ESKISEHIR OSMANGAZI UNIVERSITY ELECTRICAL AND ELECTRONICS ENGINEERING 2021-2022 OBJECT ORIENTED PROGRAMMING

K-MEANS

RAMAZAN AKKULAK, 151220162117

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Clustering is used to come to a conclusion about the structure of the data. It tries to identify subgroups in the data so that the data points in the same subgroup (cluster) are very similar, while the data points in different clusters are very different. Basically, we try to find homogeneous subgroups within the data so that the data points in each cluster are as similar as possible based on a similarity measure such as Euclidean-based distance or correlation-based distance.

INTRODUCTION

Clustering is the grouping of data showing similar characteristics in a data set. Within the same cluster, the similarities are high, and the similarities between the clusters are low. Known as Unsupervised Learning, the algorithm is not given any prior knowledge. It is the clustering algorithm commonly used in K-Means and Hierarchical Segmentation. K-Means algorithm is an unsupervised learning and clustering algorithm. The K value in K-Means determines the number of clusters and should take this value as a parameter. This is actually a disadvantage. The algorithm has a simple way of working.

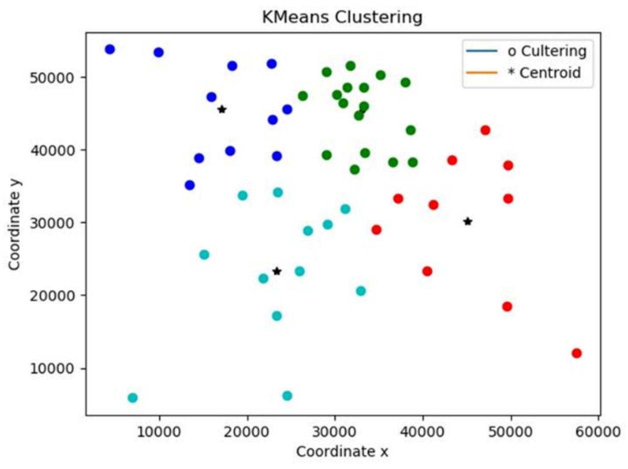

Figure 1. *K-Means Clustering*

After the K value is determined, the algorithm randomly selects K center points. It calculates the distance between each data and randomly determined center points and assigns the data to a cluster according to the nearest center point. Then, a center point is selected again for each cluster and clustering is done according to the new center points. This situation continues until the system becomes stable. The result of this situation is shown in figure 1.

In 2007, David Arthur and Sergei Vassilvitskii developed the K-Means++ algorithm, a variation of the K-Means algorithm, in order to better select the starting points [1]. After choosing a random starting point according to K-Means++, the distance between it and all other data is calculated. New starting points are selected by squaring this distance and making certain calculations. The complexity analysis of this operation is $0 \log k$.

1.1K-Means Algorithm

Each data is given as a dataset {x_1, x_2, …, x_N} as an n-dimensional real vector, and K as the number of clusters to divide. K-means clustering aims to partition N data into K pieces of S = {S_1, S_2, …, S_K} clusters in order to minimize the quadratic error. In other words,

$$\mu_i = \frac{1}{|S_j|} \sum_{x_i \epsilon S_j} x_i \tag{1}$$

where μ_i is the mean of the points in S_j

$$\underset{S}{arg\ min} \sum_{j=1}^{K} \sum_{x_i \epsilon S_j} ||x_i - \mu_i\ ||^2 \tag{2}$$

is to find.

1.1.1 How does the K-Means algorithm work?

The Kmeans algorithm is an iterative algorithm that tries to divide the dataset into different Kpre-defined non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the within-cluster data points as similar as possible, while trying to keep the clusters as distinct (away) as possible. Assigns data points to a cluster so that the sum of the squared distance between the data points and the cluster's centroid (the arithmetic mean of all data points belonging to that cluster) is minimum. The less variation we have within clusters, the more homogeneous (similar) the data points within the same cluster.

According to the working mechanism of the K-means algorithm, K objects are randomly selected to represent the center point or mean of each cluster. The remaining objects are included in the clusters with which they are most similar, taking into account their distance from the mean values of the clusters. Then, by calculating the average value of each cluster, new cluster centers are determined and the distances of the objects to the center are examined again. The algorithm continues to repeat until there is no change.

The algorithm basically consists of 4 stages:

1. Determination of cluster centers
2. Clustering the data outside the center according to their distance
3. Determination of new centers according to the clustering (or shifting of old centers to the new center)
4. Repeating steps 2 and 3 until stable state is reached.

The approach that Kmeans takes to solve the problem is called Expectation-Maximization. The e-step assigns the data points to the nearest cluster. Step Eqn 5 calculates the centroid of each cluster.

The objective function is:

$$J = \sum_{i=1}^{m}\sum_{k=1}^{K} W_{ik}\left\|x^i - \mu_k\right\|^2 \tag{3}$$

where W_{ik}=1 for data point x$_i$ if it belongs to cluster k; otherwise, W_{ik}= 0. Also, μ_k is the centroid of x$_i$'s cluster.

