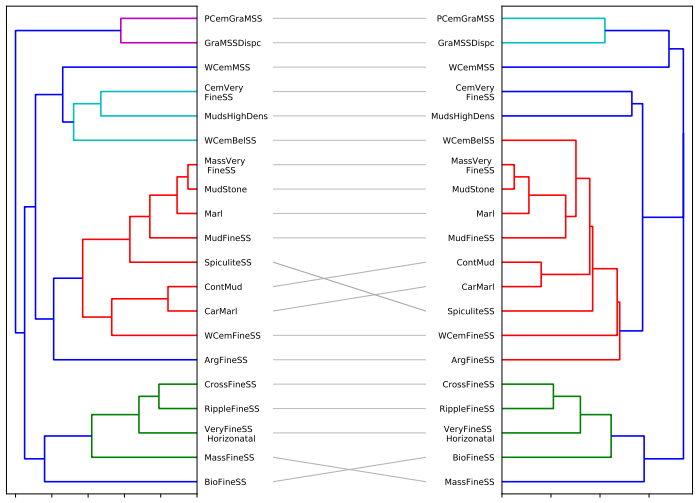


Figure 4.14: Optimized tanglegram layout of L_{GC} with entanglement of 0.055 after applying the "step2side" method.

”S&U” Method

With the aim to get smaller entanglement values than the ”step2side” method, we use the ”S&U” method to optimize the original tanglegram layouts in section 4.3.1. The results from the ”S&U” method are shown in Figures 4.15-4.17. The new entanglement of the tanglegram L_{NG} is 0.22, while that of the tanglegram L_{NC} is 0.18. The ”S&U” method reduces the entanglement of the tanglegram L_{GC} from 0.68 to 0.027.

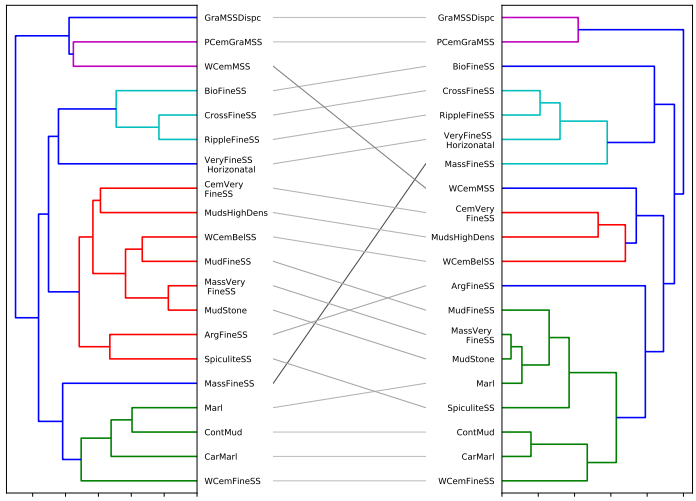


Figure 4.15: Optimized tanglegram layout of L_{NG} with entanglement of 0.22 after applying the ”S&U” method.

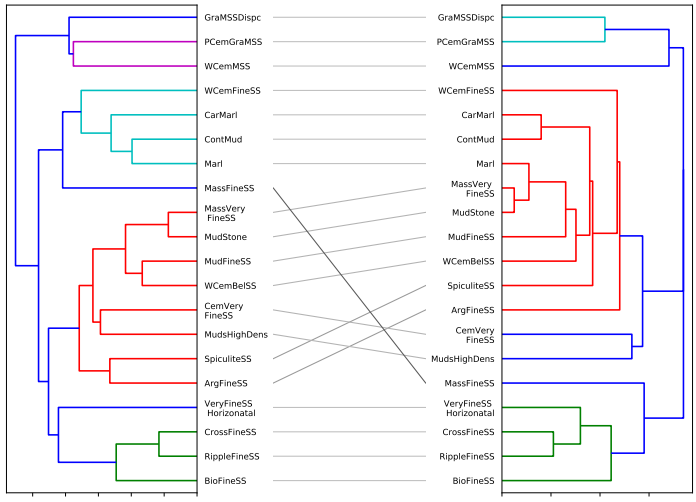


Figure 4.16: Optimized tanglegram layout of L_{NC} with entanglement of 0.182 after applying the "S&U" method.

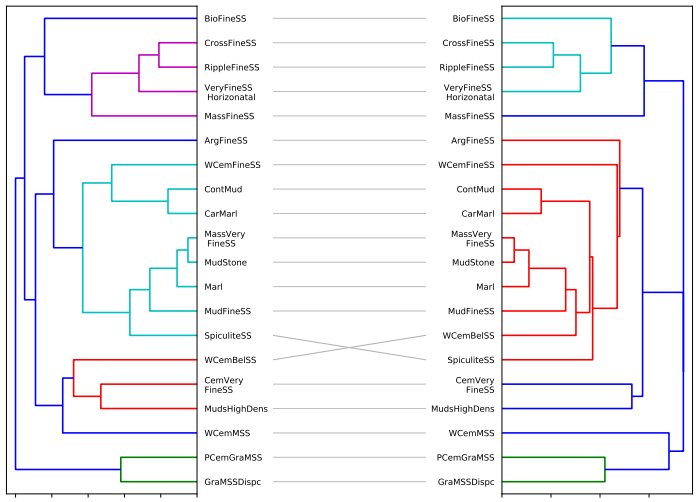


Figure 4.17: Optimized tanglegram layout of L_{GC} with entanglement of 0.027 after applying the "S&U" method.

