REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre : RTIC/M2/2020

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Réseaux et Technologies de l'information et de la communication.**

Realization of a Parallel Clustering method Application to agricultural soil management control

Par :

BOUBAKEUR OUMAIMA ANFEL

Soutenu le date 2020, devant le jury composé de :

Nom et prénom	Grade	Président
Nom et prénom	Grade	Rapporteur
Nom et prénom	Grade	Examinateur

Acknowledgment

*All my gratitude for **Allah** the Almighty who gave me the strength and courage to reach this level.*

*My deep and sincere thanks to my supervisor **PR. SAOULI Rachida** who has been source of motivation and inspiration throughout this journey, and who with her help and support gave me the opportunity to carry this project to the light: your attention has been priceless..*

*I appreciate the enormous help of **Mr.Moustefaoui Tewfik** throughout the progress of this project for providing information needed to accomplish my work, and being available and attentive to my questions.*

*My Father **Abdelhak**, my mother **Nora**, My little brother **Naoufel**: I owe you the progress I made in my study career, without you I wouldn't have reached the person I am today, the fruit of my work is dedicated to you…*

To who has always been on my side, and helped me through my journey…

To all my friends…

This work is for you

Abstract

Agricultural techniques and treatments are usually applied on a plot of land although the depth and condition of the soil is not uniform. For this, we aim in this project, to use the clustering algorithm in order to divide the soil into groups of fields according to its parameters allowing farmers to have more precise information on the crop to use. Then, we propose a parallel technic based on k-means algorithm to provide an efficient technic that subdivide the soil into parts with different properties. In this case, we use a total and partial fusion technics leading to respectively the use of hierarchical or arbitrary merging between the clusters. To implement our split soil module we based on agricultural data base provided by the Direction of the Research Division "Soil Resource Management and Utilization in Arid Regions" of the CRSTRA (Centre for Scientific and Technical Research on Arid Regions) and we used MPI (Message passing Information) API on IBN KHALDOUN HPC (High performance Computer) of university of BISKRA. We show through the obtained results that the sequential implementation is very heavy time consuming compared to the parallel on 30 values per parameter in the used dataset.

Keywords: Agricultural techniques, soil parameters, k-means algorithm, parallel calculation.

Résumé

Les techniques et traitements agricoles sont généralement appliqués sur une parcelle de terrain bien que la profondeur et les conditions du sol ne soient pas uniformes. Pour cela, nous visons dans ce projet, à utiliser l'algorithme de clustering afin de diviser le sol en groupes de champs en fonction de ses paramètres permettant aux agriculteurs d'avoir des informations plus précises sur la culture à utiliser. Pour réaliser notre objectif, nous proposons une technique parallèle basée sur l'algorithme *k-means* pour fournir une technique efficace qui subdivise le sol en parties aux propriétés différentes. Dans ce cas, nous proposons une technique de fusion totale et partielle conduisant respectivement à l'utilisation de fusions hiérarchiques ou arbitraires entre les clusters. Pour implémenter notre module *Split Soil*, nous nous sommes basés sur la base de données agricole fournie par la Direction de la Division de la Recherche «Gestion et utilisation des ressources en sols dans les régions arides» du CRSTRA (Centre de recherche scientifique et technique sur les régions arides) en utilisant l'API MPI (Message Passing Information) sur la machine cluster IBN KHALDOUN HPC (High Performance Computer) de l'université de BISKRA. Nous montrons à travers les résultats obtenus que l'implémentation séquentielle prend beaucoup de temps par rapport au parallèle sur 30 valeurs par paramètre dans la base de données utilisée.

Mots clés: Techniques agricoles, paramètres du sol, algorithme k-means, calcul parallèle.

List of Figures

List of Tables

List of Algorithms

Content

General Introduction

General Introduction

Throughout human history, our relationship with the soil has affected our ability to cultivate crops and influenced the success of civilizations. This relationship between humans, the earth, and food sources affirms soil as the foundation of agriculture. [1]

The way people use land can affect the levels of nutrients and pollution in soil. Any activity that exposes soil to wind and rain can lead to soil loss. Farming, construction and development, and mining are among the main activities that affect soil resources [2]. For that special aim, the agronomical scientists have done a classification for soils to prevent and protect the soils by grouping them.

Soil Classification concerns the grouping of soils with a similar range of properties (chemical, physical and biological) into units that can be geo-referenced and mapped.

All the data collected in the field or drawn from a soil map and interpreted with discernment allow the soils to be grouped and divided into a certain number of classes of possibilities or use values, in other words, classify the different soils in order of decreasing agricultural possibilities.

The judgment of appreciation relates to the global characteristics of the ground. These traits or qualities are « measured ».

Soils contain all naturally occurring chemical elements and combine simultaneously solid, liquid and gaseous states. Moreover, the number of physical, chemical and biological characteristics and their combinations are nearly endless. No wonder then that many different approaches have been proposed to come to a sensible grouping of different soils. [3]

This relation between humans and soil affected the agronomical nutritional properties of lands, rather than that, the misuse of agricultural fields by the farmer that comes from not knowing nor can well exploiting his soils lead to many problems that can threat any country's food security.

For this purpose, we have proposed a soil inspector method to help introduce each farmer to his field through exploiting machine learning methods in order to improve crop yields and guarantee both sustainability of cultivations and nutritional requirements of each land.

This project passed in the making by three chapters, we introduce them as follow:

- **Chapter I: State of the art.**

This chapter is the literary part of this project, where we first presented the international classifications of soils and a definition of the machine learning method: The clustering.

We have finally ended the chapter with a synthesis of previous done works in the same domain.

- **Chapter II: Cluster computing.**

The second chapter is a detailed description that provides an inside out vision of clusters, from introducing its emergence in the world of computing to detailing its structure and network components.

- **Chapter III: Conception and Implementation.**

In the third chapter, we passed from the theoretical aspect to the practical aspect when we first defined the conception of K-means algorithm in both its methods (Sequential & Parallel).

Second, we have showed a detailed implementation of the parallel program on the collected data that had been also well explained and demonstrated in examples.

Last, we shows the execution of our application on data by presenting screenshots of the process to finally giving results by illustrating graph plots and comparison tables.

Chapter I

State of the art

I.1. Introduction

Delineation or determination of management zones has been used as a method for sub-diving fields into parts with different properties for a long time. However this has been done earlier using experts and long term knowledge of the respective field which consumes a lot of time that was not always available, because of the delineation process that is required before the growing season.

For this purpose the use of Machine Learning methods took place in precision Agriculture to reduce time also to provide exact information and results. In this chapter, we will describe the precision Agriculture in the first place, second we will talk about the traditional categorised types of soils.

Thirdly a detailed presentation about the machine learning method « Clustering » will be done to pave our way to finally present a work synthesis that talks about the use of clustering in precision agriculture.

I.2. Precision Agriculture

« In fact, a recent study by Hexa Reports suggests precision agriculture is set to grow to $43.4 billion by 2025. For a concept that was born in the 1990s, that's quite impressive. » Remi Schmaltz (AgFunder networks) [4]

Figure I.1: Modern Agriculture.

Precision agriculture seeks to use new technologies to increase crop yields and profitability while lowering the levels of traditional inputs needed to grow crops (land, water, fertilizer, herbicides and insecticides). In other words, farmers utilizing precision agriculture are using less to grow more. GPS devices on tractors, for instance, allow farmers to plant crops in more efficient patterns and proceed from point A to point B with more precision, saving time and fuel. Fields can be levelled by lasers, which means water can be applied more efficiently and with less farm effluent running off into local streams and rivers. The result can be a boon for farmers and holds great potential for making agriculture more sustainable and increasing food availability. [5]

I.2.1. Precision Agronomics [4]

Precision agronomics is another important term related to the combining of methodology with technology. At its core, it’s about providing more accurate farming techniques for planting and growing crops. Precision agronomics can involve any of the following elements:

Variable rate technology (VRT) – VRT refers to any technology that enables the variable application of inputs and allows farmers to control the amount of inputs they apply in a specific location. The basic components of this technology include a computer, software, a controller and a differential global positioning system (DGPS). There are three basic approaches to using VRT – map-based, sensor-based and manual.

Computer-based applications - Computer applications can be used to create precise farm plans, field maps, crop scouting and yield maps. This, in turn, allows for the more precise application of inputs such as pesticides, herbicides, and fertilizers,thus helping to reduce expenses, produce higher yields and create a more environmentally-friendly operation.

The challenge with these software systems is they sometimes deliver a narrow value that doesn’t allow data to be used for making bigger farm decisions, especially with the support of an expert. Another concern with many software applications is poor user interfaces, and the inability to integrate the information they provide with other data sources to enrich and show significant value to farmers.

Remote sensing technology – Remote sensing technology has been in use in agriculture since the late 1960s. It can be an invaluable tool when it comes to monitoring and managing land, water, and other resources.

It can help determine everything from what factors may be stressing a crop at a specific point in time to estimating the amount of moisture in the soil. This data enriches decision-making on the farm and can come from several sources including drones and satellites.

GPS soil sampling – Testing a field’s soil reveals available nutrients, pH level, and a range of other data that is important for making informed and profitable decisions. In essence, soil sampling allows growers to consider productivity differences within a field and formulate a plan that takes these differences into account. Collection and sampling services that are worth the effort will allow the data to be used for input for variable rate applications for optimizing seeding and fertilizer.

I.2.2. Categories of soils [6]

The classification of soils based on their criteria allows their grouping into four main categories: crop specific soils, not very clean or unsuitable for cultivation and non-agricultural land. These are divided into six classes intersected in subclasses.

A. Class 1 : Very good qualitysoils

Figure I.2: Excellent soil.

In general, these soils are deep, fine to medium in texture, ranging from clay to silt, having a **good structure**, **suitable pH**, **flat topography** or **slightly slope, sufficient drainage**. All these favorable factors allow the most intensive use without special techniques. These soils are located in a physical environment suitable for growing most of the plants in the region concerned.

- **Subclasse la :** Soils of very good quality which can benefit from ordinary drainage or stone removal works and / or require the periodic contribution of calcareous or humid amendments and fertilizing elements at maintenance doses, to adapt to crop needs.

B. Class 2 : Good quality soils

Generally lighter in texture than « subclass la », suitable for cultivation, without special development measures, or even soils of very good quality, but whose culture is conditioned by certain obstacles or unfavourable physical properties, such as the presence of stones, a defect drainage or risk of slight erosion.

- **Subclass 2a:** Good quality soil, generally less fertile than « subclass la » sensitive to erosion, requiring the use of amendments or maintenance fertilizers at doses, requiring only ordinary sanitation and / or stone removal.

- **Subclass 2b:** Good quality soils, generally lighter in texture than 2a.

- **Subclass 2c:** Very good quality soils, requiring fairly high costs of stone removing.

- **Subclass 2d:** Very good quality soils, requiring major operations of sanitation.

- **Subclass 2e:** Soils of very good quality, but which require ordinary anti-erosive precautions.

- **Subclass 2i:** Very good quality soils requiring the construction of dikes to protect them against flooding.

- **Subclass 2s:** Very good soils that require appropriate contributions to correct or alleviate physical conditions.

C. Class 3: **Medium quality soils** (silty sand to gravelly sand)

Figure I.3: Medium quality soil.

Also good quality soils, but its use subordinates an important limiting factor, and / or several minor important limiting factors. Sometimes even very good soils, but which the use is conditioned by a permanent unfavourable factor.

The floors of this class, in general, still offer good cultural possibilities.

- **Subclass 3a:** Medium quality soils, with a rather light texture (silty sands or fine sands).

- **Subclass 3b:** Medium quality soils; consider significant additions of fertilizers or even a heavy use of green manures.

- **Subclass 3c:** Good quality soils where the abundance of stones requires important work and expensive stone removal.

- **Subclass 3d:** Good quality soils requiring significant and costly drainage.

- **Subclass 3e:** Good soils, but exposed to slight or moderate erosion.
- **Subclass 3f:** Soils of very good quality, but whose use requires more important anti-erosion measures than those classified in 2e.

- **Subclass 3i:** Good quality soils whose development requires flood protection.

- **Subclass 3s:** Good to medium quality soils that require tillage or appropriate inputs to correct certain physical properties.

D. Class 4: Soils of poor quality

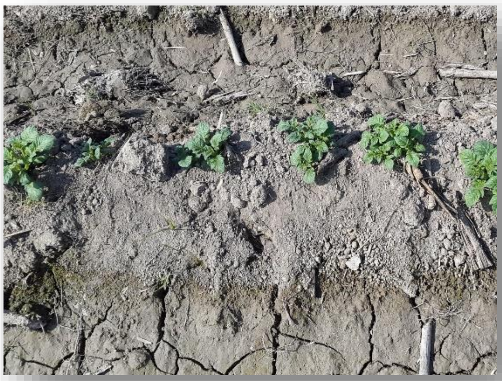


Figure I.4: Poor quality soil.

Which generally have little cultural possibilities (under special or intermittent crops these soils require significant additions of fertilizers and nutrients).

- **Subclass 4a:** Soils of average quality, but requiring simple conservation measures, or important land development works. Also, good soils and of very good quality, depending on whether they are seriously or very seriously limited by certain unfavourable factors.

- **Subclass 4b:** Soils of poor quality, which need frequent additions or important amendments and fertilizers.

- **Subclass 4c:** Medium quality soils where large and costly works stone removal are required.

- **Subclass 4d:** Medium quality soils requiring significant drainage.

- **Subclass 4e:** Medium quality soils to which simple conservation measures must be applied.

- **Subclass 4f:** Good quality soils, requiring significant protection or conservation measures.

- **Subclass 4g:** Very good quality soils, the use of which is severely limited by a very uneven topography.

- **Subclass 4i:** Medium quality soils including the erection of dikes would allow the development.

- **Subclass 4s:** Medium quality soils that require humid or calcic amendments or burial to remedy a permeability excess.

E. Class 5 :Poor to good quality soils

Where cultivation possibilities are zero either because of an excessive slope, or because of the abundance of stones or rock outcrops, either due to very advanced degradation, or too thin soil.

- **Subclass 5f**: Medium quality soils where the slope causes severe erosion.
- **Subclass 5g**: Good to medium quality soils whose use, due to very uneven topography, is restricted to grazing or forest production.
- **Subclass 5h**: Soils difficult to access, with steep or complex slopes.
- **Subclass 5m**: Thin soils on rock; subject to expert cases, but in general to restore or maintain natural vegetation or to develop pastures.

F. Class 6 :Soils to be kept under natural cover

Figure I.5: Soils of natural covers.

This group class soils that are too stony, too uneven, too prone to erosion very severe, and the wooded banks of rivers. These areas are considered, rather undercurrent conditions, from the point of view of their adaptation to the requirements of forestry than to those of agriculture.

- **Subclass 6t**: tourism and recreation.
- **Subclass 6x** : various assignments.
- **Subclass 6r**: very rough terrain.

I.3. Clustering methods in precision agriculture [7]

I.3.1. Clustering

In machine learning, we often group examples as a first step to understand a subject (data set) in a machine learning system. Grouping unlabelled examples is called clustering**.**

As the examples are unlabelled, clustering relies on unsupervised machine learning. If the examples are labelled, then clustering becomes classification**.**

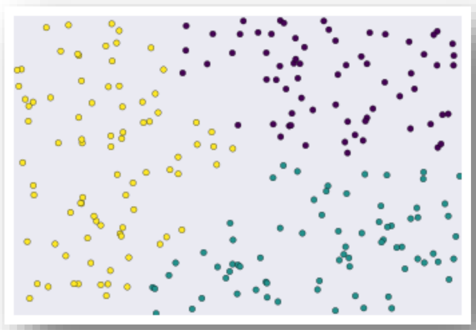


Figure I.6: Unlabelled examples grouped into three clusters.

Similarity can be measured between examples by combining the examples' feature data into a metric, called a similarity measure. When each example is defined by one or two features, it's easy to measure similarity. For example, you can find similar books by their authors. As the number of features increases, creating a similarity measure becomes more complex. We'll later see how to create a similarity measure in different scenarios.

A. Clustering Workflow

To cluster data, these steps should be followed:

1. **Prepare data**

   As with any ML problem, you must normalize, scale, and transform feature data. While clustering however, you must additionally ensure that the prepared data lets you accurately calculate the similarity between examples. The next sections discuss this consideration.

2. **Creat similarity metric**

   Before a clustering algorithm can group data, it needs to know how similar pairs of examples are. You quantify the similarity between examples by creating a similarity metric. Creating a similarity metric requires you to carefully understand your data and how to derive similarity from your features.

3. **Run clustering algorithm**

   A clustering algorithm uses the similarity metric to cluster data.

4. **Interpret results and adjust your clustering**

   Checking the quality of your clustering output is iterative and exploratory because clustering lacks "truth" that can verify the output. You verify the result against expectations at the cluster-level and the example-level. Improving the result requires iteratively experimenting with the previous steps to see how they affect the clustering.

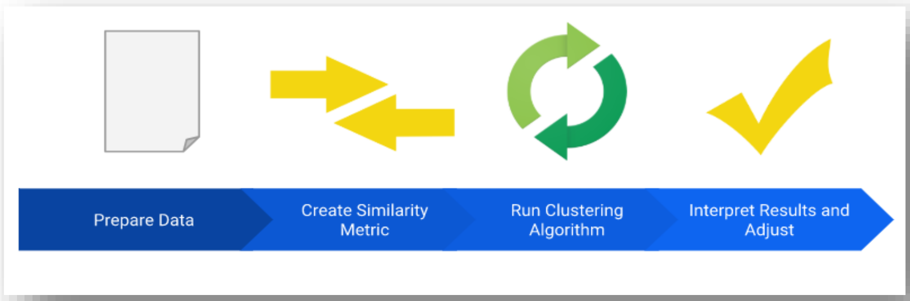


Figure I.7: Clustering workflow.

B. Clustering Methods

Clustering Algorithms are of many types. In the following only the most prominent examples of clustering algorithms are listed, as there are possibly over 100 published clustering algorithms. Not all provide models for their clusters and can thus not easily be categorized.

B.1 Density-based Clustering

Density-based clustering connects areas of high example density into clusters. This allows for arbitrary-shaped distributions as long as dense areas can be connected. These algorithms have difficulty with data of varying densities and high dimensions. Further, by design, these algorithms do not assign outliers to clusters.

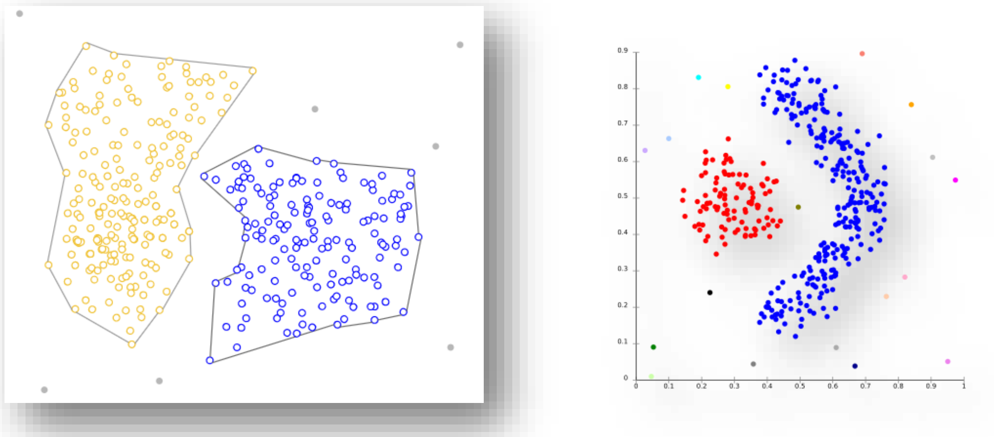


Figure I.8: Density based Clustering.

- The most well-known and used algorithm in this method is the DBSCAN.

DBSCAN algorithm requires two parameters:

1. **eps** : It defines the neighbourhood around a data point
2. **Min Pts**: Minimum number of neighbours (data points) within eps radius.

- In this algorithm, we have 3 types of data points:

Core Point: A point is a core point if it has more than Min Pts points within eps.

Border Point: A point which has fewer than Min Pts within eps but it is in the neighbourhood of a core point.

Noise or outlier: A point which is not a core point or border point.

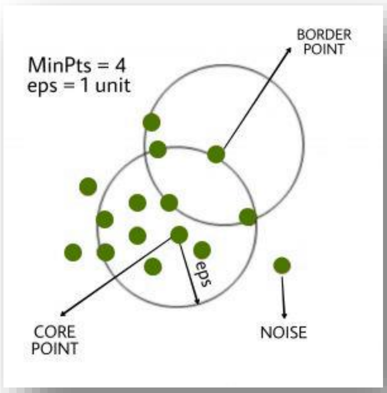


Figure I.9: DBSCAN essentials.

DBSCAN algorithm can be abstracted in the following steps:

1. Find all the neighbour points within eps and identify the core points or visited with more than Min Pts neighbours.

2. For each core point if it is not already assigned to a cluster, create a new cluster.

3. Find recursively all its density connected points and assign them to the same cluster as the core point.

4. Iterate through the remaining unvisited points in the dataset. Those points that do not belong to any cluster are noise.

