



SENIOR THESIS IN MATHEMATICS

Clustering and Spanning Trees: Consonant Approaches

Author:
Ethan Ong

Advisor:
Dr. Vin de Silva

Submitted to Pomona College in Partial Fulfillment
of the Degree of Bachelor of Arts

April 12, 2021

Abstract

Clustering is the process of grouping similar objects into groups called clusters. Spanning trees are subgraphs that connect all vertices within a graph with a minimal number of edges. In my thesis, I explore each topic separately and eventually arrive on the conclusion that single-linkage clustering and minimum spanning trees are equivalent. I then conduct further exploration into additional connections between single-linkage clustering, minimum spanning trees, and k -means clustering.

Contents

1	Introduction	1
1.1	Graph Theory	1
1.2	Clustering	2
2	Mathematics of Spanning Trees	3
2.1	Minimum Spanning Trees (MST)	3
2.1.1	Trees	3
2.1.2	Spanning Trees	6
2.1.3	Minimum Spanning Trees (MST)	8
2.2	Algorithms	9
2.2.1	Borůvka's Algorithm	9
3	Clustering	19
3.1	Hierarchical Clustering	20
3.1.1	Single-linkage clustering (SLC)	23
3.1.2	MST and SLC	28
4	Further Exploration	30
4.1	k -means clustering	31
5	Conclusion	39

Chapter 1

Introduction

Clustering and spanning trees are two topics that are often studied in different realms of mathematics. Spanning trees are commonly discussed in the context of graph theory. Graph theory being a widely studied topic that is used to map abstract relationships. On the other hand, clustering is most often associated with statistics, being widely used in various statistical and data analyses. Both topics are studied separately since the problems they seek to solve are conceptualized as fundamentally different, however, what if I were to say that the two topics are not only related but equivalent?

1.1 Graph Theory

One of the most common problems discussed and learned in graph theory is network optimization. There are multiple ways to quantify something as optimal, thus, in solving network optimization problems, there are also multiple problems to be solved. The shortest path problem is one such problem where the goal is finding the path between two vertices in a graph where the sum of edge weights is minimized [1]. In solving the shortest path problem, there are many algorithms that have been popularized. Most notably, Dijkstra's algorithm is well-studied, proposing to find the shortest path between any given pair of vertices [7, 18]. Another method to finding the shortest path involves finding the shortest-path tree [5, 2, 10]. A shortest-path tree displays the shortest paths from a fixed vertex to all other vertices. More specifically, the shortest-path tree is also known as a *minimum spanning tree*, such that the total weight of the tree is minimized [10].

One approach to find the minimum spanning tree is to find all spanning trees for a particular graph and calculate the respective total weights. As graphs get larger and larger, considering the total weight of every feasible spanning tree is not an efficient heuristic. As a result, greedy heuristics exist to efficiently find the minimum spanning tree(s) of a graph [3, 20, 15, 12].

1.2 Clustering

Clustering also known as cluster analysis is the process of putting a set of objects into groups called clusters. Due to its large relevancy in data mining, it is used in a variety of different fields for exploratory purposes such as bioinformatics and machine learning. Since clustering is a general task, several different algorithms can be used to achieve the desired outcome. Often times, a clustering algorithm may not perform as intended, thus, other algorithms are preferred and even new algorithms can be derived given the appropriate technical conditions.

Both topics are motivated to solve different problems, however, in learning more about spanning trees and clustering during the span of this thesis, I have come to find that they are more similar than they appear.

Chapter 2

Mathematics of Spanning Trees

I first read about spanning trees in Matoušek's Thirty-three Miniatures: Mathematical and Algorithmic Applications of Linear Algebra. Particularly, Matoušek elaborates on how algorithmic approaches are powerful tools in enumeration [16]. What followed was curiosity in how spanning trees are used in their mathematical applications which lead me to minimum spanning trees.

2.1 Minimum Spanning Trees (MST)

Minimum spanning trees are an important mathematical concept that can be used to solve various optimization problems. In this section, I construct a bottom-up understanding of minimum spanning trees and introduce an enlightening example.

2.1.1 Trees

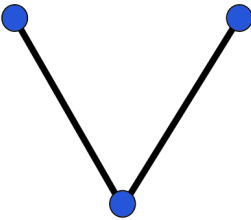
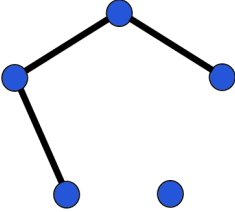
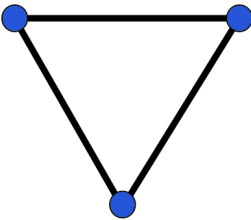
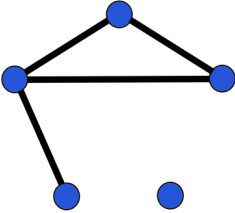
To build an understanding of trees, one must have a basic understanding of graphs. Simply put, a *graph* is a set of objects, known as *vertices*, connected by lines, known as *edges*.

Definition 2.1 (Graph). A **graph** G is an ordered pair $G = (N, E)$ where N is a set of vertices and E is a set of unordered pairs $\{a, b\}$, known as edges, where $a, b \in N$ and $a \neq b$. We refer to $\{a, b\}$ as the edge ab or ba .

Graphs can be classified in several ways and here we discuss two ways: connectivity and acyclic-ness.

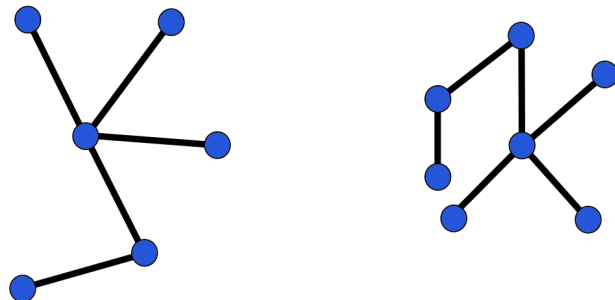
Connectivity: A graph is *connected* when the entire graph is one connected component and *disconnected* when the graph is composed of several connected components.

Acyclic-ness: A graph is *acyclic* when the graph contains no cycles and *non-acyclic* when the graph contains a cycle. Below are several examples of graphs that exhibit connectivity and acyclic-ness.

	Connected	Disconnected
Acyclic		
Non-Acyclic		

Observe that disconnected graphs are graphs with several connected components. Additionally, observe that the above non-acyclic graphs contain a cycle because each have a triangle or enclosed shape. The graphs that I will

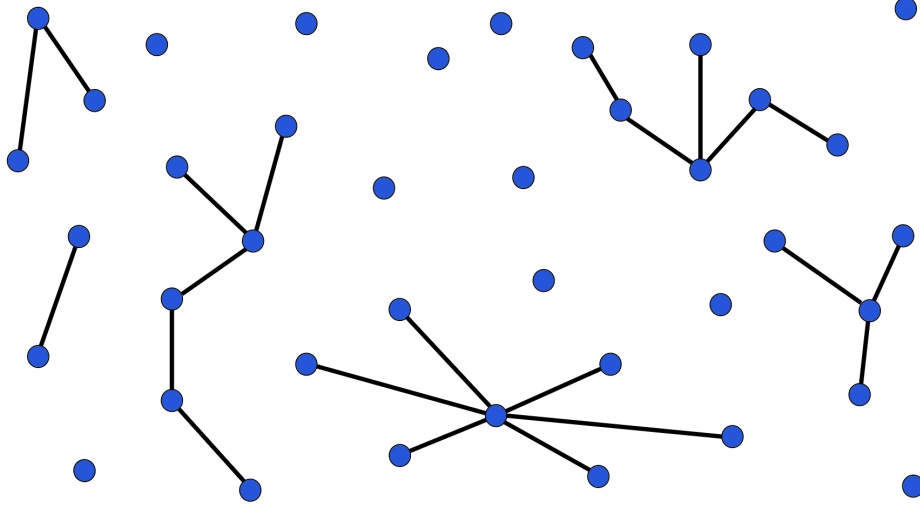
focus on are those that are connected and acyclic. In fact, graphs that are both *connected* and *acyclic* are called **trees**. Consider the following examples of trees and a theorem analogous to the 2/3 theorem of vector spaces [21].



Theorem 2.2 (2/3 Theorem). *Let T be a graph with n vertices. Then any two of the following statements implies the third.*

1. T is connected.
2. T is acyclic.
3. T has exactly $n - 1$ edges.

Lastly, important to note is that **forests** are acyclic graphs. Forests can be connected or disconnected such that a connected forest is a tree and each connected component in a disconnected forest is a tree. Observe the following to be a forest that is disconnected and acyclic.



2.1.2 Spanning Trees

Spanning trees are subgraphs containing all vertices of a graph that are also trees. Below I introduce a couple formal definitions, a lemma, and a theorem.

Definition 2.3 (Subgraph). A graph $G' = (N', E')$ is a **subgraph** of a graph $G = (N, E)$ if $N' \subseteq N$ and $E' \subseteq E$.

Definition 2.4 (Spanning Tree). Given a graph $G = (N, E)$, a **spanning tree** $T = (N_T, E_T)$ is a subgraph of G that is a tree such that $N = N_T$.

Definition 2.5 (Subforest). A **subforest** is an acyclic subgraph.

In combination with the following lemma, I claim that every connected graph has at least one spanning tree. Furthermore, I introduce two methods of identifying at least one spanning tree.