Chapter 5

Discussion

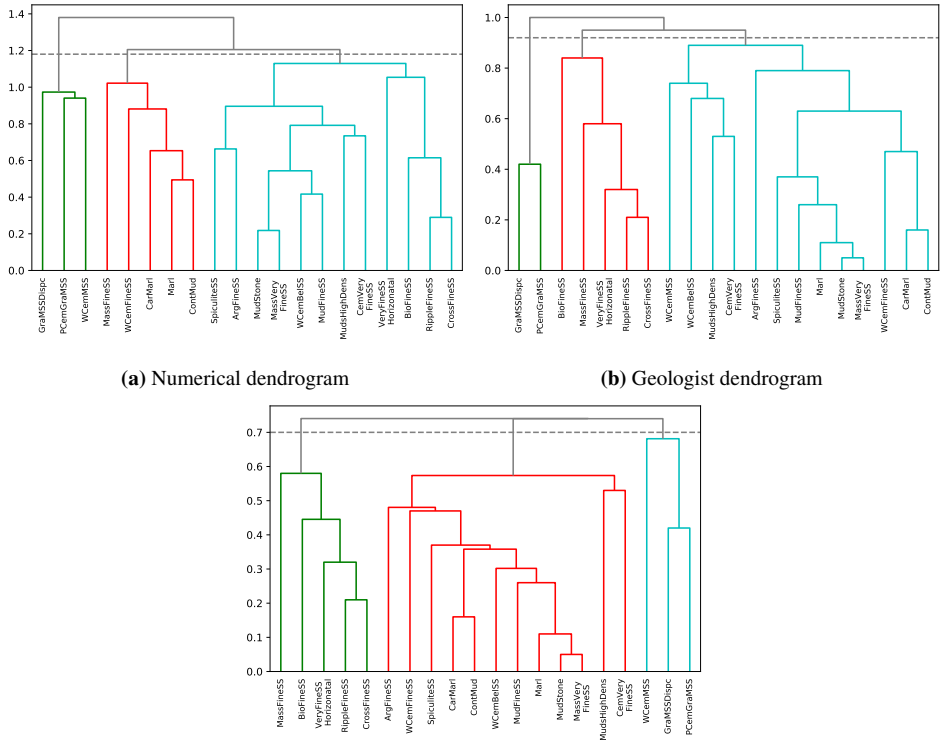
The purpose of this chapter is to discuss the applications and results in Chapter 4. One might want to know how and why the combined dendrogram is similar or dissimilar to the base dendrograms. The combined dendrogram is constructed by the MATCH and square algorithms; hence we open up a discussion on the role that these algorithms play in combining and preserving the hierarchy of the numerical and geologist dendrograms. The computational efficiency of these algorithms will be also discussed. The final discussion will centre on the effectiveness of the "step2side" method and the $S\&U$ method in terms of solving the TL problem.

5.1 Combined Dendrogram versus Base Dendrograms

First and foremost, we need to mention that the numerical and geologist dendrograms are different. As illustrated in the optimized tanglegram layouts of L_{NG} (Figure 4.15), most of the singleton clusters in two base dendrograms are merged with different clusters, causing the high entanglement (0.22 after being optimized by the $S\&U$ method) of the tanglegram containing the base dendrograms. For instance, MudFineSS and WCemBelSS are joined to create the 22nd cluster in the numerical dendrogram, while these objects are joined with other clusters before being in the same cluster which is the 26th cluster. As shortly explained in section 4.1, these differences can be accounted for in part by the different input data used to create the base dendrograms. There are several clusters that contain the same singleton clusters in these dendrograms, but they are created at distinct hierarchical levels and different partitions. In other words, the clusters that are generated at the same interior vertex in two base dendrograms might contain distinct singleton clusters. This phenomenon is thought-provoking when it comes to rotating two dendrograms at the same interior vertex to find a better tanglegram layout, which will be discussed in the next sections.

It is the differences between the two base dendrograms that leads to the need of combining them together. The combined dendrogram can be used to replace the base hierarchical clusterings because it inherits features and structures from the base dendrograms. Some

evidence of the similarity between the combined dendrogram and the other dendrograms can be found. For example, both the numerical dendrogram (Figure 4.2a) and the geologist dendrogram (Figure 4.2b) have the cluster constituting by MudStone and MasVeryFineSS or the cluster consisting of CrossFineSS and RippleFineSS, and these clusters appear in the combined dendrogram (Figure 4.7) with the same constitution. Interestingly, Figure 4.16 pinpoints that in the combined dendrogram the MassFineSS is more similar to the group of VeryFineSSHorizontal, CrossFineSS, RippleFineSS, and BioFineSS than the group of Marl, CarlMarl, ContMud, and WCemFineSS since the former relation is inherited from the geologist dendrogram. One more example is associated with the object WCemMSS. If we draw a horizontal line in each dendrogram at a hierarchical level such that the horizontal line just intersects 3 vertical edges (Figure 5.1), the horizontal line will define 3 clusters in each dendrogram, and one of these 3 clusters (the green cluster) in the numerical dendrogram consists of WCemMSS, GraMSSDispc, and PCemGraMSS, which is the same in one of 3 clusters in the combined dendrogram (the blue cluster). Whereas, the geologist dendrogram in Figure 5.1b illustrates that the object WCemMSS does not belong to the same cluster with GraMSSDispc and PCemGraMSS until the conjoint clustering is formed.



(a) Numerical dendrogram

(b) Geologist dendrogram

(c) Combined dendrogram

Figure 5.1: Dendrograms with horizontal lines.