This process should do two-part minimization. Keeping $\mu_k$and W_{ik} constant, we reduce the ratio. We then update the W_{ik} and μ_k assignments. After μ_k and W_{ik} are calculated, step Eqn 5 is performed again.

$$\begin{gathered}\frac{\partial J}{\partial W_{ik}} = \sum_{i=1}^{m}\sum_{k=1}^{K} W_{ik}\left\|x^i - \mu_k\right\|^2 \\ \rightarrow W_{ik} = \begin{cases}1 \ if\ k = argmin g_j \left\|x^i - \mu_k\right\|^2 \\ 0 \qquad\qquad otherwise\end{cases}\end{gathered} \tag{4}$$

In other words, assign the x$_i$ data point to the nearest cluster evaluated by the sum of the square of its distance from the center of the cluster.

$$\begin{gathered}\frac{\partial J}{\partial \mu_k} = 2\sum_{i=1}^{m} W_{ik}\left(x^i - \mu_k\right) = 0 \\ \rightarrow\ \mu_k = \frac{\sum_{i=1}^{m} W_{ik}x^i}{\sum_{i=1}^{m} W_{ik}}\end{gathered} \tag{5}$$

Which translates to recomputing the centroid of each cluster to reflect the new assignments. Here are a few things to note:

1. Because clustering algorithms involving Kmeans use distance-based metrics to determine similarity between data points, it is necessary to standardize the data with a mean of zero and a standard deviation of one, since features in any dataset will almost always have different units of measure.

2. Given the iterative nature of kmeans and the random initialization of centroids at the start of the algorithm, different initializations may result in different clusters, as the kmeans algorithm may get stuck at a local optimum and not converge to the global optimum. Therefore, it should be made to adjust the algorithm using different centroid and give lower squared distance sum.

$$\frac{1}{m_k}\sum_{i=1}^{m_k}\left\|x^i - \mu_c^k\right\|^2 \tag{6}$$

Algorithm 1 *k-means algorithm*

```
1: Specif the number k of clusters to assign.
2: Randomly initialize k centroids.
3: **repeat**
4:     **expectation:** Assign each point to its closest centroid
5:     **minimization:** Compute the new centoid mean of each cluster.
6: **until** The centroid positions to not change.
```

Table 1. *K-Means Algorithm*

2.1 Prim Graph Algorihm

Prim's algorithm is a Greedy algorithm. It starts with an empty spanning tree. The idea is to preserve the two sets of vertices. The first set of Minimum Spanning Tree (MST) contains vertices already found, the other set contains vertices that have not yet been included. At each step, it considers all the edges connecting the two sets and chooses the minimum weight edge from these edges. After selecting the edge, it moves the other endpoint of the edge to the cluster containing the MST.

A group of edges connecting two vertices in a graph is called a segment in graph theory. So, at each step of Prim's algorithm, we find a cut (one contains the vertices already in the MST and the other contains the remaining vertices), select the minimum weight edge from the cut, and the operations are performed by including this vertex in the MST Set. containing set).

2.1.1 How does the Prim Graph algorithm work?

The idea behind Prim's algorithm is simple, a spanning tree means all vertices must be connected. Therefore, two disjoint subsets of vertices (discussed above) must be connected to make a Spanning Tree. And to make a Minimum Spanning Tree they must be tied with a minimum weight edge.

2.1.2 Prim Graph Algorithm

1. A set of mstSet of the minutes of the peaks already found in the MstSet.
2. Assign a key value to all vertices in the value in the input. All keys as INFINITE. Assign 0 as the key for the first vertex, select it first.
3. Although mstSet does not contain all vertices
   a) A vertex value option u has no mstSet and a minimum key.
   b) u include for mstSet.
   c) Update keys of all universal vertices of u. Iterate through all the vertices to update the keys. If v is kilo for each neighbor, UV is older key value v, change UV to refresh life for now

The idea of a solution for switches is cutting to be minimal. Used for vertices whose keys are not yet included in the MST, the value of these vertices is used, specifying their minimum weight to the set of vertices not included in the MST. Algorithm similar results are below as Figure 2.

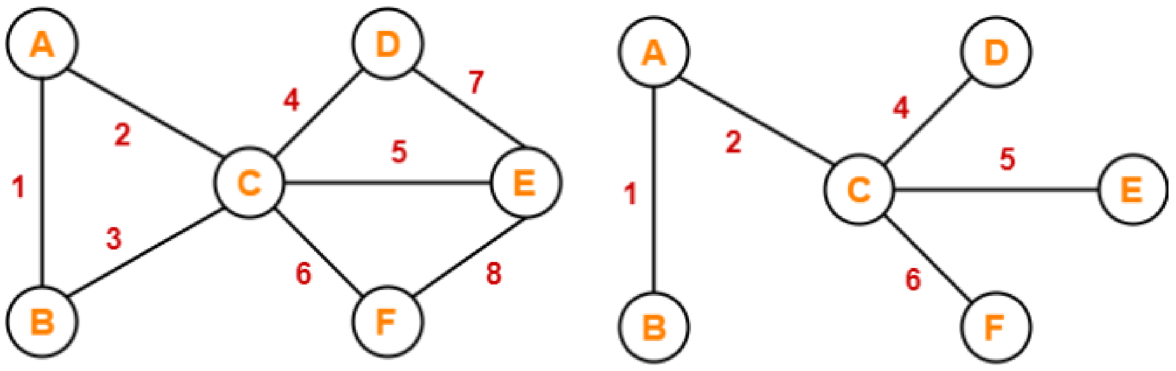

Figure 2: *Prim Graph Algorithm Example. (a) Given Graph, (b) Minimum Spanning Tree (MST)*

3.1 The Coding Hierarchy and Running

The K Means algorithm I designed consists of five classes and one inherit class. These classes are as follows:

1. Node Class (base)
2. Edge Class (base)
3. Cluster Class (base)
4. K-Means Class (base)
5. Graph Class (base)
6. Prim Class (inherit)
7. MatPlotLib.cpp

The study was developed on VS code ide. There is also the matplotlib.cpp function developed for plotting over the matplotlib library developed for python. The installation of this file is simple. This file can be accessed via github specified in table 2 step 1[3]. This should be copied to the part where the project files are.The following guidelines should be followed to run this run.

