LEBANESE AMERICAN UNIVERSITY

DG-Means – A Superior Greedy Algorithm for Clustering Distributed Data

By

Ali Assaf

A thesis Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

School of Arts and Sciences

July 2022

الجامعة اللبنانية الأميركية
Lebanese American University

THESIS APPROVAL FORM

Student Name: Ali Assaf I.D. #: 202000822

Thesis Title: DG-Means – A Superior Greedy Algorithm for ClusteringDistributed Data

Program: Computer Science

Department: Computer Science and Mathema ics

School: School of Arts and Sciences

The undersigned certify that they have examined the final electronic copy of this thesis and approved it in Partial Fulfillment of the requirements for the degree of:

Master of Science in the major of Computer Science

Thesis Advisor's Name: Ramzi A. Haraty

Signature: [redacted] Date: 26/ / 7 / 2022
Day Month Year

Committee Member's Name: Samer Habre

Signature: [redacted] Date: 26 / 7 / 2022
Day Month Year

Committee Member's Name: Sanaa Kaddoura

Signature: [redacted] Date: 26 / 7 / 2022
Day Month Year

LAU
الجامعة اللبنانية الأميركية
Lebanese American University

THESIS COPYRIGHT RELEASE FORM

LEBANESE AMERICAN UNIVERSITY NON-EXCLUSIVE DISTRIBUTION LICENSE

Name: Ali Assaf

Signature: [redacted]

Date: 14 / 07 / 2022
Day Month Year

LAU
الجامعة اللبنانية الأميركية
Lebanese American University

PLAGIARISM POLICY COMPLIANCE STATEMENT

I certify that:

1. I have read and understood LAU's Plagiarism Policy.
2. I understand that failure to comply with this Policy can lead to academic and disciplinary actions against me.
3. This work is substantially my own, and to the extent that any part of this work is not my own I have indicated that by acknowledging its sources.

Name: Ali Assaf

Signature: [redacted]

Date: 14 / 07 / 2022
Day Month Year

DEDICATION

This thesis is dedicated to my loving wife, daughter and son who suffered a lot with me and spent long times away from me to reach this fruitful outcome.

ACKNOWLEDGMENT

This work would have not been achieved if was not backed by the assistance of many people. Tons of thanks for my advisor, Dr. Ramzi Haraty, who supported me throughout this long and exhausting process. Besides, I would like to appreciate my committee, Dr. Samer Habre and Dr. Sanaa Kaddoura, for their assistance and support.

At the end, I can’t forget the support and love that was offered by my wife, family, and friends, who endured this milestone with me.

DG-Means – A Superior Greedy Algorithm for Clustering Distributed Data

Ali Assaf

ABSTRACT

Clustering is the process of dividing a set of objects into several classes in which each class is composed of similar objects. Traditional centralized clustering algorithms target those objects that are located in the same site, whereas it cannot perform on distributed objects. Distributed clustering algorithms, however, can fulfil this gap. They extract a classification model from the distributed objects even when they are in different sites and locations. In today's life, and due to the trend of storing data on different locations and sites, the popularity of distributed data is getting tremendously booming. It seems to be one of the most prevailing fields in the coming decades, especially with the huge amount of data propagating throughout the web. Even though a lot of research and work was done on this topic, it is still considered in its infantry because of the challenges that is still popping up such as bandwidth limitation, transferring data to single site and many others. In this work, we present DG-means, which is a greedy algorithm that performs on distributed sets of data. Three datasets - Wholesale dataset, Banknotes dataset, and Iris dataset are used to compare multiple distributed clustering algorithms on different matrices: runtime execution, stability, and accuracy. DG-means exhibited superior performance when compared to the other algorithms.

Keywords: Clustering, *K*-means, Distributed clustering, *G*-means, Data mining.

Table of Contents

List of Tables

LIST OF FIGURES

Chapter 1

Introduction

The consumption of data and its services have dominated the technology era we live in today [1]. Old computing methods have grown increasingly hard to process the quickly expanding data sizes due to the massive volume of info and the hardness of repetitive computations in operation. As a result, developing a clustering method for platforms with distributed nature clusters has turned into a pressing issue [2, 3]. On the other hand, more dominating challenges are being imposed on data analysis and knowledge discovery. Algorithms are commonly used in data mining to uncover the underlying meaning hidden underneath the explicit aspects of large datasets [4]. Most existing big data systems are composed of distributed computing and storage components. The big data platform's computer resources are employed to facilitate big data analysis; the most typical technique is to parallelize the algorithm [5, 6]. As a natural process of classifying vague data, clustering may be employed in the preparational stage of data as well as in the process of data mining, whether it's classic data mining or analysis of data in environment of huge data. Cluster analysis, on the other hand, confronts several obstacles in the big data context. The environment of data sometimes triggers some of these issues, while others are caused by the clustering process. The capacity to work with varied kinds of data, massive high dimensional records, and row data; the efficiency of repetitive runs of clustering methods; the technique extendable ability, and the impact of the clustering model of assessment; and different concerns have arisen as a result of these hurdles. The clustering problem may be solved using k-means [7]. Thanks to its easy principle, the simple implementation, and the speed of convergence that is considered fast, the k-means technique of clustering has been widely utilized.

Nonetheless, there are several issues with original k-means: the arbitrary picking of the initial centers helps in putting the outputs of the clustering into an optimal local solution, resulting in unstable outcomes. In addition, for big data analysis of clustering, it is proven that the efficiency of implementation is not good. Given these shortcomings, huge research efforts have been numerous researchers have been implemented to enhance and develop the clustering algorithm of k-means. When it comes to mining techniques, clustering analysis is one of the most widely used ones [8]. To cluster data into numerous groups made up of similar units, and to implement the process of clustering, k-means utilize an unsupervised learning technique. There are some common clustering analysis methods like: clustering based on partition such as k-means[9], clustering with a hierarchical approach, for example Birch[10], and clustering using density-based such as DBSCAN[11]), and grid-based clustering as Sting[12]. The kind and amount of data are expanding as the big data era progresses, and the traditional clustering method with centralized approach failed to match customers' demands to bring accurate and efficient results when analyzing big data. In the subject of clustering algorithms, improving existing clustering algorithms and implementing distributed parallel computing has become a research center.

Talking about big data context, data has massiveness, sparseness, and high dimensionality, to name a few features. Furthermore, the distributed system-based big data processing platform provides ample computational and storage resources for enormous data processing. In the complicated big data environment, one of the hot topics that has emerged is the efficient employment of processing ability for parallelized and distributed computing setups to increase the impact of old mining methods and deliver more rapid analysis services of data.

The amount of data that may be streamed is theoretically infinite, and it is neither rational nor practical to keep all of it on a single system. As a result, the algorithm's computations for streaming information could as well be done online and in one pass, and they might as well need minimal memory for execution. The need for small memory is concentrated on portable devices or sensor system hubs [13]. We aim towards the issue of rapid techniques of clustering on huge datasets that are out of core in this work, leveraging single or small attempts through the complete container of the data, away from compromising the results. The approach we followed is based on k-means clustering but with a little twist.

Macqueen invented the k-means grouping algorithm in 1967, and Hartigan improved it subsequently [14]. Bottou and Bengio illustrated how k-means computations and Algorithms techniques might coexist [15]. It's been suggested that it's particularly useful for applications that are commonly used and a frame of sensible provisions. Although the technique followed by k-means method deals with data within memory, it might efficiently be broadened to treat datasets with occupants outside the memory. k-means has a main problem in computation that it does a single query through the whole dataset on each cycle, and it takes a lot of cycles to get a good result. Due to this, it became relatively expensive to adopt, especially with very big datasets located on local drives. Various computations or approaches are used to keep track of how to reduce the attempts needed by k-means. These approaches, on the other hand, only provide educated guesses to the way of the outcomes. One of the main advantages of k-means is that it fits a local minimum, which is not correct for approximated versions [16]. In this way, an important question arises: Can we have a calculating approach that requires fewer runs through the whole dataset while producing the same convergence outcomes as the core k-means algorithm?

1.1 Scope

k-means is a technique for clustering that use the unsupervised approach. It separates data with no labels into a predefined set (the "k") of different categories. This means that k-means can detect objects with similar attributes to cluster them together. A clustering model is said to be successful when it identifies clusters with similar observations within each cluster than the clusters themselves. There are several instances when automatic data grouping can be quite beneficial. The algorithm's intuition is actually rather simple. To begin, we select a value for k (the clusters' number) and then a centroid (central coordinates) for every single cluster at random. Then we iterate the following procedure:

- Assignment step — Assign each node to it's nearest centroid.
- Update step — Update the centroids to be the center of their allocated nodes.

We keep repeating these two steps until the clusters are no longer changing. The algorithm has now reached a point of convergence, and we may extract our final clusterings. The centroid assignment is one of the common challenges that we face while dealing with k-means. The initial assignment of centroid that is done by k-means is done randomly, and this is what may lead to a totally wrong clustering. Let us see the below figure and how should the logical clusters look like along with their centroids:

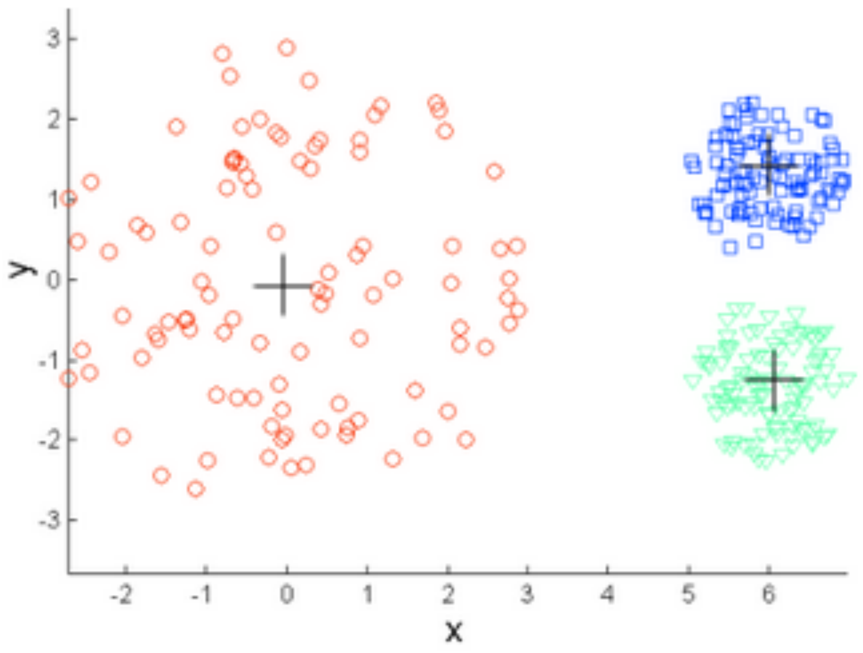

Figure 1 Ideal clustering with centroids

However, after randomly selecting the centroids, the algorithm might assign them as the following and end up in an impaired clustering (see figure 2):

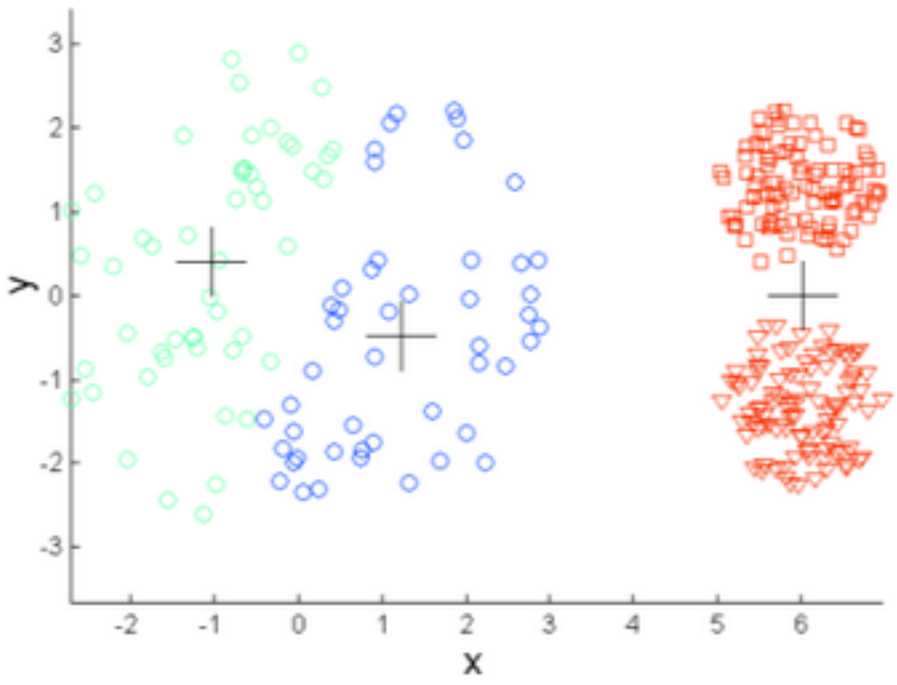

Figure 2 K-means clustering (k=3)

For that purpose, we rely on the G-means algorithm that contributes to an enhanced k-means algorithm that itself resorts to a greedy approach. This algorithm uses the same functionality as the original k-means but introduces a variation that takes into consideration the neighboring vertices. G-means initiates the centroids selection based on a greedy approach. The enhancement in this method will not impact the similarity functions used in the original k-means and will keep using them at a later stage, just like the typical k-means does. The complexity of G-means will remain the same as k-means, but the entropy, F-score, and overall running time will improve when the datasets become larger. G-means algorithm will be discussed in more detail in later sections of this study.