When we pair the combined dendrogram with each of the base dendrograms to create tanglegrams, the tanglegram of the geologist and combined dendrograms has smaller entanglement values than the tanglegram of the numerical and combined dendrograms. More specifically, after being optimized by the $S\&U$ method, the tanglegram L_{GC} has an entanglement of 0.027 while the tanglegram L_{NC} has an entanglement of 0.182. Although the entanglement is not a direct indicator of the similarity between dendrograms, we can say that the combined dendrogram is more similar to the geologist dendrogram than the numerical one. Take the sub-tree that contains MassVeryFineSS, MudStone, Marl, and MudFineSS in the combined dendrogram as an example. The geologist dendrogram also has this sub-tree with the same hierarchy. The comparison of dendrograms can be proceeded by calculating the correlation coefficients between the cophenetic matrices (or cophenetic difference matrices) of the 3 dendrograms. A correlation coefficient close to 1 indicates that the corresponding dendrograms are very similar (Podani and Dickinson, 1984). The correlation coefficients between cophenetic matrices of dendrograms are actually called the cophenetic correlations (Sokal and Rohlf, 1962). A matrix of cophenetic correlations among the cophenetic matrices resulting from the 3 dendrograms is given in Table 5.1, where Co_{CD} denotes the cophenetic matrix of the combined dendrogram. Table 5.1 demonstrates that the highest cophenetic correlation (0.83) is from the geologist dendrogram and the combined dendrogram, whereas the cophenetic correlation between the numerical and combined dendrograms is 0.61. These results strengthen that the two base dendrograms are relatively different from each other, and the combined dendrogram is more similar to the geologist dendrogram than the numerical one.

Table 5.1: Cophenetic correlations between dendrograms.

	Co_{ND}	Co_{GD}	Co_{CD}
Co_{ND}	1	0.47	0.61
Co_{GD}	0.47	1	0.83
Co_{CD}	0.61	0.83	1

5.2 MATCH versus Square Algorithms

The MATCH algorithm and the square algorithm are used in this thesis to combine two base dendrograms by aggregating their similarity based description matrices into a transitive consensus matrix. Both algorithms utilize the transitive closure approach to compute the transitive consensus matrix, but the calculations of the transitive closure are different in two algorithms.

Generally speaking, the union and max-min composition of any two relations with dimensions of $n \times n$ have computational complexities of $O(n^2)$ and $O(n^3)$, respectively. Therefore, the main calculation in the MATCH algorithm, $\overline{S} \cup (\overline{S} \circ S_i)$, has the computational complexity of $O(n^2) + O(n^3) = O(n^3)$ (Mirzaei and Rahmati, 2010). Mirzaei and Rahmati (2010) also underline that the MATCH algorithm converges in at most $n \times m - 1$ times of performing this main calculation, where m is the number of base dendrograms. Thus, the computational complexity of MATCH algorithm is $O(mn^4)$ in the worst case.

The square algorithm firstly utilizes the max-min composition to aggregate m input

matrices into a matrix $\overline{S}$, which takes $m-1$ compositions. Then it calculates the max-min composition of $\overline{S}$ with itself, and keeps calculating the max-min composition of the resulted matrix with itself until no new matrix is formed. To put it another way, the square algorithm only raises relations to the power of even numbers and ignores relations to the power of odd numbers. As the result, in the worst case, it carries out $\log n$ matrix compositions (Garmendia et al., 2009). Therefore, the computational complexity of the square algorithm is $O(mn^3 \log n)$ in the worst case.

The square algorithm has the capacity to outperform the MATCH algorithm. As anticipated, when we have two base dendrograms, the square algorithm with a time complexity of $O(2n^3 \log n)$ has lower computational complexity than the MATCH algorithm with a time complexity of $O(2n^4)$. We measure the execution time needed to execute each algorithm 10 times in Python, and the results are reported in Figure 5.2. It can be seen in Figure 5.2 that the execution time of the MATCH algorithm (the red line) is always higher than that of the square algorithm (the green line). More specifically, the average execution time of the square algorithm is about $0.195s$, while the average time to execute the MATCH algorithm is approximately 4 times higher than that. Taken together, these results suggest that the square algorithm is more computationally efficient than the MATCH algorithm.

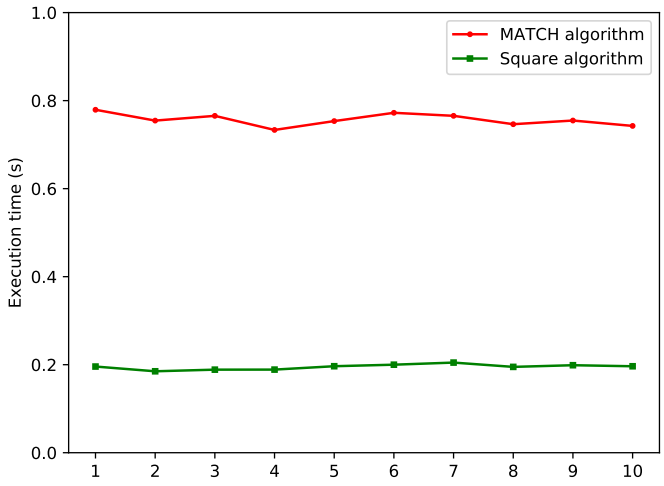


Figure 5.2: Execution time of MATCH and square algorithms.

5.3 ”Step2side” Method versus $S\&U$ Method

As expected, the tanglegram drawings created by the original dendrogram layouts are significantly tangled. In general, the entanglement of 3 original tanglegram layouts fluctuates between 0.7 and 0.8. The highest entanglement value, 0.84, belongs to the original tanglegram layout of L_{NC}. One can see that this entanglement value is close to 1 which is the entanglement of the ”worst” tanglegram layout whose leaf node vector of one dendrogram is a completely reversed vector of the other. In fact, some of the inter-tree edges in the original tanglegram of L_{NC} connect objects at the top-left corner to objects at the bottom right corner of the gap between two dendrograms, or vice versa, which generally happens in the ”worst” tanglegram layout. The high entanglement values of the original tanglegram layouts make it really hard to compare the similarity or dissimilarity between dendrograms.