(1)	https://github.com/lava/matplotlib-cpp
(2)	sudo apt-get install python-matplotlib python**X**-numpy python**X**.**X**-dev
(3)	g++ example.cpp -I/usr/include/python**X**.**X** -lpython**X**.**X**

Table 2. *MatPlotLib setups steps. (1)Github link (2) Python lib setup, (3) Example run*

The above 2 steps are performed. Before this process is performed, the python version installed on the Linux (ubuntu) operating system is important in places marked as **X** in bold. The work was run on python3.8. Then run one of the examples in the file installed from github to test the accuracy of this process.

After doing this process, you can go to the directory where you are working via the terminal on the visual code or via the ubuntu terminal with the help of the "cd" command. After navigating to the required directory:

```
g++ main.cpp Node.cpp Edge.cpp Cluster.cpp KMeans.cpp Graph.cpp Prim.cpp -
std=c++11 -I/usr/include/python3.8 -lpython3.8
```

Table 3. Terminal complier CMake

This command is typed into the terminal to view the results. The video of the study is given in [4].

```
./a.out
```

Table 4. *Output command*

4.1 Result

The results of the study are as follows.

4.1.1 Cluster = 4, Iteration = 10, K-Means Clustering – File = 50.txt

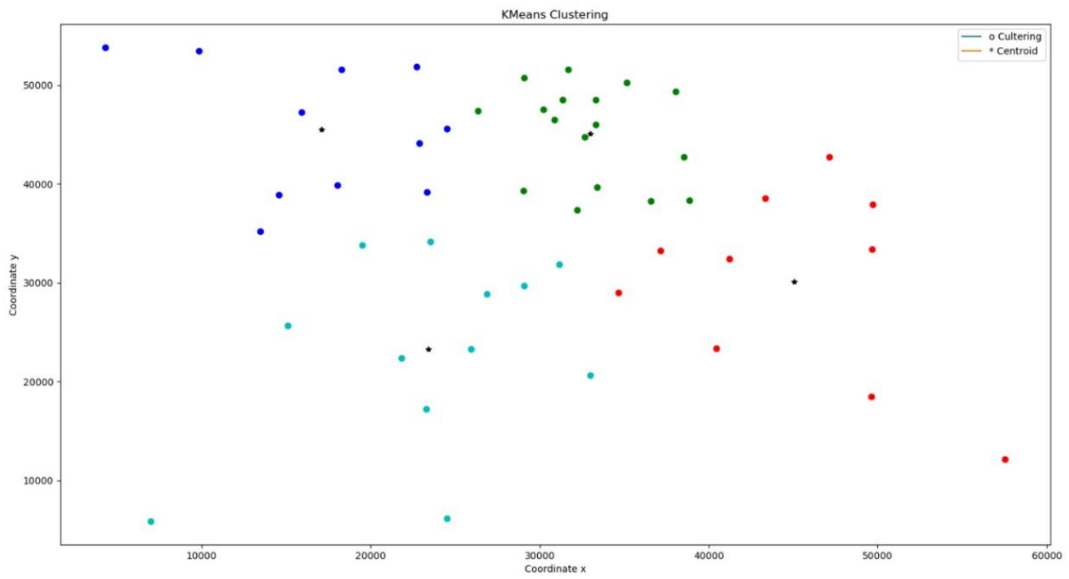

Figure 3. *Cluster = 4, Iteration = 10, K-Means Clustering – File = 50.txt*

4.1.2 Cluster = 4, Iteration = 10, Prim Graph – File = 50.txt

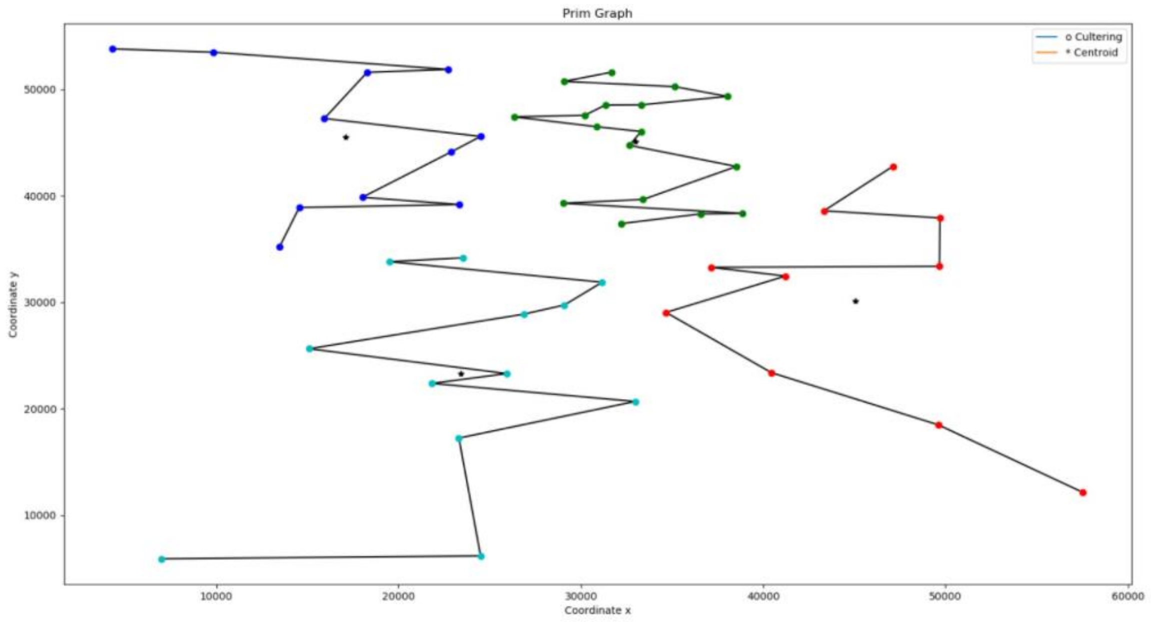

Figure 4. *Cluster = 4, Iteration = 10, Prim Graph – File = 50.txt*

4.1.3 Cluster = 3, Iteration = 20, K-Means Clustering – File = 50.txt

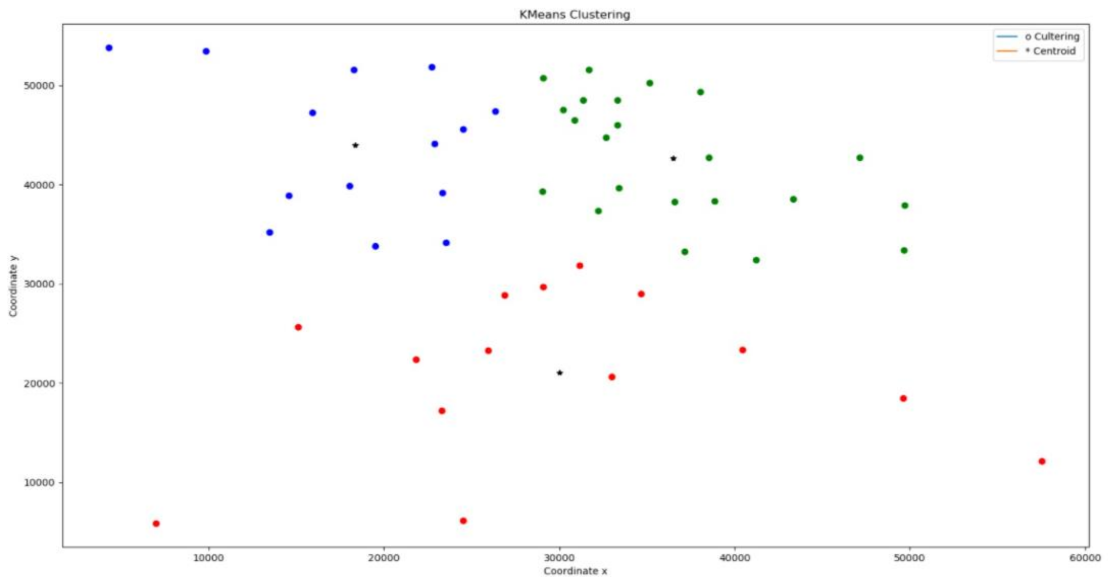

Figure 5. *Cluster = 3, Iteration = 20, K-Means Clustering – File = 50.txt*

4.1.4 Cluster = 3, Iteration = 20, Prim Graph – File = 50.txt

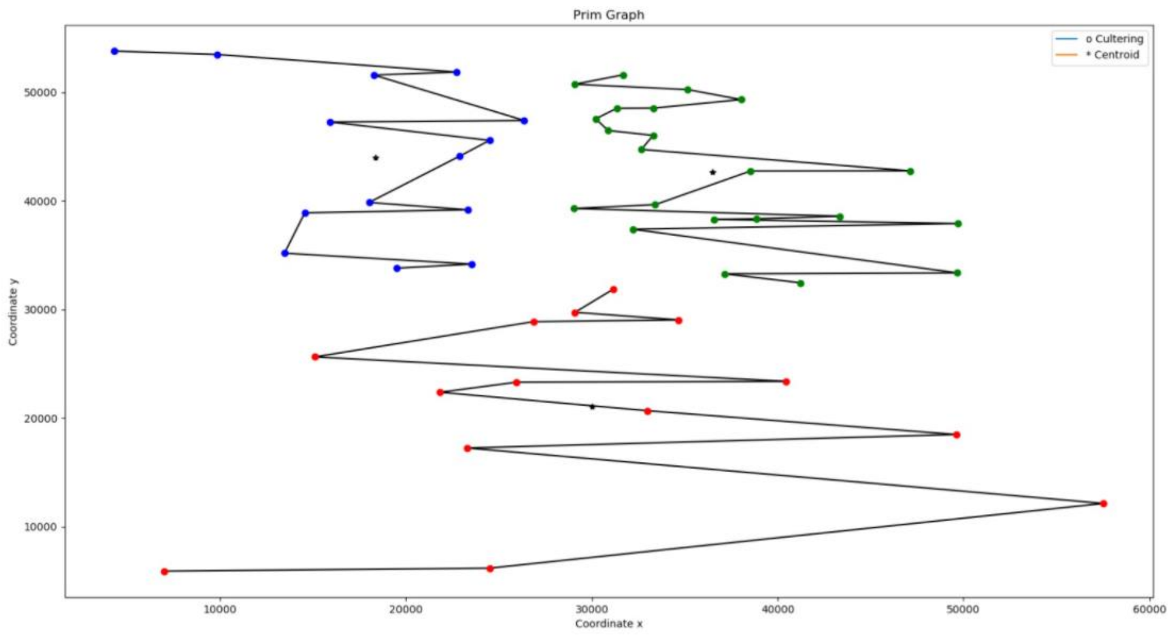