1.2 Contribution

The amount of information that individuals may access has expanded enormously as the internet has greatly developed. Among the topics that are currently booming in the theory of computer information is how to get knowledge from enormous volumes of data. Clustering has steadily gained considerable attention in recent years as an essential aspect of mining the data. One advantage of clustering is that it has the benefit of lacking previous knowledge, where information may be gathered depending on the normal distribution of data, as opposed to other data mining approaches [17]. Partition, density, stratification, grid, and model are all forms of clustering techniques. The data set may be divided into many groups using cluster analysis [18]. The k-means algorithm is well-suited to deal with vast volumes of data and high dimensions that are highly featured, and it has a low data reliance. As a result, k-means became a very popular technique of clustering [19]. However, *k*, which must be previously selected before being initiated, is decided solely by the expertise of the developer, which will have an impact on clustering efficiency and trustworthiness of the findings [20]. The initial random selection of centroids will affect the stability of results in clustering [21, 22]. Recently, and in order to perform data mining methods on distributed networks, an enormous amount of research on the level of big data was conducted. Another compelling reason was to enhance them to meet actual needs. As a result, to counter the selection of initial centroids with the functionality of distributed data, our algorithm was developed. In this work, we aim to upgrade the G-means algorithm to be applied to distributed datasets. For that, the DG-means algorithm will be introduced and extensively discussed throughout this report. The new algorithm will develop an improved version of G-means to be compatible with distributed sets, similar to many other techniques that target the trend of today, the big data.

Our contribution in this report is as follow:

- Use the modified centroid selection method using greedy approach.
- Implement this selection method on distributed data.
- Develop the new method using Spark technique.
- Prepare for wider scope for performing such methods on live data.

1.3 Organization of the Report

The report was organized as follows: Chapter 2 presents the background of k-means and G-means, where we will go in-depth with their explanation. Chapter 3 (Related work) puts further attention to G-means technicalities along with the discussion of another distributed algorithm (SOCCER), in addition to highlighting some other methods. The SOCCER algorithm was adopted for the huge benefits that it brought for our study especially when it comes to distributed datasets. Chapter 4 presents the DG-means algorithm and approach, providing an example and illustrating the complexity comparison. In chapter 5, experimental results will be demonstrated to better present the work done. At the end, chapter 6 will deal with the conclusion and future works.

Chapter 2

Background

2.1k-means Definition

In order to gain a well-developed comprehension of the structure of the data, clustering comes into the game as a famous tool for data exploration. This means that it is an exercise of extracting subgroups in a certain dataset in such a way that data points within the same group look like each other with a high dissimilarity with other points in another cluster. To express it in a different way, we try to pursue uniform clusters in the given dataset so that elements of each subgroup are extremely similar when examined on any metric of similarity, such as Euclidean-based distance. However, the application itself affects the decision of the metric of similarity to be utilized. The analysis of clustering might be practiced on either features or samples. The first is when the engine attempts to find clusters of samples, while the latter is when we need to cluster features based on samples. Unlike the supervised approach in learning, clustering follows an unsupervised technique. The absence of true values to be used for validation forced the algorithm to adopt such an approach. All that we want here is to observe the structure of the data by clustering the samples into different homogenous classes.

The method utilized by the *k*-means algorithm is the splitting of a dataset into a predefined number of clusters that are unique with no overlapping in their elements. The method tries to create different clusters with similar featured datapoints, while assuring the distinction of clusters to the maximum possible. It assigns elements for each cluster in a manner where the summation of their distances squared to the center is ultimately minimized. Then, in a cluster, the similarity of the datapoints reflects the low variance within these datapoints. That is, the relation between variance and similarity in each cluster is inversely proportional, the lower the variance means the higher the homogeneity. The following expresses how the *k*-means algorithm works:

- Assigns the number of clusters *k*.
- Set initial values for centers by iterating the dataset and then selecting random k points for the centroids.
- Keep performing until no change exists for the centroids.
- Do the calculation of the sum of the distance between all the points and the corresponding centroids.
- Allocate each datapoint to the nearest centroid.
- Calculate the average of all datapoints that belong to a certain subgroup and recompute the centroids.

Since of the nature of the *k*-means method which is iterative, and the arbitrary selection of centroids at the beginning of the process, different starting arrangements can get different clusters because the *k*-means algorithm may stick to a local minimum instead of converging into a global optimization. Accordingly, the best scenario is to have various centroids initializations then pick the run that resulted in the least sum of squared distance.

The *k*-means algorithm is widely utilized in a broad set of applications, market segmentation, clustering of documents, compression, and segmentation of images, etc. If clusters are shaped like a sphere, it does an excellent job at representing the structure of the data and can perfectly cluster accordingly. It always seeks the surround the centroid with a sphere-like shape of data points. It means that when the structure of data is complex, *k*-means does not do a good job in clustering the data. *k*-means is on top of the widely used clustering techniques, and it's often one of the first choices that people think of when they are practicing clustering applications to achieve a sensible structure of the data. The objective of *k*-means is to split data points into separate subgroups that do not overlap. Furthermore, it does not seek any advantage in learning the number of clusters from the data, but this should be assigned by the developer.

2.2G-means Definition

The aim is to briefly introduce the G-means algorithm and spot its differences from the traditional *k*-means. This algorithm has the purpose of facilitating the selection of the initial centroids by adopting a greedy technique [23]. As all know, there is the main issue with *k*-means that is the selection of the array of centroids at the beginning of the process, and this is what *G*-means will target. Hence, *G*-means will take the first attempt to do the necessary calculation for all existing datapoints having the highest degree in the given dataset. By this, we say that we set the primary setup of how the subgroups must appear. With the next run, all the single clustered centroids will be dropped, then the cluster centroids that have the highest similarity function will be selected and will be considered as true centroids. After what was performed, the iteration process on the rest of the points will begin to check if any change will occur afterward. This step will adopt the natural technique that *k*-means usually follow with the distance function as well as with similarity one [24, 25]. The major concerns that *G*-means asks are:

1. When do we perform the computation of the initial centroids with the greedy approach usage?
2. What kind of attributes will be filled in the array of centroids?
3. Will these attributes contribute to saving computational time on dataset reading?
4. Are we going to detect indications of reaching identical results as traditional *k*-means without the need to involve the randomization of *k*-means?

These questions along with some further exploration of *G*-means algorithm will be highlighted in the next chapter of this study. It will be conducted with other methods before introducing the enhancement that is the outcome of this work.

Chapter 3

Related Work

3.1*G*-means

The naming convention behind *G*-means was adopted due to the usage of a greedy approach for the traditional k-means algorithm, taking *G* from Greedy and the term means from *k*-means. This new algorithm will inherit the same distance and similarity functions from *k*-means [23]. The basic input of this problem is a graph G(*V, E*) with *k* as a constant, the set of vertices is represented by *V*, and *E* represents the edges or connections between vertices. In addition, we must indicate that this graph is of the undirected type and what the maximized clusters must be outputted. There are seven computation phases of this algorithm:

1. This is the greedy part of the algorithm by which it passes through every single element and identifies those that have the highest degrees.
2. These elements will be compared and considered with respect to the number of clusters *k*.
3. Goes by each element and calculates the similarity and distance to the corresponding centroid of the given cluster.
4. After reading, every node has one option of the below:

   a. It can be assigned to the same cluster of the centroid, meaning that the algorithm will not go through it again.
   b. It will be visited again because the distance function indicates that it might belong to a different cluster.
   c. Be assigned as a centroid due to its advantage over the given centroid in terms of similarity and distance function. This case is not common, but it could happen because we are considering the highest degree elements to be assigned as centroids.
5. Keeps the iteration process but takes into consideration part b of the algorithm, where this set of points might eventually be considered as “Boundary points”. They will have special terms that we will deal with them in a different approach.
6. Double-check if the iteration does not do any modifications to the array of centroids, if so, then it will not proceed further and will be declared as being converged and prints the array of centroids.
7. In order to print the clusters, we need to iterate once the centroids array to unify the color of centroids to the color of the elements allocated to the corresponding cluster.

The algorithm’s procedure and how it works will be demonstrated using the following example. This example was extracted for [23] to elaborate on this method:
Figure 3 illustrates phase 1, that is the original distribution of the graph; the circles in this graph represent the elements within the given space.

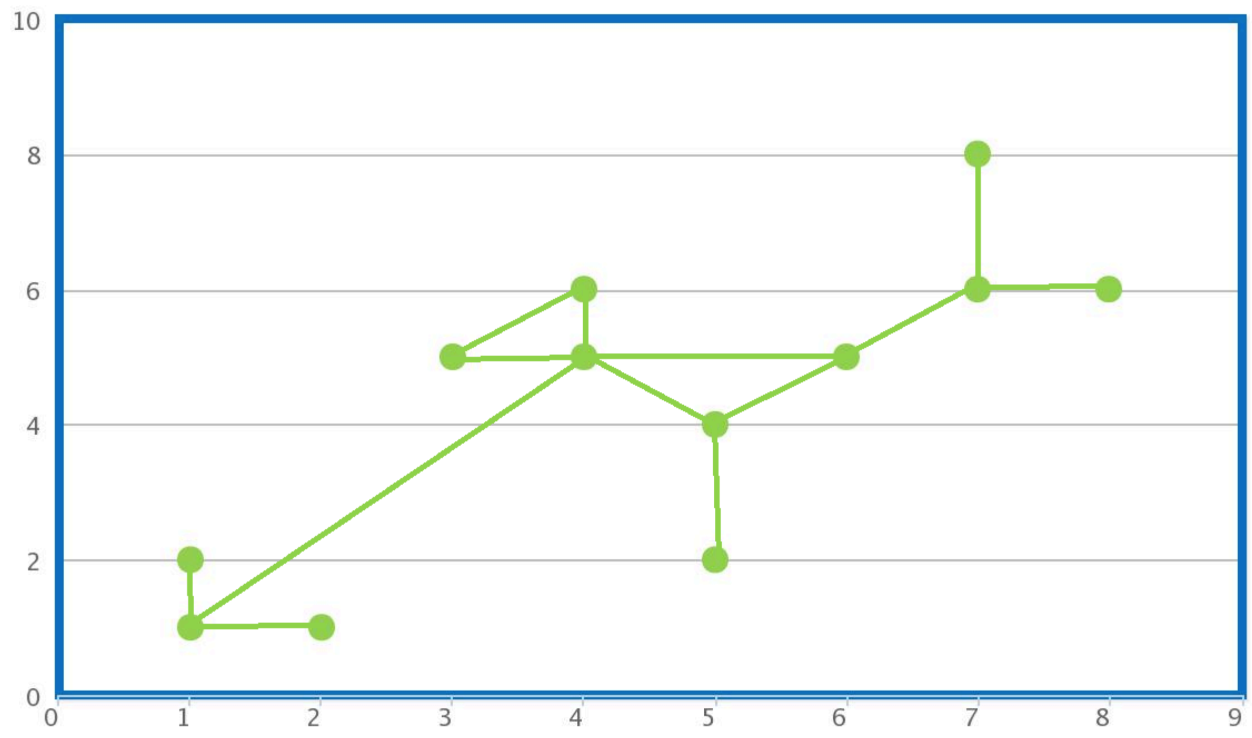

Figure 3 Initial distribution [23]

In the second phase, and regardless of the selected k [23], the algorithm will pick the elements with the highest degree. Figure 4 illustrates these selections with red marks on them.