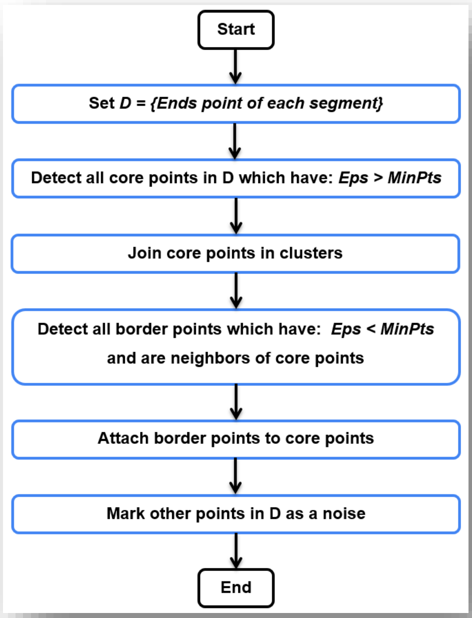

Start
Set *D* = *{Ends point of each segment}*
Detect all core points in D which have: *Eps* > *MinPts*
Join core points in clusters
Detect all border points which have: *Eps* < *MinPts* and are neighbors of core points
Attach border points to core points
Mark other points in D as a noise
End

Figure I.10: DBSCAN Flowchart.

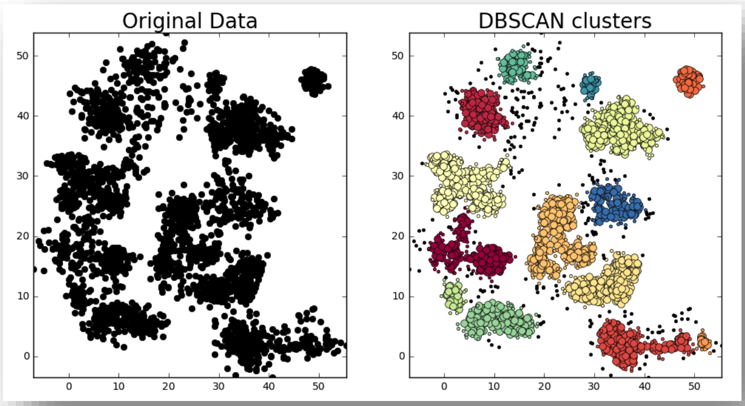


Figure I.11: DBSCAN Clusters.

B.2. Distribution based Clustering

This clustering approach assumes data is composed of distributions. The distribution-based algorithm clusters data into three Gaussian distributions. As distance from the distribution's center increases, the probability that a point belongs to the distribution decreases. The bands show that decrease in probability. When you do not know the type of distribution in your data, you should use a different algorithm.

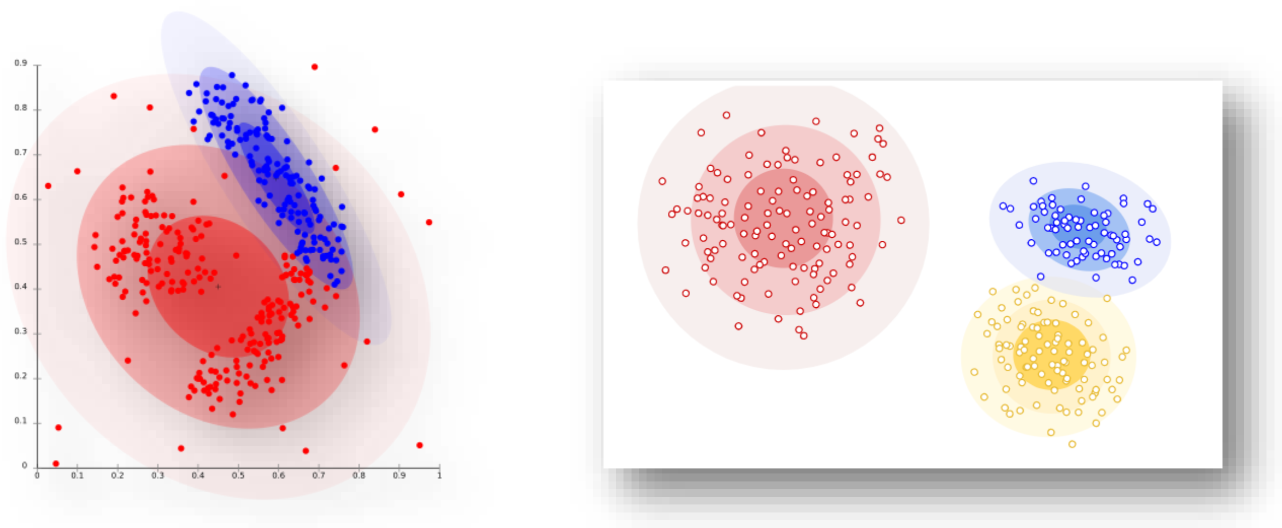


Figure I.12: Distribution based clustering.

- The most well-known and used algorithm for this method is the Expectation-Maximization Algorithm

- This algorithm has mainly four steps :

1. Given a set of incomplete data, consider a set of starting parameters.
2. **Expectation step (E – step):** Using the observed available data of the dataset, estimate (guess) the values of the missing data.
3. **Maximization step (M – step):** Complete data generated after the expectation (E) step is used in order to update the parameters.
4. Repeat step 2 and step 3 until convergence.

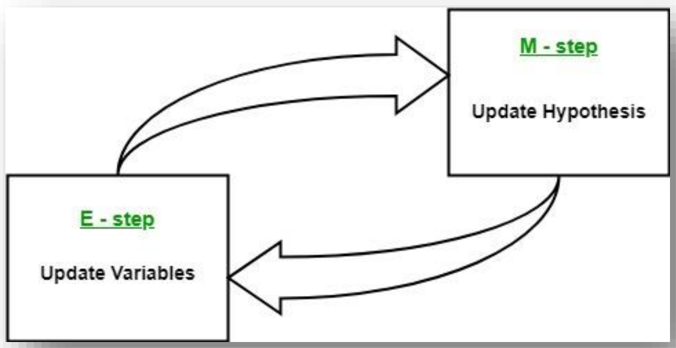


Figure I.13: EMA Algorithm.

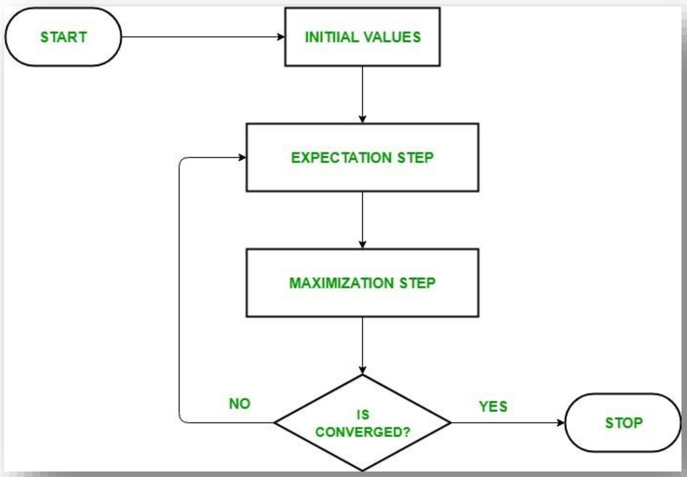


Figure I.14: EMA Algorithm flowchart.

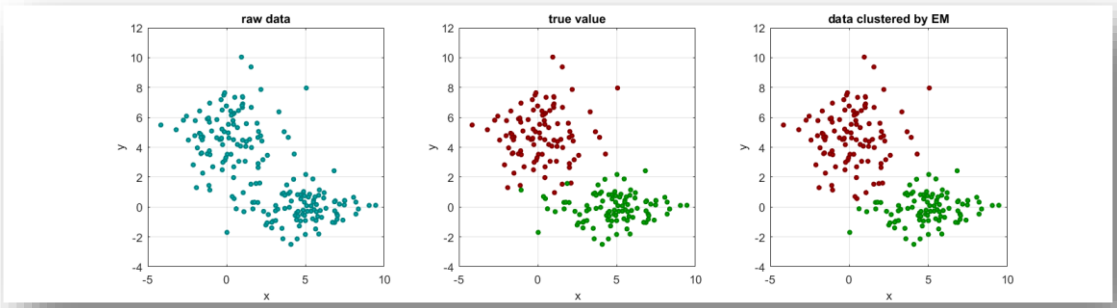


Figure I.15: Results of applying EM Algorithm.

B.3. Connectivity based Clustering

The core idea of connectivity based model is basically defining clusters on the basis of closeness of data points .Here we work on a notion that the data points which are closer have similar behaviour as compared to data points that are farther. It is not a single partitioning of the data set, instead it provides an extensive hierarchy of clusters that merge with each other at certain distances. The choice of distance function is subjective. These models are very easy to interpret but it lacks scalability.

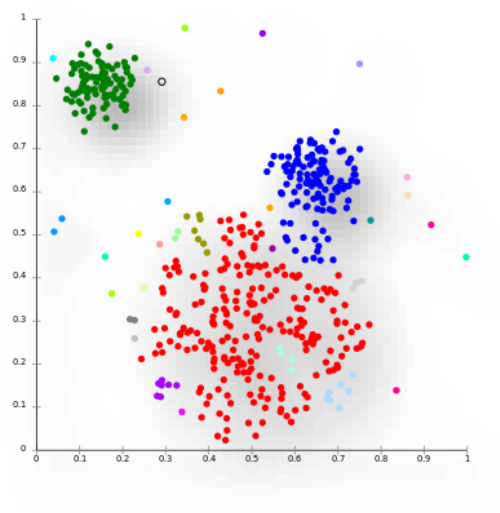


Figure I.16: Connectivity based Clustering.

- Its most used and well known algorithm is the hierarchical algorithm and its variants.

B.4. Hierarchical Clustering (known as the hard clustering)

This method mainly deals with the task of partitioning a set of entities into a number of homogenous clusters, with respect to a suitable similarity measure.

B.4.1. Hierarchical agglomerative clustering

Also known as the bottom-up approach, that deals with partitioning tasks.

Starts by placing each object in its own cluster .Than merge these clusters into successively larger clusters, until all of the objects are in a single cluster or a certain termination condition is satisfied.

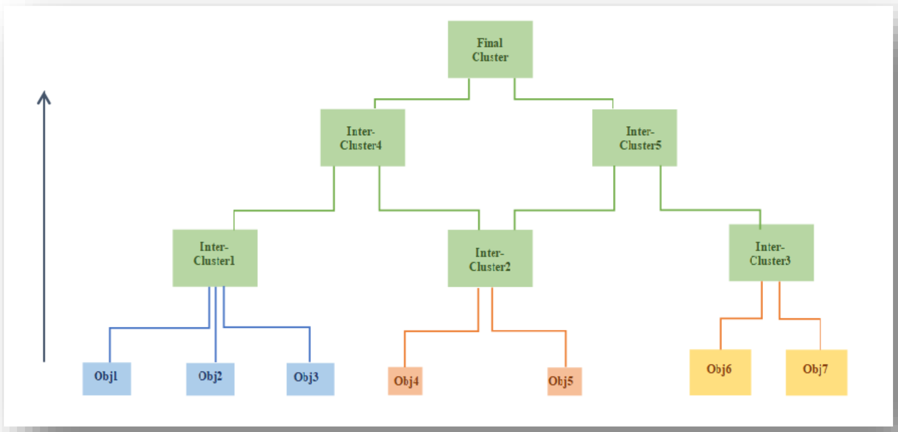


Figure I.17: Hierarchical agglomerative clustering.

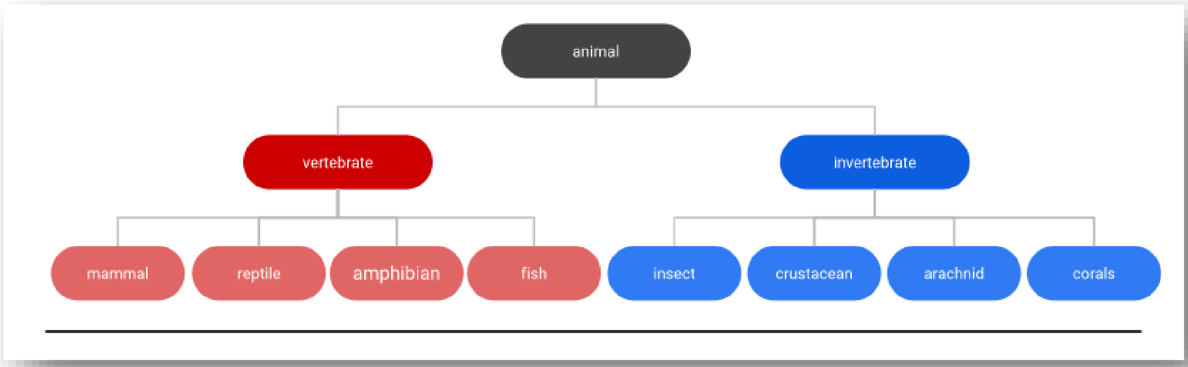


Figure I.18: Example of hierarchical agglomerative clustering.

B.4.2. Hierarchical Divisive Clustering

Known to be the top-down approach, starts with all objects within the same cluster.

In successive iterations, each cluster is split up into smaller clusters until each object is placed in its own cluster or a termination condition holds.

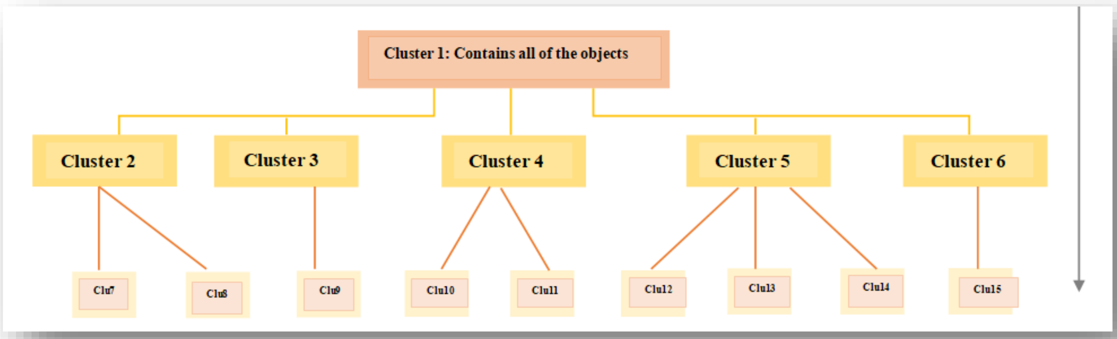


Figure I.19: Hierarchical divisive clustering.

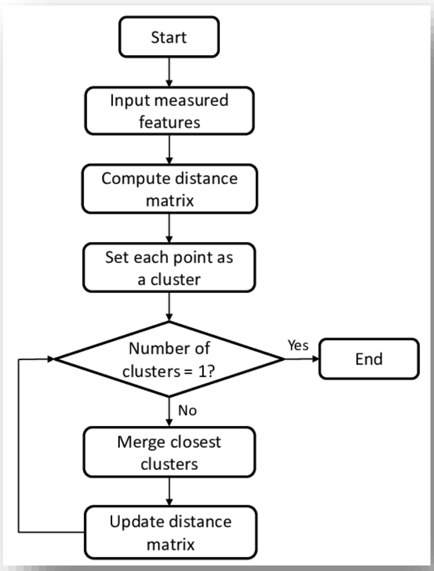


Figure I.20: Flowchart of Hierarchical clustering.

B.5. Centroid-based Clustering

Centroid-based clustering organizes the data into non-hierarchical clusters, in contrast to hierarchical clustering defined below. K-means is the most widely-used centroid-based clustering algorithm. Centroid-based algorithms are efficient but sensitive to initial conditions and outliers.

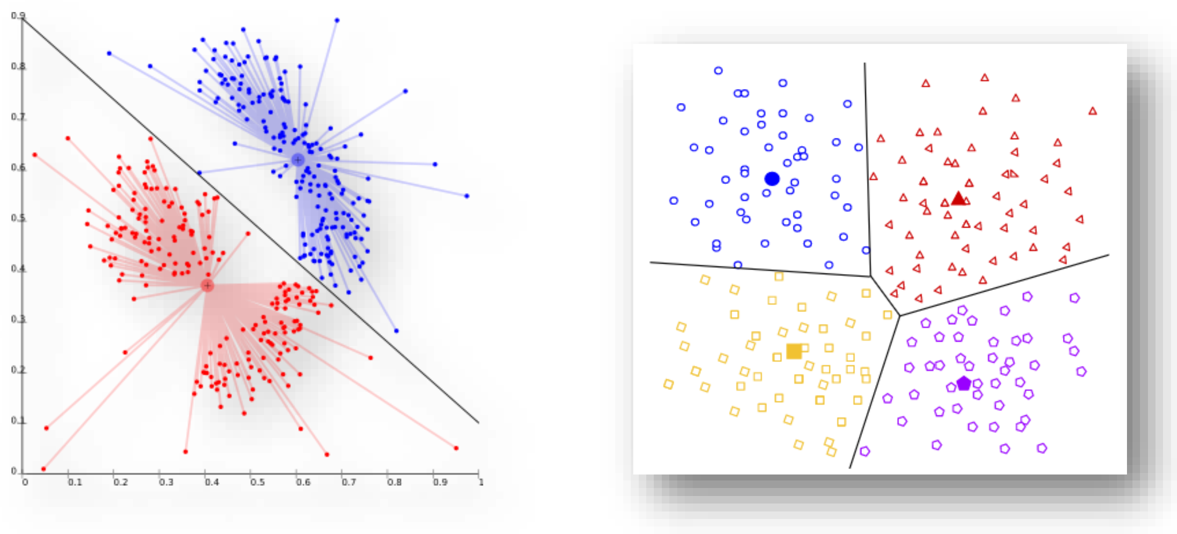


Figure I.21: Centroid-based Clustering.

The most well-known algorithm in the centroid based clustering is k-means algorithm.

A target number *k* should be defined, which refers to the number of centroids needed in the dataset. A centroid is the imaginary or real location representing the center of the cluster .Every data point is allocated to each of the clusters.

In other words, the K-means algorithm identifies *k* number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.

The *'means'* in the K-means refers to averaging of the data; that is, finding the centroid.

- Steps of k-means algorithm:

1. Specify number of clusters *K*.
2. Initialize centroids by first shuffling the dataset and then randomly selecting *K* data points for the centroids without replacement.
3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.

4. Compute the sum of the squared distance between data points and all centroids.
5. Assign each data point to the closest cluster (centroid).
6. Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

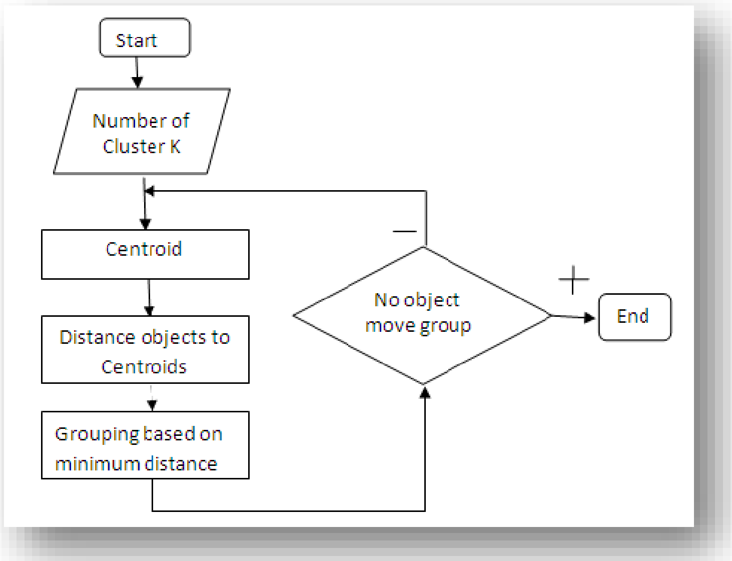

Start
Number of Cluster K
Centroid
Distance objects to Centroids
Grouping based on minimum distance
No object move group
—
+
End

Figure I.22: Flowchart of k-means clustering algorithm.

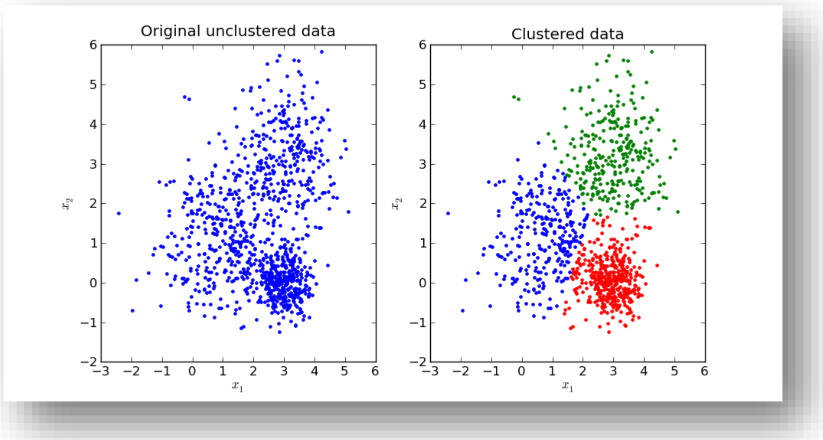


Figure I.23: Example of applying k-means clustering algorithm.