The ”step2side” untangle method and the proposed untangle method prove clearly beneficial to solving the tanglegram layout problem. Both methods help to remarkably reduce the entanglement of the original tanglegram embeddings. As detailed in Figure 5.3, two tanglegrams, L_{NG} and L_{NC}, have the optimized entanglements fluctuated between 0.18 and 0.24. These numbers are nearly 3 times smaller than the original entanglements. Especially, the tanglegram of the geologist and combined dendrograms has the entanglements after optimization smaller than 0.05, and the number of crossings is just 1 or 3.

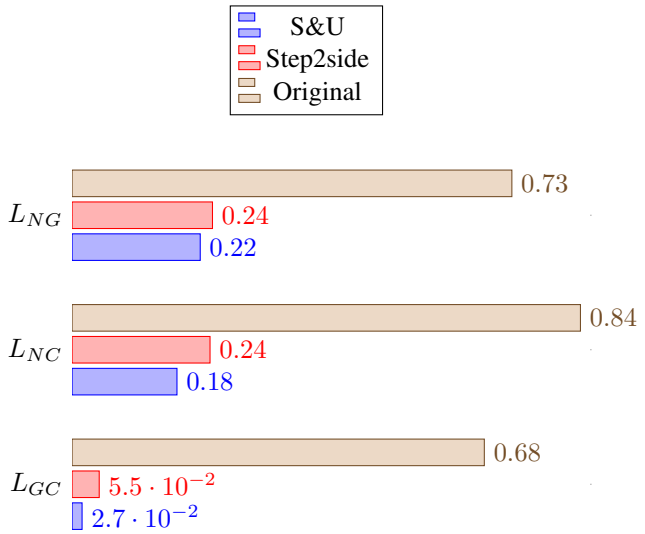
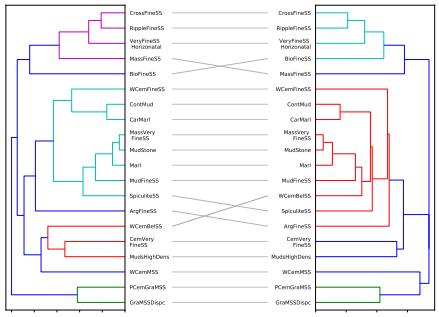


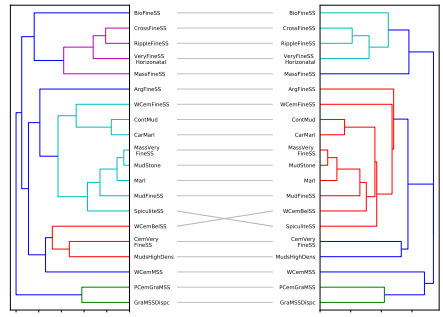
Figure 5.3: Entanglement values of three tanglegrams.

Notwithstanding the fact that the "step2side" was successful in reducing the number of crossings (or the entanglement value), there are still some crossings that could be untangled. For instance, in the tanglegram layout of L_{GC} after applying the "step2side" method (Figure 4.13), if we rotate both the numerical dendrogram and the combined dendrogram at the interior vertex whose one of two children clusters is the singleton cluster GraMSS-Dispc, we can untangle the crossing at the top of this tanglegram. One more example is related to the optimized tanglegram layout of L_{GC} (Figure 4.14). The crossing at the bottom of this tanglegram layout can be eliminated if one rotates both trees at the interior vertex generated by merging the singleton cluster BioFineSS with another cluster. Yet, the "step2side" fails to untangle such crossings due to the fact that the "step2side" method does not rotate two dendrograms simultaneously. Instead, it keeps the left dendrogram unchanged and rotates the right one and vice versa.

Even if one rotates both trees at the same interior vertex at the same time, which is the task of the coarse optimization in the $S\&U$ method, it might not help to eliminate some particular crossings. A great deal of attention must be paid when two interior vertices of the same index in both dendrograms may not contain the same singleton clusters, which is already highlighted in the previous section. Therefore, in the $S\&U$ method, beside the coarse optimization, we came up with the fine optimization to assist the coarse optimization in solving this problem. In the fine optimization, we find the interior vertex that consists of a given object as a singleton cluster in both dendrograms of a tanglegram and rotate the dendrograms at these interior vertices. Occasionally, the indices of these interior vertices are different. To highlight this, we shall optimize the tanglegram of the geologist and combined dendrograms, L_{GC}, by using the the $S\&U$ algorithm but without the fine optimization, and show the result in Figure 5.4a. The most striking observation to emerge from this attempt is that there exists a crossing caused by BioFineSS and MassFineSS in the tanglegram layout optimized by the $S\&U$ algorithm lacked the fine optimization, and this crossing can be eradicated if one rotates both dendrograms at the interior vertex whose one child cluster is the singleton cluster BioFineSS (or MassFineSS) (Figure 5.4b). The coarse optimization can not perform this rotation as the indices of the above interior vertex are different in both trees. Similarly, the crossing caused by ArgFineSS and WCemBelSS in the tanglegram presented in Figure 5.4a can be eliminated with the fine optimization. To do this entails rotating both dendrograms at the interior vertex formed by merging the singleton cluster ArgFineSS with another cluster. This finding reinforces the usefulness of the fine optimization in the $S\&U$ algorithm.