Figure 6. *Cluster = 3, Iteration = 20, Prim Graph – File = 50.txt*

4.1.5 Cluster = 5, Iteration = 50, K-Means Clustering – File = 50.txt

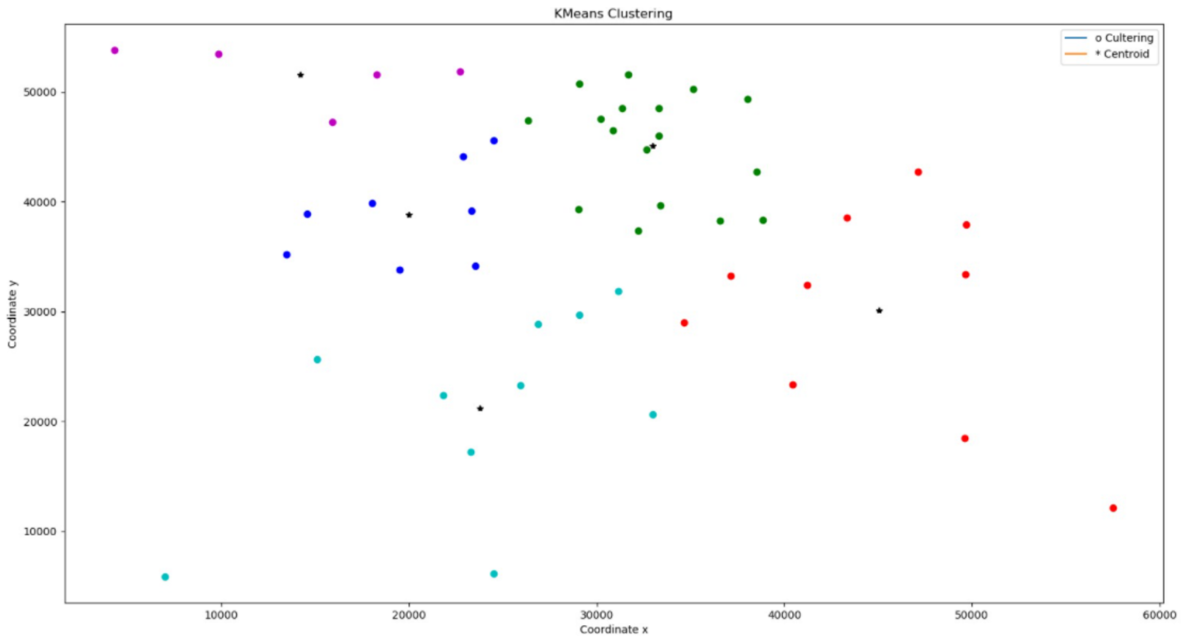

Figure 7. *Cluster = 5, Iteration = 50, K-Means Clustering – File = 50.txt*

4.1.6 Cluster = 5, Iteration = 50, Prim Graph – File = 50.txt

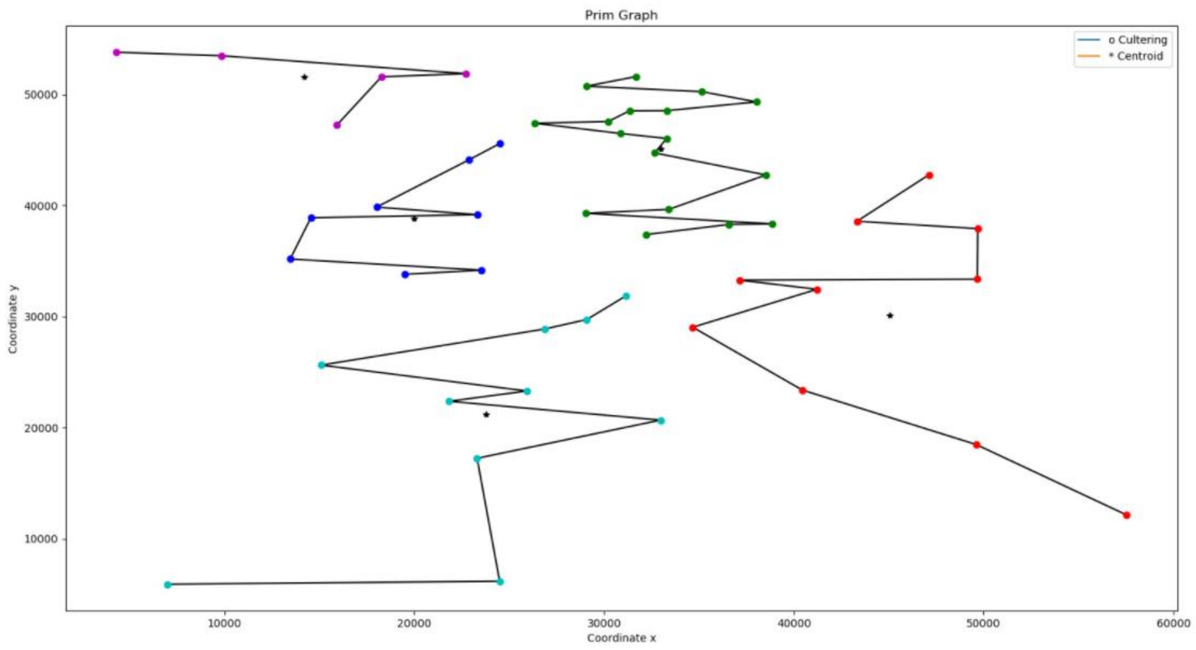

Figure 8. *Cluster = 5, Iteration = 50, Prim Graph – File = 50.txt*

4.1.7 Cluster = 5, Iteration = 50, K-Means Clustering – File = 20.txt

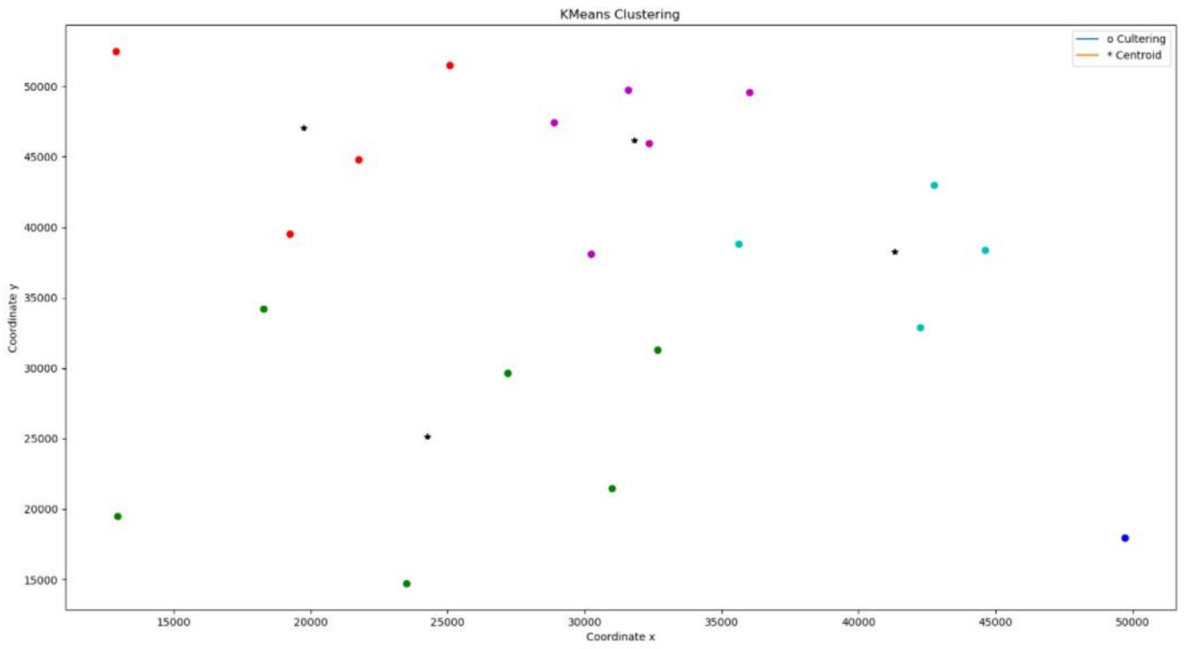

Figure 9. *Cluster = 5, Iteration = 50, K-Means Clustering – File = 20.txt*