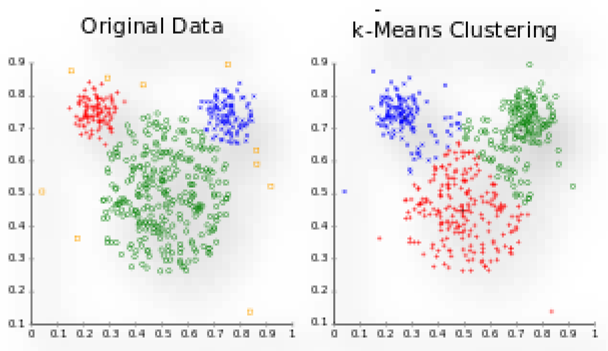


Figure I.24: k-means clustering algorithm applied on “mouse” dataset.

I.4. Work synthesis

I.4.1. Fuzzy Clustering in plant disease detection

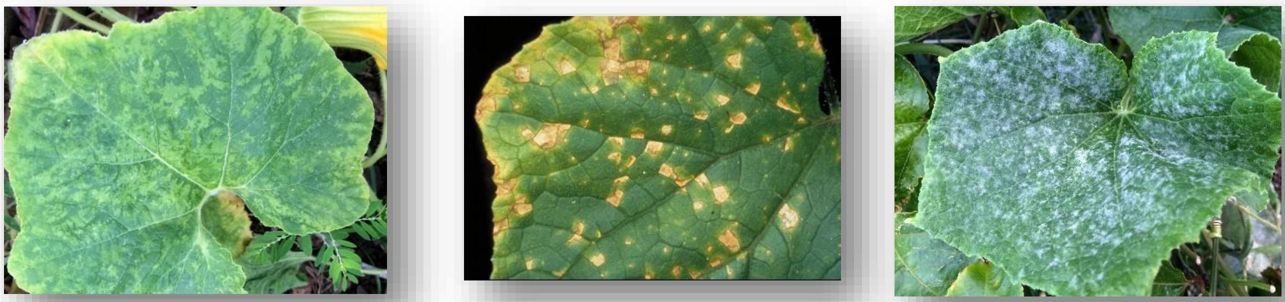


Figure I.25: Samples of healthy and mildew affected cucumber leaves.

The work referenced in [8] is an application of fuzzy clustering in detection of plant disease by using image analysis of cucumber leafs.

This method was initiated to investigate cucumber leafs diseases and identifying leaf batches in the crop.

To do so, 90 images were taken and categorised into 3 main mildews found in the cucumber leaf, which are:

1. Powdery Mildew.

2. Downy Mildew.

3. Leaf miner.

Than fuzzy c-mean algorithm was applied on those 3 classes following this flowchart, the reason of using fuzzy clustering is due to the fuzzy nature of plant leaves.

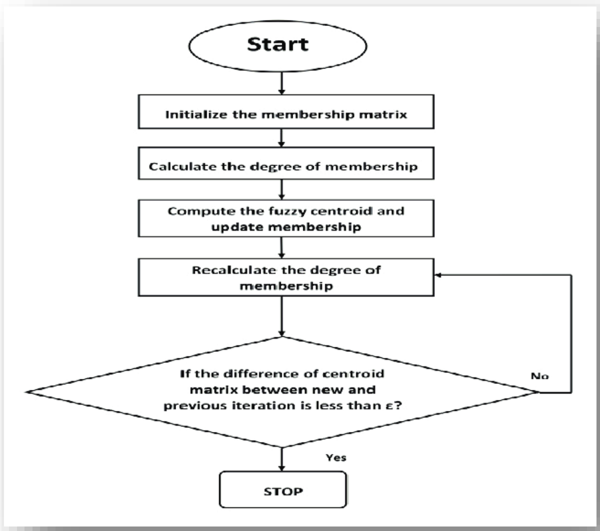

Start
Initialize the membership matrix
Calculate the degree of membership
Compute the fuzzy centroid and update membership
Recalculate the degree of membership
If the difference of centroid matrix between new and previous iteration is less than ε?
No
Yes
STOP

Figure I.26: Flowchart of fuzzy c-means algorithm.

The success of the application depended on adapting the right input parameters which are :

- Features of data set.
- Optimal number of clusters.
- Degree of fuzziness.

I.4.2. Hierarchical Agglomerative Clustering in soil delineation

Figure I.27: Samples of delineated soil maps.

The work proposed in [9] is an application of Agglomerative Clustering for management zone delineation in precision agriculture

In this approach, a two-step process has been developed:

First Step: an overlayed grid to achieve spatially data partitioning. Due to the irregularities in the field shape and gapes also data density, they used a K-means algorithm (that is mentioned previously) on the coordinates of the data set points to provide a more flexible and close solution to the initial tessellation.

Second Step: Repetitively merging of two zones according to 2 main conditions:

Similarity in attributes of the tow to be merged zones: to assure homogeneity of subdiving.

The zones to be merged must be direct neighbours in geographical space: to check contiguousity.

- **How to merge spatially ?**

The merging process is done by generating a list of neighbours for each cluster and updating it following 3 ways:

-Single linkage: determine the data records which are most similar in tow candidate clusters and merge the clusters containing these records.

- Complete linkage: determine the data records which are most dissimilar in tow candidate clusters and merge those clusters.

-Average Linkage: determine the average vector in data records in one cluster and would then merge those clusters which are closest according to the attribute distance.

I.5. Conclusion

In this chapter, we have presented details about the agronomic and agricultural part of this project, and defined the clustering with revealing its most used and well-known methods.

This literary section was a starter to know best the environment of our work, to pave the way to the next chapters into introducing the practical parts.

Chapter II

Cluster Computing

II.1 Introduction

The first inspiration for cluster computing was developed in the 1960s by IBM as an alternative of linking large mainframes to provide a more cost effective form of commercial parallelism. Cluster computing did not gain momentum until the convergence of three important trends in the 1980s: high-performance microprocessors, high-speed networks, and standard tools for high performance distributed computing. A possible fourth trend is the increasing need of computing power for computational science and commercial applications coupled with the high cost and low accessibility of traditional supercomputers. In this section, we describe the faster computers that allows to solve larger problems, and to find solutions more quickly, with greater accuracy, and at a lower cost.

II.2 Modern Computing

When computing, there are three basic approaches to improving performance use a better algorithm, use a faster computer, or divide the calculation among multiple computers. First, consider what we are trying to calculate. All too often, improvements in computing hardware are taken as a license to use less efficient algorithms, or to perform meaningless or redundant calculations rather than carefully defining the problem. Selecting appropriate algorithms is a key way to eliminate instructions and speed up a calculation.

The parallelism lead to execute instructions simultaneously. There is a variety of ways to achieve this. At one end of the spectrum, parallelism can be integrated into the architecture of a single CPU. At the other end of the spectrum, you may be able to divide the computation up among different computers on a network, each computer working on a part of the calculation, all working at the same time [7].

II.2.1 Uniprocessor Computers

The traditional classification of computers based on size and performance, i.e., classifying computers as microcomputers, workstations, minicomputers, mainframes, and supercomputers, has become obsolete. The ever-changing capabilities of computers means that today's microcomputers now outperform the mainframes of the not-too-distant past. Furthermore, this traditional classification scheme does not readily extend to parallel systems and clusters but. Nonetheless, it is worth looking at the capabilities and problems associated with traditional computers, since these will be used to assemble clusters.

Regardless of where we place them in the traditional classification, most computers today are based on an architecture often attributed to the Hungarian mathematician John von Neumann. The basic structure of a von Neumann computer is a CPU connected to memory by a communications channel or bus. Instructions and data are stored in memory and are moved to and from the CPU across the bus. The overall speed of a computer depends on both the speed at which its CPU can execute individual instructions and the overhead involved in moving instructions and data between memory and the CPU.

Several technologies are currently used to speed up the processing speed of CPUs. The development of reduced instruction set computer (RISC) architectures and has led to more uniform instruction sets. This eliminates cycles from some instructions and allows a higher clock-rate. The use of RISC technology and the steady increase in chip densities provide great benefits in CPU speed. Superscalar architectures and pipelining have also increased processor speeds. Superscalar architectures execute two or more instructions simultaneously. Pipelining overlaps the different phase of instruction execution like an assembly line.

Improvements in memory bandwidth have not kept up with CPU improvements. It doesn't matter how fast the CPU is theoretically capable of running if you can't get instructions and data into or out of the CPU fast enough to keep the CPU busy. Consequently, memory access has created a performance bottleneck for the classical von Neumann architecture.

Computer architects and manufacturers have developed a number of techniques to minimize the impact of this bottleneck. Computers use a hierarchy of memory technology to improve overall performance while minimizing cost. Frequently used data is placed in very fast cache memory, while less frequently used data is placed in slower but cheaper memory. Another alternative is to use multiple processors so that memory operations are spread among the processors. If each processor has its own memory and its own bus, all the processors can access their own memory simultaneously.

II.2.2 Multiple Processors

Traditionally, supercomputers have been pipelined, superscalar processors with a single CPU. In recent years, we have come to augment that definition to include parallel computers with hundreds or thousands of CPUs, otherwise known as multiprocessor computers. Multiprocessor computers fall into two basic categories—centralized multiprocessors (or single enclosure multiprocessors) and multi-computers.

- **Centralized multiprocessors**

With centralized multiprocessors, there are two architectural approaches based on how memory is managed—uniform memory access (UMA) and non-uniform memory access (NUMA) machines. With UMA machines, also called symmetric multiprocessors (SMP), there is a common shared memory. Identical memory addresses map, regardless of the CPU, to the same location in physical memory. Main memory is equally accessible to all CPUs, as shown in Figure II.1 To improve memory performance, each processor has its own cache.

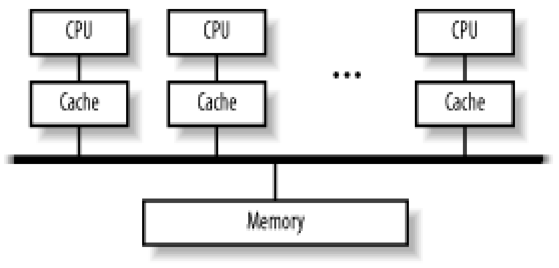


Figure II.1: UMA architecture

There are two closely related difficulties in designing a UMA machine. The first problem is synchronization. Communications among processes and access to peripherals must be coordinated to avoid conflicts. The second problem is cache consistency. If two different CPUs are accessing the same location in memory and one CPU changes the value stored in that location, then how is the cache entry for the other

CPU updated. Several techniques are available such as snooping where each cache listens to all memory accesses. If a cache contains a memory address that is being written to in main memory, the cache updates its copy of the data to remain consistent with main memory.

A closely related architecture is used with NUMA machines. Roughly, with this architecture, each CPU maintains its own piece of memory, as shown in Figure 2. Effectively, memory is divided among the processors, but each process has access to all the memory. Each individual memory address, regardless of the processor, still references the same location in memory. Memory access is non uniform in the sense that some parts of memory will appear to be much slower than other parts of memory since the bank of memory "closest" to a processor can be accessed more quickly by that processor. While this memory arrangement can simplify synchronization, the problem of memory coherency increases.

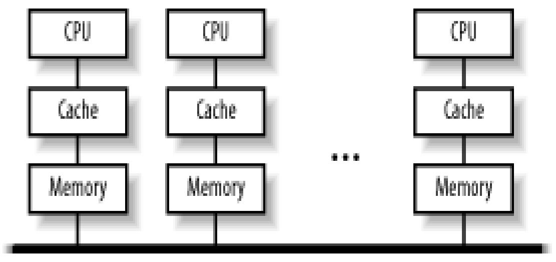


Figure II.2: NUMA architecture

Operating system support is required with either multiprocessor scheme. Fortunately, most modern operating systems, including Linux, provide support for SMP systems, and support is improving for NUMA architectures.

When dividing a calculation among processors, an important concern is granularity, or the smallest piece that a computation can be broken into for purposes of sharing among different CPUs. Architectures that allow smaller pieces of code to be shared are said to have a finer granularity (as opposed to a coarser granularity). The granularity of each of these architectures is the thread. That is, the operating system can place different threads from the same process on different processors. This implies that, if computation generates only a single thread, then that thread can't be shared between processors but must run on a single CPU. If the operating system has nothing else for the other processors to do, they will remain idle and we will see no benefit from having multiple processors.

A third architecture worth mentioning in passing is processor array, which, at one time, generated a lot of interest. A processor array is a type of vector computer built with a collection of identical, synchronized processing elements. Each processor executes the same instruction on a different element in a data array. Numerous issues have arisen with respect to processor arrays. While some problems map nicely to this architecture, most problems do not. This severely limits the general use of processor arrays. The overall design doesn't work well for problems with large serial components. Processor arrays are typically designed around custom VLSI processors, resulting in much higher costs when compared to more commodity-oriented multiprocessor designs. Furthermore, processor arrays typically are single user, adding to the inherent cost of the system. For these and other reasons, processor arrays are no longer popular.

- **Multi computers**

A multicomputer configuration, or **cluster**, is a group of computers that work together. A cluster has three basic elements—a collection of individual computers, a network connecting those computers, and software that enables a computer to share work among the other computers via the network. Since each computer in a cluster has its own memory (unlike a UMA or NUMA computer), identical addresses on individual CPUs map different physical memory locations. Communication is more involved and costly.

II.3 Emergence of cluster computers

The NOW [AND 95] and Beowulf [STE 95] projects in the 1990s launched the idea of aggregating hundreds of standard machines in order to form a high-power computing cluster. The initial interest lay in the highly beneficial performance/price relationship because aggregating standard materials was a lot cheaper than purchasing the specialized supercomputers that existed at the time. Despite this concept, achieving high computing power actually requires masking the structure of a cluster, particularly the time- and bandwidth-consuming communications between the different nodes. Many works were therefore carried out on the improvement of these communications in conjunction with the particular context of parallel applications that are executed on these clusters [9].

II.4 Cluster structure

It's tempting to think of a cluster as just a bunch of interconnected machines, but when we begin constructing a cluster, we will need to give some thought to the internal structure of the cluster. This will involve deciding what roles the individual machines will play and what the interconnecting network will look like.

The simplest approach is a symmetric cluster. With a symmetric cluster (Figure II.3) each node can function as an individual computer. This is extremely straightforward to set up. A subnetwork can be created with the individual machines (or simply add the computers to an existing network) and add any cluster-specific needed software. This is the architecture typically expect to see in a network of workstations NOW, where each machine must be independently usable.

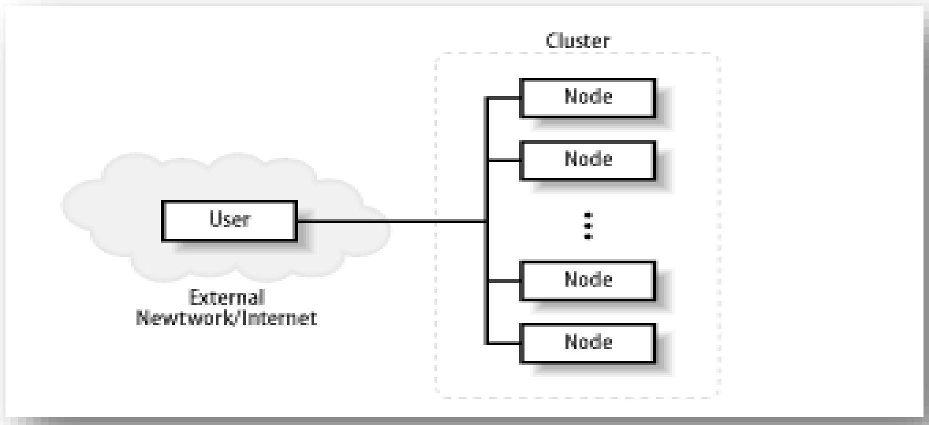


Figure II.3: Symmetric clusters

There are several disadvantages to a symmetric cluster. Cluster management and security can be more difficult. Workload distribution can become a problem, making it more difficult to achieve optimal performance.

For dedicated clusters, an asymmetric architecture is more common. With asymmetric clusters (**FigureII.4**) one computer is the head node or frontend. It serves as a gateway between the remaining nodes and the users. The remaining nodes often have very minimal operating systems and are dedicated exclusively to the cluster. Since all traffic must pass through the head, asymmetric clusters tend to provide a high level of security.

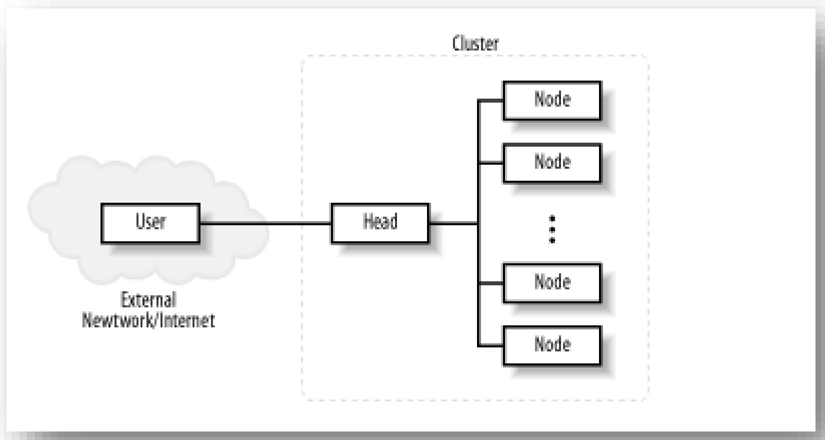


Figure II.4: Asymmetric clusters

The head often acts as a primary server for the remainder of the clusters. Since, as a dual-homed machine, it will be configured differently from the remaining nodes, it may be easier to keep all customizations on that single machine. This simplifies the installation of the remaining machines.

The primary disadvantage of this architecture comes from the performance limitations imposed by the cluster head. For this reason, a more powerful computer may be used for the head. While beefing up the head may be adequate for small clusters, its limitations will become apparent as the size of the cluster grows. An alternative is to incorporate additional servers within the cluster. For example, one of the nodes might function as an NFS server, a second as a management station that monitors the health of the clusters, and so on.

In addition, I/O represents a particular challenge. It is often desirable to distribute a shared file system across a number of machines within the cluster to allow parallel access. Figure II.5 shows a more fully specified cluster.

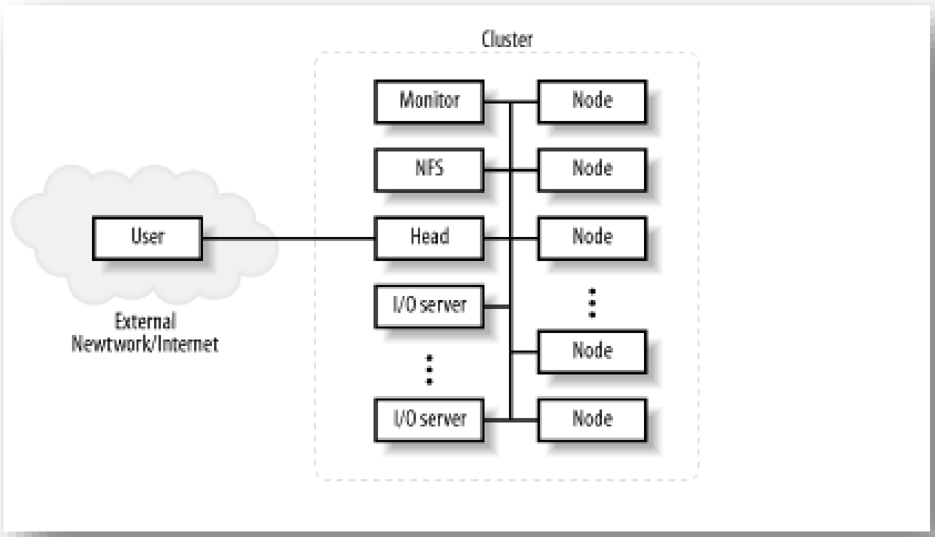


Figure II.5: Expanded cluster

Network design is another key issue. With small clusters, a simple switched network may be adequate. With larger clusters, a fully connected network may be prohibitively expensive. Numerous topologies have been studied to minimize connections (costs) while maintaining viable levels of performance. Examples include hyper-tree, hyper-cube, butterfly, and shuffle-exchange networks.

Heterogeneous networks are not uncommon. Although not shown in the figure, it may be desirable to locate the I/O servers on a separate parallel network. For example, some clusters have parallel networks allowing administration and user access through a slower network, while communications for processing and access to the I/O servers is done over a high-speed network.