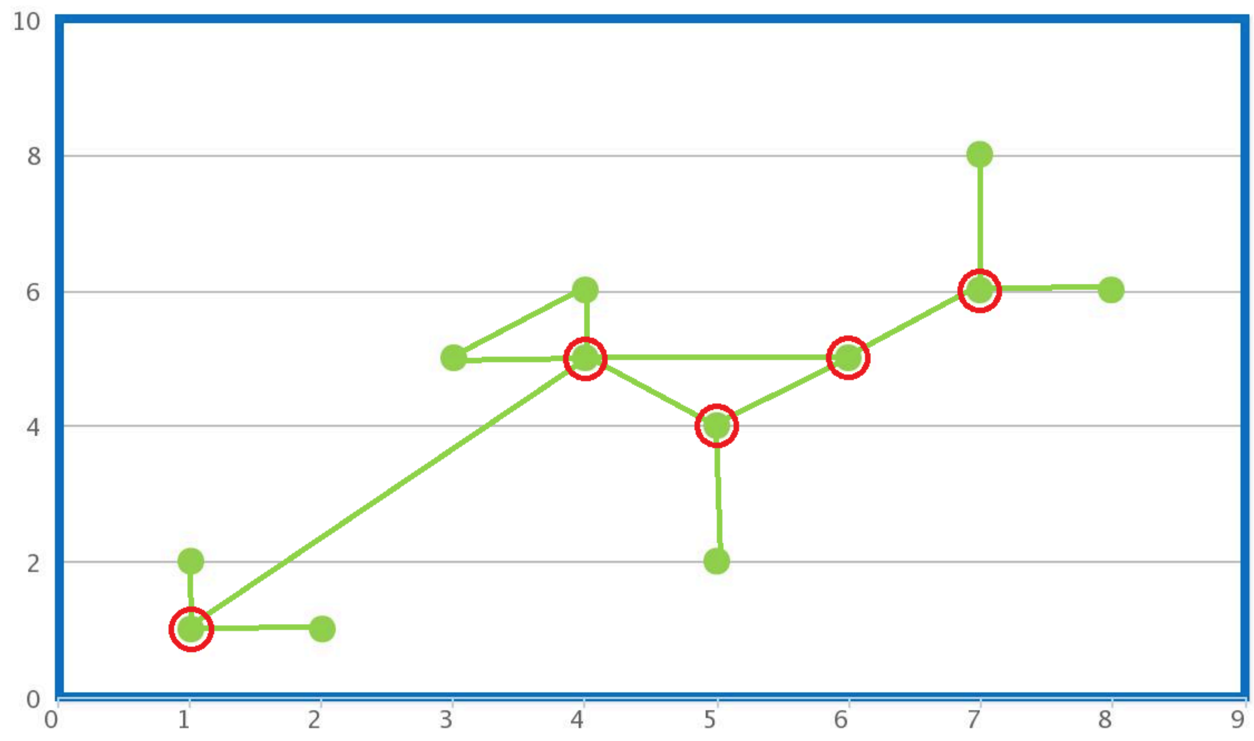

Figure 4 Selection of initial centroids [23]

At this stage, a comparison between the selected centroids with the k is assigned to return a specific number of elements within the array of centroids. *k*=3 will be assigned for the sake of simplification.

Figure 5 shows the centroids selection when *k*=3. Taking the distance function into account, the selection was conducted in accordance with the highest number of nodes in each cluster.

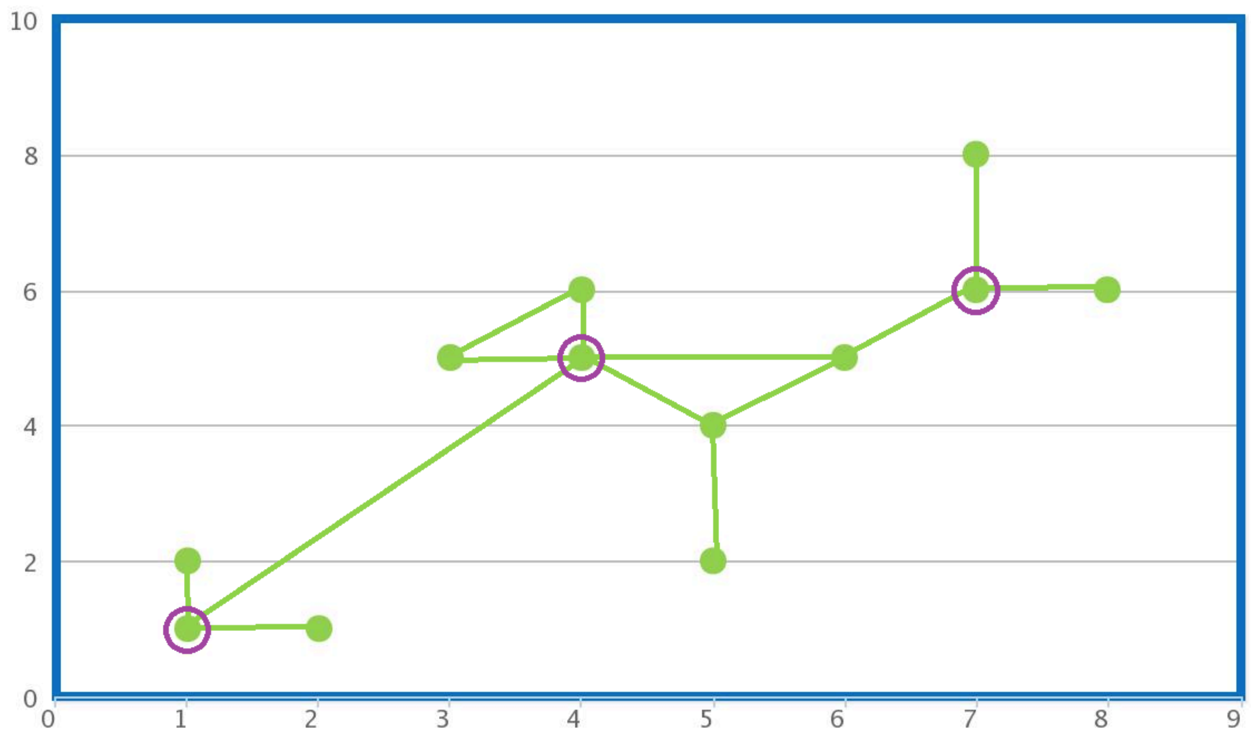

Figure 5 Centroid selection, k=3 [23]

For simplicity purposes, let us presume that the centroids will stay static. Thus, the elements will be colored with the same color as the centroid of their cluster.

Figure 6 demonstrates the convergence of the algorithm illustrating three clusters, with the three centroids having black circles around them.

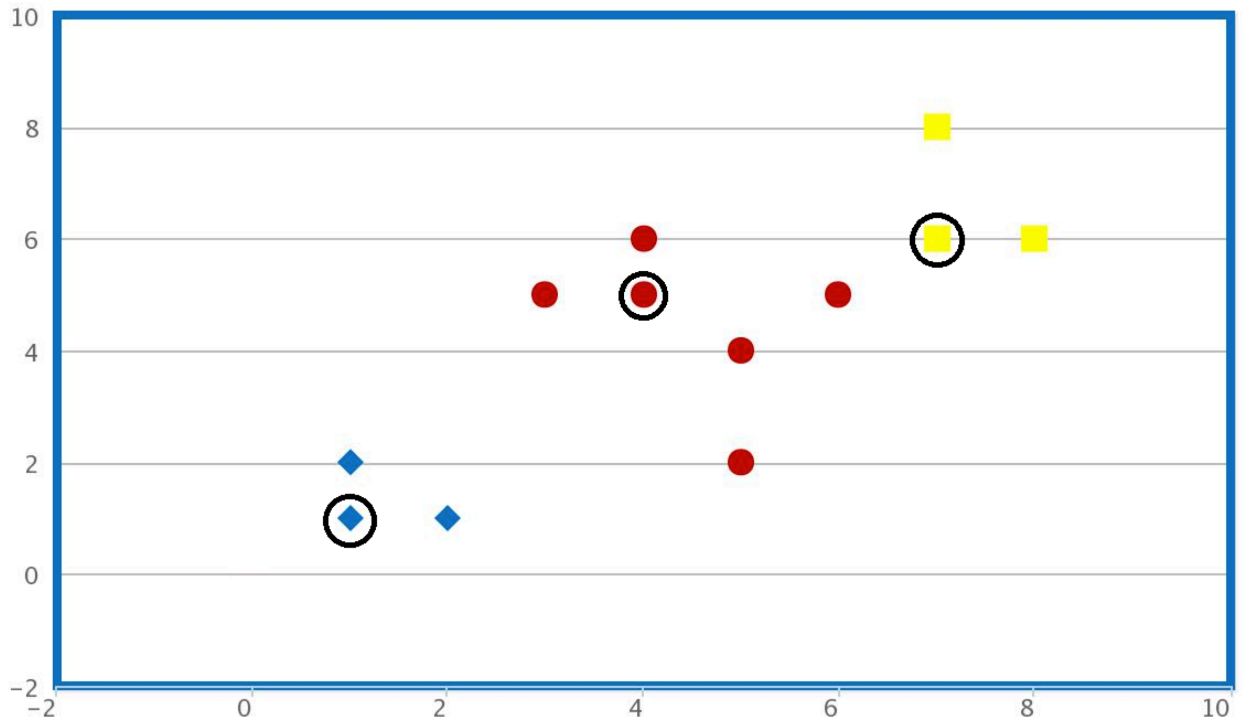

Figure 6 Convergence of the algorithm [23]

3.2SOCCER

SOCCER works by performing a loop, by which, at every iterative attempt, the coordinator will receive sub-samples of the points from every machine. In return, the machines will receive from the coordinator the assigned datapoints along with a certain threshold selected. After that, the machines will be responsible for the removal of the points that have a smaller distance from the assigned threshold from their data, each on a separate mode. This process will keep repeating until each machine satisfies the minimum volume needed before fully storing the remaining points back in the coordinator. The sub-samples that the coordinator receives from the machines will be used to play to the role of input to the *k*-means clustering technique, which is a black box for the coordinator. Afterwards, an estimation of the truncated cost for *k*-means will be computed for the centroids that were picked from the whole data. The threshold that will be used by the machines for the purpose of points removal from their datasets will heavily resort to the previously estimated figure. This whole process of cost estimation and threshold calculation is in reference to a method proposed first by Hess et al. [26]. This technique approached a new method of clustering in the application of centralized clustering. The estimation followed by Hess et al. was utilized when dealing with a set of points, to highlight the on the fly centroid selection. In spite of resorting to a whole new objective, analysis revealed that such estimation can enhance the performance when working on distributed setup.