4.1.8 Cluster = 5, Iteration = 50, K-Means Clustering – File = 20.txt

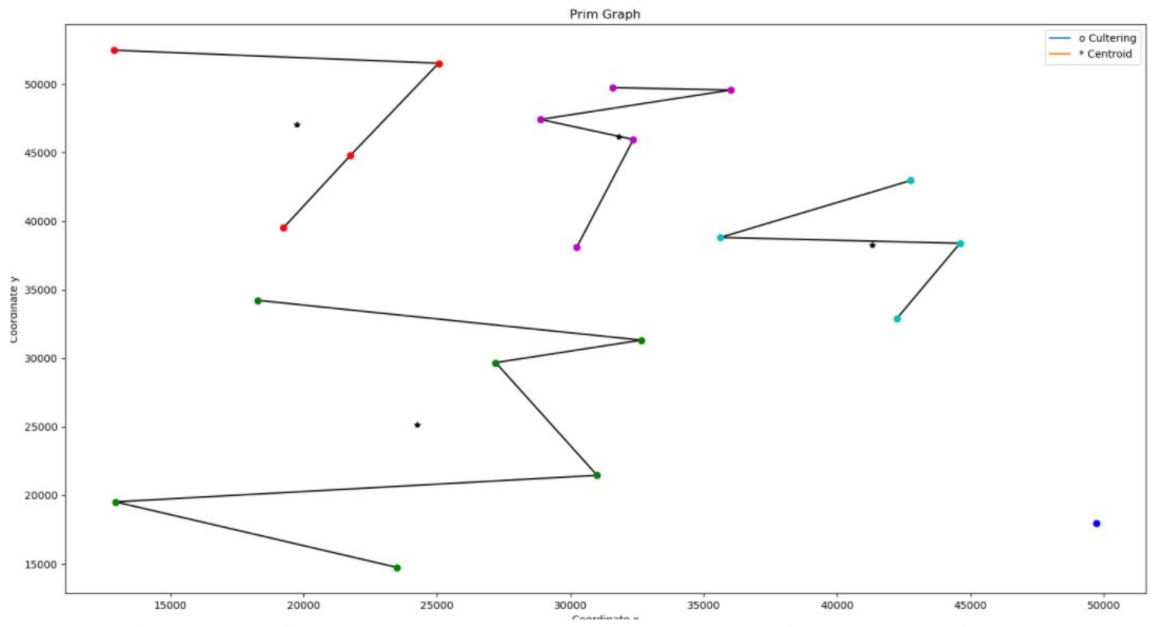

Figure 10. *Cluster = 5, Iteration = 50, K-Means Clustering – File = 20.txt*

4.1.9 Cluster = 2, Iteration = 100, K-Means Clustering – File = 20.txt

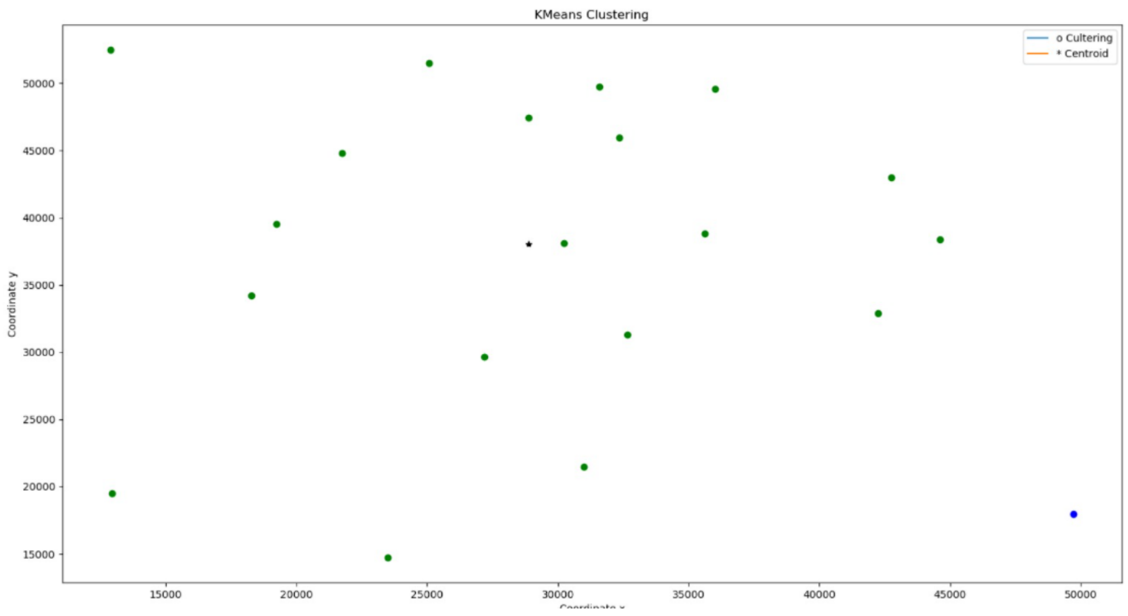

Figure 11. *Cluster = 2, Iteration = 100, K-Means Clustering – File = 20.txt*

4.1.10 Cluster = 2, Iteration = 100, Prim Graph – File = 20.txt

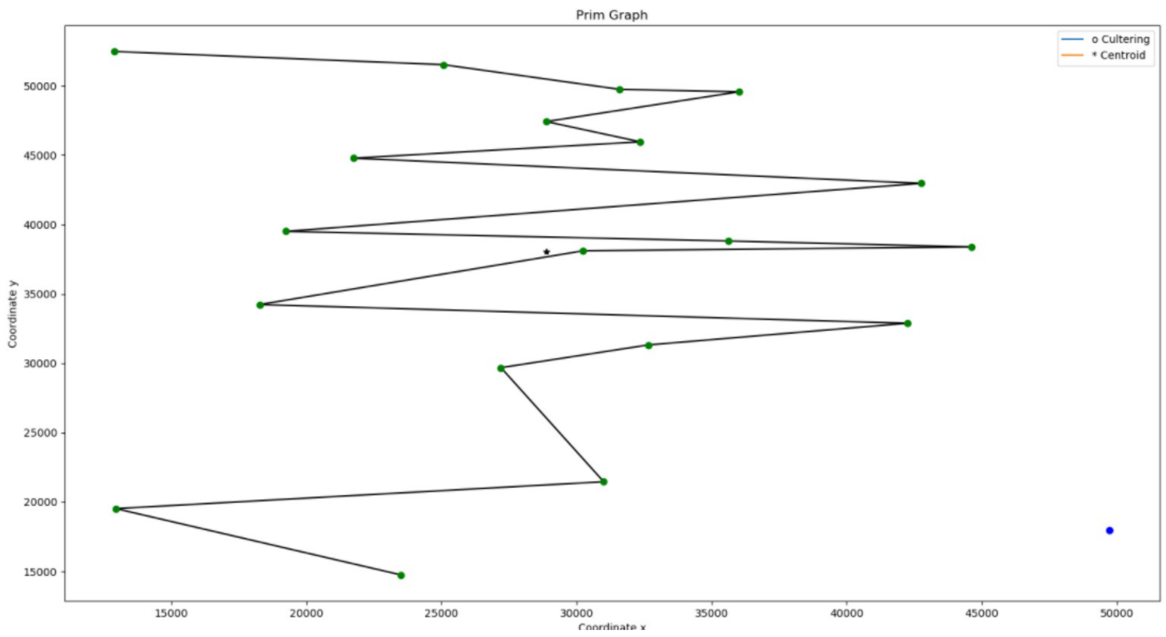

Figure 12. *Cluster = 2, Iteration = 100, Prim Graph – File = 20.txt*

4.1.11 Cluster = 2, Iteration = 100, K-Means Clustering – File = 50.txt

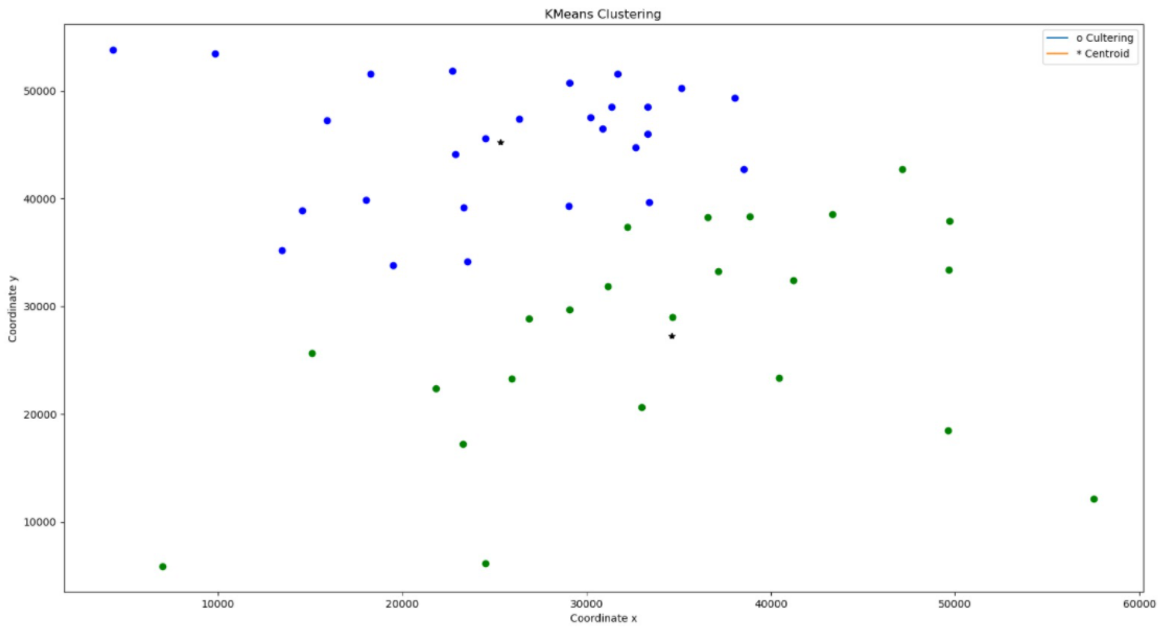