II.5 High performance Network

Clusters need to incorporate fast interconnection technologies in order to support high-bandwidth and low latency inter-processor communication between cluster nodes. Slow interconnection technologies had always been a critical performance bottleneck for cluster computing. Today, improved network technologies help realize the construction of more efficient clusters.

Selecting a cluster interconnection network technology depends on several factors, such as compatibility with the cluster hardware and operating system, price, and performance. There are two metrics to measure performance for interconnects: **bandwidth** and **latency**. Bandwidth is the amount of data that can be transmitted over the interconnect hardware in a fixed period of time, while latency is the time to prepare and transmit data from a source node to a destination node [8].

- **Gigabit Ethernet**

Provides a reasonably high bandwidth given its low price, but suffers from relatively high latency, thus restricting Gigabit Ethernet as a good choice. However, the low price of Gigabit Ethernet is appealing to building clusters.

The 10 Gigabit Ethernet Standard is an extension of the basic IEEE 802.3* standard protocols to a wire speed of 10Gbps. As an extension, 10GbE is still fully Ethernet compatible and retains the key Ethernet architecture, including the Media Access Control (MAC) protocol, the Ethernet frame format, and the minimum and maximum frame size. Just as Gigabit Ethernet followed the standard Ethernet model, 10GbE continues the evolution in speed while using virtually the same architecture used in other Ethernet specification.

As shown in **Figure II.6**, the Intel®82597EX 10GbE Controller connects to the adapter's PMD by means of four transmit (Tx) channels and four receive (Rx) channels:

Each channel has a 3.125 Gbps bandwidth so that the aggregate bandwidth in the Tx and Rx directions exceeds 10Gbps each way. The separate Tx and Rx channels also allow the server adapter and 10GbE traffic to operate in full-duplex mode (simultaneous Tx and Rx). By contrast, Ethernet standards prior to 10GbE operated in simplex mode and required use of a Carrier-Sensing Multiple-Access with Collision Detection (CSMA/CD) protocol to avoid contention between network devices seeking access to the server. By using full-duplex operation and eliminating CSMA/CD processing, 10GbE can provide faster, lower-latency responses to network transaction requests

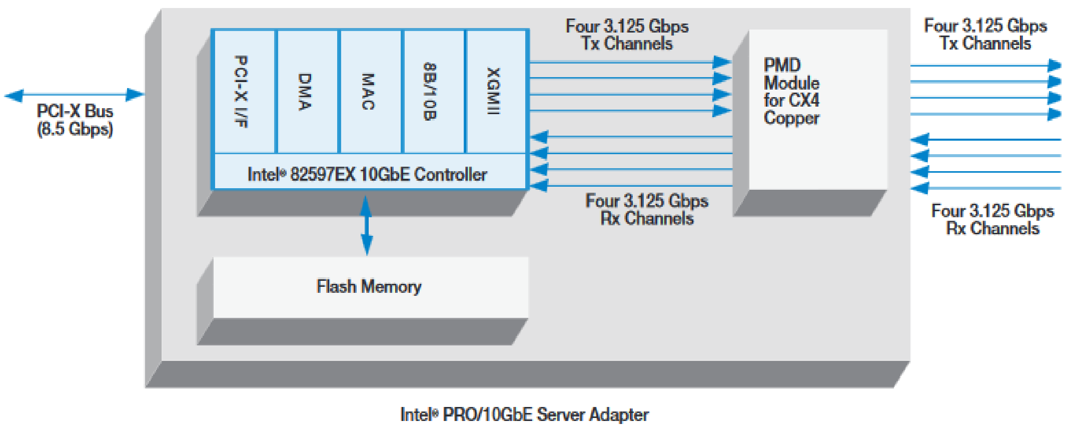


Figure II.6: Intel PRO/10GbE Architecture

- **InfiniBand**

InfiniBand (IB) is the more recent computer-networking communications standard used in high-performance computing that features very high throughput and very low latency. It is used for data interconnect both among and within computers. Mellanox, now usually produces the processors of InfiniBand interface cards. The very attractive standard led to re-purchasing of most of the initial start-ups by large companies, which has led the current market to be dominated by Mellanox, Voltaire, Qlogic and Cisco [9]. The standard's specifications detail both hardware and software implementation. Each retailer can distribute its own software network-access software layers that more or less respect the norm.

InfiniBand is also used as either a direct or switched interconnect between servers and storage systems, as well as an interconnect between storage systems. InfiniBand uses a switched fabric topology, as opposed to early shared medium Ethernet. All transmissions begin or end at a channel adapter. Each processor contains a host channel adapter (HCA) and each peripheral has a target channel adapter (TCA) . These adapters can also exchange information for security or quality of service (QoS).

Host Channel Adapter (HCA) : Device that terminates an IB link and executes transport - level functions and support the verbs interface.

Switch : A device that routes packets from one link to another of the same IB Subnet

Router : A device that transports packets between IBA subnets.

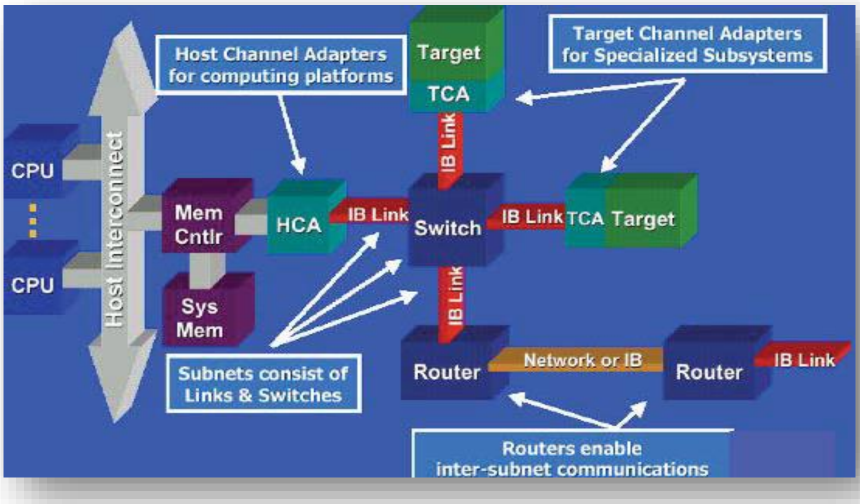

Host Interconnect
CPU
CPU
Host Channel Adapters for computing platforms
Target
TCA
Target Channel Adapters for Specialized Subsystems
IB Link
Mem Cntlr
HCA
IB Link
Switch
IB Link
TCA
Target
Sys Mem
IB Link
Subnets consist of Links & Switches
Router
Network or IB
Router
IB Link
Routers enable inter-subnet communications

Figure II.7: Infiniband components

InfiniBand is a pervasive, low-latency, high-bandwidth interconnect which requires low processing overhead and is ideal to carry multiple traffic types (clustering, communications, storage, management) over a single connection. It is now used in thousands of **high-performance** compute clusters and beyond that scale from small scale to large scale:

Infiniband type	LInk Speed	Data Speed	Max Bandwidth (at application level)
SDR 4x	10 Gigabit	8 Gigabit	1 GB/sec
DDR 4X	20 Gigabit	16 Gigabit	2GB/sec
QDR 4X	40 Gigabit	32 Gigabit	4GB/sec

Table II.1: Infiniband values

II.6 Conclusion

We have examined in this chapter the cluster-specific components and the interconnection technology Infiniband which is the more recent technology (after Scalable Coherent Interface (SCI) first, then Myricom and Quadrics). The rapid research and developments of cluster hardware and software components has enhanced the usage of cluster computing for a wide variety of applications both in scientific and commercial domains. In chapter III, we present our parallel split soil application and how clusters **[Ibn-Khaldoun]** are used to implement it.

Chapter III

Conception and Implementation

III.1 Introduction

We have devoted this chapter to explain the practical aspect of our project where we divided it into three sections.

The first section of this chapter is a detailed study case conception of our parallel k-means algorithm.

The second section of this chapter the implementation of our parallel k-means where the last section would present the results of this implementation starting from the execution concluding with the interpretations on the field.

III.2 Conception of Parallel k-means algorithm

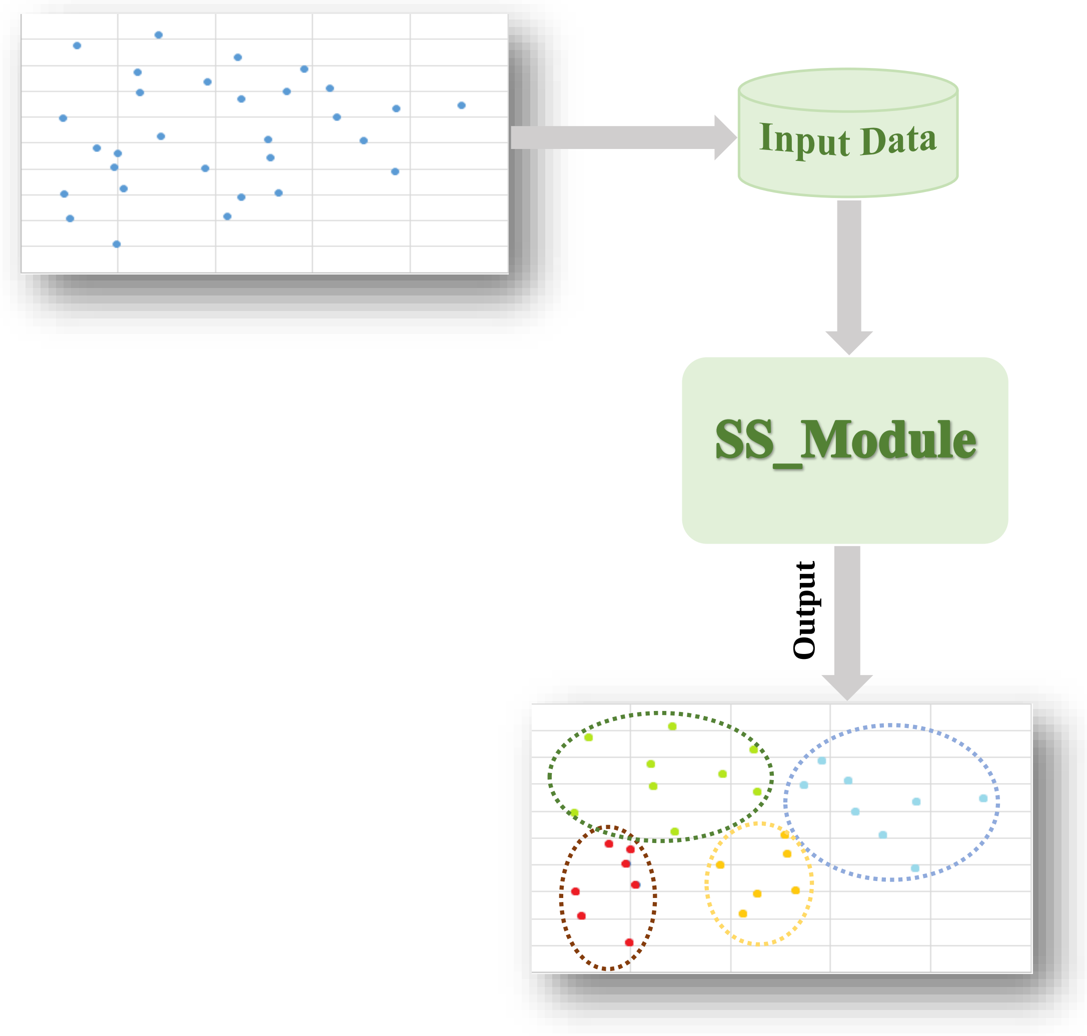


Figure III.1. General Architecture.

As shown in the Figure III.1 above, we aim to implement a parallel split soil module to obtain groups of soils with certain parameters from raw soil data.

- **Input Data:** is a collected soil samples translated into points of different parameters in an Excel data base represented by the plot graph above in Figure III.1.

  Each plot represents a sample with different entries called soil parameters that will be described in the implementation section later.
- **Output:** is a set of grouped original data provided after applying the split soil module on raw dataset.

  Resulted groups (clusters) are samples that have similarity measures.
- **SS_Module**: is the Soil Split module based on Parallel K-means algorithm

III.3 Split Soil Module

The cycle of split soil module is divided to three parts as following:

a. A parallelised version of the sequential k-means algorithm: Parallel_ K-means_SS.

b. Application on raw soil data to group and obtain results.

c. Comparison of results between both sequential and parallel implementations.

III.3.1. Sequential_ K-means_SS algorithm principle:

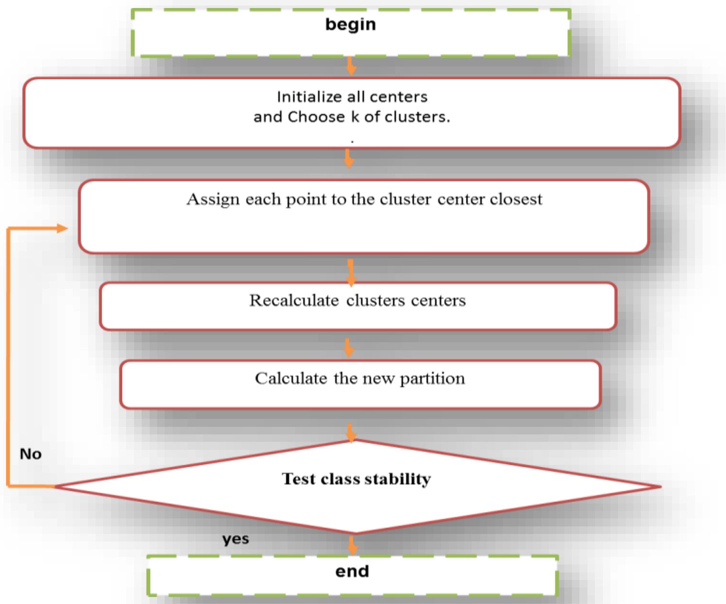

begin
Initialize all centers
and Choose k of clusters.
Assign each point to the cluster center closest
Recalculate clusters centers
Calculate the new partition
No
Test class stability
yes
end

Figure III.2: Sequential k-means flowchart.

III.3.2. Parallel K-means_SS algorithm principle:

We assume $N + 1$ processors $\boldsymbol{P_0} \dots \boldsymbol{P_N}$:

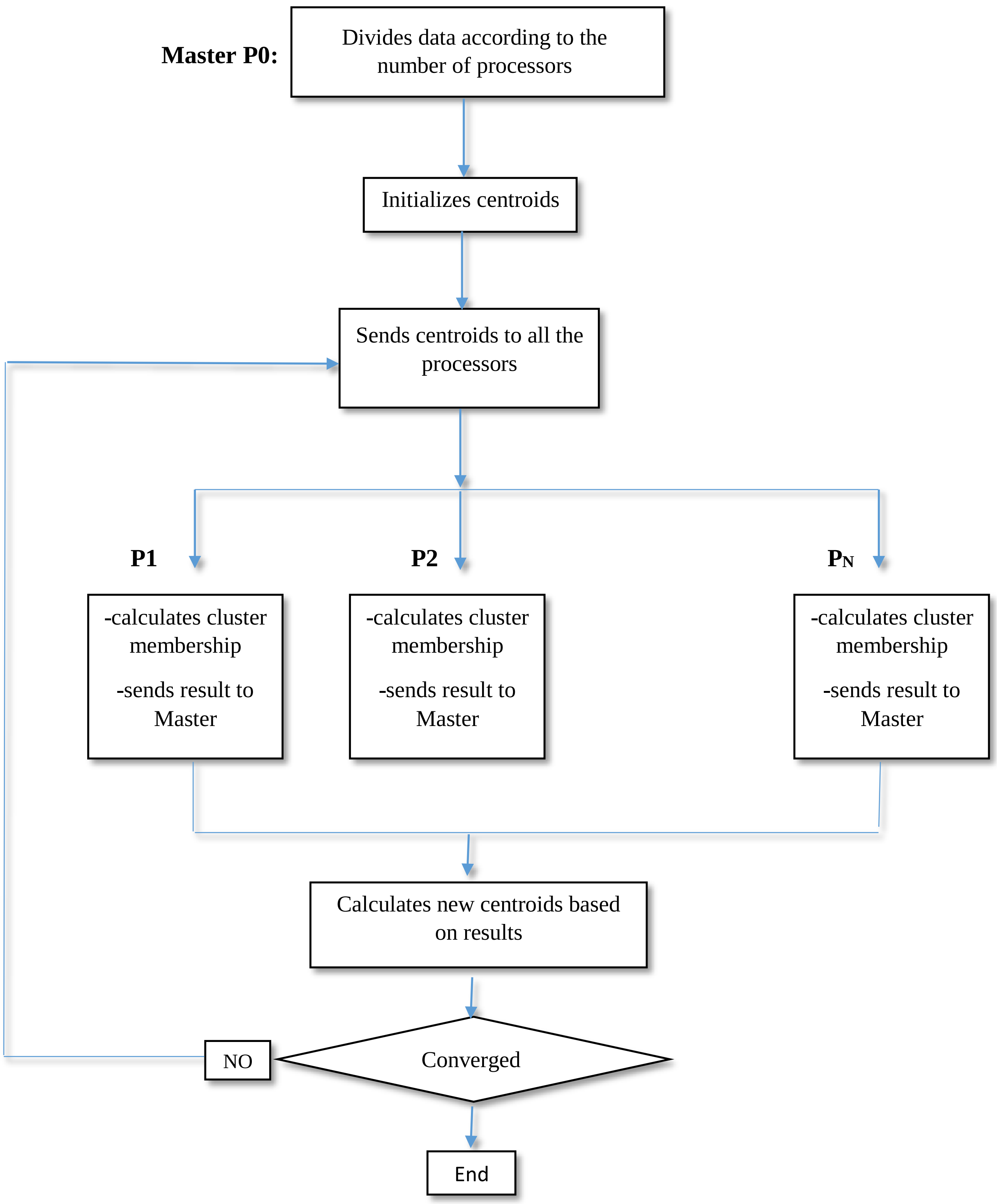

Master P0:
Divides data according to the number of processors
Initializes centroids
Sends centroids to all the processors
P1
P2
PN
-calculates cluster membership
-sends result to Master
-calculates cluster membership
-sends result to Master
-calculates cluster membership
-sends result to Master
Calculates new centroids based on results
NO
Converged
End

Figure III.3: Parallel k-means_SS flowchart.

- **Calculating cluster membership** is a two-step process:
  1- Calculating distance between clusters:
  We have used the Euclidian distance following this equation:[13]

$$\sqrt{\sum_{j=1}^{m}\left(x_j - y_j\right)^2}$$

  Which in general calculates the root between tow square co-ordinates of pair of objects, and in our case computes the distance between two points x and y with k dimensions.

  2- Comparing the distances between data points and centroids to assign membership, where the point with minimum distance to a certain cluster centroid is its new member.

III.3.3. FUSION

The split soil module is not only based on the parallel k-means algorithm, but also on the method of fusion, and it is divided into two sections depending on the results we want to obtain on reality.

III.3.3.1 Total Fusion

- **Description :**
  1- Application of the parallel k-mean on the input file.
  2- Checking the output file than using the result to create a second input file.
  3- Applying the k-means algorithm on the new input file to create an elite cluster of the parameter in progress.
  4- Finally, gathering all the elite clusters values into one input file and applying the parallel k-means for a third round to result a final and elite cluster.

The merging of all the parameters into one whole cluster is what we call a total fusion because the elite cluster in this case will contain all of the soil parameters at once plus it will have the feature of the even values of each one of them.

The following demonstration will clear the sight:

- Step 1: Example of applying the PKM on one of the soil parameters.

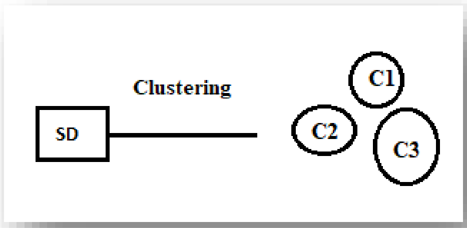


Figure III.4: Clustering process1 (1).

- Step 2&3: Reapplying the PKM on the result clusters from step1 to obtain an elite Cluster.

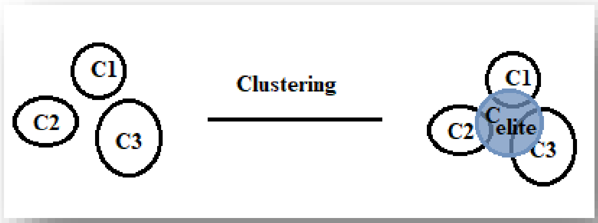


Figure III.5: Clustering process1 (2).

- Step 4: Applying PKM for a final round on the elite resulting clusters from each parameters to obtain as a result a total elite cluster of the soil batch we are working on.