While iterating, the SOCCER algorithm urges the machines to initiate two sub-samples for each of them and each from its own dataset. Namely *P1j* and *P2j* machine *j*, for example. These sub-samples will be shared with the coordinator. They will be drawn in an independent manner and randomly from the machine's current dataset. The sizes of these samples are determined in a way that the total number of nodes forwarded to the coordinator by all the machines is η(ϱ). After that, these pairs of sub-samples will be merged together by the coordinator into sets *P1* and *P2*. Next, the method starts calculating k-means clustering on *P1* by the use of *A*, denoted *C (iter)*, and with the use of the trimmed cost of *C(iter)* on *P2* it computes a threshold: For the sets *S* and *T* where $S \subseteq X$ and $T \subseteq X$ in addition to an integer *l*, the cost *l(S, T)* is the I-truncated cost on S by T, this denotes the overall cost of the cluster after dropping the *l* points in *S* that were responsible for most of the cost. The addition of *C(iter)* to the output set *C(out)* is performed by the coordinator, where it also sends *v* and *C(iter)* to every single machine. Afterward, the machines will take care of their own datasets by removing the nodes that have distances that are less than or equal to $\sqrt{v}$ from the *C(iter)*. Thus, to guarantee the ultimate approximate factor, we consider those points that are close enough (at most *√v)* to the certain center in *C(iter).* Finally, the loop will terminate and sends the remaining points to the coordinator as a result of sufficient points removal in each machine.

The coordinator will then calculate a *k*-cluster on each of the machines and insert the output centroids to *C(out)*. We realize that the major obstacle in the calculation is when the machines need to compute the distance of their stored datapoints to the points that were announced by the coordinator. Thus, for this challenge to be manageable, the set of broadcasted nodes should be as minimum as possible. For huge data sets, the prerequisites for calculation for machines in SOCCER are less in terms of magnitude orders than those of Hess et al. [26]. Other factors like *C(iter)* and *v* are computed in a similar way to the algorithm approached by Hess et al. [26], which was mentioned above. The constants' enhancement has a high significance and importance in practice: SOCCER uses these constants. If they would be of high volume, as is the method of Hess et al., then the practice of SOCCER will not meet the expectations. For example, the removed outliers were a lot when computing the truncated cost with Hess et. al. However, with SOCCER, if the same figures were used, it would have implemented a reduction in points removed, which will lead to a huge number of iterations. In addition, without thorough examination, it is hard to modify these constants because of the independent manner they have. In order to make enhance the practicality of the algorithm by deploying the appropriate practice for constants, many challenging quantities with a sophisticated balance is needed.

3.3PSO (Particle Swarm Optimization) Algorithm

The author in [27] proposed Particle Swarm Optimization (PSO) technique for extracting the ideal performance of k-means clustering, and to enhance the clustering accuracy. For improving the execution efficiency, it uses the approach of Hadoop and MapReducce in distributing and calculating nodes and distances. However, the method is prone to local optimum, and it is required to tackle the issue of the determination of parameters. Also, the *k*-means algorithm must be called and generated repeatedly, and every attempt must get to the dataset, while memory storage is barely provided by the MapReduce framework, so the improvement of the degree of efficiency is limited.

It was emphasized in [28] that PSO is a technique that relies on a stochastic approach. It imitates the social behavior of animals, mainly insects and small other animals. These swarms conform to a collaborative technique to look for what to eat, and everyone in this group updates the method that they are building while fetching in accordance to what is learned by experience and the neighbors as well. Typical illustrational thought of PSO is mainly related to few researches: First is the algorithm of revolution, where PSO utilizes a swarm approach that makes it to fetch a huge area, at the same time, in the pool of solutions offered by an objective function that was optimized. Second is fabricated life, along with the traits of life, it examines the synthetic structure. Millonas [29] suggested a list of fundamentals in the construction of the unreal swarm setup while examining the attitude of animal's social behavior in his theory, in cooperation with computational assistance:

1) Proximity: relative allocation and timings must be held by the swarm.

2) Quality: in terms of quality change in the environment, the swarm must be sensible in response to it.
3) Diverse response: limiting the way to narrow the scope of getting the is prohibited for the swarm.
4) Stability: the swarm should not change its behavior mode with every environmental change.
5) Adaptability: they must be able to amend the trend when this amendment is necessary and has advantages.

Take into account that the last two principles have correlated connection to each other. The manufactured system has basic features that are well presented in the five discussed attributes. Elements In PSO can update their locations and speed with respect to any update in the surrounding, meaning that the proximity and quality are well met here. Additionally, there is no limitation to the movement of the swarm in PSO, yet consistently fetching for the ideal resolution amongst what is possible. Nodes might maintain a consistence move while in the search stage and in order to meet the trend in the swarm. Therefore, the five principles are well achieved by the swarm systems.

3.4DK-Means

k-means clustering technique is extensively utilized due to many factors: simple when being implemented, ease of premise, and fast time in converging. Nonetheless, the normal k-means faces many challenges: the arbitrary initial centroids picking puts the algorithm in the danger of sticking to a local minimum, which may result in poor clustering; also, dealing with big data is not efficient enough due to the poorly implemented methods. Due to these flaws, the huge number of studies are stressing to enhance and optimize this method. A different technique of k-means initial clustering center selection was suggested by Xie Xiujuan et al. [30] that resorts to density.

The density-based *k*-means distributed clustering method (shortly named DK-means) is a type of method that may be used to cluster data having a uniform density distribution. The algorithm's main idea is to compute the distance among the nodes in a data set, determine the density of all the nodes using the function of density, choose the node with the highest return as its primary centroid, and drop the nodes that are near to it from the dataset, reiterate the process to find *k* centroids. To increase clustering accuracy, the algorithm evaluates the density of nodes and selects those with the highest density as the centroids. The parallel technique is achieved by merging the MapReduce architecture with the Hadoop platform to boost clustering execution efficiency. Using DK-means for clustering datasets with different densities of distributions, there is a possibility to choose numerous samples from the high dense cluster as initial centroids, but that is clearly not the best choice. At the same time, while hitting the data set multiple times in the phase of iterating, the unresolved k-means method consumes a lot of I/O, resulting in low clustering efficiency.

Before initiating the matrices of similarities, the algorithm calculates the mean similarity among all objects. The main object is the one with a similarity that exceeds the threshold indicated. Select the first core object as the primary centroid, the top unsimilar core object as the second, and keep doing this until k centroids are discovered, and by monitoring the centroids with the best quality we increase the accuracy. Such an approach, however, strictly enforced constraints to determining similarity criterion. The level of the first cluster center selection is directly proportional to the quality of the setting. Simultaneously, another similarity threshold is required for every case or exercise meaning that it is not global.

3.5Density-based Clustering Algorithm for Distributed Datasets using Mutual *k*-Nearest Neighbors

Distributed Databases is a database saved on many machines that are located in one physical address or maybe spread over a network of connected machines. To the contrary to parallel techniques, where CPUs are well connected to form a single database system, a distributed database system comprises slightly connected locations with the absence of any physical elements. Talking about the distributed setting, we can say that database *D* has an implicit definition in *n* databases that are explicitly connected D_is stored in *n* separate locations. Database D_i is modeled at the *ith* location by a relationship that contains many tuples. Databases with local storage can be granted calculation, whereas the normalization of data does not necessarily perform for this setting. The collection of tuples formed by the joining function applied on all D_is is the superclass of the implicit database *D,* where all the calculations will take place. In addition, turning the tuples of *D* into explicitly can't be feasible at any location because the whole local DBs, D_is, can't be transferred to a single place. Thus, it is a must to specify in an implicit way the tuples of *D*; this results in an issue that was addressed by the preservable mining algorithm that deals with privacy. Undoubtedly, it is required to have an algorithm that is effectively developed to deal with the clustering assignment of data points lying in a space that has locations with various densities, in specific those that are located in locations that are geographically far. The majority of algorithms were implemented and configured to suite the datasets that are located on the same site. It is hard for these techniques to counter data that is located in multiple sites, and also the transfer of data towards single site is another exhausting task. Due to many considerations like the volume of data, the ownership, privacy, and security issue, we say that transferring data is a hard exercise. When dealing with training datasets, nodes that belong to a certain group might have a different density from nodes that are part of another group. Therefore, in a similar scenario, it is complicated to classify the whole dataset to a specific class because of the different densities for each group. To perform a proper classification, a smart classifier is needed that is sensitive to this identity that we have mentioned. With the current classifying techniques, there are problems from this aspect. Before ideal outputs are to be generated, these classifiers have a lot of parameters to configure. Looking at Figure 7, the data space has two classes. The class that contains *R1*, which is very dense, and the other class that contains only point *Q*, which is away from that dense class. Performing the original *k*-nearest neighbor classifier, and having *k* as *1* to classify point *P* then *P* is considered a member of the class that has point *R1* in it because it is closer to *P* than any other point. However, it is obvious that this point does not belong the that class but to the other one that contains point *Q*. KNN will classify points *P* and *Q* in same class.

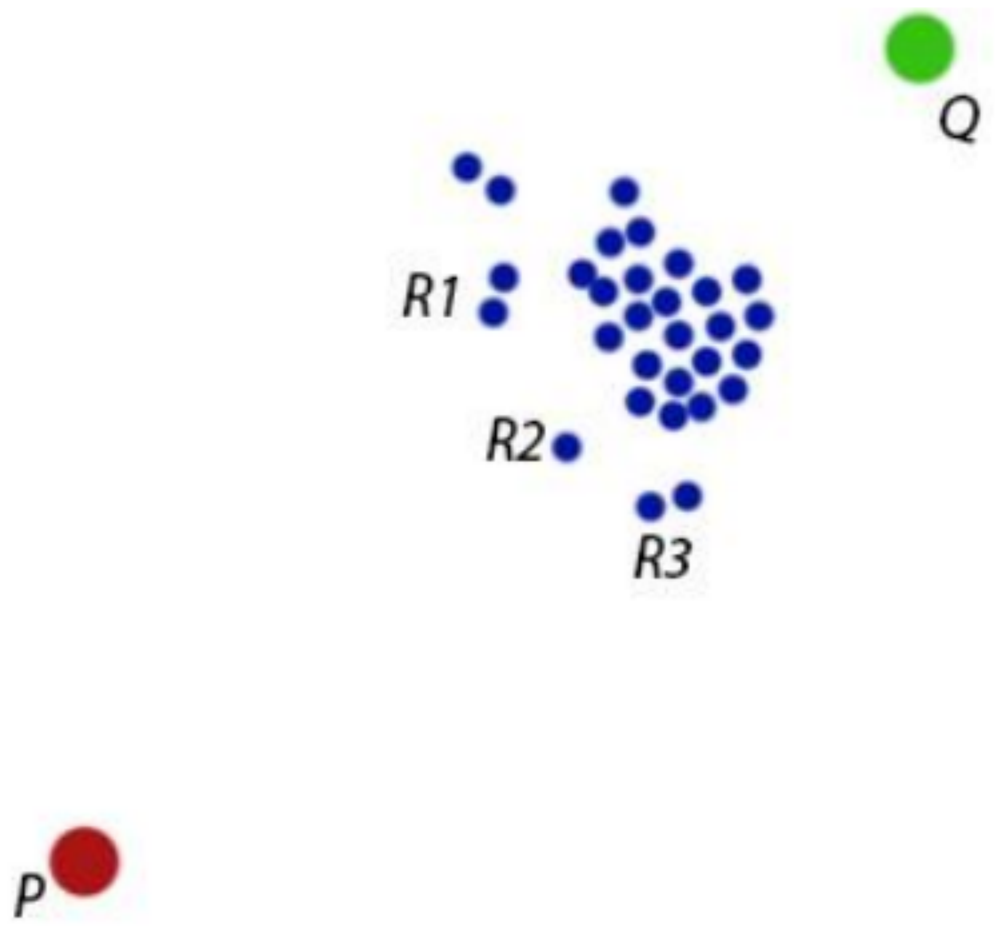

Figure 7 Class1 contains blue points, Class 2 is green point and P is Query

Looking at similar problems requires a design of an algorithm that can deal with the clustering from the density aspect. This will definitely impose a new challenge. With the often appearance in real life of having multi-dimensional points, it is a fact that the task is harder than expected.