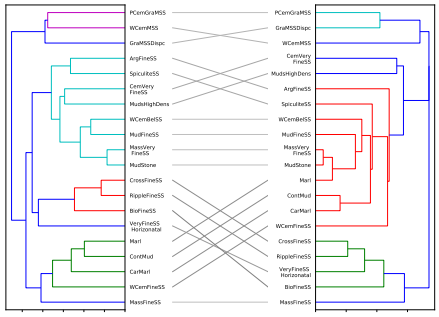
(a) The optimized tanglegram layout of L_{GC} without the fine optimization, entanglement of 0.055



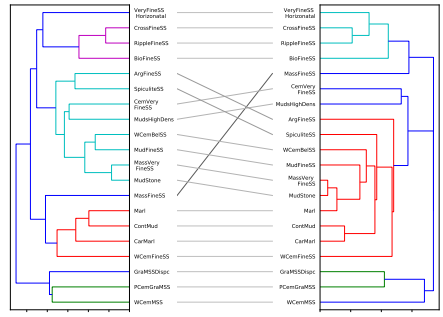
(b) The optimized tanglegram layout of L_{GC} after including the fine optimization, entanglement of 0.027

Figure 5.4: The performance of the $S\&U$ method with and without the fine optimization on the tanglegram L_{GC}.

Based on these observations, we are of the opinion that the results from the ”step2side” method can be improved by the $S\&U$ method. We try to use a tanglegram layout of L_{NC} obtained by the ”step2side” method (Figure 5.5a) as the input for the $S\&U$ algorithm. The result of this attempt can be found in Figure 5.5b, which verifies that the tanglegram after being further optimized by $S\&U$ algorithm is a clear improvement on the result of the ”step2side” method. Some crossings, for example, the crossing related to GraMSSDispc, ContMud, CarMarl, and so on, in the input tanglegram have been removed by the $S\&U$ algorithm.



(a) Input tanglegram layout: the optimized tanglegram layout of L_{NC} with the ”step2side2 method, entanglement of 0.236

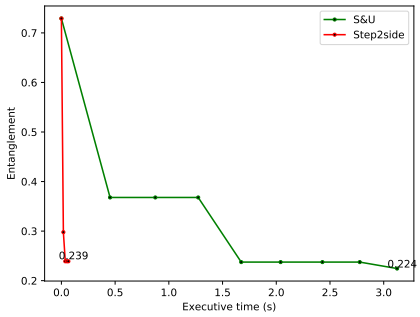


(b) Output tanglegram layout, entanglement of 0.182

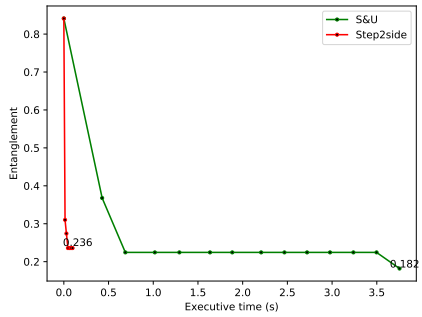
Figure 5.5: Input and output of the $S\&U$ algorithm to illustrate that the result from the ”step2side” method can be improved.

Figure 5.3 highlights that all optimized tanglegram layouts from the $S\&U$ method have smaller entanglements than optimized layouts from the "step2side" method. However, the $S\&U$ method takes more time to execute than the "step2side" method. It is probable that the reason for this is that the "step2side" method only optimizes the original tanglegram layout, while the $S\&U$ method first generates a set of tanglegram layouts from the original tanglegram drawing by rotating the dendrograms at one or several interior vertices from the root, and then optimizes every single tanglegram layout in the set. Broadly speaking, while optimizing, we tend to rotate the dendrograms from the first interior vertex to the root. This could conceivably lead to a problem that rotating both dendrograms at the same interior vertex near the root might not contribute to generating a better drawing. In other words, when the tanglegram is already optimized by going through several first interior vertices which mainly constituted by singleton clusters, rotating the same interior vertex of the last vertices which normally contain a large amount of objects is more likely to make the tanglegram more tangled. It is getting even worse if the same interior vertex does not compose of the same singleton clusters. We encountered this problem in the tanglegram of the 16 Iris flower samples (Figure 2.11). In the tanglegram shown in Figure 2.11, it is pointless to rotate both trees at the root because it aggravates the current tanglegram layout by causing more crossings. Nonetheless, if we rotate the right dendrogram at the root and the left dendrogram at the second last interior vertex, the new formed tanglegram layout will have a smaller entanglement. This stresses just how important the shuffle step in the $S\&U$ algorithm is. We should also mention that the number of layouts in the set of tanglegram layouts $\mathcal{L}$ increases exponentially with the number of rotated interior vertices, which can be modelled by a function of $y = 4^x$, where y is the number of tanglegram layouts in the set, and x is the number of interior vertices to be rotated. If we want to shuffle the original tanglegram layout from the root to the third last interior vertex, the number of different tanglegram layouts in the set $\mathcal{L}$ will be 64. This is not a huge number, but optimizing 64 tanglegram layouts may require 64 times as much than optimizing only one layout. Furthermore, with each tanglegram layout in the set, the $S\&U$ method performs the coarse and fine optimizations which go through all other interior vertices and all objects, respectively. Whereas, the "step2side" method just goes through all the interior vertices. Correspondingly, the $S\&U$ method is more complicated than the "step2side" method.

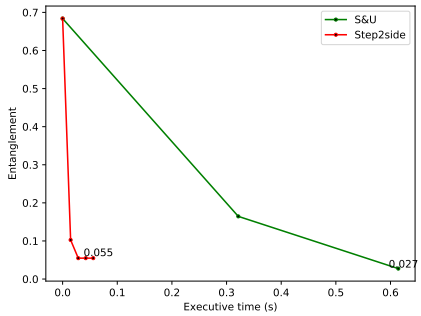
After employing the two untangle methods, we already know the smallest entanglements of the 3 tanglegrams we can get, which is revealed by Figure 5.3. Therefore, one can consider these smallest entanglements as the targets for solving the TL problem. The $S\&U$ and "step2side" algorithms are implemented again in Python, and they terminate when the targeted entanglements are found or no more improvement can be made. We measure the execution times that these algorithms need to execute in Python and compare them in Figure 5.6. This figure highlights that the "step2side" algorithm has low execution time (less than $0.1s$), but it fails to reach the targets. Whereas the $S\&U$ algorithm finds the targeted entanglements although its execution time is large. The $S\&U$ algorithm terminates after around $3.5s$ when optimizing the tanglegrams L_{NG} and L_{NC}. For the tanglegram L_{GC}, it takes about $0.6s$ to get the entanglement value of 0.027.