Figure 13. *Cluster = 2, Iteration = 100, K-Means Clustering – File = 50.txt*

4.1.12 Cluster = 2, Iteration = 100, Prim Graph – File = 50.txt

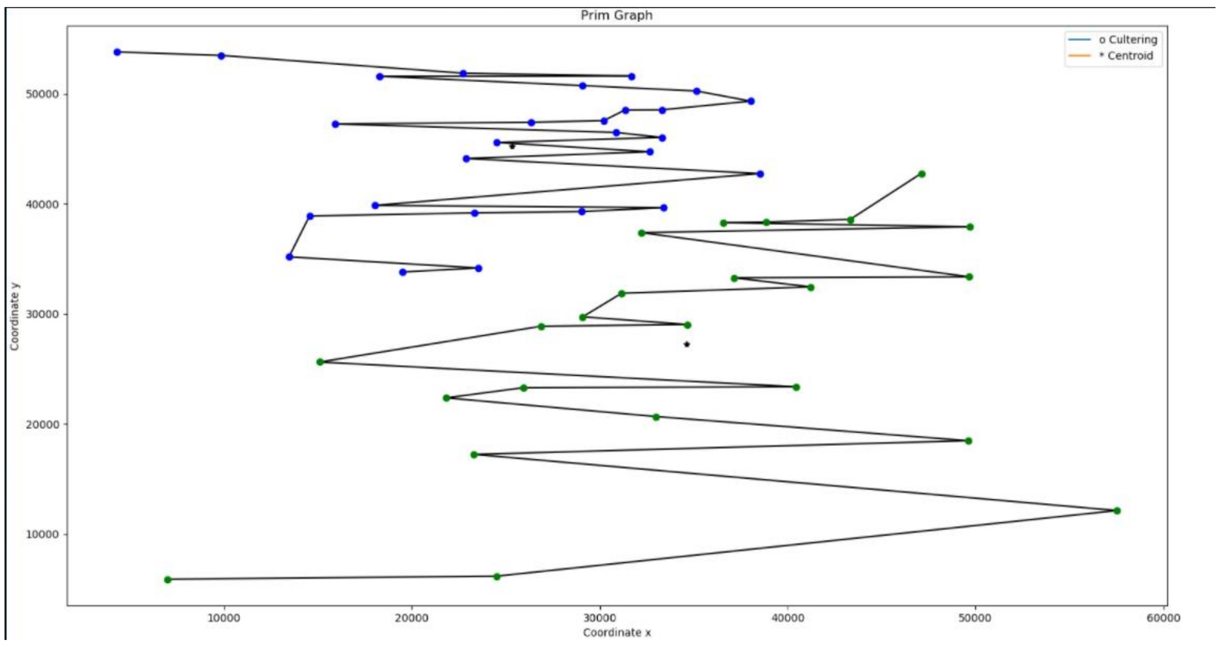

Figure 14. *Cluster = 2, Iteration = 100, Prim Graph – File = 50.txt*

5.1 References

[1] Arthur, D., & Vassilvitskii, S. (2006). *k-means++: The advantages of careful seeding*. Stanford.

[2] Zhong, C., Malinen, M., Miao, D., & Fränti, P. (2015). A fast minimum spanning tree algorithm based on K-means. *Information Sciences*, *295*, 1-17.

[3] https://github.com/lava/matplotlib-cpp

[4] https://youtu.be/PxEkcAIync0