A suggested approach to handle this difficulty is to transfer all the data points to a central location and then to execute the same algorithm. With the fact of how applicable and feasible this technique is, the issue is in the cost of transferring the data and how exhausting it is in terms of computation. In addition, another concern is privacy, where some remote locations might not agree to move their data due to some security concerns. The size of the data is also another challenge. Therefore, what is required is a powerful algorithm that deals with further computations that occur at each distinct location and then transfers the minimum info needed to a centralized machine meaning the reduction of the cost of communication. The novel algorithm makes sure to preserve the privacy and security of data at each location because of the minimum information needed to be transmitted and being exposed to other locations. This algorithm puts an extra load on remote locations than the work that is assigned to the central machine. Hence, this is what will be called the decentralized method.

Density-based clustering has never seen an algorithm that inherits this technique. Some algorithms deal with clustering partitioned databases but have limitations on the partitioned data, such as [31,32]. Some other present methods for clustering when we have locations that comprise distinct features. The work in [32,33] treats *k*-means from a privacy-preserving aspect for distributed databases.

3.6Parallel *k*-means clustering algorithm

The author of [34] relied on cuckoo search in ad adaptive approach to propose a parallel *k*-means clustering algorithm. The improvement of the cuckoo search technique helps in the adjustment of the search process by fixing the size of each step, in addition to the integration of such a search method to the k-means traditional approach. The optimal center of clustering is found with the iterative-based k-means, and then the clustering task will be done. The algorithm was also enhanced also by the combination of the Map reduction framework and the Hadoop platform parallel structure. For this algorithm to function, it requires couple of inputs such as: the number of clusters needed, the discovered probability, the max iteration number, and the maximum and minimum step size. The value of clustering and its quality are impacted directly by the qualities of these inputs, in addition to some human factors that might also come into the game.

The cuckoo search algorithm adopts the behavior of some cuckoo breeds as well as some other birds. Cuckoo search has a novel method of intensification, which means that it searches for better solutions just around the solution that it has in hand. In addition, it also provides a new way of diversification, meaning that it assures the algorithm can investigate the search space in an efficient manner [35].

To simplify the behavior of the cuckoo, the below list of three rules can be formed [36]:

1) Every bird lays an egg at a given time, then puts the egg in a random nest;
2) The following generations will be formed from the highest quality eggs that were found in the top nests;

3) There is a fixed amount of the nests that will host the eggs, and the egg given by any random cuckoo will be monitored by the bird that is hosting this egg with a probability from [0,1].

Based on what is given, it is the choice of the hosting bird to decide if it wants to let go of the laid egg or to leave the nest and start building a fully new nest. Based on those rules, the pseudocode of the Cuckoo Search algorithm was proposed, see Figure 8 below.

```
Objective function f(x); $x = (x1, x2,…, xd)^T$
Generate initial population of n host nests xi (i=1, 2, …, n)
while (t < MaxGeneration) or (stop criterion)
Get a cuckoo randomly by Lévy flights
Evaluate its Quality/Fitness Fi
Choose a nest among n (say, j) randomly
If (Fi > FJ)
Replace j by the new solution
end
A fraction (pa) of worst nests are abandoned and new ones are built
Keep the best solutions (or nests with quality solutions)
Rank the solutions and find the current best
End while
Post process results and visualization
```

Figure 8 Cuckoo search algorithm pseudo-code, captured from [37].

A preliminary study was published in 2011 [38]. In order to produce new solutions, this study combines a real representation of solutions (centroids), Euclidian distance, Davies-Bouldin index as the fitness function, and local improvement of solutions based on SSE across iterations, and ultimately, an adaption of Lévy flights (eggs). The data clustering findings are encouraging, but they do not include cluster labeling or the maximum processing time available to complete the task. They also don't deal with the dead unit problem that Lévy flights are likely to cause (in order to improve the algorithm's efficacy).

A comparison of the Genetic Algorithm, Particle Swarm Optimization, and Cuckoo Search (CS) over numerous clustering issues was carried out recently [39]. In terms of average classification error % and execution time, CS outperforms. In order to run all algorithms in this work, a predefined value of clusters must be employed; hence, this concept is impractical for online document clustering. Furthermore, it ignores cluster labels, noise, and other success criteria relevant to the study topic.

3.7EDC-CSK

Another new algorithm, named Web Document Clustering based on the Cuckoo Search Algorithm (WDC-CSK), is an algorithm based on descriptions [40] for web clustering, where it was influenced by the Cuckoo Search algorithm, which itself is a meta-heuristic approach [36]. This technique tends to combine strategies with global and local means of searching a whole space [40]. It resorts to the *k*-means algorithm to enhance the global solution. To develop better diversification in the population and ensure the prevention of quick convergence to a local minimum, the levy flights were substituted by the split and merge methods. Finally, for the purpose of automatically finding the number of clusters, Bayesian Information Criterion was deployed as a fitness function. Figure 9 illustrates the primary operations for the execution of the WDC-CSK algorithm.

```
01 Initialize algorithm parameters
02 Document preprocessing
03 Execute in parallel a specific number (I_max) of Islands
04 Initialize population of nests; create randomly a set of nests (population of nests) from the current island
05 Execute k-means (local optimizer) for each nest in population from the current island
06 Calculate fitness values (BBIC or BIC) according to (4) or (5) for all nests in population from the current island
07 Repeat
08 Create a new nest using abandon, split or merge operations (methods) based on a randomly selected nest (current nest) from the current island
09 Execute k-means (local optimizer) for the new generated nest
10 Calculate fitness value (BBIC or BIC) according to (4) or (5) for the new generated nest
11 Store best solution, if the new generated nest is better than another randomly selected nest, this last nest is replaced in the population for the new generated nest
12 Until stopping conditions are satisfied (MNN parameter is reached or MET parameter is reached)
13 Select the best nest in the population of nest from the current island
14 End on parallel execution
15 Select the best nest from all islands
16 Assign labels to clusters in the best nest based on the frequent phrases in each cluster.
```

Figure 9 WDC-CSK algorith summary, taken from [37]

01: Initialize algorithm parameters. Here it was intended to minimize the Bayesian Information Criterion. The algorithm required some inputs to operate such as integer I_max, integer PS, enumeration value OF, real value PA, integer TFT, another integer MNCK, and finally, MNN. These parameters are well explained and discussed in [37].

As also discussed in [41, 42], to enhance the quality of the solution encouraged, parallelism is required where it is not only for the resolution time reduction. This is due to the fact that in various cases, the deployment of search progress is different especially when approaching a parallel meta-heuristic method [43].

02: Document preprocessing. Here, we will rely on Lucene to handle the stage of preprocessing documents. This phase comprises: tokenizing, lower case filtering, stop word removal, Porter's stemming algorithm, and the building of TDM. Also the removal of columns that are zeros.

The wide usage of the TDM matrix in the structuring of documents where it is inspired by the vector space model [6,12] made it necessary in this model. Documents are structured as sets of words; a matrix of D terms crossed by n documents represents the documents. To measure the degree of similarity among documents or between a document and a centroid, the representation through a vector of normalized frequencies is necessary.

04: Initialize the population of the nest. WDC-CSK algorithm operates in a way where nests are utilized for the representation of the solution. The number of clusters in each nest is different from one another, centroids list, and a value for the objective function, with respect to BIC. In the beginning, every centroid represents a random document that was picked by the TDM matrix. The minimum number of documents can be assigned by users in order to run the clustering method.

05: Execute *k*-means. The execution of lines 2 and 5 from the figure is done with respect to the centroids' registration on the corresponding nests. Due to the previous selection of the centroids, line 1 is not really needed. Lines 02 to 05 are repeated MNCK times. MNCK parameter, which handles the repetition of lines 2 and 5, is also responsible for the control of the level of exploitation of the algorithm. When this parameter is assigned as 1, there will be the only organization of the documents to the suitable centroids rather than any enhancement or improvement in the solution set. However, when this value gets higher, the method will develop some improvements locally. It should also be considered that lines 2 and 5 might be finished a couple of cycles before the convergence of *k*-means.

08: Create a new nest. The algorithm executes one of the operations that are abandoned, merged, or split. After selecting a random centroid from the TDM matrix, the method then tends to create a new nest with a predefined probability. This technique leads to abandonment, and it is influenced by the scenario where the host bird discovers the cuckoo egg. In this case, and in the concerned island, a whole novel nest will be initiated for the population of nests to be completed. It also helps the prevention of the quick convergence as well as the diversity it ensures. The merge or the split functions will execute with a certain probability. The replacement of the levy flight commands in the main cuckoo search algorithm is guaranteed by the mentioned steps. The random initial selection of a nest from the current population is followed in both operations. Then a base nest is generated by copying the nest that resulted previously. Afterward, from the base nest, we pick the two most similar centroids in order to join them. On the other hand, the most far centroids will be selected to be divided using the split operation.

13: Select the best nest. In this stage, the algorithm proceeds in finding and the selecting the ideal solution from the set of nests on the given island. It is considered as the best solution when this nest has minimized the BIC value. It then returns the value of the best solution from the island we are dealing with.

16: Assign labels to clusters. To label the clusters, this algorithm applies FPH, which is the frequent phrases approach. This stage is similar to what is called ‘‘Frequent Phrase Extraction’’ in Lingo [44] but with some adjustments to the original one. However, In WDC-CSK, this technique is utilized after the generation of every cluster in the best solution set, thus labeling each and every resulting cluster.

3.8IMR-KCA (improved K-means algorithm)

The author in [45] proposed an improved *k*-means algorithm (imr-kca). In the analysis of the flaws in the traditional k-mean redundancy, the mean-based technique suggested a model of selection that aims toward the simplification of the way we compute when having various centroids. Simultaneously, the Manhatten distance method replaced the Euclidean one to better suit the clustering when dealing with data related to the medical fields, and it uses the MapReduce framework to handle the parallel computations. The suggested algorithm is relatively powerful when dealing with medical data, but it is not much promising for universal usage.

Chapter 4

Algorithm and Approach

This chapter will shed light on the developed algorithm and the purpose behind this work. Our algorithm is an enhanced version of G-means on distribution bases. G-Means is not a popular clustering approach; it was developed on the basis of degree with the ability the identification of better initial centroids. The distributed version of it is another challenge due to a certain order when dealing with data and accesses. Besides, when it comes to the program with MPI, data engineers need to manage how nodes can communicate with each other; it is actually real defiance. We unveil a novel Distributed G-Means based on the Spark framework.

We suggest a distributed parallel algorithm that implements G-Means with Spark. The algorithm starts by extracting information from the HDFS (Hadoop Distributed File System), forming RDDs (Resilient Distributed Datasets), then moves to the transformation phase of these RDDs into data points. The procedure will be certainly executed by the Spark engine. Those formed RDDs are then sent to multiple executors. After that, and in each of these executors, sub-groups will be formed to be forwarded after each statement to Spark. To avoid shuffling operations between executors and the high cost that might be imposed, each executor avoids communication with others by performing its computation. So, additional points are introduced at the level of every sub-cluster. When all these sub-clusters get gathered by the power of the accumulator that is variably shared, the clusters will be identified by the algorithm and will be merged by the newly introduced points. Thanks to the ability of the variables shared in the

Spark drive; the merge will occur in the Spark framework. The experimental results conducted at the level of machine, which is memory distributed, reveal that scalable performance could be obtained based on the proposed algorithm.