(a) Optimization of tanglegram L_{NG}



(b) Optimization of tanglegram L_{NC}



(c) Optimization of tanglegram L_{GC}

Figure 5.6: Execution time comparison between $S\&U$ algorithm and ”step2side” algorithm.

Chapter 6

Conclusion

In this master thesis, we performed hierarchical clustering combination and addressed the tanglegram layout problem. From the results and discussions, we arrive at the following conclusions.

First of all, this thesis has given an account of how one can work with dendrograms stemming from different perspectives. To manipulate a dendrogram as an object, it is required to work with its description matrices (descriptors). A dendrogram description matrix is a summarization of the relative position of each pair of objects in a dendrogram, and each descriptor relies on different criteria to define the relative position between objects. Among six types of descriptors, we chose the cophenetic different descriptor (cophenetic matrix) to be the summary representation to the base dendrograms on account of the fact that it is ultrametric, and its entries are the hierarchical levels of the corresponding dendrogram. The ultrametric property of a description matrix plays a very important role in combining and reconstructing dendrograms, as it guarantees the one-to-one correspondence between the description matrix and its dendrogram. After being mapped in the $[0, 1]$ interval, a ultrametric anti-reflexive descriptor becomes a ultrametric dissimilarity matrix, which can be transformed into a transitive similarity matrix. A transitive similarity matrix is a membership matrix of a fuzzy equivalence relation. Therefore, this thesis outlines the one-to-one correspondence between a fuzzy equivalence relation (or a transitive similarity relation) and a dendrogram.

The main focus of our thesis was then on checking first, and then improving the performance of the MATCH algorithm, known to be the state-of-the-art in this specific case. Through the use of the MATCH and square algorithms, we have succeeded in aggregating the similarity-based description matrices of input hierarchical clusterings into a transitive consensus matrix from which the combined hierarchical clustering is formed. The calculation of the transitive closure of a relation is a fundamental basement at which these algorithms are build up. To a certain extend, the final dendrogram inherits structural features from the two base dendrograms. Then we believe that the combined dendrogram can be considered as a proper representative of the two original dendrograms, and it will probably be a useful aid for decision makers in further classifying or narrowing down the number

of lithofacies classes. In our view, the tanglegrams and their entanglement values after the optimization are helpful in comparing the closeness of dendrograms. Our study provides additional support for aggregating dendrogram description matrices besides the MATCH algorithm. The square algorithm has a computational complexity of $O(mn^3 \log n)$ while the MATCH algorithm's computational complexity is $O(mn^4)$, where m is the number of base dendrograms. These results point towards the idea that the square algorithm advances the MATCH algorithm by being more computationally efficient. Nevertheless, the square algorithm is applicable exclusively to reflexive relations.

Our contribution has then been to introduce (and characterize) an alternative to address the tanglegram layout problem. Our work has led to the conclusion that the characteristics of tanglegrams regarding the role of the last interior vertices in solving the TL problem and the distinct children clusters at the same interior vertex between two dendrograms have not been dealt with in depth in the "step2side" method. Thus, we understood that there was the need for improving the "step2side" method. With this in mind, we tried to tackle the issues described above by designing, coding and testing the what we call "$S\&U$ method", and in this way discovered an innovative solution to the TL problem. The small entanglements of the tanglegram layouts generated by employing the here developed method confirm the superiority of our approach over the "step2side" method in terms of finding a better layout for a given tanglegram. From a visual perspective, this is due to the fact that we simultaneously rotate both dendrograms of a tanglegram at each interior vertex and take into consideration the different clusters merged at the same interior vertex in both trees. However, the here proposed $S\&U$ algorithm is computationally demanding, especially with big tree sizes. Despite this, our findings do nevertheless suggest that this method proves to be helpful in untangling crossings that the "step2side" fails to eliminate. In other words, this study has gone some way towards enhancing our understanding of the reason why the "step2side" method is supposed to provide the local minimum of the entanglement optimization. In summary, we are confident that though being time consuming, our $S\&U$ method has the ability to produce better results for the geologists than the "step2side" method does.

Chapter 7

Future Work

There are some tasks and experiments that have been left for the future due to the lack of time. Future work concerns deeper analysis of the $S\&U$ algorithm and new proposals to try different methods to find a transitive relation of a given relation.

Firstly, there is an idea that we would have liked to try during studying the fuzzy relations in Chapter 2. Zadeh (1971) proved that any fuzzy relation R can be expressed by its α-cuts which are basically crisp relations R_α containing all pairs of elements in R having grades of membership greater than or equal to α. The set of α-cuts of a relation forms a nested sequence of crisp relations, which can be illustrated by the partition tree (Zadeh, 1971). In other words, a partition tree is the diagrammatic representation of a nested sequence of partitions. Zadeh (1971) noted that a partition tree is closely related to the concept of the hierarchical clustering. Additionally, a dendrogram can be generated from the transitive consensus matrix by calculating its α-cuts (Mirzaei and Rahmati, 2010). Therefore, to further our research, we are planning to investigate the link between a partition tree of a fuzzy equivalence relation and a dendrogram.

Secondly, the calculation of the transitive consensus matrix in the hierarchical clustering combination task in this thesis relies on the transitive closure approach. In Chapter 3, we mentioned that calculating the transitive closure belongs to 1 of 3 main groups of algorithms to turn a relation to a transitive relation. For reasons of time, the other algorithms are not considered in this work. In fact, there are some algorithms that can perform better than the algorithms proposed by Klir et al. (1997), but they are not the focus of this study. Therefore, it is interesting to consider an algorithm with time complexity $O(n^2)$ introduced by Dunn (1974) or an algorithm formulated by Kundu (2000).