Lemma 2.6. *Let G be a graph. Let e be an edge in G that lies on a cycle. Then $G - e$ has the same connected components as G .*

Theorem 2.7. *For every connected graph G , there exists at least one spanning tree T .*

Method 1: A minimal connected subgraph is a spanning tree.

Proof. Given a connected graph G , consider all connected subgraphs which contain all vertices in G . There exists at least one connected subgraph as G fits this criteria. Let G' be such a subgraph with a minimum number of edges. We claim that G' is a spanning tree.

We know G' is connected. Suppose G' contains a cycle. Then by Lemma 2.6 we can remove one edge in the cycle and the resulting subgraph will still be connected. This contradicts the minimality of G' . Thus, G' must be a spanning tree after all. \square

Method 2: A maximal acyclic subgraph is a spanning tree.

Proof. Given a connected graph G , consider all acyclic subgraphs which contain all vertices in G . There exists at least one acyclic subgraph as the empty graph with all vertices and no edges fits this criteria. Let G' be such a subgraph with a maximal number of edges. We claim that G' is a spanning tree.

We know G' is acyclic. Suppose G' is not connected. Since G is connected, it must contain at least one edge e that connects two distinct components of G' . Let $G'' = G' + e$ be the subgraph obtained by adding e to G' . If we can show that G'' is acyclic, then this contradicts the maximality of G' .

Suppose G'' contains a cycle. Since G' is acyclic, it follows that e must be on the cycle. Then Lemma 2.6 implies that G' and G'' have the same connected components. This contradicts the fact that e was chosen to connect two connected components of G' . Thus, G'' is indeed acyclic.

We have contradicted the maximality of G' . Thus, G' is connected after all and is then a spanning tree. \square

Considering the two methods above, I informally state that a spanning tree can be constructed by subtracting edges from the original graph or by adding edges to the empty subgraph containing all vertices and no edges. As a matter of fact, given a graph with n vertices, the spanning tree will have $n - 1$ edges.

We have established that we know given any connected graph, there exists at least one spanning tree. It is helpful to note that *spanning* in spanning tree can be related to the fact that spanning trees encompass all vertices in their original graph. In a sense, the spanning tree *spans* all vertices of the original graph.

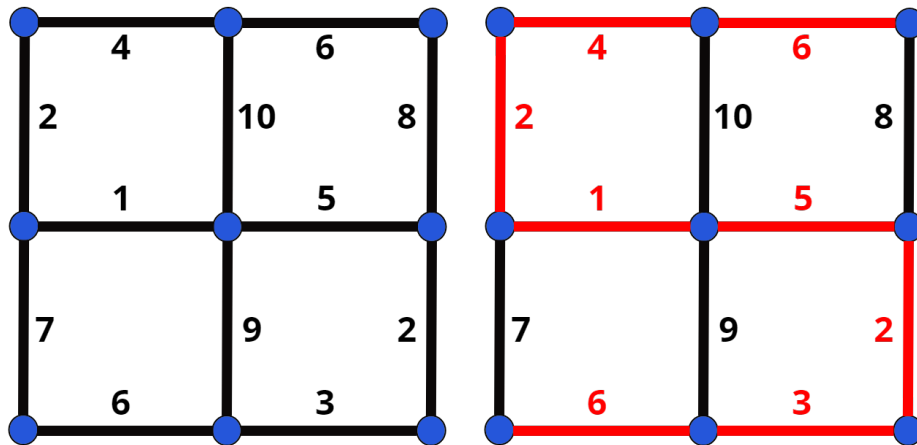
2.1.3 Minimum Spanning Trees (MST)

In this section, we consider the notion of a minimum spanning tree which occurs when we have a weighted graph. From the previous section, it is helpful to takeaway that a spanning tree of a connected graph G is a connected subgraph with minimum edges. However, minimum spanning trees have relevance when we are considering a graph with weighted edges. Consider the following example in conjunction with a definition.

Definition 2.8 (Weighted Graphs). A **weighted graph** is a graph in which a number (weight) is assigned to each edge.

Example 2.9 (Neighborhood Road Expansion). Suppose you have a neighborhood of houses that are connected by a network of roads. You want to expand certain roads such that every house is accessible for an extremely large moving truck. You are considering many different expansion plans because expanding each road has an associated labor cost. How can you do this such that you are minimizing the labor costs for expansion?

The solution to this problem is to find the spanning tree where the labor cost of expansion plan or sum of edge weights is minimized. Provided below is the neighborhood of houses in consideration represented as a graph and the cheapest expansion plan represented as a minimum spanning tree.



Definition 2.10 (Minimum Spanning Tree). A **minimum spanning tree** is a spanning tree where the sum of edge weights is as small as possible.

2.2 Algorithms

With increasingly larger graphs, finding a minimum spanning tree becomes difficult to manually compute. Fortunately, many greedy algorithms exist to find minimum spanning trees through an iterative process. I introduce one algorithm here and prove the algorithm's correctness.

2.2.1 Borůvka's Algorithm

Borůvka's Algorithm is one algorithm that allows one to find a minimum spanning tree. Initially published by Otakar Borůvka in 1926, Borůvka derived the method in order to construct efficient electricity networks [3]. Similar to the other algorithms in existence, Borůvka's Algorithm is a greedy algorithm that is optimal at every iteration. In using this algorithm, consider the following assumption.

Assumption 2.11. *The edge weights of the graph must be distinct.*

The intuition here is that an ordering of priority must be imposed on the edges of the graph. If there are multiple edges with the same weight, given $\varepsilon > 0$, multiples of ε can be added to each of the distinct weights. In this way, we impose an order of priority for edges with distinct weights.

The steps of the algorithm can be classified into two steps that are iterated on until a minimum spanning tree is constructed.

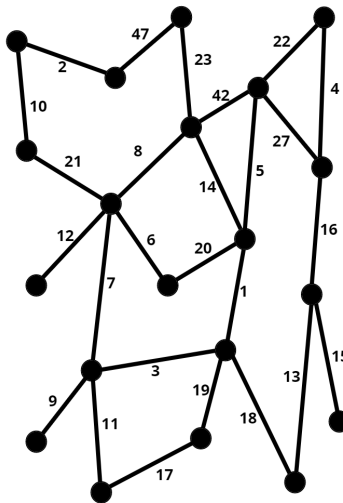
Notes:

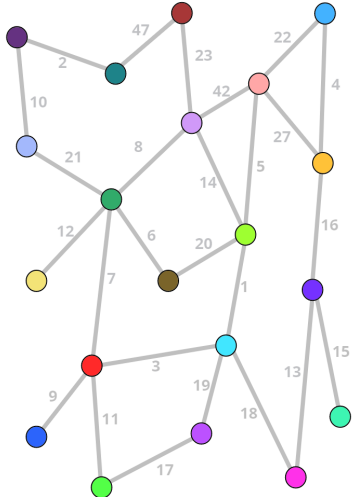
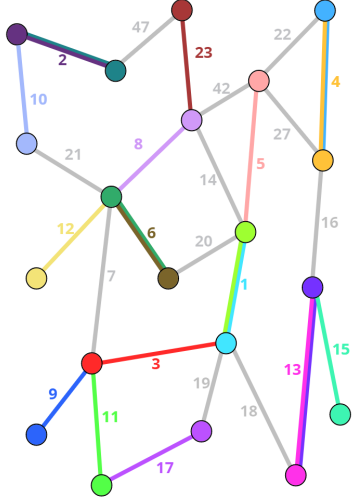
- Edge weights must be distinct.
- If there exists non-distinct edge weights, add multiples of ε such that a priority is imposed on the distinct edge weights.

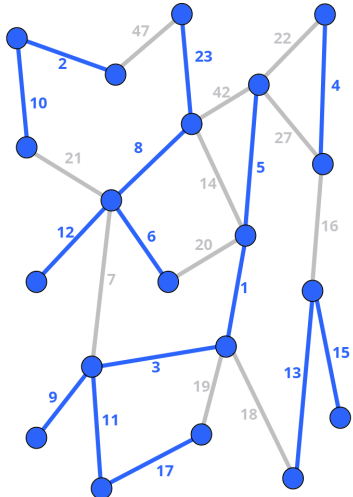
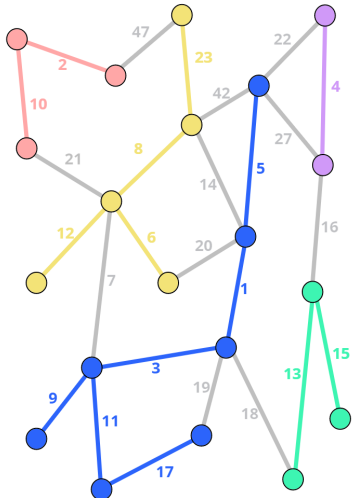
Steps:

1. Initialization: Let F be a subforest of all vertices and no edges.
2. While F has more than one component:
 - For each component, find the lowest weight edge connecting it to another component of the forest.
 - Add these edges to F .

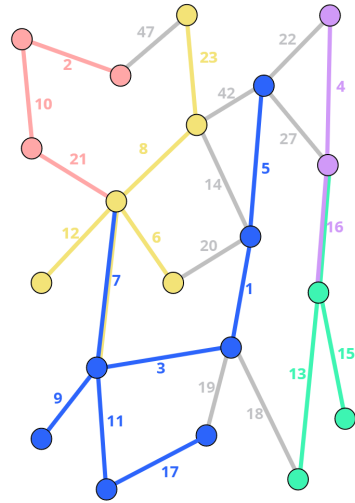
Consider the following weighted graph in implementing Borůvka's Algorithm.