A running Spark application, at a high level, includes one driver process communicating with multiple executing frameworks, assigning some jobs to them then taking care of the collection of what they have done. Initially, a Spark application will construct what is called Spark Context drive, that instructs Spark on the way to connect to a given cluster. It will then extract a file or more from HDFS and then do the processing to them as RDDs, which are collections of items that are segmented among elements and then to be processed following a parallel approach. The fundamental abstract that Spark technique offers is RDD. We would like to stress out that Spark can access data not just through its own APIs, but also through the Hadoop API. TaskScheduler uses the Resource manager to assign tasks to executors. Executors will communicate back to the driver what they got or print to the external storage the output after completing their job. The Spark framework incorporates all of MapReduce's key capabilities. It also has the following characteristics.

- Computations that are performed entirely in memory. RDDs represent the initial abstraction in Spark, which lets developers run computations that are based in memory on huge clusters. Algorithms with iterating behavior and interacting data mining are two sorts of applications that MapReduce struggles to perform efficiently [46]. Several attempts with map reduction executors are to be done in order to handle iterative algorithms, which is inefficient since the results of the intermediate map must be printed first on the local drives and then to the remote ones retrieved to reduction actors, and the input/output commands of the disk are particularly costly with such an instance. The ability to execute iterative computations at breakneck rates is one of the benefits of keeping everything in memory with Spark.

- Supporting complex analytics, real-time analysis, and data streaming. We were not able to use MapReduce to perform real-time analysis in a beneficial way, so this powerful feature was put down at the time being. However, it will be part of the future work to make the best of Spark with our algorithm.

- The quick recovery from faults. Traditionally in Mapreduce, when no heart beat from a task tracker is sent to the job tracker for a certain stint, the job tracker will conclude that the agent allocated to the task tracker has stopped. In such a case, the job tracker is responsible for setting a new schedule for the tasks that were being in progress along with those that were about to start to a new task tracker, since the information that belongs to the corrupted task tracker will no longer stay available [47]. Otherwise, and with correspondence to linear trend, Spark will construct RDDs again to tackle the exercise. If we compare this method to the replica technique, it turns out that reconstructing RDDs consumes less time [48].

Despite the high efficiency of Spark, and the automated parallelization it offers, extra attention is required for the avoidance of the shuffling process. Therefore, in this implementation, another challenge was to avoid all-to-all communication.

Figure 10 illustrates the flow of data within the Spark session. First of all, when the interpreter calls a Spark session, Spark will create an operator to manipulate and interpret the code to best suit the need. When an Action is called on Spark RDD at higher level, Spark will then submit the graph to the DAG Scheduler (Direct Acyclic Graph Scheduler), which will fragment the operator into stages of tasks, and each stage contains task(s) based on the data input partitioning. The stages then are passed to the Task Scheduler by the TaskSet. These steps are all executed within the Spark Context which will then forward to the Spark Driver, also known as cluster manager. This manager or driver is the brain of the whole technique and will take care and manage the process of distributing tasks on different executors or workers, listen back to them to collect their results. After that, this manager will also treat the returned results to deliver the best cluster in accordance with all the executors.

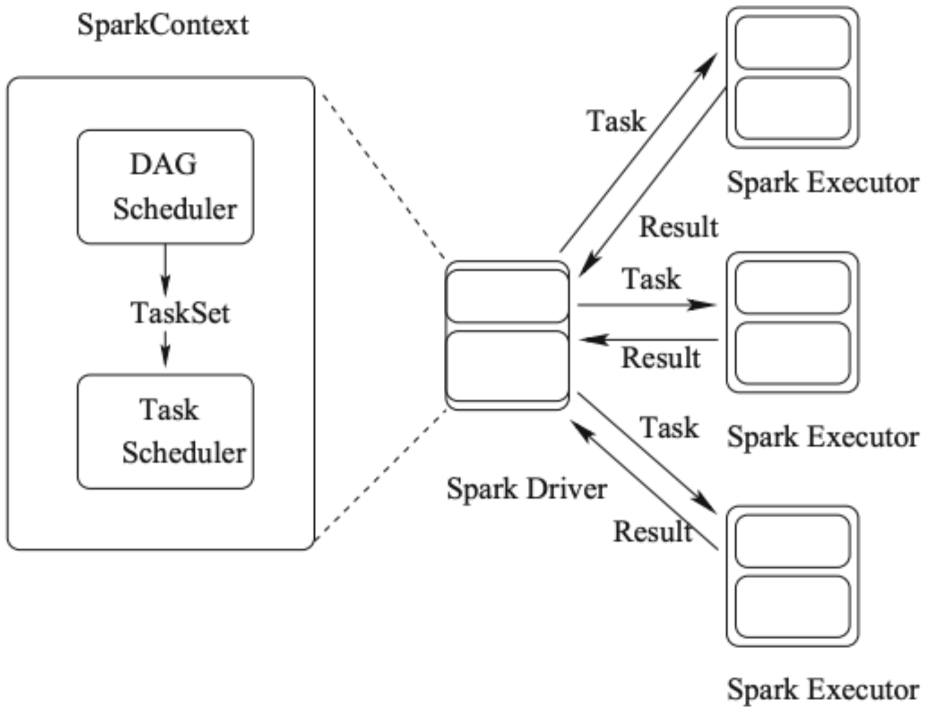

Figure 10 Data flow overview in Spark

4.1Pseudocode

The following represents the pseudocode of the D-G-means algorithm with Spark.

```
Input (epsi, minpt, D)
Output (an array of clusters)
1.  reads input file from HDFS, generates RDDs from data
2.  transforms current RDDs to suitable RDD points
3.  distributes these RDDs into execs
4.  for each start
5.      if point p not in hash table
6.          select neighbors of p
7.          insert all suitable neighbors into Queue N
8.          if (size of N less than minpt
9.              define p to become noise
10.         else
11.             create new cluster C then add p to C
12.             while (N not empty)
13.                 remove p' from N
14.                 insert p' index in hash table
```

```
15.             if (p’ is not visited)
16.                 note p’ visited
17.                 mark N’ as epsi-neighbor to p’
18.                 if (N’greater than or equal minpt)
19.                     + suitable pts into N
20.                 End if
21.             End if
22.             if (p’ doesn’t belong to cluster)
23.                 + p’ to C
24.             End if
25.         End while
26.     End if
27.   End if
28.   if (point = last in closure)
29.       send part-cluster to drive
30.   end if
31.   end for each
32.   search all part-clusters then merge
```

4.2Algorithm Complexity

From the first part of the algorithm, the driver reads RDDs from HDFS, and the suitable data points should be processed by executors and constructed corresponding trees. So we assume *O(n*logn)*. As for the other part, the generation of the local sub-clusters takes place in the executors. Ideally, the worst case it will be *O(n1-1/d+K),* as researchers have reported [49]. In the final segment of the algorithm, after all sub-clusters are sent back, the Spark engine will merge them and produce the universal cluster. The search operation here takes *O(n),* and the merging step takes K_m, both resulting in $O(n+ K_m)$.

Therefore, the final complexity would be:

$O(n*logn) + O(n1\text{-}1/d+K) + O(n+ K_m) \rightarrow O(n*logn + n1 – 1/d + K + n + K_m)$

Chapter 5

Experimental Results

5.1Hardware

We prepared an environment with Spark 2.4.2 and Hadoop 2.5 cluster. We developed all the algorithms in Python. The development IDE that we used is Jupyter Notebook for Python and with a lot of libraries of various technologies and methods. A computer with a 2.26GHz Core i7 processor with 8GB memory, 500GB solid-state drive, and windows 10 64 bits operating system was used for the experimental analysis.

5.2Datasets

For the evaluation of k-means, DG-Means, and SOCCER algorithms three separate databases were used as follows:

- Data set 1: CSV is the format of this data. It consists of whole sale clients' data based on three variables. It consists of 10,000 records.
- Data set 2: The format of this file is also CSV. This dataset is to check the genuine and forged banknotes. It also consists of three variables, the standard deviation of the image transformed, the image's entropy, and the class. It consists of about 13,000 records.
- Data set 3: It is the famous data set of IRIS, and it is a CSV file. We use three attributes, sepal length, sepal width, and petal length. 4500 data points are included.

These three datasets were selected because of their availability and the features that well suit the algorithms that we are working on. First, for whole sale dataset there are a lot of work done for clustering it using k-means and other unsupervised algorithms, these works motivated the selection of this dataset. The banknotes dataset on the other hand has some work as well on clustering which helped us in featuring its characteristics and have an insight to what it might converge. Finally, the popular IRIS dataset is one of the most used datasets in the field of clustering for the ease and cleanliness it provides. It actually returns a clear image on how good or bad the algorithm might perform on small datasets.

5.3Performance Evaluation Results

For this assignment, the execution runtime of our developed distributed Greedy algorithm DG-Means, k-means, and SOCCER algorithms are observed and discussed by the change of data. The experiments rely on to distributed Spark cluster. A discussion of the results of these tests based on the convergence, in addition to how stable are these methods is illustrated in the following part:

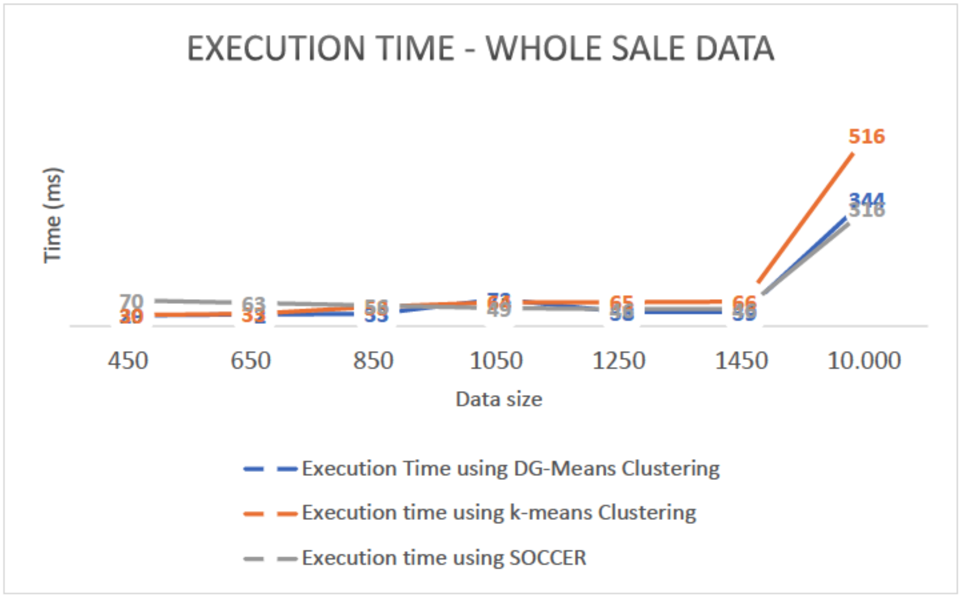

Figure 11 Execution time - whole sale data

On the WholeSale dataset, we can see how each algorithm execution time varies based on the data volume itself. In general, the behavior of DG-means is the best among the three algorithms. However, although its poor execution time on small data volume, SOCCER performed better than DG-means when dealing with bigger data (the case of 10,000 records of data). k-means on the other hand has the worst performance against other algorithms. In conclusion, and based on this dataset, we can order the execution time of the algorithms as follows: DG-means, then SOCCER, and finally k-means.