Finally, we are aware that this thesis is just a preliminary attempt to develop a new solution for the TL problem. Our research suggests that the $S\&U$ algorithm so far has encouraging results, but it has one limitation which is the high complexity. Future work, which takes the optimization of the $S\&U$ algorithm into account, will need to be undertaken. It requires the comprehension of the worst-case time complexity of the proposed algorithm and its behaviour with respect to the tree size and number of base dendrograms. We hope that further studies will make the $S\&U$ method more computationally efficient.

Bibliography

Bandler, W., Kohout, L.J., 1988. Special properties, closures and interiors of crisp and fuzzy relations. Fuzzy Sets and Systems 26, 317–331. URL: https://www.sciencedirect.com/science/article/pii/0165011488901261, doi:https://doi.org/10.1016/0165-0114(88)90126-1.

Bansal, M.S., Chang, W.C., Eulenstein, O., Fernández-Baca, D., 2009. Generalized binary tanglegrams: Algorithms and applications, in: Rajasekaran, S. (Ed.), Bioinformatics and Computational Biology, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 114–125.

Buchin, K., Buchin, M., Byrka, J., Nöllenburg, M., Okamoto, Y., Silveira, R., Wolff, A., 2008. Drawing (complete) binary tanglegrams, pp. 324–335. doi:10.1007/s00453-010-9456-3.

Chawshin, K., Berg, C.F., Varagnolo, D., Lopez, O., 2021. Lithology classification of whole core ct scans using convolutional neural networks. SN Applied Sciences 3, 668. URL: https://doi.org/10.1007/s42452-021-04656-8, doi:10.1007/s42452-021-04656-8.

Chernoff, H., 1975. Cluster analysis for applications (michael r. anderberg). SIAM Review 17, 580–582. URL: https://doi.org/10.1137/1017065, doi:10.1137/1017065, arXiv:https://doi.org/10.1137/1017065.

Dietterich, T.G., 2000. Ensemble methods in machine learning, in: Multiple Classifier Systems, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 1–15.

Dunn, J.C., 1974. A graph theoretic analysis of pattern classification via tamura's fuzzy relation. IEEE Transactions on Systems, Man, and Cybernetics SMC-4, 310–313. doi:10.1109/TSMC.1974.5409141.

Fisher, R.A., 1936. The use of multiple measurements in taxonomic problems. Annals of Eugenics 7, 179–188. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-1809.1936.tb02137.x, doi:https://doi.org/10.1111/j.1469-1809.1936.tb02137.x.

Galili, T., 2015. dendextend: an r package for visualizing, adjusting and comparing trees of hierarchical clustering. Bioinformatics 31, 3718 – 3720.

Garmendia, L., Campo, R., López, V., Ferrés, J., 2009. An algorithm to compute the transitive closure, a transitive approximation and a transitive opening of a fuzzy proximity .

Gezlaw, A., Speckenmeyer, E., Porschen, S., 2012. Generalized k-ary tanglegrams on level graphs: A satisfiability-based approach and its evaluation. Discrete Applied Mathematics 160, 2349–2363. URL: https://www.sciencedirect.com/science/article/pii/S0166218X1200234X, doi:https://doi.org/10.1016/j.dam.2012.05.028.

Jain, A.K., Dubes, R.C., 1988. Algorithms for Clustering Data. Prentice-Hall, Inc., USA.

Jr., J.H.W., 1963. Hierarchical grouping to optimize an objective function. Journal of the American Statistical Association 58, 236–244. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1963.10500845, doi:10.1080/01621459.1963.10500845.

Kandasamy, W.B.V., Smarandache, F., 2003. Fuzzy Cognitive Maps and Neutrosophic Cognitive Maps. Xiquan.

King, B., 1967. Step-wise clustering procedures. Journal of the American Statistical Association 62, 86–101. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1967.10482890, doi:10.1080/01621459.1967.10482890.

Klir, G.J., St. Clair, U., Yuan, B., 1997. Fuzzy Set Theory: Foundations and Applications. Prentice-Hall, Inc., USA.

Kundu, S., 2000. An optimal o(n2) algorithm for computing the min-transitive closure of a weighted graph. Information Processing Letters 74, 215 – 220. URL: http://www.sciencedirect.com/science/article/pii/S0020019000000648, doi:https://doi.org/10.1016/S0020-0190(00)00064-8.

Meyer, H., Naessens, H., De Baets, B., 2004. Algorithms for computing the min-transitive closure and associated partition tree of a symmetric fuzzy relation. European Journal of Operational Research 155, 226–238. doi:10.1016/S0377-2217(02)00730-0.

Mirzaei, A., Mohammad, R., Ahmadi, M., 2008. A new method for hierarchical clustering combination. Intell. Data Anal. 12, 549–571. doi:10.3233/IDA-2008-12603.

Mirzaei, A., Rahmati, M., 2010. A novel hierarchical-clustering-combination scheme based on fuzzy-similarity relations. IEEE Transactions on Fuzzy Systems 18, 27–39.

Mitchell, T.M., 1997. Machine Learning. McGraw-Hill, New York.

Mohri, M., Rostamizadeh, A., Talwalkar, A., 2018. Foundations of Machine Learning. 2nd ed., The MIT Press.

Naessens, H., Meyer, H., De Baets, B., 2002. Algorithms for the computation of t-transitive closures. Fuzzy Systems, IEEE Transactions on 10, 541 – 551. doi:10.1109/TFUZZ.2002.800654.

de Oliveira, J., Pedrycz, W., 2007. Advances in fuzzy clustering and its applications doi:10.1002/9780470061190.