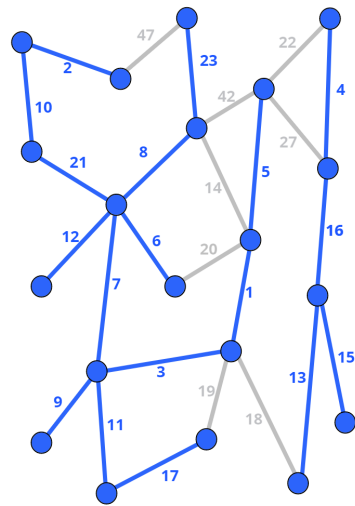
<p>Initialization of subforest F of all vertices and no edges.</p>	
<p>Add the lowest weight edge connecting each component to another component to subforest F.</p>	

<p>Update F.</p>	
<p>Check to see if F has more than one connected component.</p>	

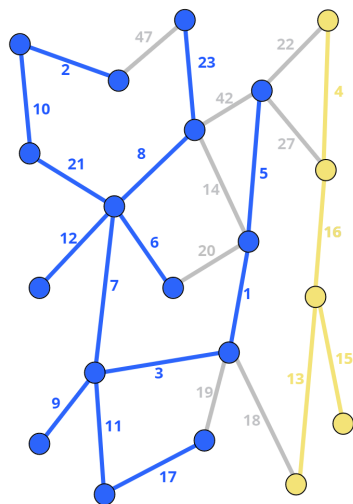
Add the lowest weight edge connecting each component to another component to subforest F .



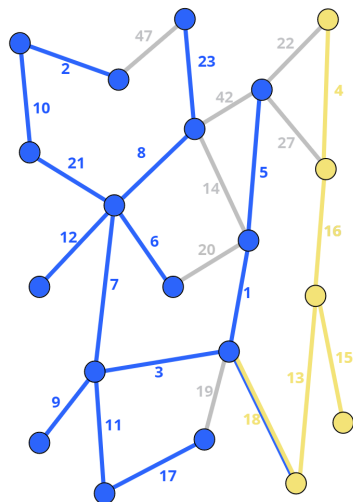
Update F .



Check to see if F has more than one connected component.



Add the lowest weight edge connecting each component to another component to subforest F .



Now we will prove the correctness of Borůvka's Algorithm. The claims of importance are:

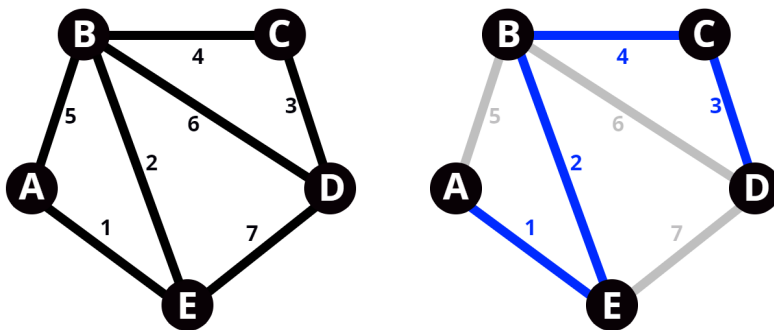
1. Borůvka's Algorithm gives a tree which is minimum.
2. Every edge added in Borůvka's Algorithm is in every minimum spanning tree.

Consider Theorem 2.2 and the following lemma in proving the correctness of Borůvka's Algorithm.

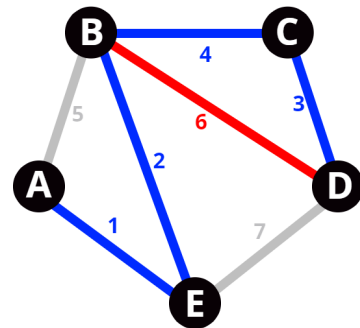
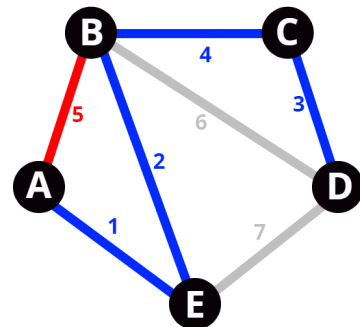
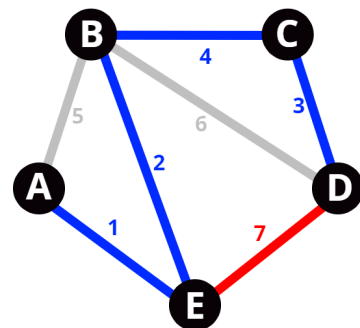
Lemma 2.12. *Let T be a minimum spanning tree of G . Let $e = \{x, y\}$ be an edge connecting vertices x and y in G that is not in T . Then $T + e$ creates a cycle C where e is the heaviest edge in C .*

First, observe the following proof by example for lemma 2.12.

Proof By Example. Given the graph on the left G , observe the subgraph on the right T to be the MST of G .



Consider any edge not in T to be an edge $e = \{x, y\}$, where e connects vertices x and y in G . Note that with each possible edge e , $T + e$ creates a cycle where e is the heaviest edge in the cycle.

$e = \{B, D\}$	
$e = \{A, B\}$	
$e = \{D, E\}$	

□

Now, we formally prove lemma 2.12.

Proof. There exists a path connecting any vertices x and y in T . This path together with e forms a cycle C . Now we must show that e is the heaviest edge in the cycle C . If e were lighter than some edge f in the cycle, then we claim that $T + e - f$ is a minimum spanning tree of lighter weight. In fact, $T + e - f$ has $n - 1$ edges and is connected. Thus, $T + e - f$ is a minimum spanning tree that is of lighter weight than T . Thus, T is not a minimum spanning tree which is a contradiction. \square

Remark 2.13. If we assume the graph has distinct weights, then it follows that the added edge is *strictly* heavier than all the other edges in the cycle.

Claim 2.14 (Correctness of Borůvka's Algorithm). *Consider a minimum spanning tree T_0 . Every iteration of Borůvka's Algorithm is a subgraph of T_0 .*

Proof. We will prove the above claim by induction.

Base Case: After the first initialization, the empty graph of all vertices and no edges is a subgraph of T_0 .

Inductive Step: After the n^{th} iteration of Borůvka's Algorithm, I claim we still have a subgraph of T_0 . If Borůvka's Algorithm adds an edge e not in T_0 , Lemma 2.12 says that $T_0 + e$ has a cycle C where e is the heaviest edge in C . Hence, e should not have been added in the first place and every edge that Borůvka's Algorithm adds is in T_0 . Thus, the n^{th} iteration is a subgraph of T_0 . \square

Furthermore, if the subgraph Borůvka's Algorithm constructs is not connected, the algorithm does not terminate. With that, I end on the below result that follows from the correctness of Borůvka's Algorithm.

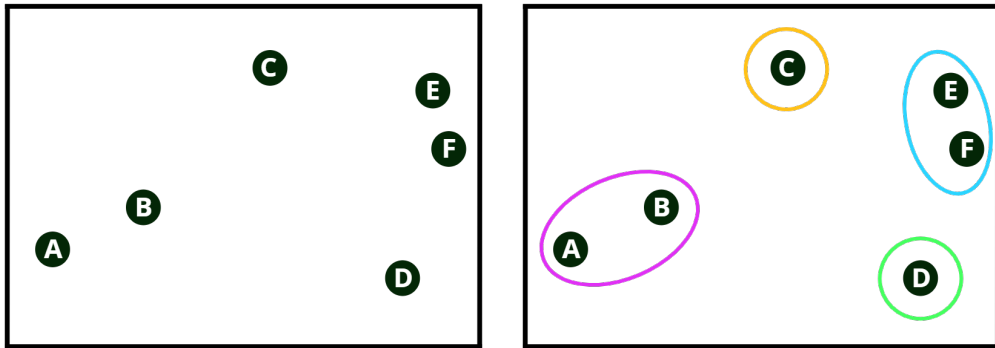
Corollary 2.15. *The minimum spanning tree is unique.*

Proof. Assuming the edges are distinct, suppose we know there exists two distinct minimum spanning trees $MST_1 \neq MST_2$. By the correctness of Borůvka's Algorithm, we know that the algorithm does not add edges not in MST_1 and MST_2 . In other words, Borůvka's Algorithm converges to MST_1 and MST_2 . Hence, $MST_1 = MST_2$. \square

Chapter 3

Clustering

Clustering, also known as cluster analysis, is a process of grouping similar objects in groups called clusters. The underlining goal is to group elements together that have more in common with each other than with elements in other clusters. It is important to note that clustering is not a single task but an iterative process that can be approached using various algorithms. For instance, given the following set of elements, one might want to visually partition the elements like so.



As one might expect, the task of clustering is applicable in so many areas of exploration. As a result, many different approaches to clustering exist.