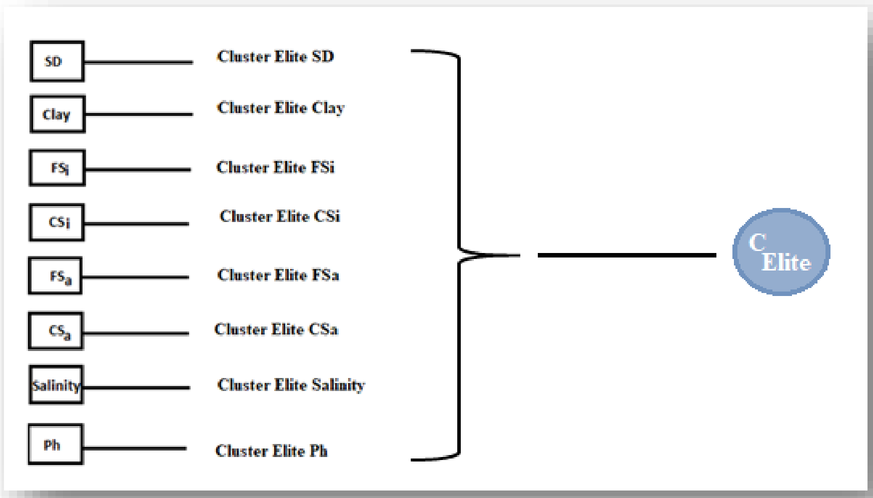


Figure III.6: Clustering process1 (3).

Note: We can obtain more than one final elite cluster of the soil, the number of elite clusters depend on how many excellent spots we want to cultivate.

III.3.3.2 Partial Fusion

- **Description :**
  1- Application of the parallel k-mean on the input file.
  2- Finishing the process on the rest of parameters.
  3- Gathering the results of each output file in one file to be the new input file.
  4- Deciding how many clusters needed, than applying the parallel k-means for another time on the new input file.
  5- The result would be a defined number of clusters that contains not all the parameters but most of them, which is a good quality cluster.

The good quality cluster is a merging between the parameters, it may as it may not contain all of them at once, but the fact that it gathers the most of them is already a good result. In this case the fusion is partial.

The process is as follow:

- Step 1:

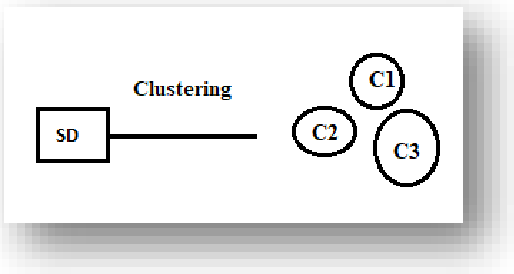


Figure III.7: Clustering process2 (1).

- Step [2, 5]:

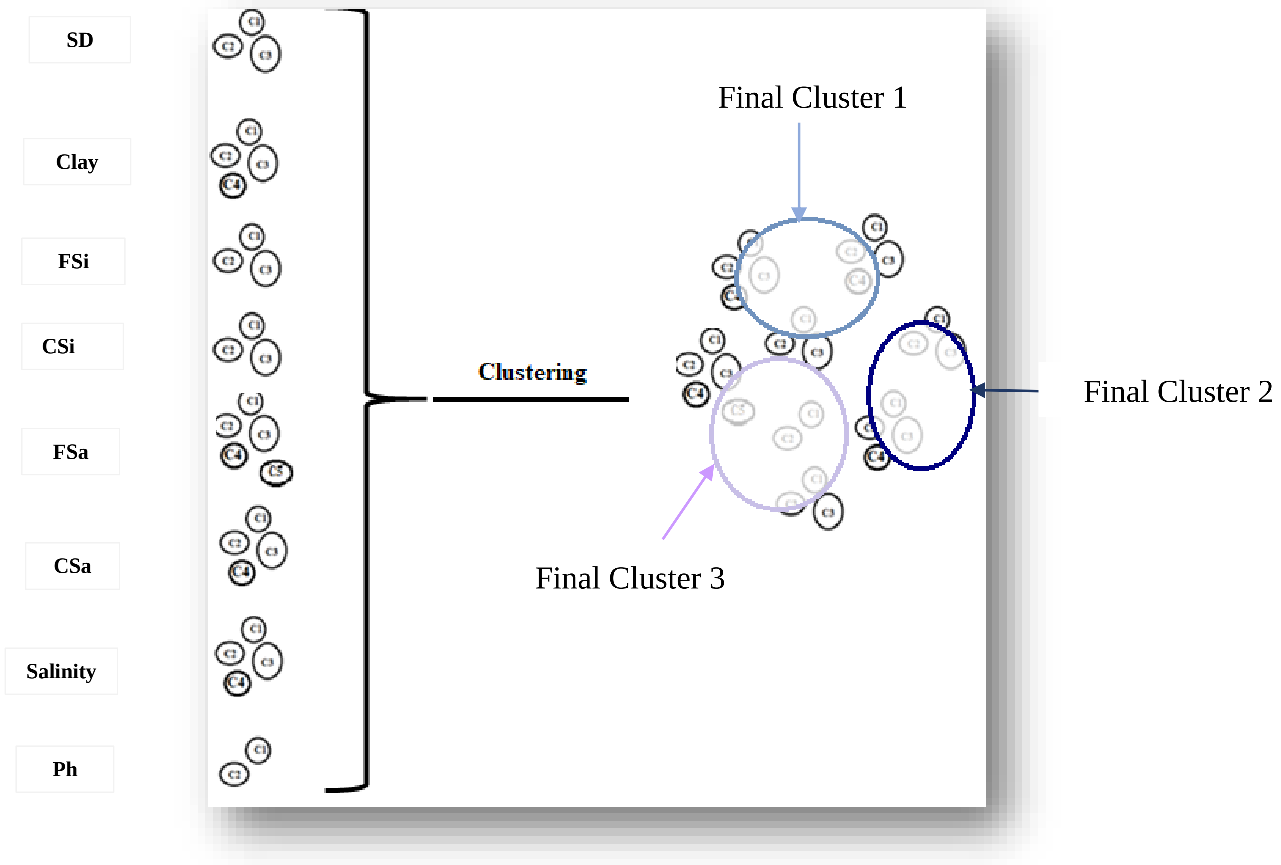


Figure III.8: Clustering process2 (2).

III.4 Implementation of parallel k-means algorithm

III.4.1 Description

The implementation of the parallel split soil module was done in C programming language using MPI (Message passing Information), and was both compiled and executed on **IBN KHALDOUN HPC** (High performance Computer) that belongs to the **university of BISKRA.**

The application of this module was on an agricultural data base that the Director of the Research Division **"Soil Resource Management and Utilization in Arid Regions" of the CRSTRA (Centre for Scientific and Technical Research on Arid Regions)** provided us.

III.4.1.1 MPI

The message passing interface (MPI) is a standardized means of exchanging messages between multiple computers running a parallel program across distributed memory.[11]

In parallel computing, multiple computers -- or even multiple processor cores within the same computer -- are called nodes. Each node in the parallel arrangement typically works on a portion of the overall computing problem. The challenge then is to synchronize the actions of each parallel node, exchange data between nodes and provide command and control over the entire parallel cluster. The message-passing interface defines a standard suite of functions for these tasks.[14]

III.4.1.2 C Programming Language:

C is a general-purpose programming language that is extremely popular, simple and flexible. It is machine-independent, structured programming language, which is used extensively in various applications. [15]

C was the basic language to write everything from operating systems (Windows and many others) to complex programs like the Oracle database, Git, Python interpreter and more.

III.4.1.3 HPC

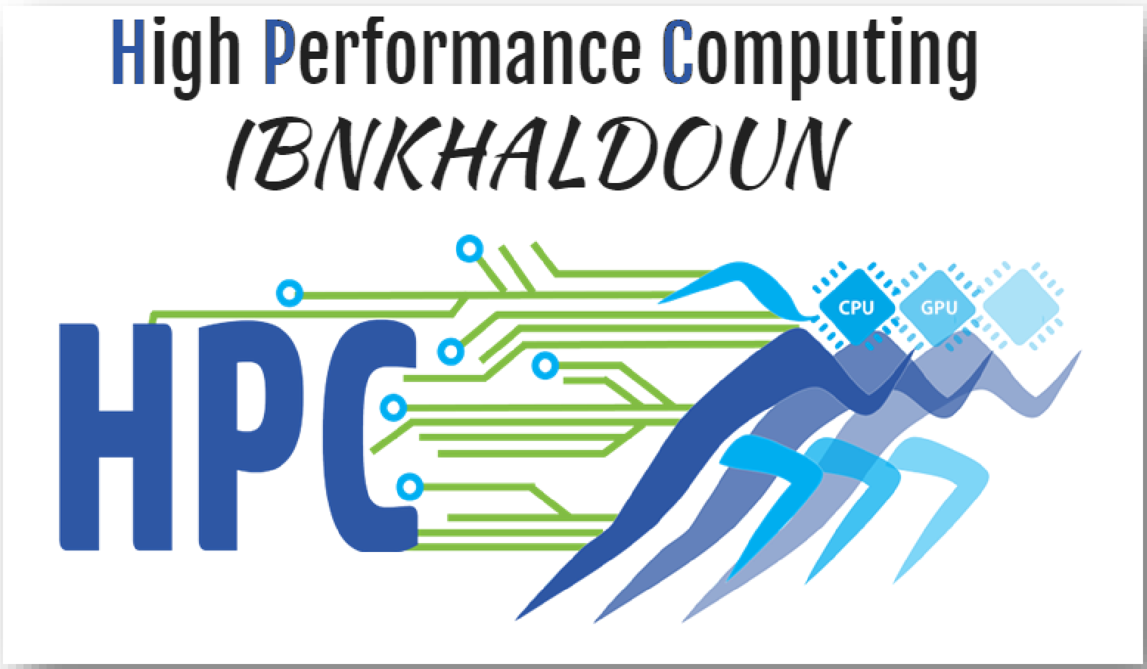


Figure III.9: Logo of HPC IBNKHALDOUN Biskra University.

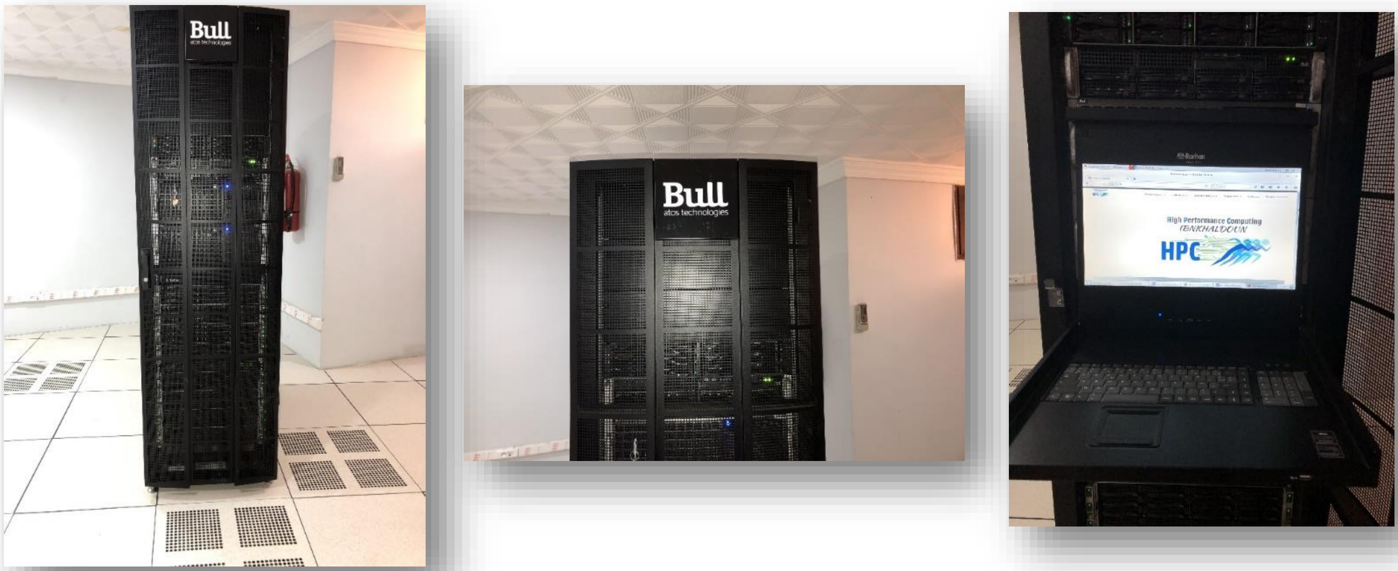


Figure III.10: Presentation of the HPC.

The Mohamed Khider University of Biskra provided students with the platform for High Performance Computing (HPC) IBNKHALDOUN.

This platform is intended for users from various fields. The IBNKHALDOUN platform is a cluster consisting of 32 bi-processor nodes with 10 cores each.

- HPC's Characteristics:[16]

The HPC- IBNKHALDOUN intensive computing unit has:

- A management node (ibnkhaldoun0)

Processor: Intel Xeon (R) E5-2660 v3 @ 2.60 GHz x 20,

Memory: 64 GB RAM.

Disk: 2 TB HDD

OS: Red Hat Enterprise Linux Server 7.2

OS Type: 64 bit

- A visualization server with a high-performance graphics card

- An NFS Storage Management Server

- 32 Calculation nodes, each node has the following characteristics:

10 physical cores

Storage capacity: 500 GB HDD.

Available memory: 64 GB.

- One Infinite Band switch

- An Ethernet switch

- Inverter 20KVA

- Scheduling software: Slurm

- Installed software:

Open source: MPI, MPICH2, MVAPICH,

Licensed: Intel MKL, TBB, IPP.

III.4.1.4 CRSTRA (Centre for Scientific and Technical Research on Arid Regions):

The CRSTRA is in charge of developing scientific and technical research and keeping an ecological watch through continuous observation and the establishment of a scientific and technical database on arid and semi-arid territories (Saharan and steppe regions) covering more than 2 million km2 or more than 85% of the Algerian national territory.[17]

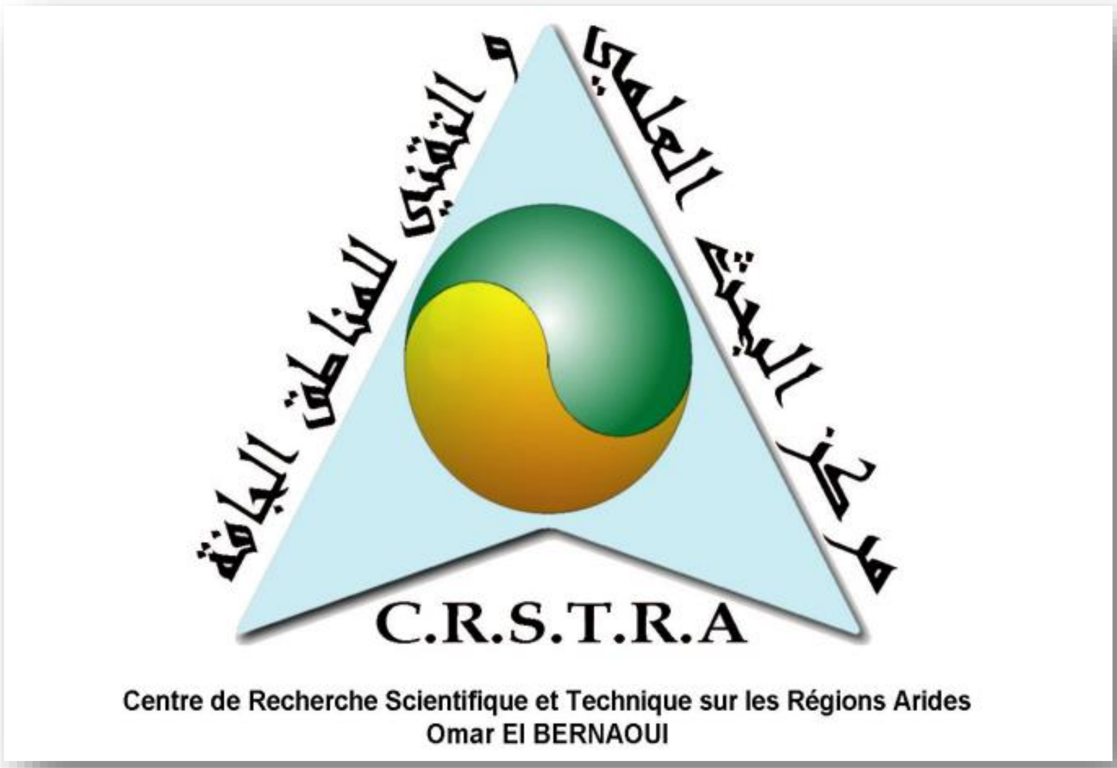

مركز البحث العلمي و التقني للمناطق الجافة
C.R.S.T.R.A
Centre de Recherche Scientifique et Technique sur les Régions Arides
Omar El BERNAOUI

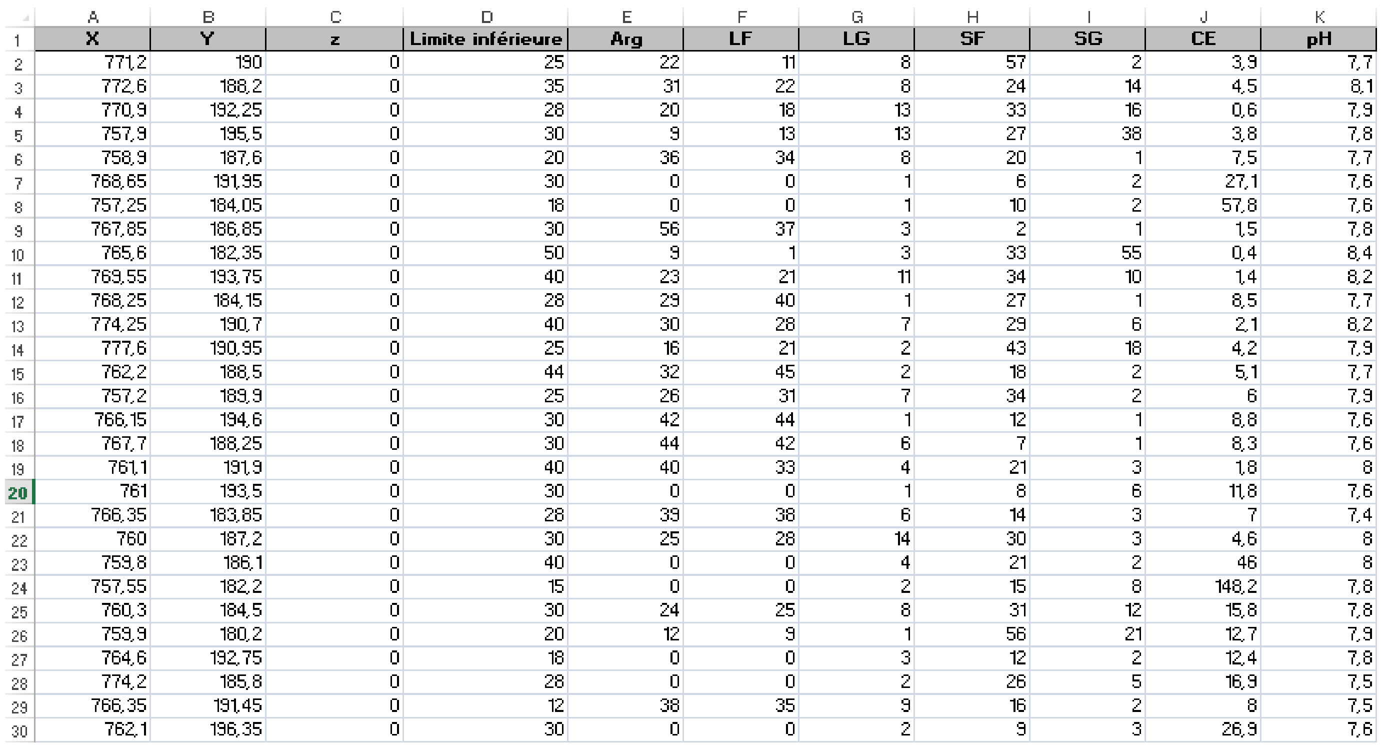
Figure III.11: Logo of CRSTRA.

III.4.2 Data description:

The used data is a collaboration of work provided by the CRSTRA center crew, where they have taken 30 soil samples from a batch field in the OUTAYA region in BISKRA province, which is dedicated for agronomical studies of arid and semi-arid regions.

Every sample went through laboratories to analyse and extract the soil characteristics which found to be 8 values, than characterised by 2D Cartesian geographic coordinates (X, Y).