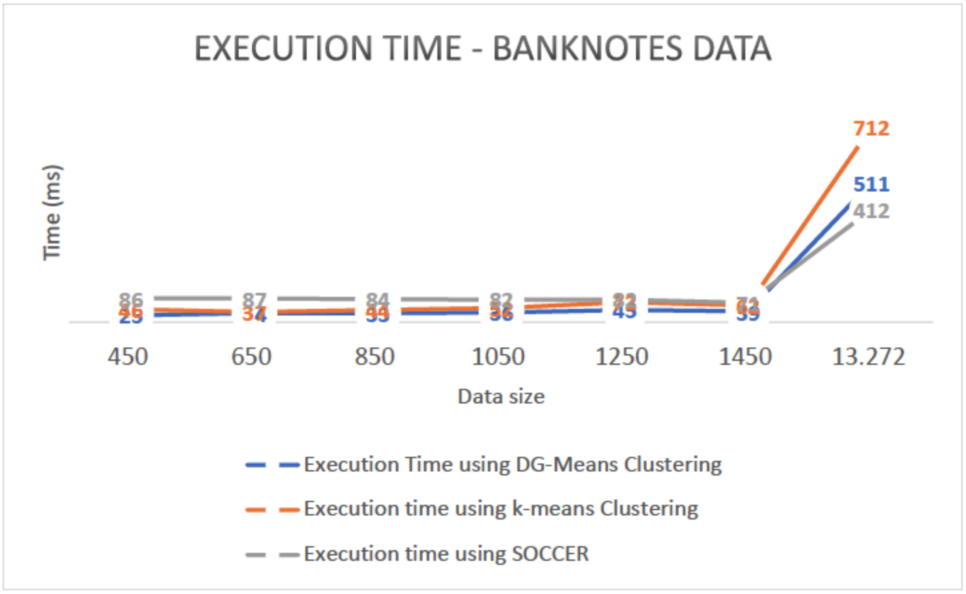

Figure 12 Execution time - banknotes data

Moving to the Banknotes dataset, we can also find that the execution time of DG-means is the best among the three algorithms in general. In a similar manner to the previous dataset, the SOCCER algorithm performs better when data gets bigger, and it even surpasses DG-means when reaching 13272 records of data. k-means has the lowest performance despite the advantage it has over SOCCER on a small number of records. We summarize this sample by ordering the execution time of algorithms as D-G-means, SOCCER, then finally comes k-means.

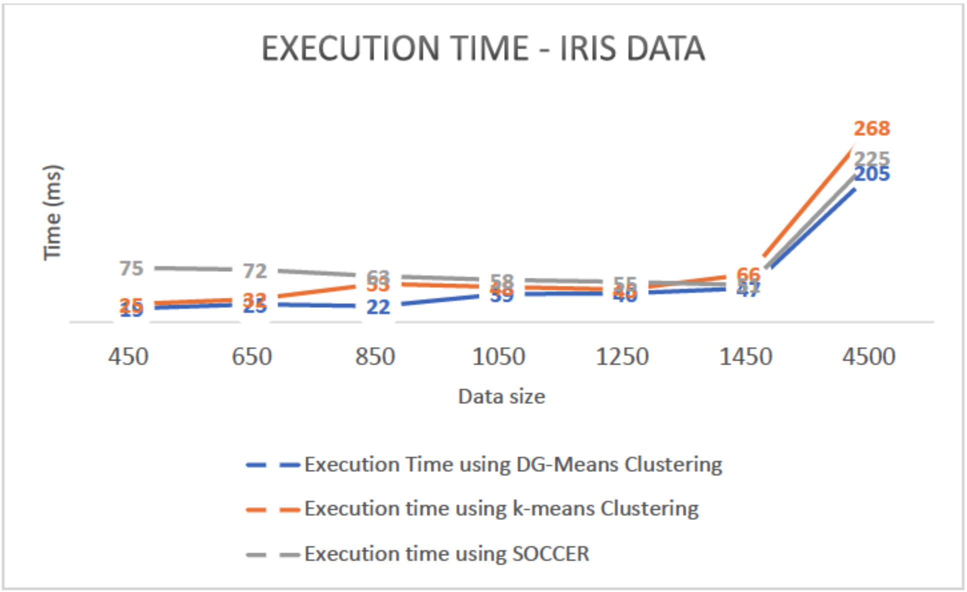

Figure 13 Execution time - IRIS data

In the third dataset, the IRIS dataset, we realize that the three algorithms are very close in terms of execution time. This might be true due to the fact of the small data records compared to the two previous datasets. They all ended up within the same range of time when dealing with 4500 records. Within this slight range, DG-means as usual started better than the rest, and SOCCER had the poorest performance. As data grew, the same pattern was followed. SOCCER started to enhance its performance following DG-means, and k-means had the poorest as we went bigger. To summarize, DG-means, followed by SOCCER, and then k-means, was the order of execution time order for the three algorithms on the IRIS dataset.

To evaluate the stability of each algorithm, we will address the parameters that every algorithm needs. Also, for simplicity, we will show the evaluation based on IRIS dataset only, although the other datasets can be easily managed as well.

For k-means algorithm, we need to find an ideal number of clusters. We cannot introduce straight forward answer for choosing *k* value. We will resort on the Elbow method to do this. First, we calculate the SSE to a set of values of *k*, after that we display values of *k* against the sum of squared error graph.

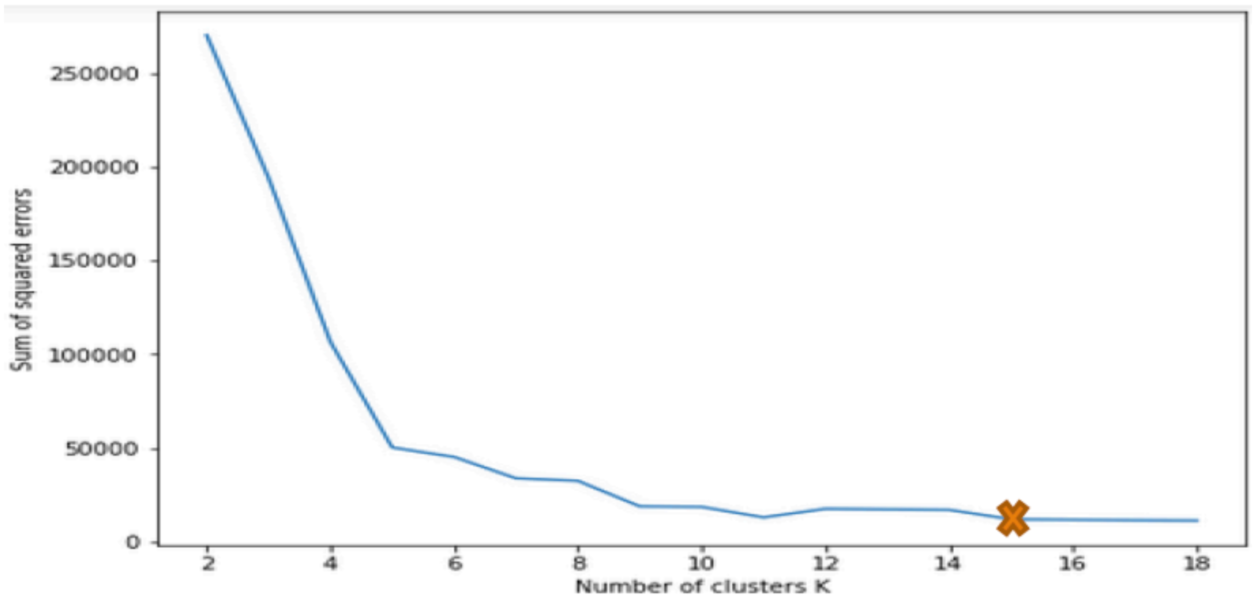

Figure 14 Sum of square error vs K - IRIS Dataset

On the other side, for DG-means, ε and MinPts parameters are required, so to evaluate the stability of this clustering algorithm, we set up ε with different values for several iterations of ε, all this based on IRIS Dataset.

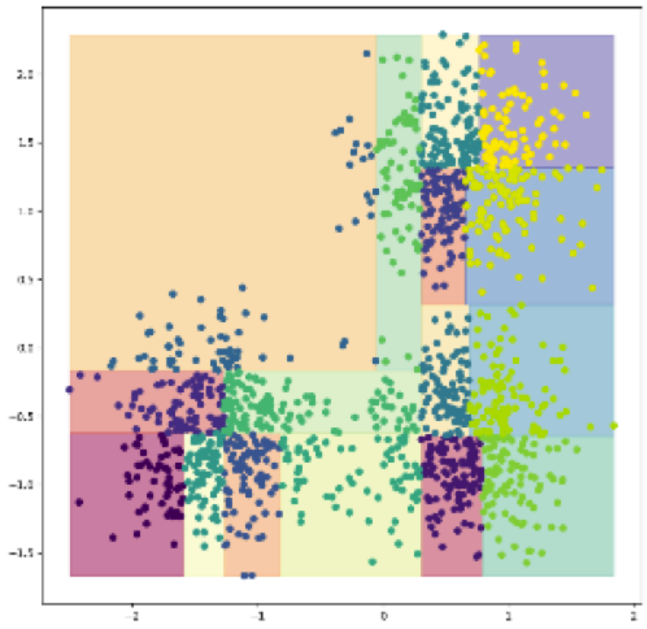
Figure 15 Minimum points = 5, Eps = 0.7

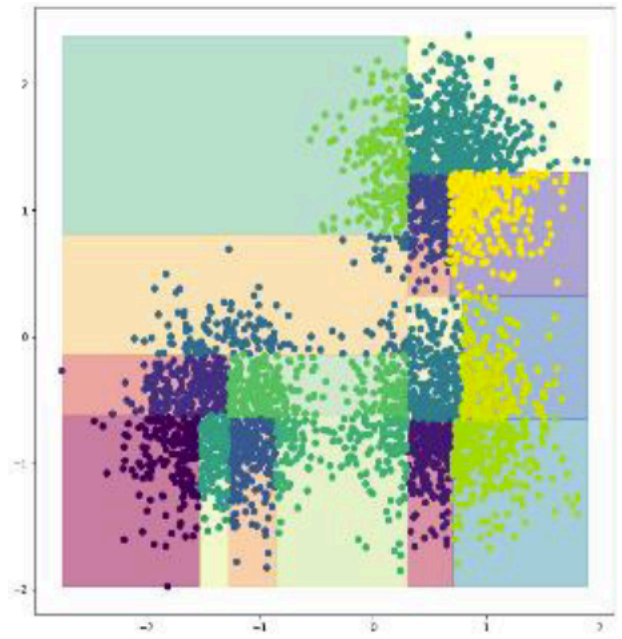
Figure 16 Minimum points =8, Eps = 0.5

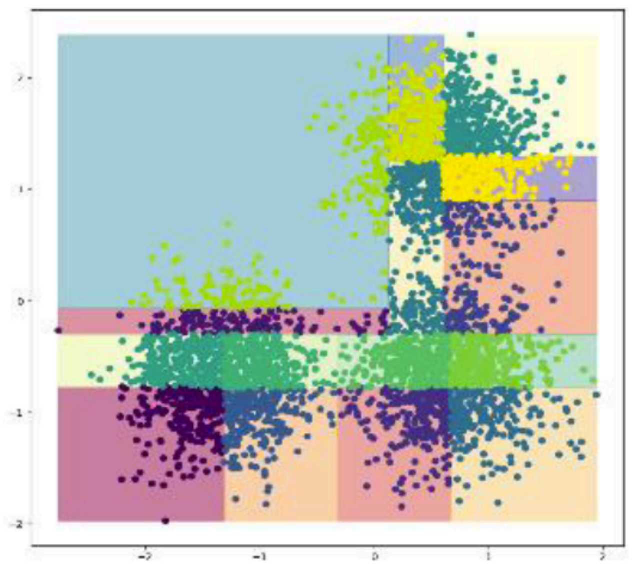

Figure 17 Minimum points = 3, Eps = 0.3

Another set of experiments was performed with WholeSale, Banknotes and IRIS datasets. Ten rounds were conducted on these three datasets to investigate the accuracy. The average accuracy and average iterations of these ten experiments were considered and illustrated in table 3 below.