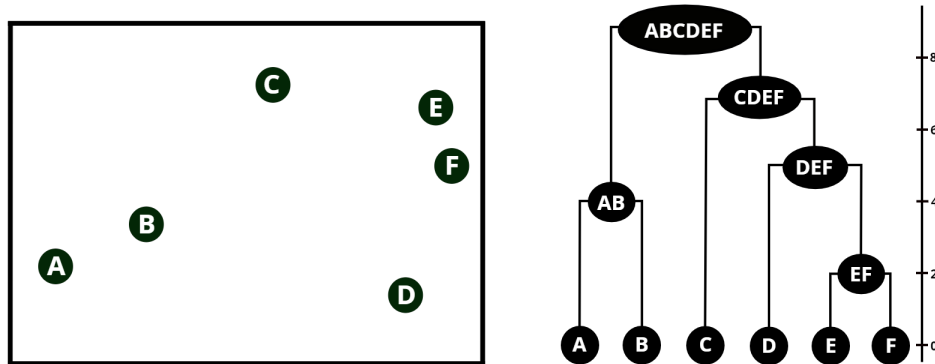
3.1 Hierarchical Clustering

Definition 3.1 (Hierarchical clustering). **Hierarchical clustering** is a type of clustering that builds a hierarchy of clusters.

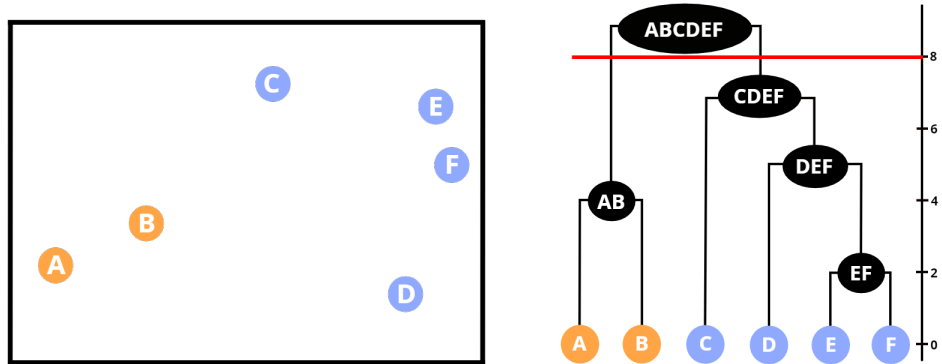
- At any given threshold, we have a clustering.
- Lowering the threshold results in a refinement of the previous clustering.

Definition 3.2 (Dendrogram). A **dendrogram** is a ‘tree-like’ diagram illustrating the series of steps taken by the method in proceeding from n single member clusters to a single group containing all n individuals [6].

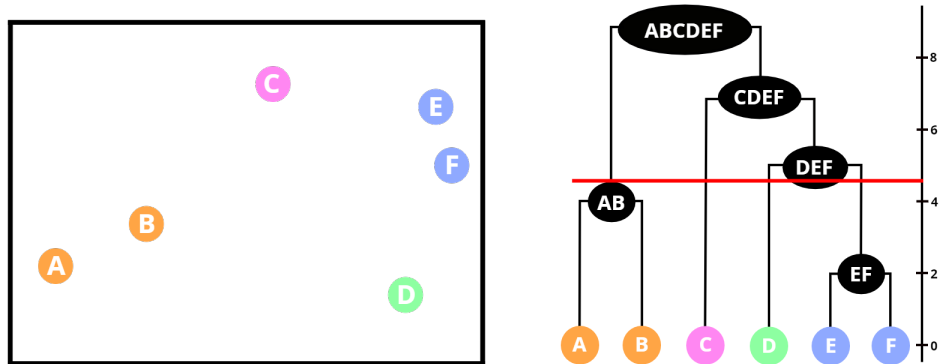
Hierarchical clustering is one approach to the process of cluster analysis. The output of a hierarchical clustering algorithm is a hierarchy of clusters. This can be viewed as a dendrogram such as the one below.



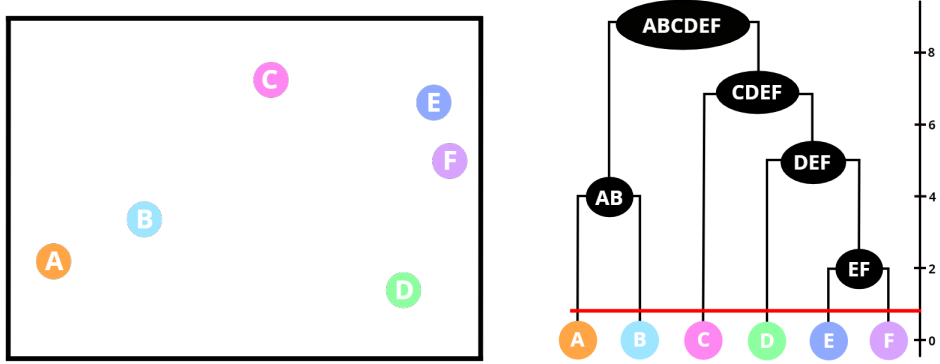
Note that thresholds correspond to values on the vertical axis such that at every threshold, there exists a clustering. Observe that at a threshold of 8, the following clustering results: $\{AB, CDEF\}$.



Lowering the threshold from 8 to 5 results in a refinement of the previous clustering: $\{AB, C, D, EF\}$. Observe that the clustering at a threshold of 5 not only has more clusters, the newly formed clusters are a subset of a cluster in a previous clustering. In this case, clusters $\{C, D, EF\}$ are all subsets of the cluster $CDEF$ at a threshold of 8.



Necessarily, lowering the threshold from 5 to 1 results in a further refinement of the previous clustering. In this case, each element is now its own cluster.



Methods of hierarchical clustering are algorithmic and can either be agglomerative or divisive. In other words, the dendrograms that depict the hierarchy of clustering can either be built from the *bottom-up* or from the *top-down*. Hierarchical clustering algorithms are based on the assumption that objects are more similar to nearby objects than to objects farther away. As a result, at various thresholds, hierarchical clustering algorithms form different clusters.

Furthermore, hierarchical clustering algorithms differ in how the distance metrics are calculated and the linkage criterion that is used. The choice of **metric** can affect the overall shape of clusters and can be tuned appropriately. The **linkage criterion** determines in what priority a hierarchical clustering algorithm merges or splits its clusters. As a result, depending on how the distances are computed and the selected linkage criterion, hierarchical clustering algorithms can output different hierarchies of the same set of elements.

Remark 3.3. Suppose H is a set of n elements such that elements $i, j \in H$. The **input** for a hierarchical clustering algorithm is a $n \times n$ matrix where each entry (i, j) is the distance between elements i and j . The **output** of a hierarchical clustering algorithm is a $n \times n$ matrix where each entry (i, j) is the threshold at which elements i and j split or merge.

3.1.1 Single-linkage clustering (SLC)

Single-Linkage Clustering is one hierarchical clustering method that can be used in cluster analysis. Below I present two definitions for single-linkage clustering that are equivalent.

Definition 3.4 (Single-linkage clustering I). In **single-linkage clustering**, at any threshold d , two elements belong to the same cluster if and only if they are connected by a path of edges with weight less than or equal to d .

We will now discuss an alternative definition that is more algorithmic such that we can implement single-linkage clustering in an agglomerative way.

Definition 3.5 (Single-linkage clustering II). In **single-linkage clustering**, the distance between any two clusters is the minimum distance between any element in the first cluster and any element in the second cluster. This can be described by the following linkage criterion where A and B are any two elements or clusters and $d(a, b)$ represents the distance metric between two elements a and b .

$$D(A, B) = \min_{a \in A, b \in B} d(a, b)$$

The steps of single-linkage clustering can be described as two steps that are iterated on until all elements are in one cluster.

Notes:

- D , an $n \times n$ distance matrix is required as input for a set N with n elements.

Steps:

1. Initialize a disjoint clustering where each element is a singleton cluster and L , an empty set of thresholds.
2. While there is more than one cluster:
 - Find a and b where

$$d(a, b) = \min d(i, j) \quad \forall i, j \in N$$

.

- Merge a and b into a new cluster and add $d(a, b)$ to L .
- Delete the rows and columns corresponding to a and b while adding a new row and column for (a, b) such that

$$d[(a, b), c] = \min \{d(a, c), d(b, c)\} \quad \forall c \in N$$

.

As mentioned in introducing hierarchical clustering, the results from single-linkage clustering can be represented as a dendrogram. From the algorithmic steps above, note that the set of thresholds, L is required to plot at which distances the merging of clusters occur. What follows is that the original distance matrix can be referenced for every distance $l \in L$ in producing a proper dendrogram.