Podani, J., Dickinson, T., 1984. Comparison of dendrograms: a multivariate approach. Botany 62, 2765–2778.

Rehman, S., Shafique, L., Ihsan, A., Liu, Q., 2020. Evolutionary trajectory for the emergence of novel coronavirus sars-cov-2. Pathogens 9.

Rohlf, F.J., Sokal, R.R., 1981. Comparing numerical taxonomic studies. Systematic Zoology 30, 459–490. URL: http://www.jstor.org/stable/2413054.

Schaeffer, J., 2008. One Jump Ahead: Computer Perfection at Checkers. 2nd ed., Springer Publishing Company, Incorporated.

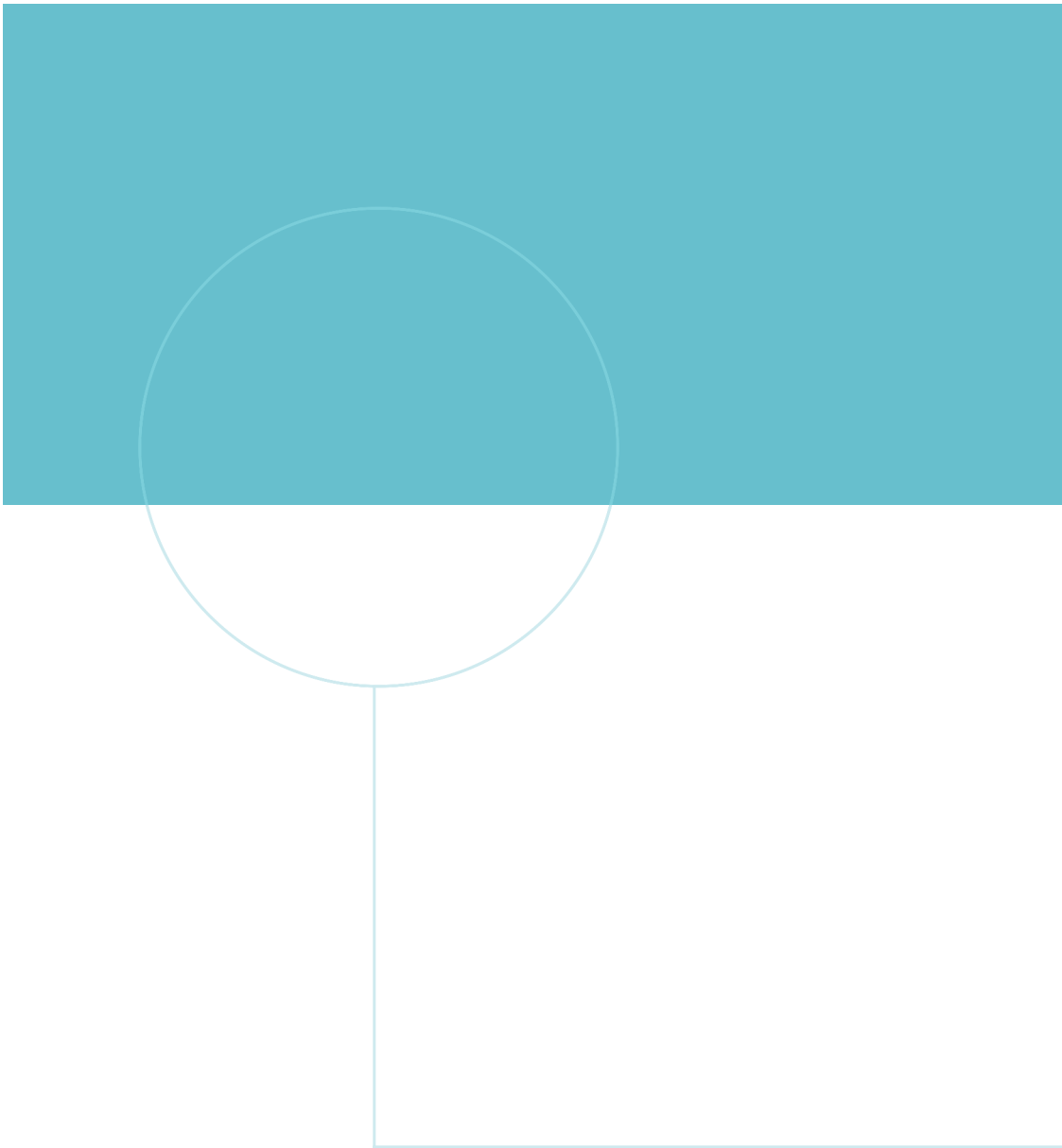
Sneath, P.H.A., Sokal, R.R., 1973. Numerical Taxonomy: The Principles and Practice of Numerical Classification. W. H. Freeman and Co.

Sokal, R., Rohlf, F., 1962. Sokal rr, rohlf fj. the comparison of dendrograms by objective methods. taxon 11: 33-40. Taxon 11, 33–40. doi:10.2307/1217208.

Vienne, D., 2018. Tanglegrams are misleading for visual evaluation of tree congruence. Molecular biology and evolution 36. doi:10.1093/molbev/msy196.

Zadeh, L., 1965. Fuzzy sets. Information and Control 8, 338 – 353. URL: http://www.sciencedirect.com/science/article/pii/S001999586590241X, doi:https://doi.org/10.1016/S0019-9958(65)90241-X.

Zadeh, L., 1971. Similarity relations and fuzzy orderings. Information Sciences 3, 177 – 200. URL: http://www.sciencedirect.com/science/article/pii/S0020025571800051, doi:https://doi.org/10.1016/S0020-0255(71)80005-1.



NTNU

Norwegian University of
Science and Technology