Table 1 Comparison of accuracy and iterations on three Datasets

Parameter	Dataset	DG-means Clustering	K-means Clustering	SOCCER
Accuracy	WholeSale	96.10%	89.50%	95.50%
Number of Iterations		7	8.3	8
Accuracy	Banknotes	70.90%	66.90%	69.80%
Number of Iterations		6	8.6	9
Accuracy	IRIS	88.70%	83.60%	88%
Number of Iterations		4	6.9	5

From table 1, we can notice that the accuracy of DG-means is higher than that of both SOCCER and traditional k-means. In addition, DG-means has a smaller number of iterations, which means that it has a faster convergence speed and higher efficiency. Thus, the proposed DG-means is superior to k-means and SOCCER algorithms in clustering accuracy and efficiency. Next to it comes the SOCCER algorithm, which exceeds k-means in accuracy on all three datasets, with lower or similar number of iterations. From this standpoint, we also concluded that the order of the three algorithms on the three different datasets we have dealt with is the following: First one is DG-means, followed by SOCCER, and finally comes the traditional *k*-means algorithm.

Chapter 6

Conclusion and Future Work

Many papers dealt with clustering based on similarity metrics, but not much work was done based on the greedy approach before G-means was proposed by [23], where the author presented a technique that clustered huge data that was not likely to be performed by traditional *k*-means. From that point on, we took that G-means algorithm processed the distributed mode on it relying on Spark methods and came up with the distributed version of it that was called, DG-means. This mode allowed spread data or even spread computational power through machines placed in different locations to deal with the same data simultaneously by clustering and dealing with a given dataset.

The execution time of DG-means and *k*-means can be conducted in an easy manner. The figures in the previous section can clearly exhibit it. It can be easily seen that the DG-Means clustering algorithm is faster than the k-means clustering. One can observe the contradiction easily, especially if we are to consider huge datasets.

In the next phase, it could be a point of concern dealing with the analysis of clusters of Spark Apache framework, especially the complex analytics, real-time analysis, and data streaming. In addition, we shall introduce bigger data to the fields of study to express the ability of this tool in dealing with big data experiments.

Another area of expanding this research can go towards examining the ability of this algorithm to actually perform on different machines at the same time. Although this feature is really embedded within Spark and could have been implemented, we were not able to establish such an environment due to various reasons. Therefore, this part can be taken further by setting up the appropriate scenario for such an interesting field of practice.

Finally, another approach can be a dedicated kernelization of the clustering algorithm. Although this was proposed as well by [23], it was not feasible at the current stage and can still be considered as a potential future work, especially on the level of preprocessing of the data as well as the preparation of the algorithm to meet that well-suited data. This can definitely enhance the performance, and the results of the clustering techniques followed.

References

[1] W. Yi and J. Yan, “Energy consumption and emission influences from shared mobility in China: a national level annual data analysis,” Applied Energy, vol. 277, Article ID 115549, 2020.

[2] S. G. Anton and A. E. Afloarei Nucu, “The effect of financial development on renewable energy consumption. A panel data approach,” Renewable Energy, vol. 147, pp. 330–338, 2020.

[3] P. Pei, Z. Huo, O. S. Mart´ınez, and R. G. Crespo, “Minimal green energy consumption and workload management for data centers on smart city platforms,” Sustainability, vol. 12, no. 8, p. 3140, 2020.

[4] T. Enokido and M. Takizawa, “the power consumption model of a server to perform data access application processes in virtual machine environments, advanced information networking and applications,” in Proceedings of the International Conference on Advanced Information Networking and Applications, pp. 184–192, Springer, Toronto, ON, Canada, May 2020.

[5] Q. Zhou, S. Guo, and H. Lu, “Falcon: addressing stragglers in heterogeneous parameter server via multiple parallelism,” IEEE Transactions on Computers, vol. 70, no. 1, pp. 139–155, 2020.

[6] K. G. Miller, R. P. Lee, A. Tableman et al., “Dynamic load balancing with enhanced shared-memory parallelism for particle-in-cell codes,” Computer Physics Communications, vol. 259, Article ID 107633, 2021.

[7] K. P. Sinaga and M.-S. Yang, “Unsupervised K-means clustering algorithm,” IEEE Access, vol. 8, pp. 80716–80727, 2020.

[8] N. Altman, M. Krzywinski. “Points of Significance: Clustering.J. Nature Methods.”, 14(6):545-546, 2017.

[9] M. Wu, X. Li, C. Liu, et al. “Robust global motion estimation for video security based on improved k-means clustering”. Journal of Ambient Intelligence & Humanized Computing., 10(2): 439-448, 2018.

[10] B. Lorbeer, A. Kosareva, B. Deva, et al. “Variations on the Clustering Algorithm” BIRCH.J. Big Data Research., 11:44-53, 2018.

[11] P. Zhang, Y. Wang, L. Liang, et al. “Short-Term Wind Power Prediction Using GABP Neural Network Based on DBSCAN Algorithm Outlier Identification”. Processes., 8(2):157, 2020.

[12] V. Bureva, E. Sotirova, S. Popov, et al. “Generalized Net of Cluster Analysis Process Using STING: A Statistical Information Grid Approach to Spatial Data Mining”. International Conference on Flexible Query Answering Systems. Univ Westminster, London, ENGLAND. 239-248, 2017.

[13] S. Guha, R. Rastogi, & K. Shim. “Cure: An efficient clustering algorithm for large databases. Proceedings from ACM SIGMOD International Conference on Management of Data”. Snowbird, UT, 1998.

[14] J. MacQueen. “Some methods for classification and analysis of multivariate observations. Proceedings from The Fifth Berkeley Symposium on Mathematical Statistics and Probability”. Berkeley, CA, 1967.

[15] L. Bottou and Y. Bengio. “Convergence Properties of the KMeans Algorithm, Advances in Neural Information Processing Systems” 7, 585-592, MIT Press, Denver, 1995.

[16] S. Parthasarathy and M. Ogihara. “Clustering Distributed Homogeneous Datasets. Proceedings from The Fourth European Conference on Principles of Data Mining and Knowledge Discovery”. Springer-Verlag, London, 2000.

[17] A. K. Sangaiah, A. E. Fakhry, and M. Abdel-Basset, “Arabic text clustering using improved clustering algorithms with dimensionality reduction,” Cluster Computing, vol. 22, no. 2, pp. 1–15, 2019.

[18] S. SoukainaMjahed, K. Bouzaachane, A. Taher Azar, S. El Hadaj, and S. Raghay, “Hybridization of fuzzy and hard semi supervised clustering algorithms tuned with ant lion optimizer applied to Higgs boson search,” Computer Modeling in Engineering & Sciences, vol. 125, no. 2, pp. 459–494, 2020.

[19] J. J. Jay, J. Eblen, and Y. Zhang, “A systematic comparison of genome-scale clustering algorithms,” Bmc Bioinformatics, vol. 13, no. 10, pp. 1–12, 2012.

[20] M. S. Yang and K. P. Sinaga, “A feature-reduction multi-view k-means clustering algorithm,” IEEE Access, vol. 9, p. 1, 2019.

[21] J. Song, X. Li, and Y. Liu, “An optimized k-means algorithm for selecting initial clustering centers,” International Journal of Security and Its Applications, vol. 9, no. 10, pp. 177–186, 2015.

[22] H. B. Zhou and J. T. Gao, “An improved initial clustering center selection method for K-means algorithm,” Advanced Materials Research, vol. 1022, pp. 337–340, 2014.

[23] R. Haraty, M. Dimishkieh and M. Masud, “An Enhanced k-Means Clustering Algorithm for Pattern Discovery in Healthcare Data”, June 2015.

[24] P. Berkhin. “Survey of clustering data mining techniques”. (pp 25-71). Sunnyvale, CA. Grouping Multidimensional Data, 2006.

[25] F. Samatova, G. Ostrouchov, A. Geist, and A. Melechko. “RACHET: An Efficient Cover-Based Merging of Clustering Hierarchies from Distributed Datasets”. TN, United States, 2002.

[26] T. Hess, M. Moshkovitz, and S. Sabato. “A constant approximation algorithm for sequential no-substitution k-median clustering under a random arrival order”. arXiv preprint arXiv:2102.04050, 2021.

[27] J. Judith, j. Jayakumari. “Distributed Document Clustering Analysis Based on a Hybrid Method”. China Communications., 14(02):131-142, 2017.

[28] R.C. Eberhart, J. Kennedy. “A new optimizer using particle swarm theory” Proceedings of the 6th international symposium on micro machine and human science, pp 39–43, Nagoya, Japan, Mar 13–16, 1995.

[29] F. Van Den Bergh. “An analysis of particle swarm optimizers”. Ph.D. dissertation, University of Pretoria, Pretoria, South Africa, 2001.

[30] X. Xie, X. Li, L. Mo. “Microblog public opinion analysis based on improved kmeans algorithm”. Computer engineering and science., 40 (01): 155-158, 2018.

[31] J.Vaidya, C. Clifton. “Privacy-Preserving K-Means Clustering Over Vertically Partitioned Data[C] Proceedings of ACM SIGKDD03” pp. 206-215, 2003.

[32] X. Lin, C. Clifton, M. Zhu. “Privacy Preserving Clustering with Distributed EM Mixture Modeling[J]. Knowledge and Information Systems” 8(1): 68-81, 2005.

[33] L. Xiong, S. Chitti and L. Liu, “Mining Multiple Private Databases Using a kNN Classifier, AMC SAC”, pp 435-440, 2007.

[34] B. Wang, X. Yu. “Parallel K-means clustering algorithm for adaptive cuckoo search”. Computer application research., 35 (3): 675-679, 2018.

[35] X.S. Yang, “Nature-Inspired Metaheuristic Algorithms”, Luniver Press, 2008.

[36] Y. Xin-She, S. Deb, “Cuckoo search via lévy flights”, World Congress on Nature & Biologically Inspired Computing, 2009.

[37] C. Cobos et al. “Clustering of web search results based on the cuckoo search algorithm and Balanced Beyasian Information Criterion”. June 2014.

[38] S. Goel, A. Sharma, P. Bedi, “Cuckoo search clustering algorithm: a novel strategy of biomimicry”, World Congress on Information and Communication Technologies (WICT), 2011, pp. 916–921, 2011.

[39] J. Senthilnath, V. Das, S.N. Omkar, V. Mani, “Clustering using levy flight cuckoo search”, J.C. Bansal, P. Singh, K. Deep, M. Pant, A. Nagar (Eds.), Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012), Springer, India, 2013.

[40] Q.H. Nguyen, Y.S. Ong, N. Krasnogor, “A study on the design issues of Memetic Algorithm”, IEEE Congress on Evolutionary Computation, 2007.

[41] E. Alba, “Parallel Metaheuristics: A New Class of Algorithms”, Wiley-Interscience, 2005.

[42] E. Alba, M. Tomassini, “Parallelism and evolutionary algorithms”, IEEE Trans. Evol. Comput. 2002.

[43] G. Luque, E. Alba, “Parallel Genetic Algorithms: Theory and Real World Applications”, Springer Berlin Heidelberg, 2011.

[44] S. Osin´ ski, D. Weiss, “A concept-driven algorithm for clustering search results”, Intell. Syst. 20, 2005.

[45] Z. Tang, K. Liu, J. Xiao, et al. “A parallel k-means clustering algorithm based on redundance elimination and extreme points optimization employing MapReduce”. Concurrency and Computation., 29(20):e4109.1-e4109.18, 2017.

[46] M. Zaharia, et al. “Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing”. In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (pp. 2–2). USENIX Association, 2012.

[47] T. White. “Hadoop: The Definitive Guide”. O’Reilly Media, 2011.

[48] M. Zaharia. “An Architecture for Fast and General Data Processing on Large Clusters”. Technical Report UCB/EECS-2014-12, University of California, Berkeley (Technical Report), 2014.

[49] H. M. Kakde. “Range Searching using Kd Tree” (Online). http://www.cs.utah.edu/lifeifei/cs6931/kdtree.pdf, August 25, 2005.