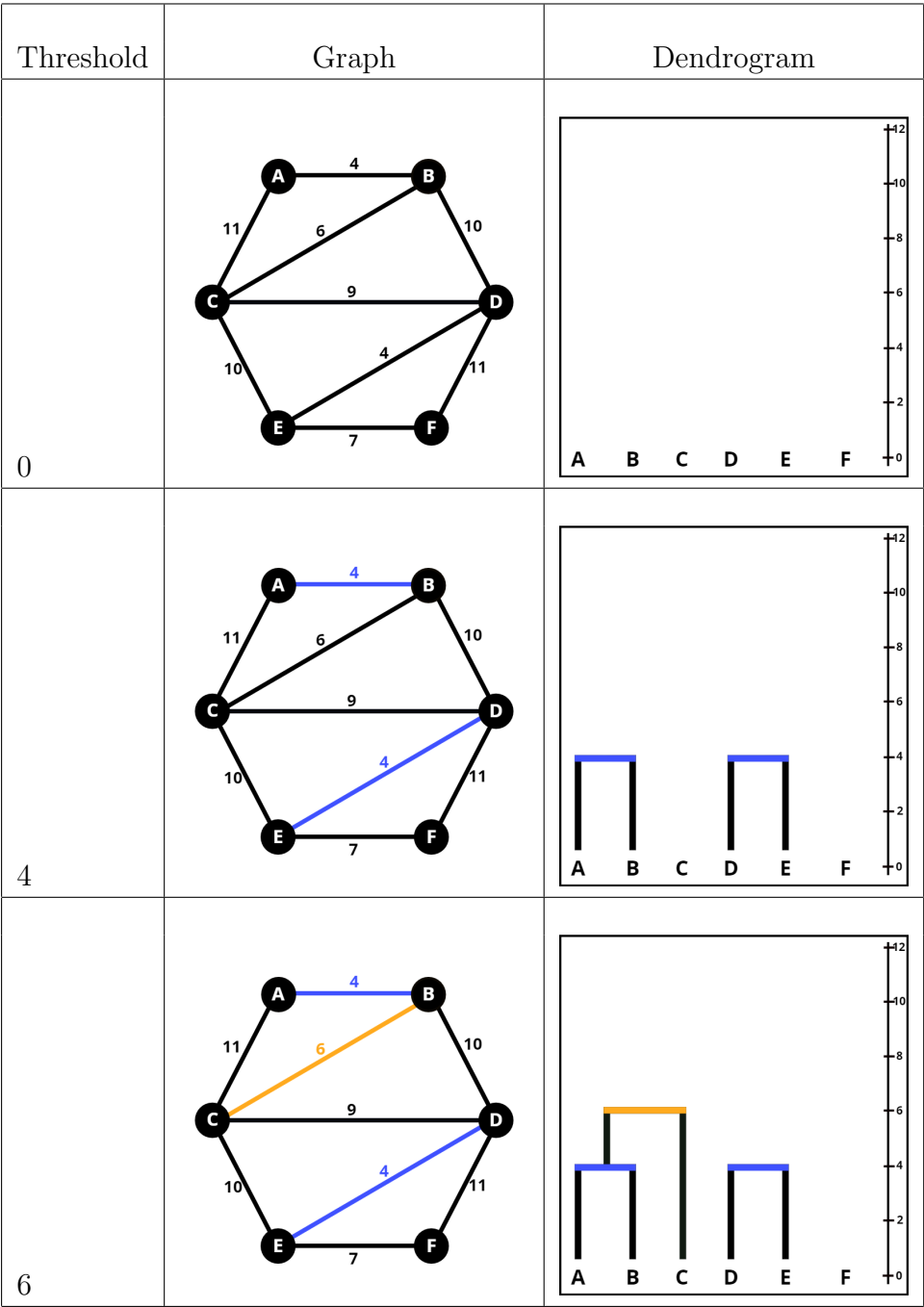
At this point, consider every weighted graph to have a single-linkage clustering where the edge weights correspond to a measure of similarity between nodes. Similarity metrics include but are not limited to Euclidean distances. Note that in an absence of an edge, the edge weight should be considered to be ∞ .

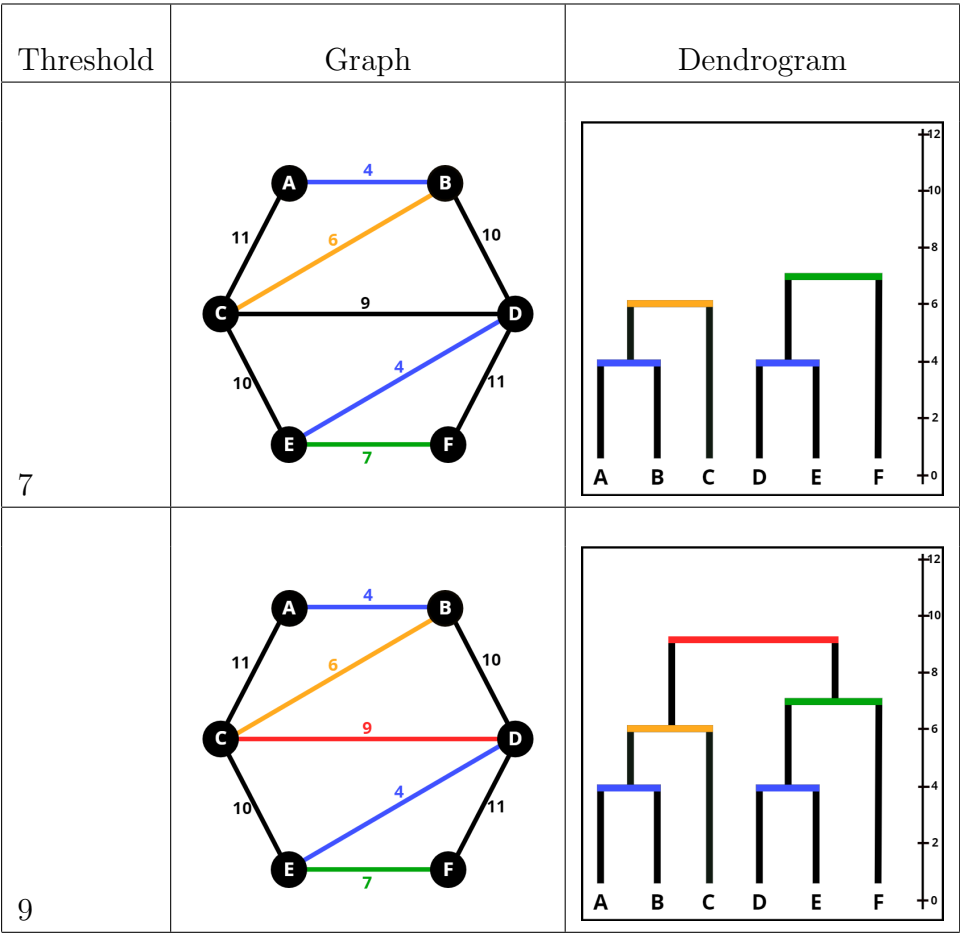
Example 3.6. (A weighted graph) Consider the following weighted graph where the absence of an edge is equivalent to an edge weight of ∞ .

Recall the single-linkage linkage criterion:

$$D(A, B) = \min_{a \in A, b \in B} d(a, b)$$

The single-linkage linkage criterion prioritizes minimum distances that connect two elements that are not in the same cluster. Since single-linkage clustering is an agglomerative process, one starts with all elements as singleton clusters and can build the single-linkage dendrogram by raising the threshold from 0. This allows one to see at what thresholds do cluster merges occur. In the case of a weighted graph, minimum edges that connect separate components are prioritized. Observe the single-linkage dendrogram that results from the following weighted graph.





3.1.2 MST and SLC

Recall Definitions 2.4 and 2.10 of minimum spanning trees and the dendrogram shown in Example 3.6. Considering nodes $A - F$ as singleton nodes, note that reading the dendrogram bottom-up allows us to construct a minimum spanning tree where each edge in the minimum spanning tree corresponds to a merge in the single-linkage dendrogram.

At this point, I state the following equivalence relation.

Theorem 3.7. *Given a weighted graph G and a threshold d . The single-linkage clustering (SLC) of the minimum spanning tree (MST) of G is equal to the single-linkage clustering of G .*

Proof. To confirm the above theorem, I will show the equivalence both ways for any vertices x and y in G . For the purpose of this proof, I first define a d -path.

Definition 3.8 (d -path). Given a threshold d , such a path is called a **d -path** if every edge in that path is less than or equal to d .

If a path exists between x and y in the MST of G , the path exists in G and is a d -path as well.

Conversely, if a path exists between vertices x and y in G , for every edge $\{p, q\}$ in the path that does not belong to the MST, Lemma 2.12 says we can replace $\{p, q\}$ with the unique path that connects vertices p and q within the MST. By Lemma 2.12, every edge in this new path has weight less than or equal to the weight of $\{p, q\}$. Thus, the path is a d -path and exists in the MST of G as well. \square

If the SLC of the MST of a weighted graph G is equal to the SLC of G , the outputs must be equivalent as well. Thus, I state the following corollary.

Corollary 3.9. *Each edge in the minimum spanning tree corresponds to a merge in the single-linkage dendrogram.*

Similarly, recall in Section 2.2.1, I conclude the MST to be unique. This result applies to dendrograms as the following.

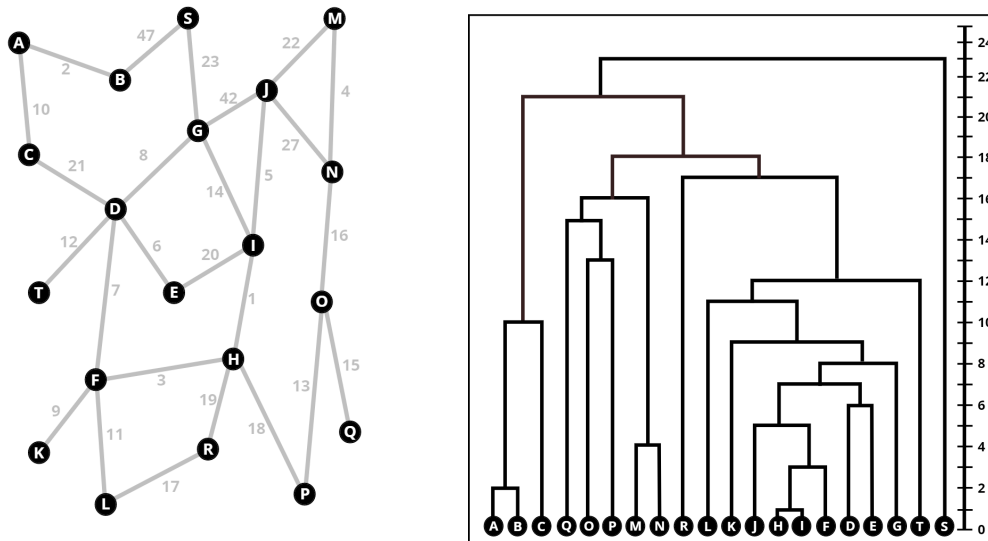
Theorem 3.10. *If two graphs have the same minimum spanning tree with identical weights, they have the same dendrogram.*