The data base is as following:



	A	B	C	D	E	F	G	H	I	J	K
1	X	Y	z	Limite inférieure	Arg	LF	LG	SF	SG	CE	pH
2	771,2	190	0	25	22	11	8	57	2	3,9	7,7
3	772,6	188,2	0	35	31	22	8	24	14	4,5	8,1
4	770,9	192,25	0	28	20	18	13	33	16	0,6	7,9
5	757,9	195,5	0	30	9	13	13	27	38	3,8	7,8
6	758,9	187,6	0	20	36	34	8	20	1	7,5	7,7
7	768,65	191,95	0	30	0	0	1	6	2	27,1	7,6
8	757,25	184,05	0	18	0	0	1	10	2	57,8	7,6
9	767,85	186,85	0	30	56	37	3	2	1	1,5	7,8
10	765,6	182,35	0	50	9	1	3	33	55	0,4	8,4
11	769,55	193,75	0	40	23	21	11	34	10	1,4	8,2
12	768,25	184,15	0	28	29	40	1	27	1	8,5	7,7
13	774,25	190,7	0	40	30	28	7	29	6	2,1	8,2
14	777,6	190,95	0	25	16	21	2	43	18	4,2	7,9
15	762,2	188,5	0	44	32	45	2	18	2	5,1	7,7
16	757,2	189,9	0	25	26	31	7	34	2	6	7,9
17	766,15	194,6	0	30	42	44	1	12	1	8,8	7,6
18	767,7	188,25	0	30	44	42	6	7	1	8,3	7,6
19	761,1	191,9	0	40	40	33	4	21	3	1,8	8
20	761	193,5	0	30	0	0	1	8	6	11,8	7,6
21	766,35	183,85	0	28	39	38	6	14	3	7	7,4
22	760	187,2	0	30	25	28	14	30	3	4,6	8
23	759,8	186,1	0	40	0	0	4	21	2	46	8
24	757,55	182,2	0	15	0	0	2	15	8	148,2	7,8
25	760,3	184,5	0	30	24	25	8	31	12	15,8	7,8
26	759,9	180,2	0	20	12	9	1	56	21	12,7	7,9
27	764,6	192,75	0	18	0	0	3	12	2	12,4	7,8
28	774,2	185,8	0	28	0	0	2	26	5	16,9	7,5
29	766,35	191,45	0	12	38	35	9	16	2	8	7,5
30	762,1	196,35	0	30	0	0	2	9	3	26,9	7,6

Figure III.12: El Outaya soil parameters data base.

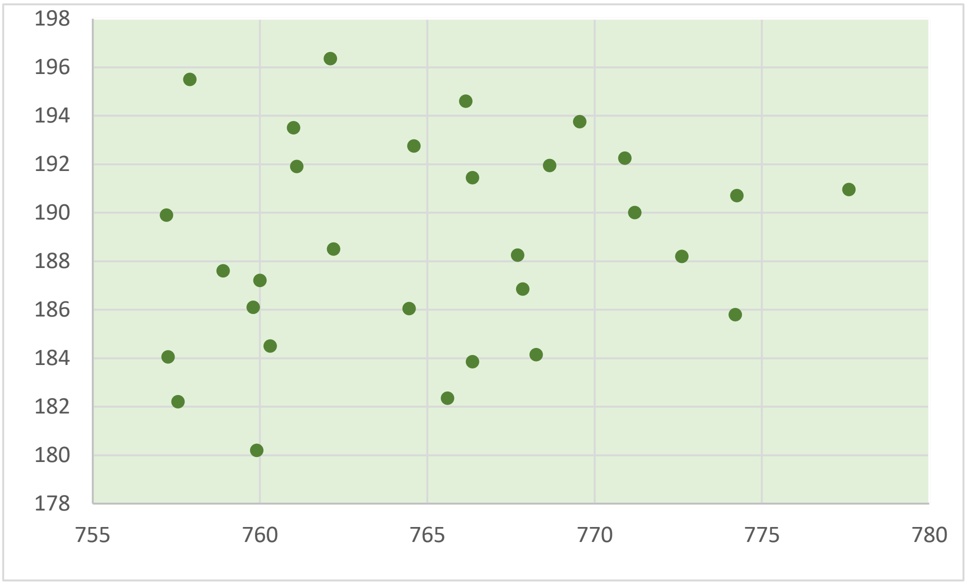


Figure III.13: Localisation of the soil samples

The following is a description of each column of the data set figure showed above, each parameter name is sited as provided than well explained and demonstrated in nature:

Limite inférieure: Soil depth

- It defines the root space and the volume of soil from where the plants fulfil their water and nutrient demands.[18]

Arg (Argile): Clay

- Clay denotes a natural rocky material based on silicates or hydrates aluminosilicates of lamellar structure, generally derived from the weathering of silicates with a three-dimensional framework, such as feldspars.[19]

Figure III.14: Types of Clay in nature.

LF (Fine Silt**), LG (**Coarse Silt**):**

- Silt is a solid, dust-like sediment that water, ice and wind transport and deposit.
- Silt is made up of rock and mineral particles that are larger than clay but smaller than sand. [20]

Figure III.15: Types of Silt in nature.

SF (Fine Sand**), SG (**Coarse Sand**):**

- Sand is a granular material composed of finely divided rocks and mineral particles. Defined by size, being finer than gravel and coarser than silt. [21]

Figure III.16: Types of Sand in nature.

Ce:

- electrical conductivity that is known in the agronomics by Soil salinity.

Figure III.17: Soil salinity in nature.

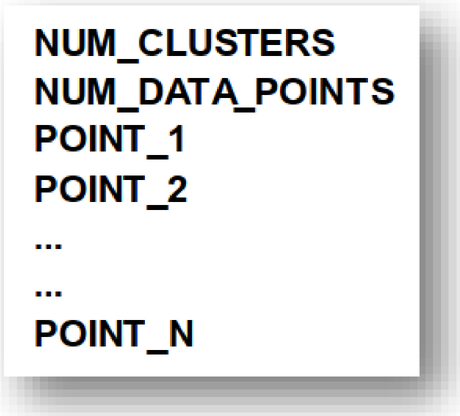
Ph.: soil PH is considered a master variable in soils as it affects many chemical processes. It specifically affects plant nutrient availability by controlling the chemical forms of the different nutrients and influencing the chemical reactions they undergo. The optimum pH range for most plants is between 5.5 and 7.5

III.4.3. Parallel split soil

What we did in this project is that we have applied the parallel k-means algorithm on the data values only one at a time, in other words, we clustered each of the characteristics by itself first and got an elite cluster from each one of them than we have done a fusion at the end to extract the best spots of the batch soil that gives the best results after planting and harvesting.

- In this program, we enter an input text file that contains the data values and it results an output file with the centroids of the generated clusters, for that aim we have separated the columns (soil parameters) of the data shown above each by itself and inserted the values in an input text file, as a total we started with 8 input files each represents a different parameter.

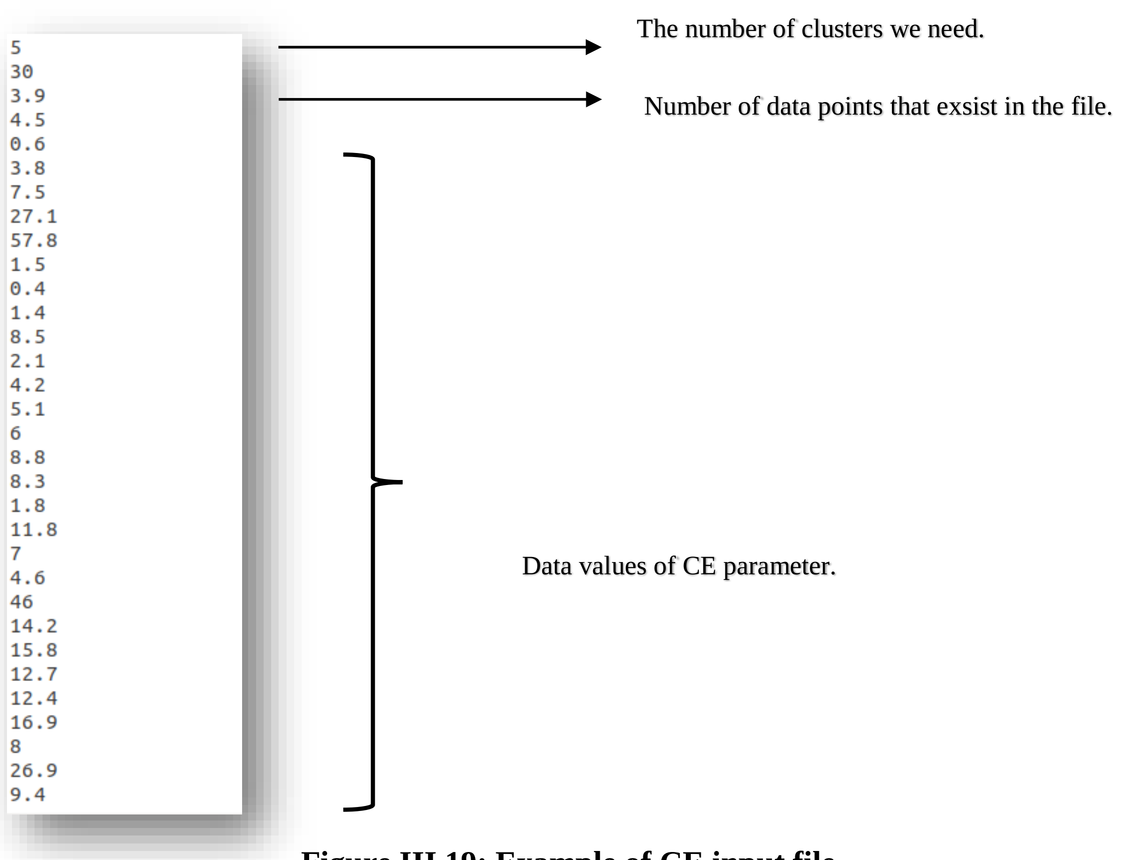
The form of the input file is the following:



```
NUM_CLUSTERS
NUM_DATA_POINTS
POINT_1
POINT_2
...
...
POINT_N
```

Figure III.18: Form of the input file.

- **Num_clusters:** this input is an Integer number, which consists of how many groups the user wants after applying the split model on his data.
- **Num_Data_Points:** also an integer number that presents the total number of points existing in the input file.
- **Point_1... Point_N:** the values of data, which in our case is every column of the excel figure above.

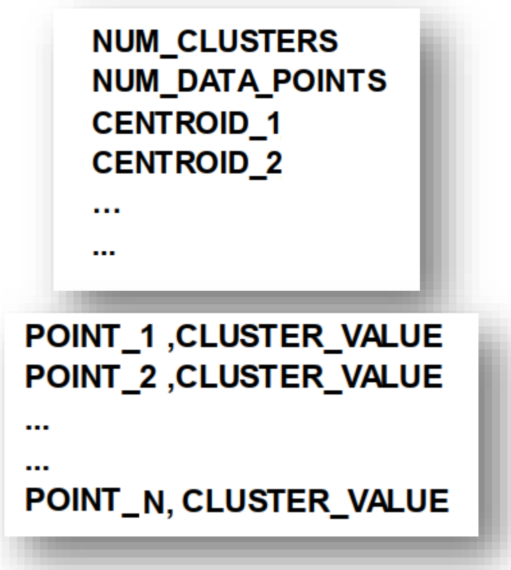


```
5
30
3.9
4.5
0.6
3.8
7.5
27.1
57.8
1.5
0.4
1.4
8.5
2.1
4.2
5.1
6
8.8
8.3
1.8
11.8
7
4.6
46
14.2
15.8
12.7
12.4
16.9
8
26.9
9.4
```

The number of clusters we need.
Number of data points that exsist in the file.
Data values of CE parameter.

Figure III.19: Example of CE input file.

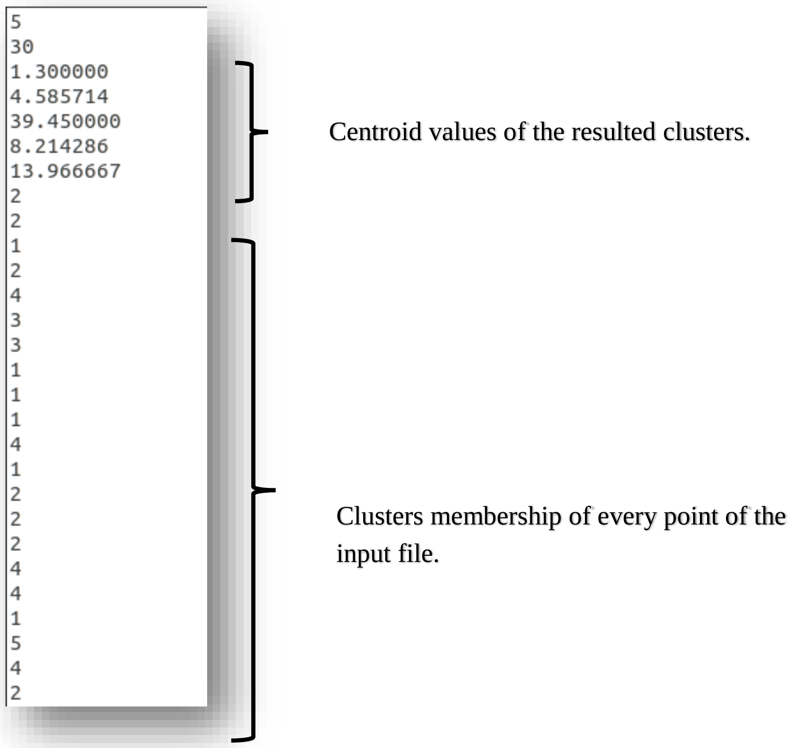
And the form of the output file:



```
NUM_CLUSTERS
NUM_DATA_POINTS
CENTROID_1
CENTROID_2
...
...
POINT_1 ,CLUSTER_VALUE
POINT_2 ,CLUSTER_VALUE
...
...
POINT_N, CLUSTER_VALUE
```

Figure III.20: Form of the Output file.

- **Centroid_1:** this output is the value of centroid of the first resulted cluster.
- **Point_1, Cluster value**: is the membership of cluster, which means that each position of point values of the input file is replaced by the number of cluster that point belongs to.



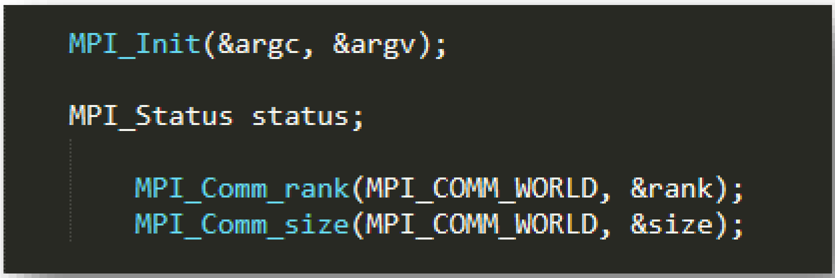
```
5
30
1.300000
4.585714
39.450000
8.214286
13.966667
2
2
1
2
4
3
3
1
1
1
4
1
2
2
2
4
4
1
5
4
2
```

Centroid values of the resulted clusters.
Clusters membership of every point of the input file.

Figure III.21: Example of CE Output file.

- This project is developed with considering the power plays of both C language and MPI environment. For the MPI side, these functions are the essentials that based on developing our parallel split soil module.
-

Creating the MPI Environment



```
MPI_Init(&argc, &argv);

MPI_Status status;

    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
```

MPI_comm_rank ()

- Determines the rank of the calling process in the communicator.

MPI_comm_size ()

- Determines the size of the group associated with a communicator.

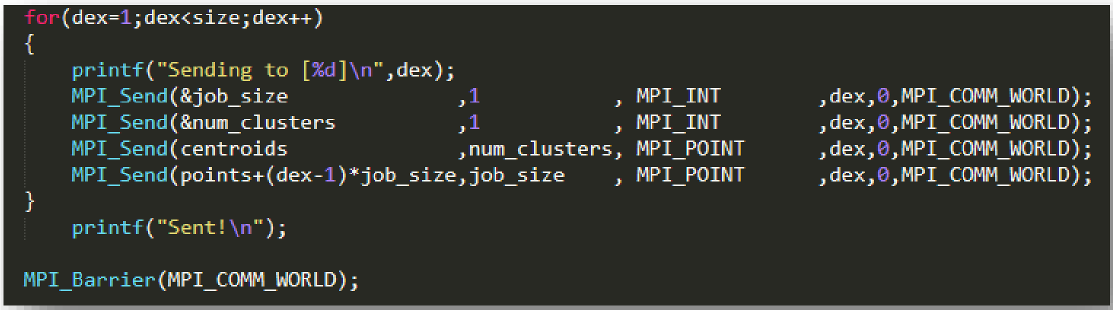
Algorithm 1: Master Processor

If (rank==MASTER)

{

/*inputting from file

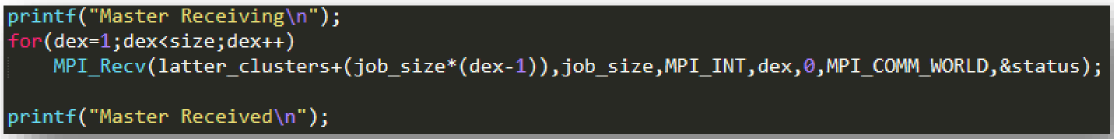
/*sending the essential data to slave processors



```
for(dex=1;dex<size;dex++)
{
    printf("Sending to [%d]\n",dex);
    MPI_Send(&job_size              ,1           , MPI_INT      ,dex,0,MPI_COMM_WORLD);
    MPI_Send(&num_clusters          ,1           , MPI_INT      ,dex,0,MPI_COMM_WORLD);
    MPI_Send(centroids              ,num_clusters, MPI_POINT    ,dex,0,MPI_COMM_WORLD);
    MPI_Send(points+(dex-1)*job_size,job_size    , MPI_POINT    ,dex,0,MPI_COMM_WORLD);
}
    printf("Sent!\n");

MPI_Barrier(MPI_COMM_WORLD);
```

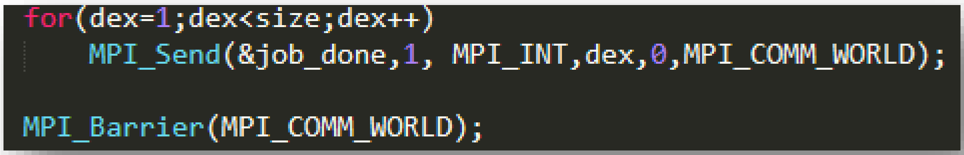
```
While (1)
{
        /*Receiving essential data from slave processors
```



```
printf("Master Receiving\n");
for(dex=1;dex<size;dex++)
    MPI_Recv(latter_clusters+(job_size*(dex-1)),job_size,MPI_INT,dex,0,MPI_COMM_WORLD,&status);

printf("Master Received\n");
```

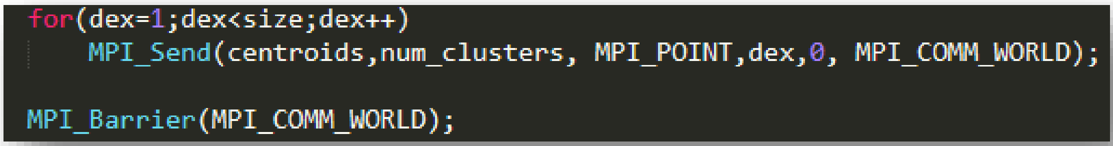
```
            CalculateNewCentroids ();
               If (check Convergence () ==0)
    {
                    /*Converged
                        /*informing slaves that no more job to be done
```



```
for(dex=1;dex<size;dex++)
    MPI_Send(&job_done,1, MPI_INT,dex,0,MPI_COMM_WORLD);

MPI_Barrier(MPI_COMM_WORLD);
```

```
                    /*sending the recently created centroids
```



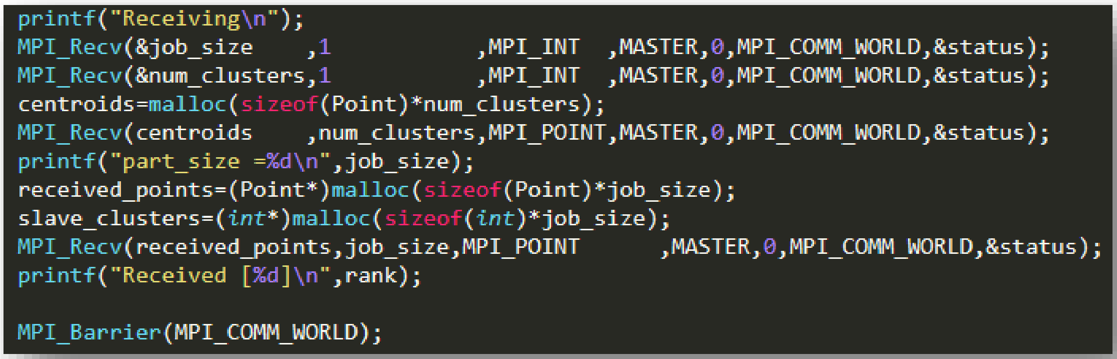
```
for(dex=1;dex<size;dex++)
    MPI_Send(centroids,num_clusters, MPI_POINT,dex,0, MPI_COMM_WORLD);

MPI_Barrier(MPI_COMM_WORLD);
```

```
            Else
                    /* Not converged
      }
  }
}
```