Proof. Suppose two graphs G_1 and G_2 have the same minimum spanning tree with identical distinct weights, yet they have different dendrograms. Then the sequence of merges for the single-link clustering of G_1 and G_2 differ. But then the edge weights of G_1 and G_2 must differ which is a contradiction. \square

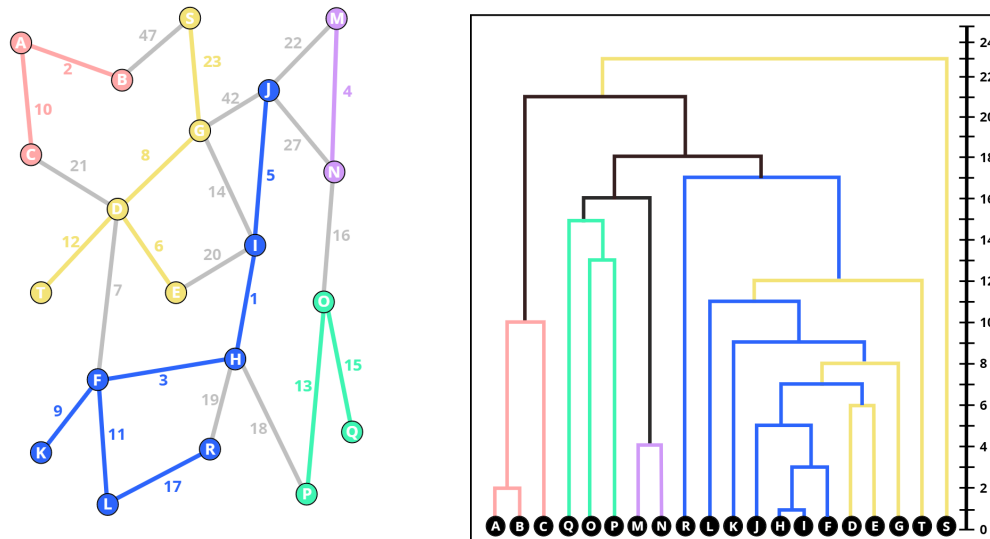
Chapter 4

Further Exploration

In the previous section, I stated that MSTs and SLC are equivalent. In fact, previous research in this intersection has reported that clusters can be found in MSTs and that MSTs contain information about the underlying clusters of an original weighted graph [24]. I also came to this conclusion in examining the connected components of the weighted graph in Section 2.2.1 after the first iteration of Borůvka's Algorithm and the SLC dendrogram. Below I have the weighted graph and it's SLC dendrogram displayed.



Coloring the thresholds in the SLC dendrogram with the corresponding connected components after the first iteration of Borůvka's Algorithm *almost* results in 5 somewhat defined clusters.



Notice how coloring the thresholds with the corresponding connected components does not result in the exact number of defined clusters, however, it does get quite close. This finding that MSTs contain information about the underlying clusters in the original weighted graph led me to another type of clustering, k -means clustering.

4.1 k -means clustering

Definition 4.1 (k -means clustering). **k -means clustering** is a method of clustering that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean [14].

The k -means clustering algorithm is a popular clustering algorithm used in data analysis. All the algorithm needs is the parameter k , which specifies the number of clusters to partition the observations into. Unlike hierarchical clustering, k -means clustering does not construct a hierarchy. In fact, when k -means clustering is run on a particular set of observations, the clustering may differ every run even if the parameter k is kept constant. A common drawback to k -means clustering is that one does not know the optimal number of clusters, to be specified when applying the algorithm. Many methods of selecting an optimal k for k -means clustering exist [19, 25], which involve selecting a k which minimizes classification error.

Since k -means clustering can result in a different clustering every time it is run, there exists several *boundary* observations that can often be misclassified. Thus, I pose the following question.

Motivating Question 4.2. Does a clustering algorithm exist which takes a weighted graph to say if two points are likely to be in the same clustering or not?

I explored this briefly, using Jupyter Notebook [13], with the 80 Cereals data set [4] describing 80 different cereals and their nutritional breakdowns. This data set includes many variables describing nutritional content, however, in order to plot the data in two dimensions, I only consider a cereal's sugar content and rating. I have omitted 4 of the 80 cereals due to missing information. Depending on the variables selected, the cereals plotted in two dimensions could look very different.

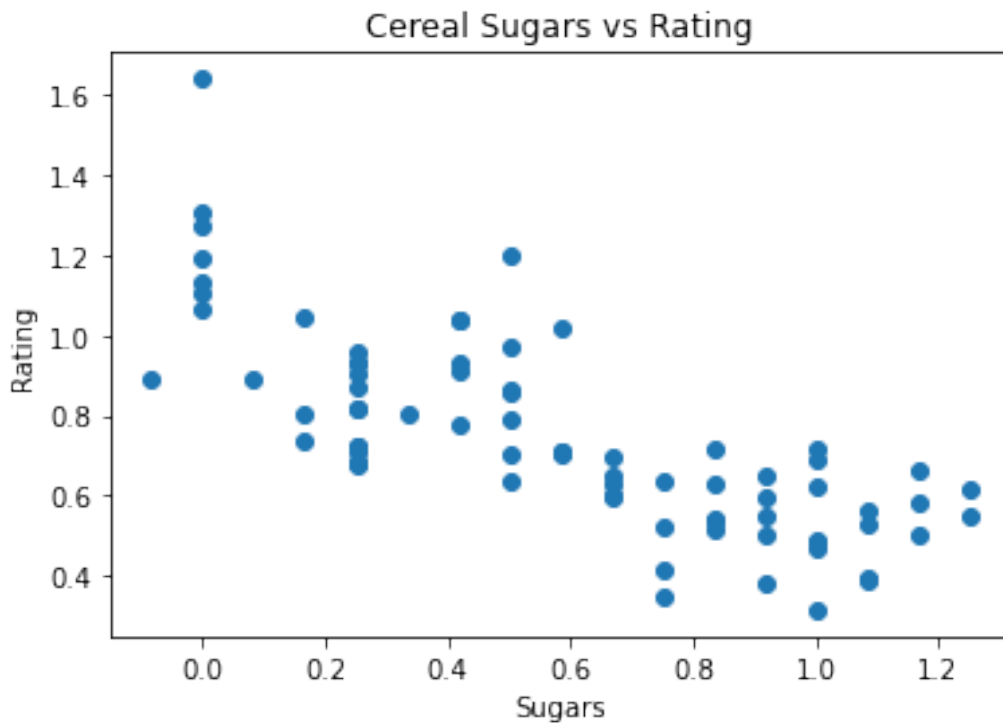


Figure 4.1: The ranges of both axes have been normalized using an 85th-percentile normalization [11].

When $k = 3$, notice that executing k -means several times results in slightly different clustering variations.

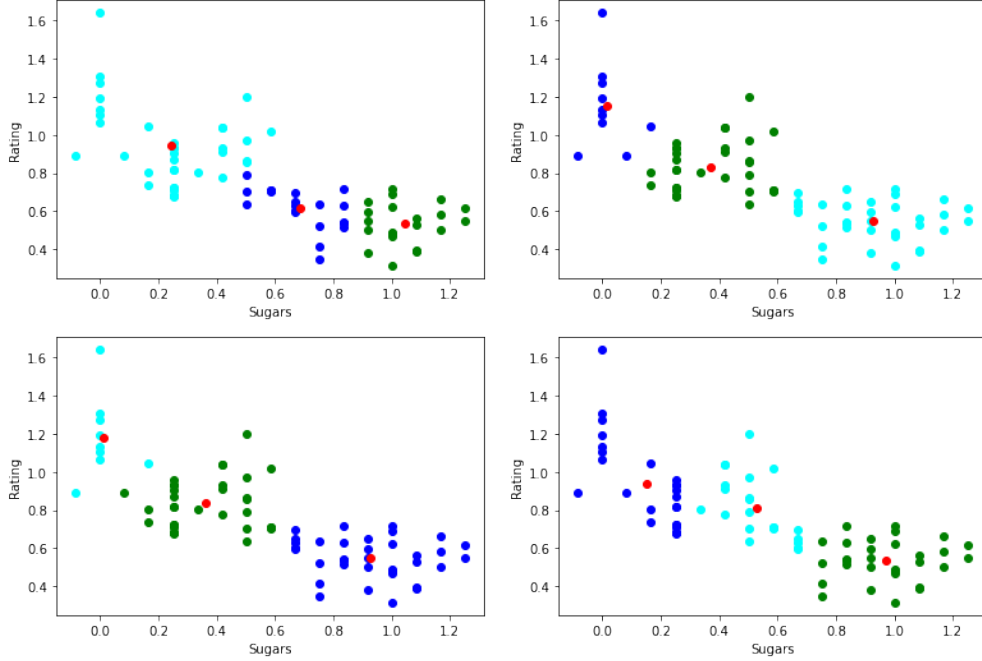


Figure 4.2: Each of the above plots is a clustering variation resultant of one iteration of k -means where $k = 3$ [11, 17, 9].

The observations that differ in classification can be examined to see how stable they are in comparison to others. To do this, I run k -means on the 76 observations plotted in two dimensions by sugars and rating t times. After each iteration of k -means, I create a $n \times n$ binary matrix which records a 1 if two observations are in the same cluster and a 0 if they are not. These $n \times n$ matrices are then summed and averaged such that we are left with one averaged matrix τ , where each entry (i, j) is a value between 0 and 1. Applying the following function then separates the observations by clustering consistency.

$$f(\tau) = \tau(1 - \tau)^{k-1} \frac{k^k}{(k - 1)^{k-1}} \quad (4.1)$$

Below I have plotted the stability function, $f(\tau)$, at different values of k . Note that when an entry (i, j) in the averaged matrix is 0 or 1, observations i and j are either consistently not in the same cluster or consistently in the same clustering respectively. As a result, the stability function transforms stable entries to 0 and unstable entries to 1.

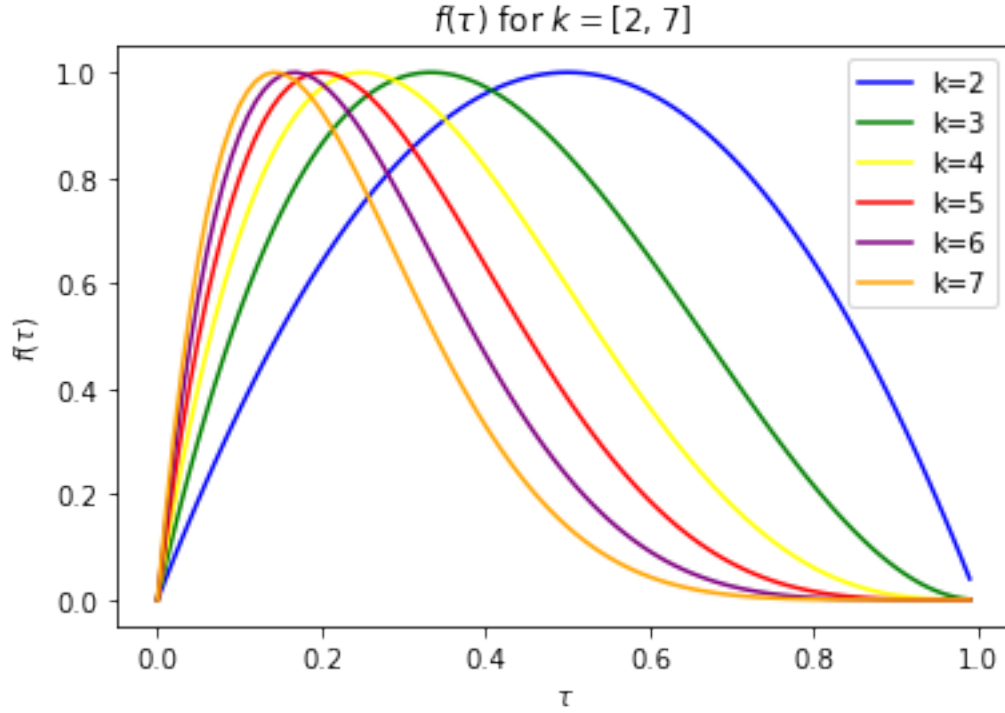


Figure 4.3: Entries (i, j) that are consistently clustered are pushed to 0 while entries that are inconsistently clustering are pushed to 1 [11, 9].

The transformed matrix can then be visualized on a heat map where a black square indicates that an observation is clustered consistently and a light beige square indicates that an observation is clustered inconsistently. Below I have plotted the transformed matrix onto a heat map and reordered the entries on the vertical axis to group entries that reflect consistency and inconsistency together.

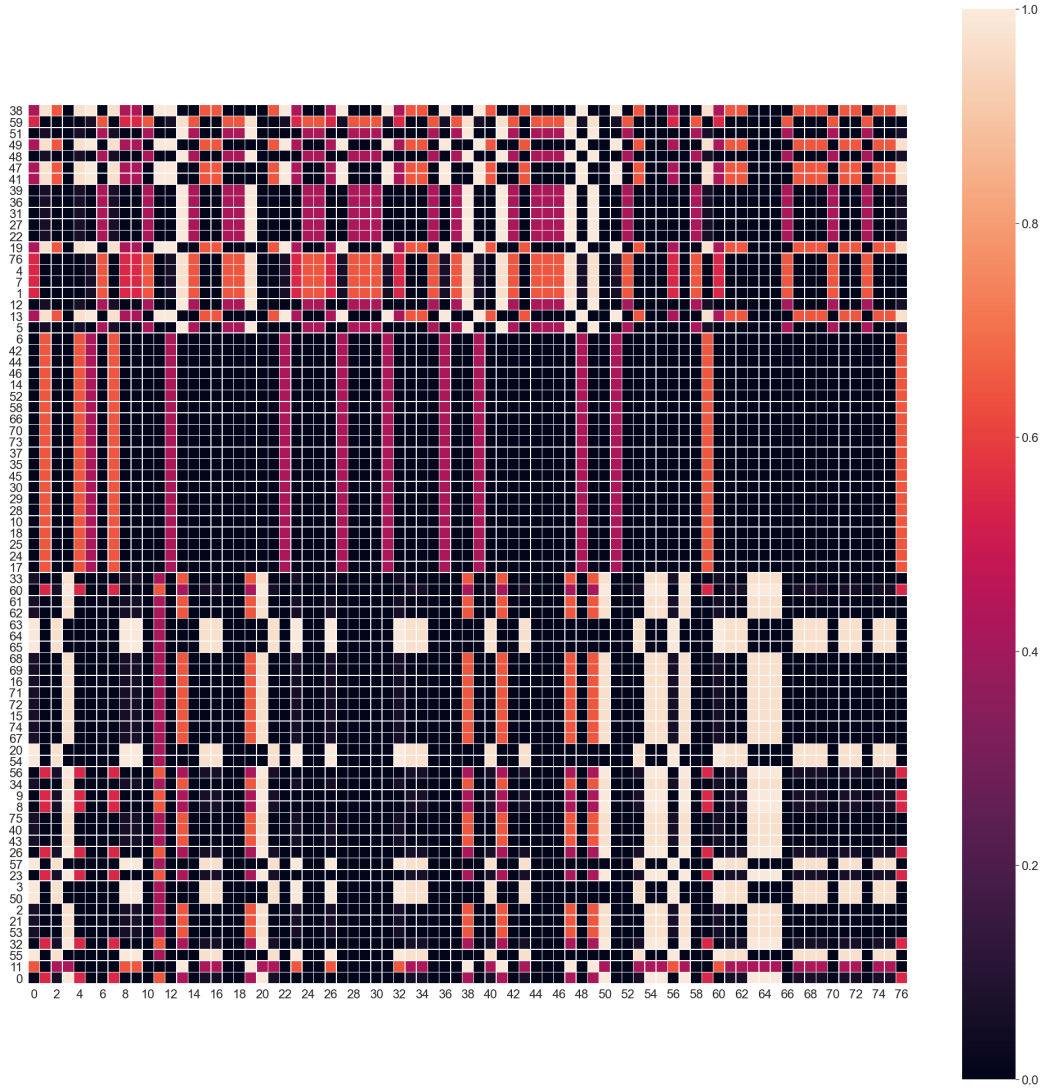


Figure 4.4: The observations on the vertical axis have been reordered to group entries by consistency [9, 23, 11].

We can then map the transformed matrix to a weighted graph where the edge weight of edge (i, j) corresponds to the entry (i, j) in the transformed matrix.

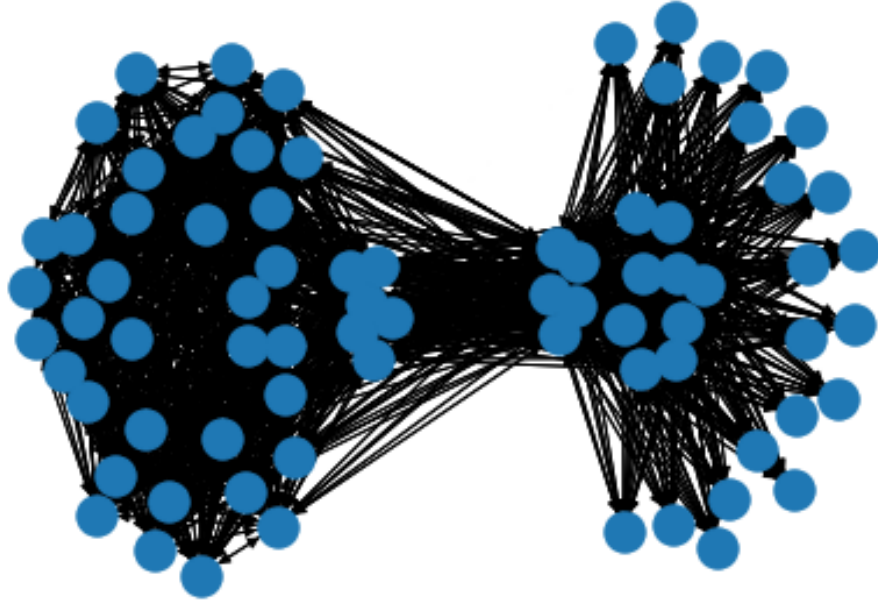


Figure 4.5: The edge weight and vertex labels have been omitted here for visual clarity [8].

As stated previously, every weighted graph has a single-linkage clustering. Thus, we can take the weighted graph given in Figure 4.5 and build the single-linkage dendrogram displayed below [11, 22].