Algorithm 2: Slave Processors

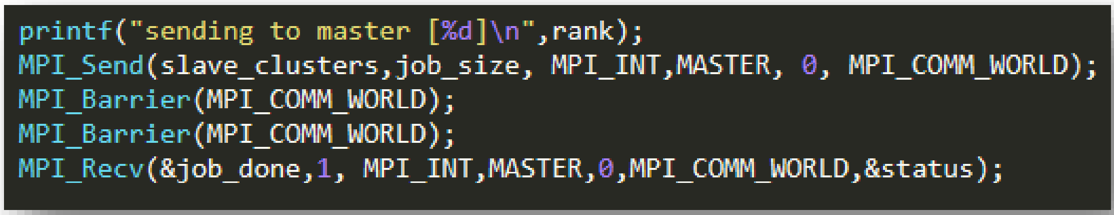
```
Else
{
        /*Receiving the essential data from Master
```



```
printf("Receiving\n");
MPI_Recv(&job_size    ,1           ,MPI_INT  ,MASTER,0,MPI_COMM_WORLD,&status);
MPI_Recv(&num_clusters,1           ,MPI_INT  ,MASTER,0,MPI_COMM_WORLD,&status);
centroids=malloc(sizeof(Point)*num_clusters);
MPI_Recv(centroids    ,num_clusters,MPI_POINT,MASTER,0,MPI_COMM_WORLD,&status);
printf("part_size =%d\n",job_size);
received_points=(Point*)malloc(sizeof(Point)*job_size);
slave_clusters=(int*)malloc(sizeof(int)*job_size);
MPI_Recv(received_points,job_size,MPI_POINT      ,MASTER,0,MPI_COMM_WORLD,&status);
printf("Received [%d]\n",rank);

MPI_Barrier(MPI_COMM_WORLD);
```

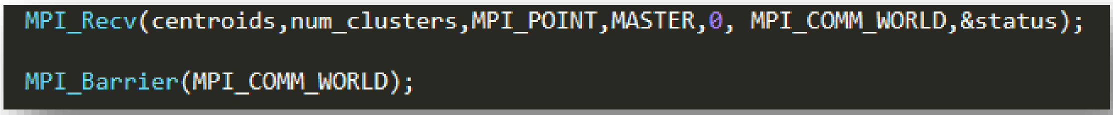
```
        While (1)
    {
            CalculateNewClusters ();
            /* sending to Master
```



```
printf("sending to master [%d]\n",rank);
MPI_Send(slave_clusters,job_size, MPI_INT,MASTER, 0, MPI_COMM_WORLD);
MPI_Barrier(MPI_COMM_WORLD);
MPI_Barrier(MPI_COMM_WORLD);
MPI_Recv(&job_done,1, MPI_INT,MASTER,0,MPI_COMM_WORLD,&status);
```

```
                        If (job_done == 1)

          /* Receiving new centroids created by Master
```



```
MPI_Recv(centroids,num_clusters,MPI_POINT,MASTER,0, MPI_COMM_WORLD,&status);

MPI_Barrier(MPI_COMM_WORLD);
```

```
    }
}
MPI_finalize ();
```

III.5 Results

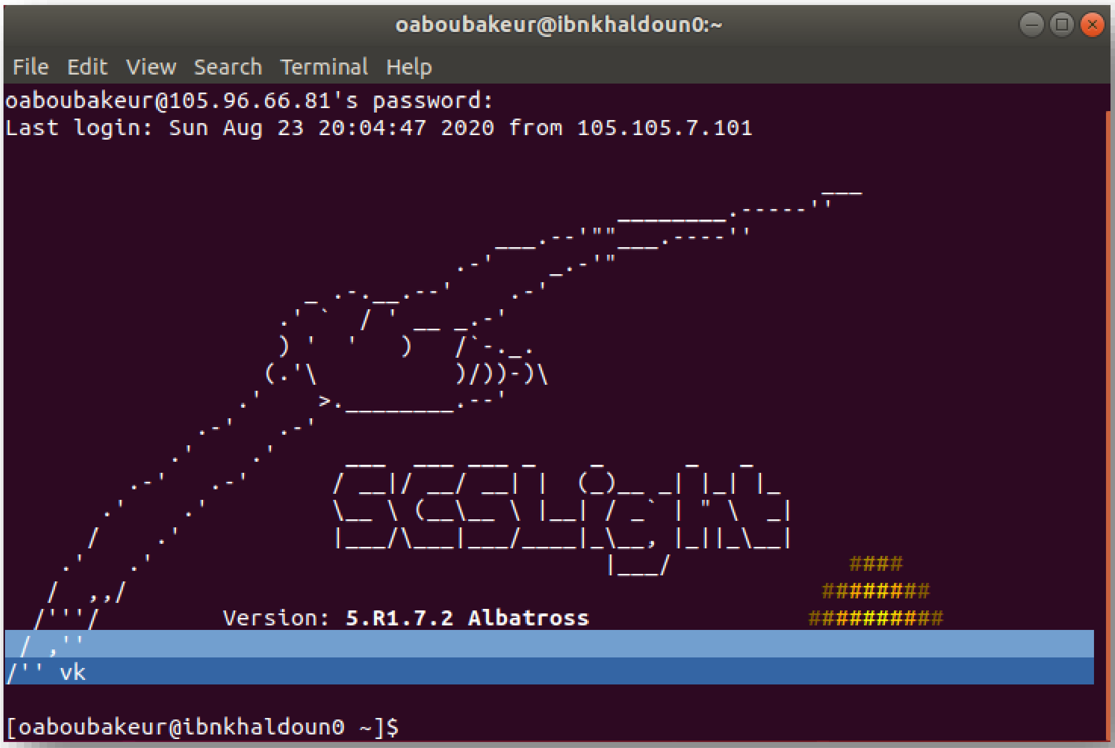
III.5.1 Execution Process

The execution of the parallel split soil module took place in the IBN KHALDOUN high process computer of Biskra University.

In order to code in HPC or execute a parallel program, a creation of an account is important through filling a chart and sending it to the responsible.

When the account is created and ready to use, the steps of executing the program are as following:

1- Connecting to the HPC account:
Ssh userId@105.96.66.81 -p 9922
Password *****


oaboubakeur@ibnkhaldoun0:~
File Edit View Search Terminal Help

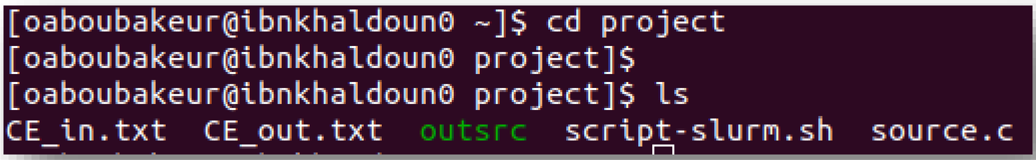
```
oaboubakeur@105.96.66.81's password:
Last login: Sun Aug 23 20:04:47 2020 from 105.105.7.101

                  Version: 5.R1.7.2 Albatross
/'' vk

[oaboubakeur@ibnkhaldoun0 ~]$
```

Figure III.22: IBN KHALDOUN HPC Interface.

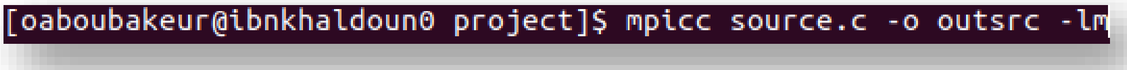
2- A folder named project was created in order to collect all of the files from code file to inputs and outputs, we run command “ls” to show the components of this folder.



```
[oaboubakeur@ibnkhaldoun0 ~]$ cd project
[oaboubakeur@ibnkhaldoun0 project]$
[oaboubakeur@ibnkhaldoun0 project]$ ls
CE_in.txt  CE_out.txt  outsrc  script-slurm.sh  source.c
```

Figure III.23: Project Folder.

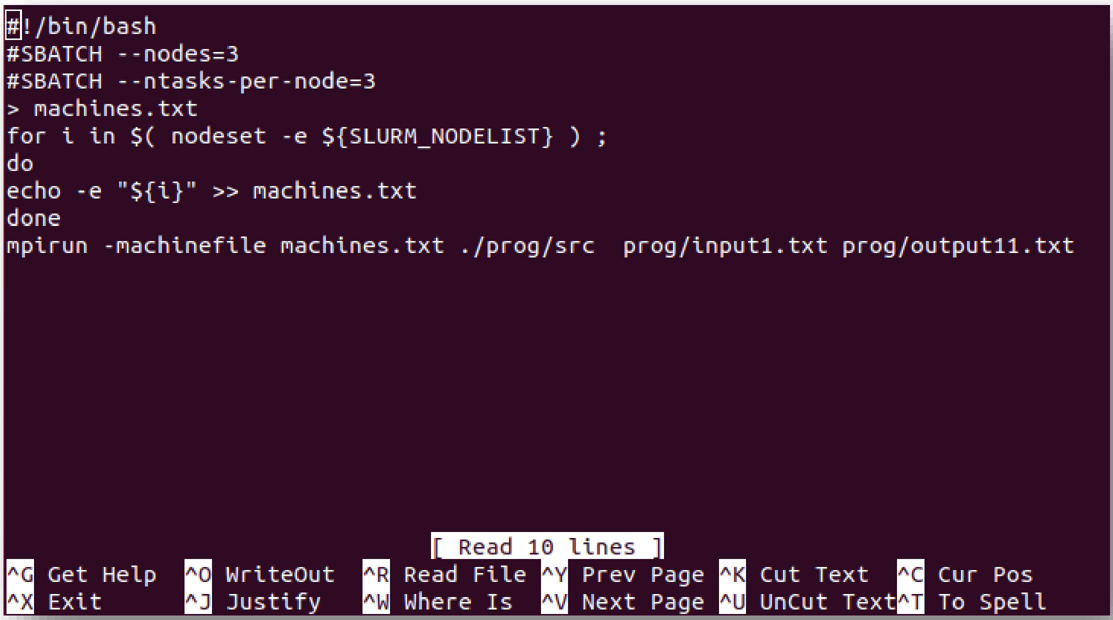
3- We compile first our parallel program using the following command "**mpicc source.c -o outsrc -lm**"



```
[oaboubakeur@ibnkhaldoun0 project]$ mpicc source.c -o outsrc -lm
```

Figure III.24: compilation of the source code.

4- In aim to execute the parallel split soil program, we created a script file as following:



```
#!/bin/bash
#SBATCH --nodes=3
#SBATCH --ntasks-per-node=3
> machines.txt
for i in $( nodeset -e ${SLURM_NODELIST} ) ;
do
echo -e "${i}" >> machines.txt
done
mpirun -machinefile machines.txt ./prog/src  prog/input1.txt prog/output11.txt
```

Figure III.25: script file.

The component of this script file are:

Nodes=3: indicates the number of nodes to use in parallel. **Ntasks-per-node**= 3: indicates the number of tasks per node. **machines.txt:** to empty the contents of the file machines.txt.

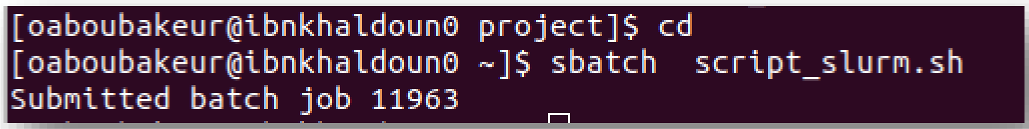
- The machines.txt file contains the machines on which the parallel program.
- The for loop is used to fill the machines.txt file by the machines reserved by SLURM.

Mpirun: command that executes a parallel program written in mpi.

Machinefile machines.txt: indicates where the machines on which we execute the program are.

.pgm: indicates the program that will be executed in parallel.

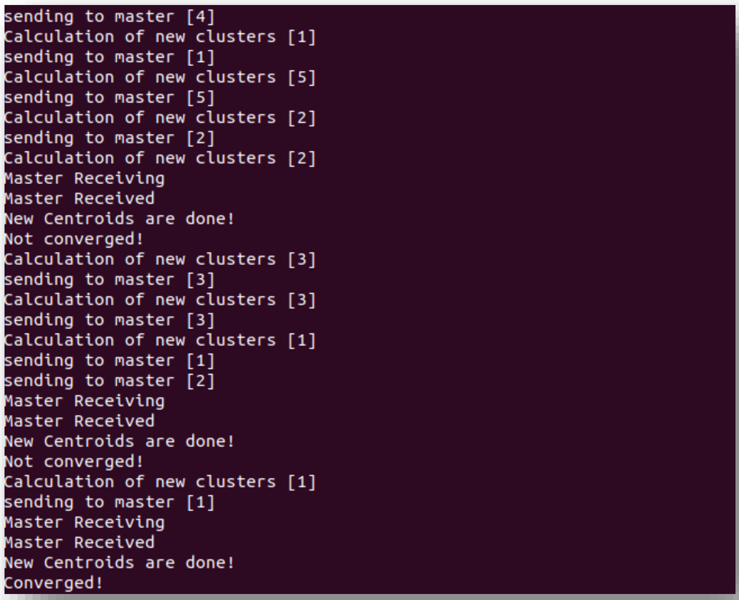
5- After launching the script by the following command line "**sbatch script_slurm.sh**" a job is created with an ID identifier.



```
[oaboubakeur@ibnkhaldoun0 project]$ cd
[oaboubakeur@ibnkhaldoun0 ~]$ sbatch  script_slurm.sh
Submitted batch job 11963
```

Figure III.26: Job creation.

6- To visualise the execution we command "**cat slurm-jobID.out**".



```
sending to master [4]
Calculation of new clusters [1]
sending to master [1]
Calculation of new clusters [5]
sending to master [5]
Calculation of new clusters [2]
sending to master [2]
Calculation of new clusters [2]
Master Receiving
Master Received
New Centroids are done!
Not converged!
Calculation of new clusters [3]
sending to master [3]
Calculation of new clusters [3]
sending to master [3]
Calculation of new clusters [1]
sending to master [1]
sending to master [2]
Master Receiving
Master Received
New Centroids are done!
Not converged!
Calculation of new clusters [1]
sending to master [1]
Master Receiving
Master Received
New Centroids are done!
Converged!
```

Figure III.27: Execution steps.

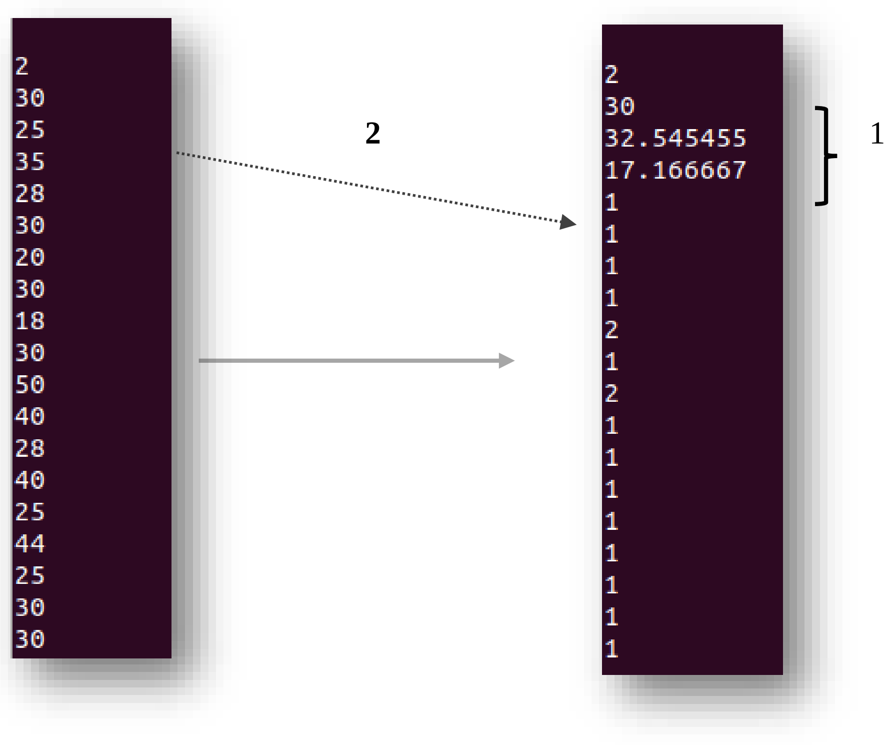
III.5.2 Execution Results

III.5.2.1 Clustering Results

This section shows the results of the parallel split soil module on each parameter by itself through inserting execution images of each input and output file.

- First line of each input and output files is the number of clusters.
- Second line of each file is the number of values.

1- Soil depth



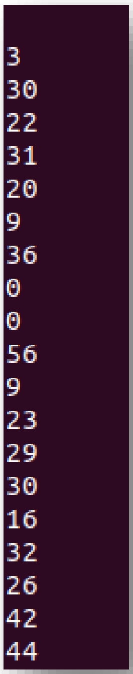
```
2
30
25
35
28
30
20
30
18
30
50
40
28
40
25
44
25
30
30
```

```
2
30
32.545455
17.166667
1
1
1
1
2
1
2
1
1
1
1
1
1
1
1
```

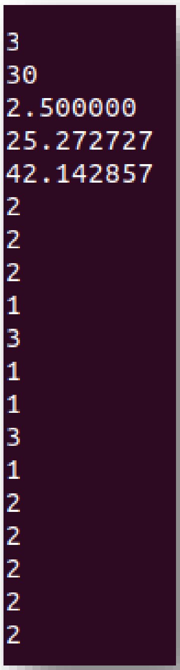
Figure III.28: Input and output of soil depth.

- 1: The values of centroids of the clusters created.
- 2: Each value of the input file corresponds to the number of cluster that exists in the output file of the same position. (Value 25 corresponds to cluster number 1).

2- Clay



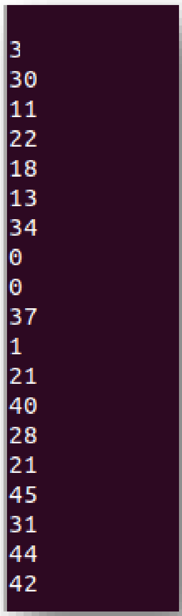
```
3
30
22
31
20
9
36
0
0
56
9
23
29
30
16
32
26
42
44
```



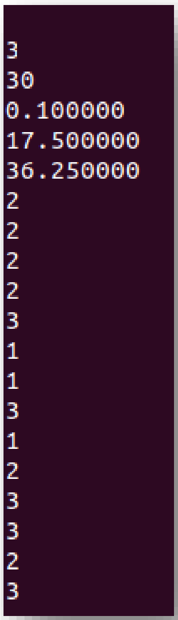
```
3
30
2.500000
25.272727
42.142857
2
2
2
1
3
1
1
3
1
2
2
2
2
2
```

Figure III.29: Input and output of Clay.

3- Fine silt



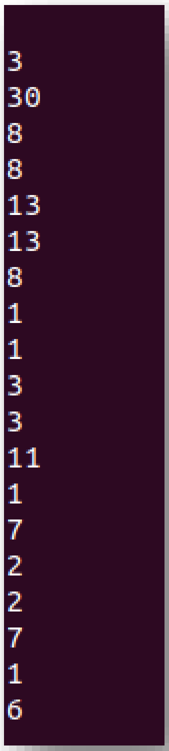
```
3
30
11
22
18
13
34
0
0
37
1
21
40
28
21
45
31
44
42
```



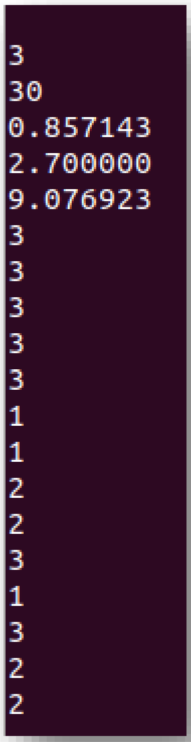
```
3
30
0.100000
17.500000
36.250000
2
2
2
2
3
1
1
3
1
2
3
3
2
3
```

Figure III.30: Input and output of Fine Silt.

4- Coarse silt



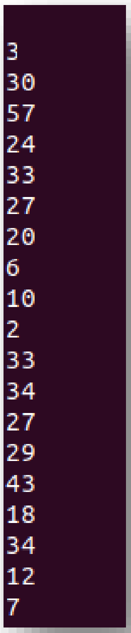
```
3
30
8
8
13
13
8
1
1
3
3
11
1
7
2
2
7
1
6
```



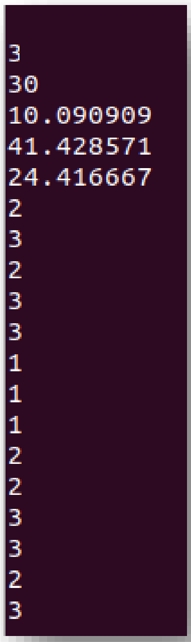
```
3
30
0.857143
2.700000
9.076923
3
3
3
3
3
1
1
2
2
3
1
3
2
2
```

Figure III.31: Input and output of Coarse Silt.