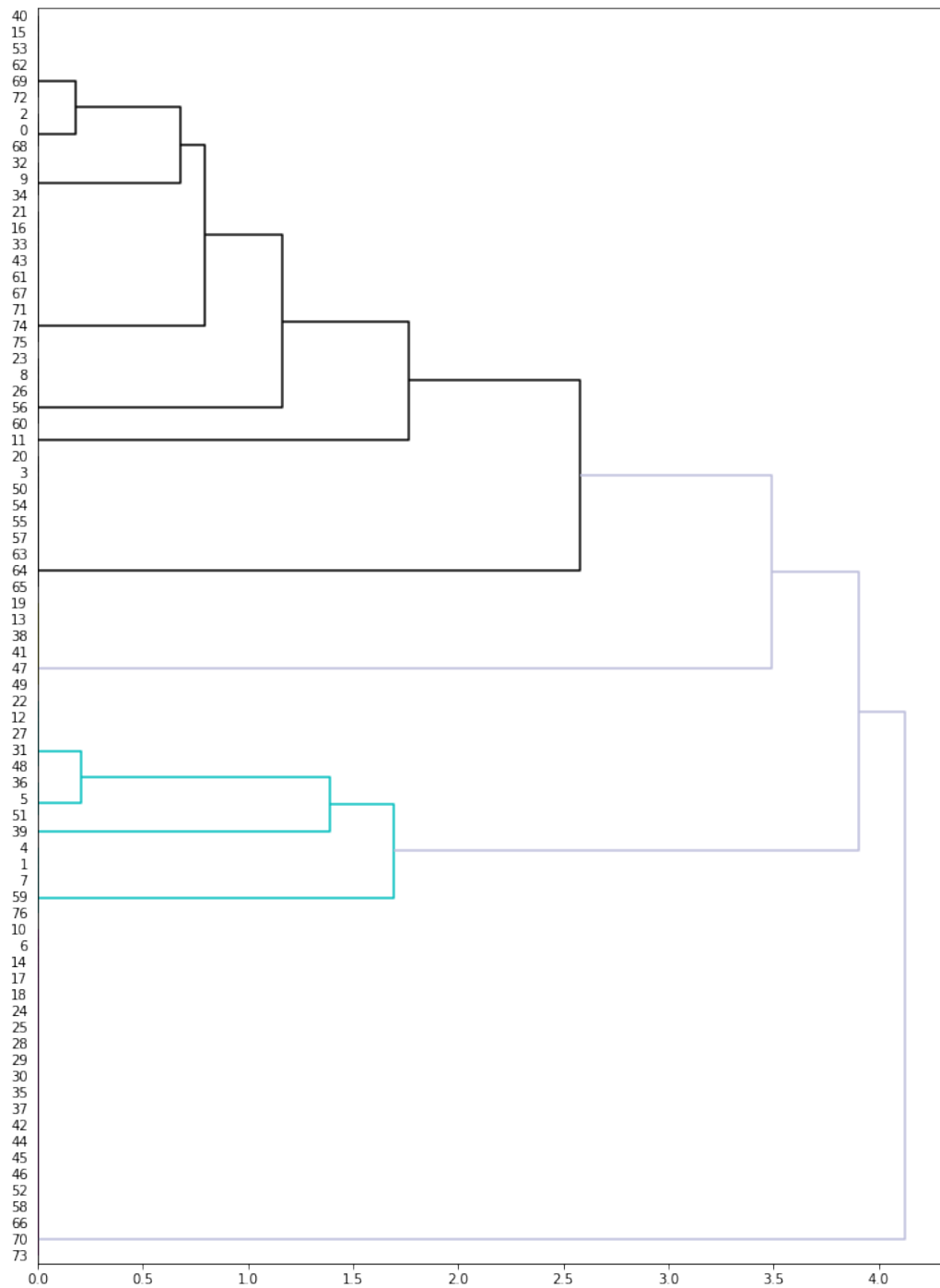


Figure 4.6: This single-linkage dendrogram is constructed from the weighted graph displayed in Figure 4.5.

Displayed in the dendrogram above, it appears that there are underlying clusters in the weighted graph (Figure 4.5). This lends motivation to further exploration into what clusters in the transformed stability matrix represent.

In summary, we can take a data set and identify which observations are more stable than others. We do this by calculating an averaged binary matrix over several iterations of k -means clustering then apply the stability function $f(\tau)$. We can then visualize which clustering relationships are more stable than others in a heat map and a weighted graph. Lastly, we can perform clustering on the weighted graph using the transformed stability matrix as input. From here, many clustering algorithms, not exclusive to SLC and k -means, can be considered to explore the question of interest.

Chapter 5

Conclusion

Despite the fact that I do not arrive on an answer to the question I pose in my further exploration, it is encouraging that connections exist to say topics in graph theory and statistics are more similar than they seem. I explored minimum spanning trees and single-linkage clustering separately and eventually found them to be equivalent. This lead me to explore k -means clustering to see what connections could be found between the three topics. I end here with potentially another exciting question for exploration. *Does a clustering algorithm exist which takes a weighted graph to say if two points are likely to be in the same clustering or not?*

Acknowledgements

I would like to thank Professor Vin de Silva for all his help in constructing this senior thesis. His knowledge and understanding of mathematics have not only helped me understand various mathematical topics but also have helped me in becoming a better writer and speaker. I would also like to thank Professor Ghassan Sarkis, Jo Hardin, and Sarah Cannon (CMC) for their advice, mentorship, and encouragement in research endeavors during my time at Pomona College.

Bibliography

- [1] Revindra K Ahuja, K Mehlhorn, and JB Orlin. Re tar jan,”. *Faster Algorithms for the Shortest Path Problem*,” *JACM*, 37:213–223, 1990.
- [2] Huynh Thi Thanh Binh, Pham Dinh Thanh, and Ta Bao Thang. New approach to solving the clustered shortest-path tree problem based on reducing the search space of evolutionary algorithm. *Knowledge-Based Systems*, 180:12–25, 2019.
- [3] O Borůvka. O jistém problému minimálním,(about a certain minimal problem) práce moravské přírodovědecké společnosti v brně (Acta Societ. Scienc. Natur. Moraviae), 3 (3), 37–58, 1926.
- [4] Chris Crawford. 80 cereals: Nutrition data on 80 cereal products, 2018. Available from <https://www.kaggle.com/crawford/80-cereals>.
- [5] Andrei Eremeev. The spanning tree based approach for solving the shortest path problem in social graphs. 2016.
- [6] Brian Everitt and Anders Skron dal. *The Cambridge dictionary of statistics*, volume 106. Cambridge University Press Cambridge, 2002.
- [7] Andrew Goldberg and Tomasz Radzik. A heuristic improvement of the Bellman–Ford algorithm. Technical report, Stanford University, Department of Computer Science, 1993.
- [8] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

- [9] Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Pícus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357—362, 2020.
- [10] MR Hassan. An efficient method to solve least-cost minimum spanning tree (LC-MST) problem. *Journal of King Saud University-Computer and Information Sciences*, 24(2):101–105, 2012.
- [11] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science Engineering*, 9(3):90–95, 2007.
- [12] David R Karger, Philip N Klein, and Robert E Tarjan. A randomized linear-time algorithm to find minimum spanning trees. *Journal of the ACM (JACM)*, 42(2):321–328, 1995.
- [13] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90. IOS Press, 2016.
- [14] Trupti M Kodinariya and Prashant R Makwana. Review on determining number of cluster in k -means clustering. *International Journal*, 1(6):90–95, 2013.
- [15] Joseph B Kruskal. J., 1956. on the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, pages 48–50.
- [16] Jiří Matoušek. *Thirty-three miniatures: Mathematical and Algorithmic applications of Linear Algebra*. American Mathematical Society Providence, RI, 2010.

- [17] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.
- [18] Seth Pettie and Vijaya Ramachandran. A shortest path algorithm for real-weighted undirected graphs. *SIAM Journal on Computing*, 34(6):1398–1431, 2005.
- [19] Duc Truong Pham, Stefan S Dimov, and Chi D Nguyen. Selection of k in k -means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 219(1):103–119, 2005.
- [20] Robert Clay Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401, 1957.
- [21] Shahriar Shahriari. Retro linear algebra. unpublished, 2018.
- [22] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.
- [23] Michael Waskom, Olga Botvinnik, Drew O’Kane, Paul Hobson, Saulius Lukauskas, David C Gemperline, Tom Augspurger, Yaroslav Halchenko, John B. Cole, Jordi Warmerhoven, Julian de Ruiter, Cameron Pye, Stephan Hoyer, Jake Vanderplas, Santi Villalba, Gero Kunter, Eric Quintero, Pete Bachant, Marcel Martin, Kyle Meyer, Alistair Miles, Yoav Ram, Tal Yarkoni, Mike Lee Williams, Constantine Evans, Clark Fitzgerald, Brian, Chris Fonnesbeck, Antony Lee, and Adel Qalieh. mwaskom/seaborn: v0.8.1 (September 2017), September 2017.

- [24] Meichen Yu, Arjan Hillebrand, Prejaas Tewarie, Jil Meier, Bob van Dijk, Piet Van Mieghem, and Cornelis Jan Stam. Hierarchical clustering in minimum spanning trees. *Chaos: An Interdisciplinary Journal of Non-linear Science*, 25(2):023107, 2015.
- [25] Chunhui Yuan and Haitao Yang. Research on k -value selection method of k -means clustering algorithm. *J—Multidisciplinary Scientific Journal*, 2(2):226–235, 2019.