5- Fine sand



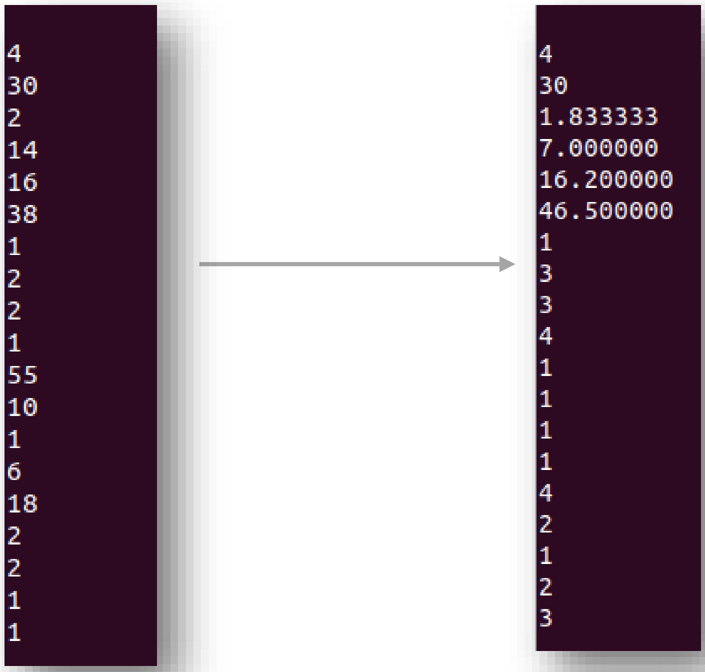
```
3
30
57
24
33
27
20
6
10
2
33
34
27
29
43
18
34
12
7
```



```
3
30
10.090909
41.428571
24.416667
2
3
2
3
3
1
1
1
2
2
3
3
2
3
```

Figure III.32: Input and output of Fine Sand.

6- Coarse sand

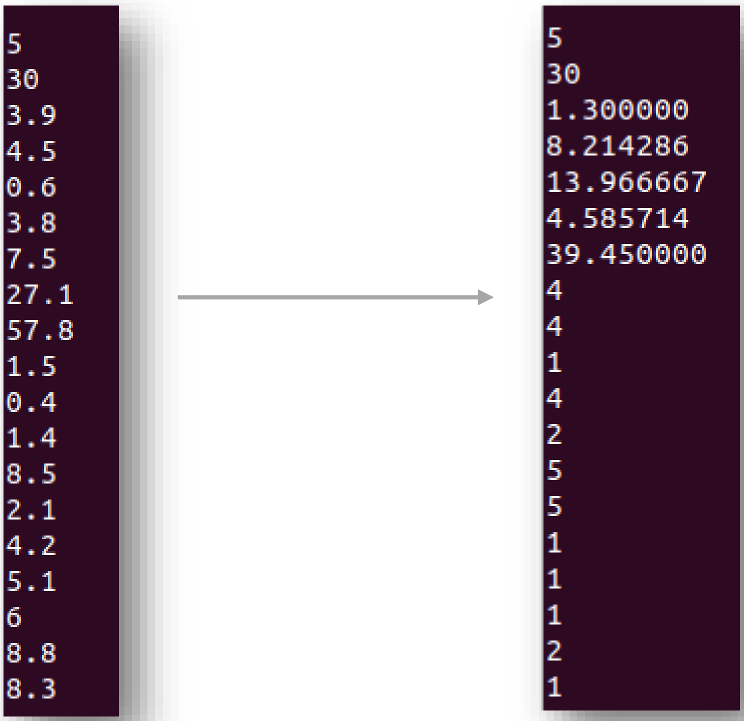


```
4
30
2
14
16
38
1
2
2
1
55
10
1
6
18
2
2
1
1
```

```
4
30
1.833333
7.000000
16.200000
46.500000
1
3
3
4
1
1
1
1
4
2
1
2
3
```

Figure III.33: Input and output of coarse sand.

7- Salinity

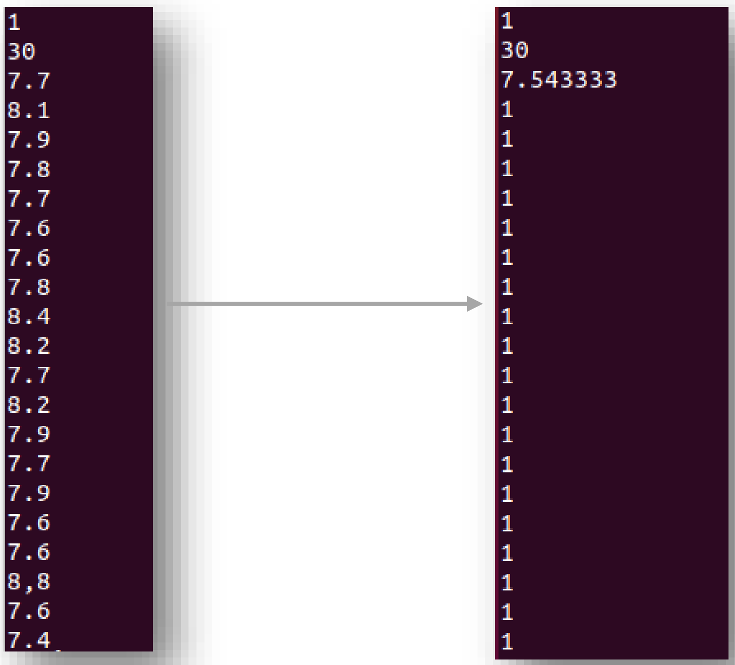


```
5
30
3.9
4.5
0.6
3.8
7.5
27.1
57.8
1.5
0.4
1.4
8.5
2.1
4.2
5.1
6
8.8
8.3
```

```
5
30
1.300000
8.214286
13.966667
4.585714
39.450000
4
4
1
4
2
5
5
1
1
1
2
1
```

Figure III.34: Input and output of salinity

8- Ph



```
1
30
7.7
8.1
7.9
7.8
7.7
7.6
7.6
7.8
8.4
8.2
7.7
8.2
7.9
7.7
7.9
7.6
7.6
8,8
7.6
7.4
```

```
1
30
7.543333
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
```

Figure III.35: Input and output of Ph.

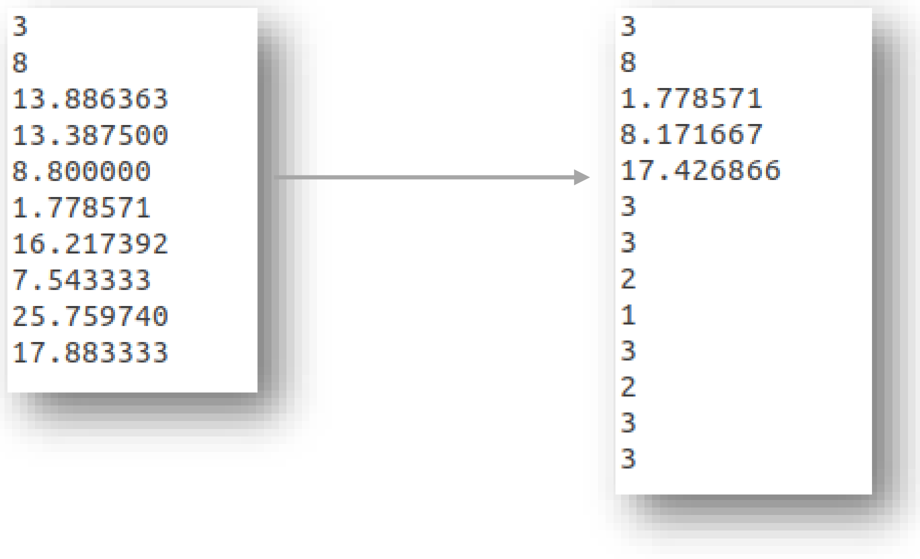
III.5.3 Fusion Results

- **Total fusion**

As mentioned in the last chapter, the total fusion aims to extract the best spots of the field or the batch soil in terms of soil parameters.

For that aim, we clustered each parameter as shown in the figures above than we re-clustered the centroids generated into only one cluster.

As a result from this step, we obtained the best spot of each parameter to finally do the clustering for a third round on the totality of parameters to obtain the elite cluster or clusters that contain/s all of the parameters homogenously.



```
3
8
13.886363
13.387500
8.800000
1.778571
16.217392
7.543333
25.759740
17.883333
```

```
3
8
1.778571
8.171667
17.426866
3
3
2
1
3
2
3
3
```

Figure III.36: Input and output of total fusion.

1- 3 Elite clusters:

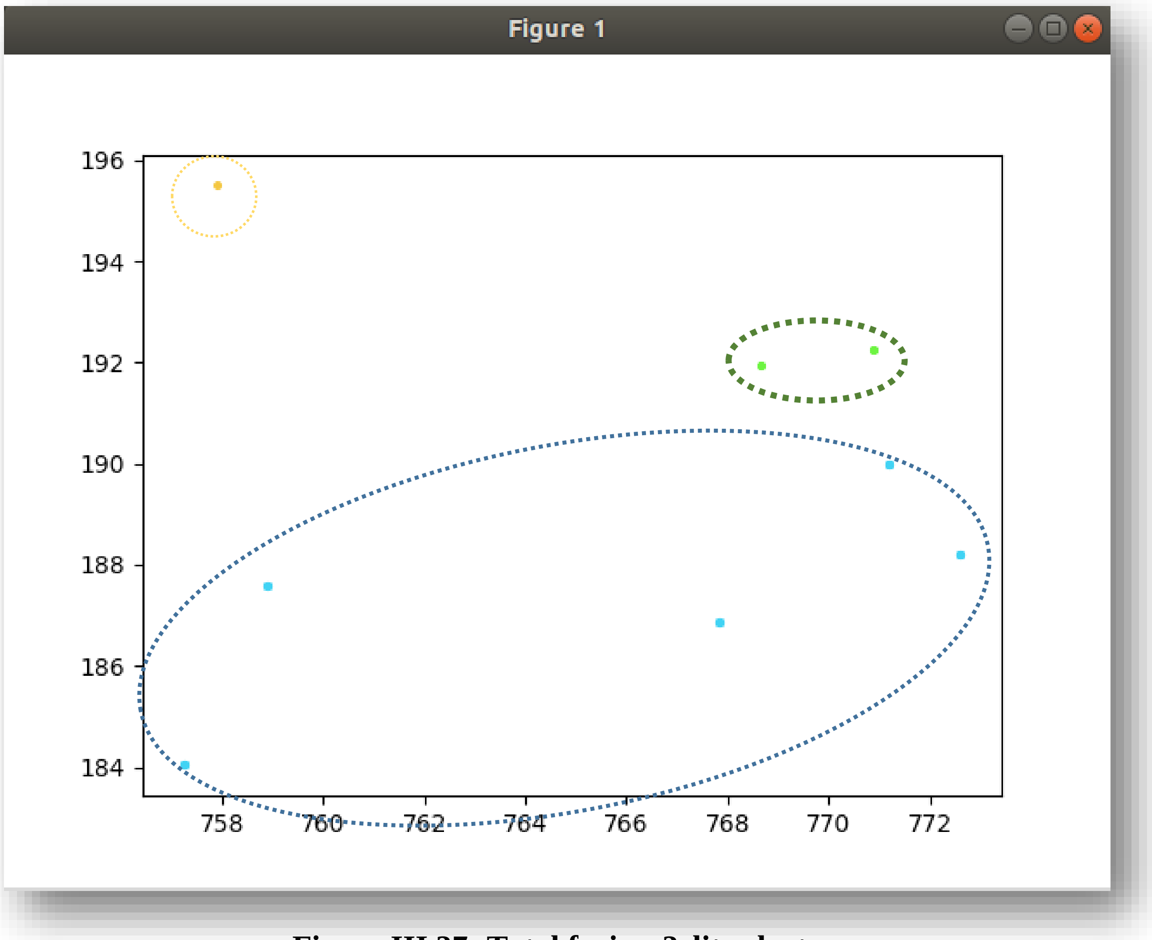


Figure III.37: Total fusion-3elite clusters.

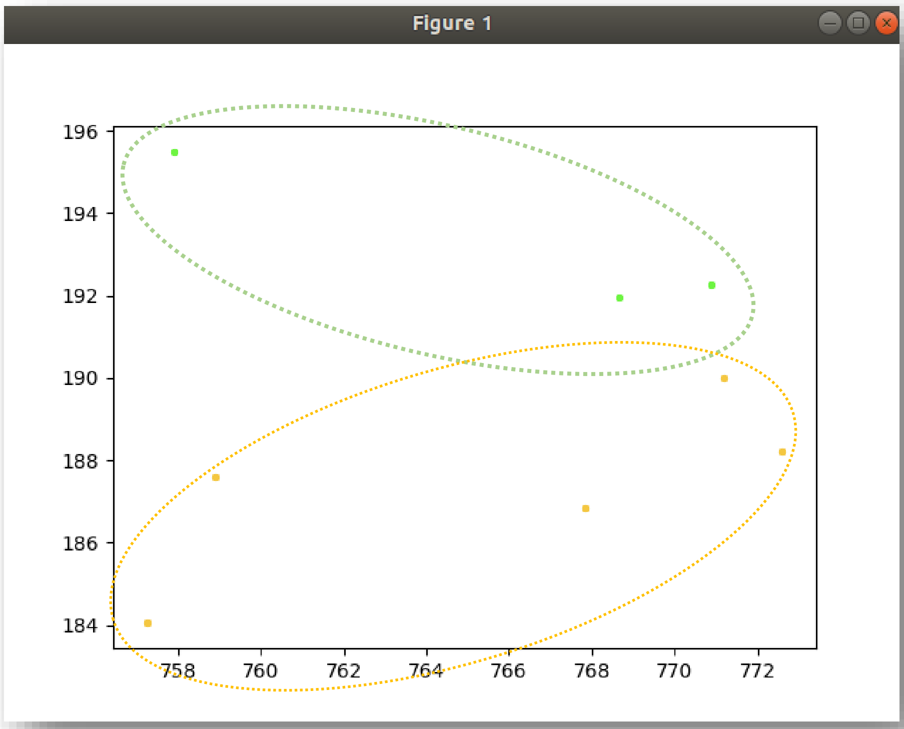


Figure III.38: Total fusion-2elite clusters.

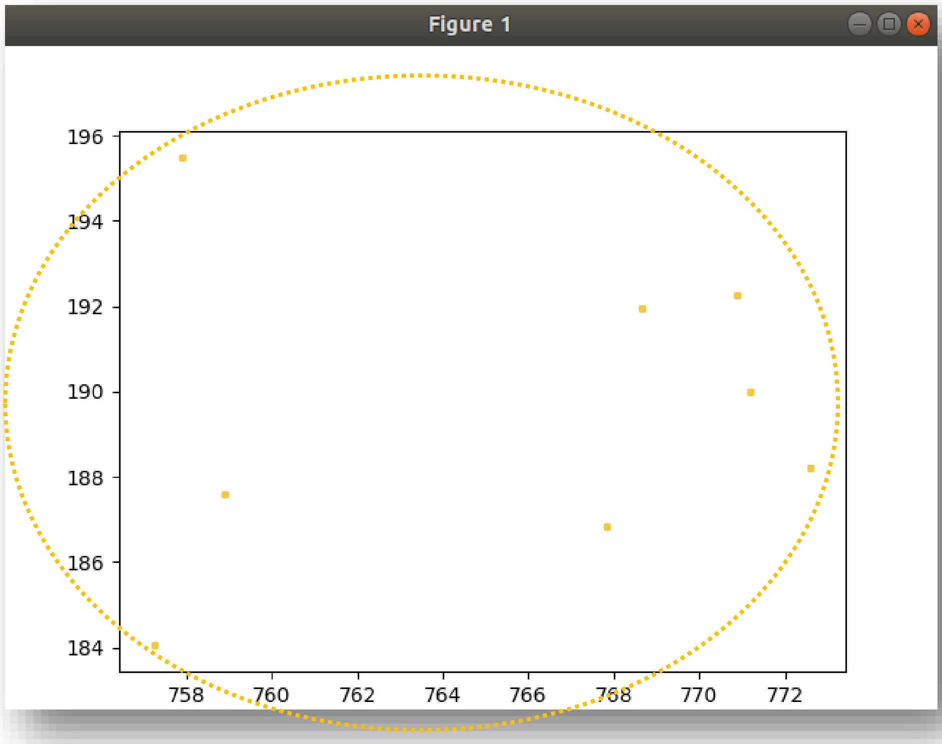


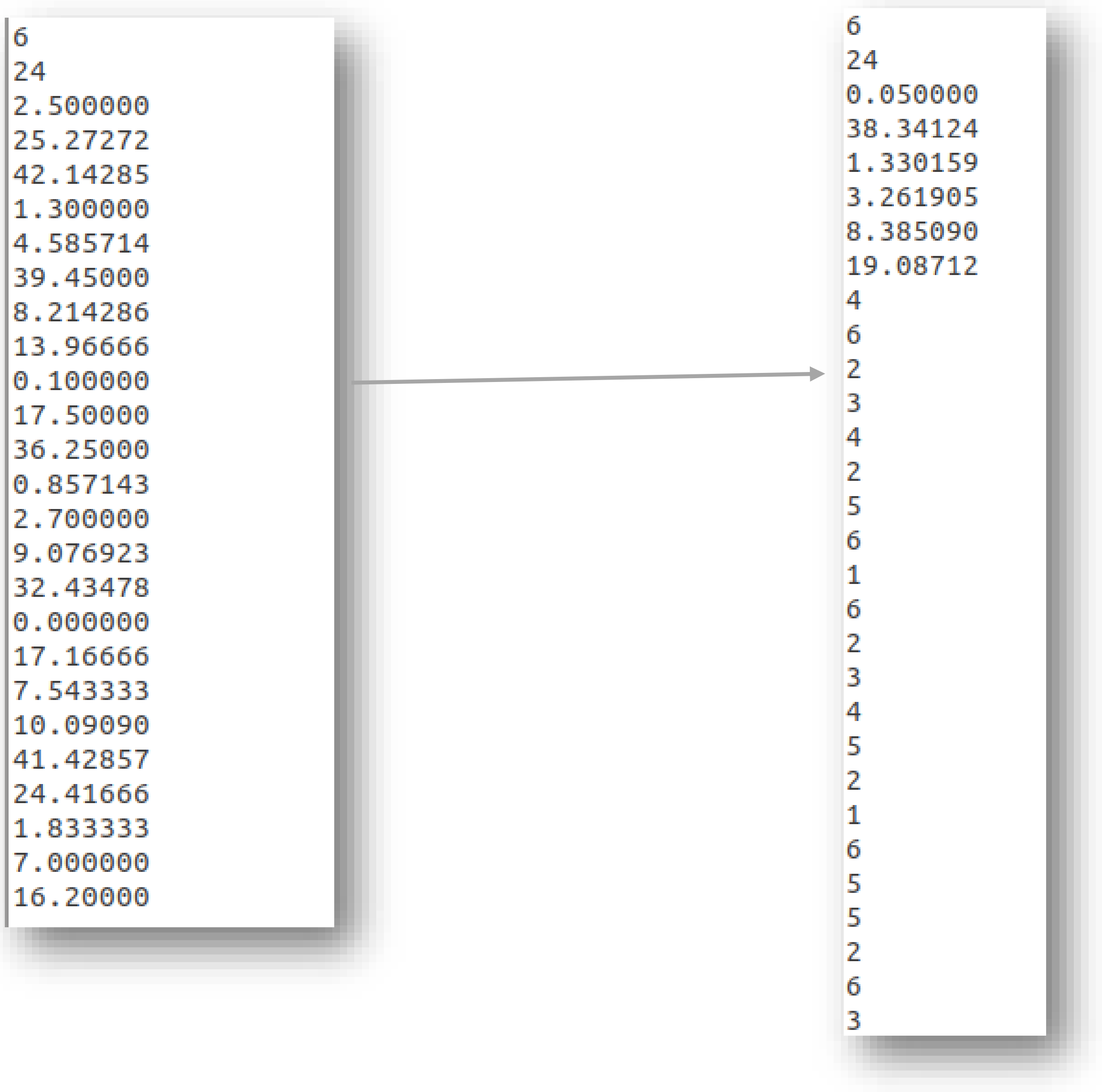
Figure III.39: Total fusion-1elite clusters.

- **Partial fusion**

The partial fusion is no different from the total fusion, both methods extracts the best spots in the batch field.

This method relays on the results of clustering of each parameter, the centroids generated are gathered into one input file and splatted to as many spots as needed.

As following:



```
6
24
2.500000
25.27272
42.14285
1.300000
4.585714
39.45000
8.214286
13.96666
0.100000
17.50000
36.25000
0.857143
2.700000
9.076923
32.43478
0.000000
17.16666
7.543333
10.09090
41.42857
24.41666
1.833333
7.000000
16.20000
```

```
6
24
0.050000
38.34124
1.330159
3.261905
8.385090
19.08712
4
6
2
3
4
2
5
6
1
6
2
3
4
5
2
1
6
5
5
2
6
3
```

Figure III.40: Partial fusion input and output.

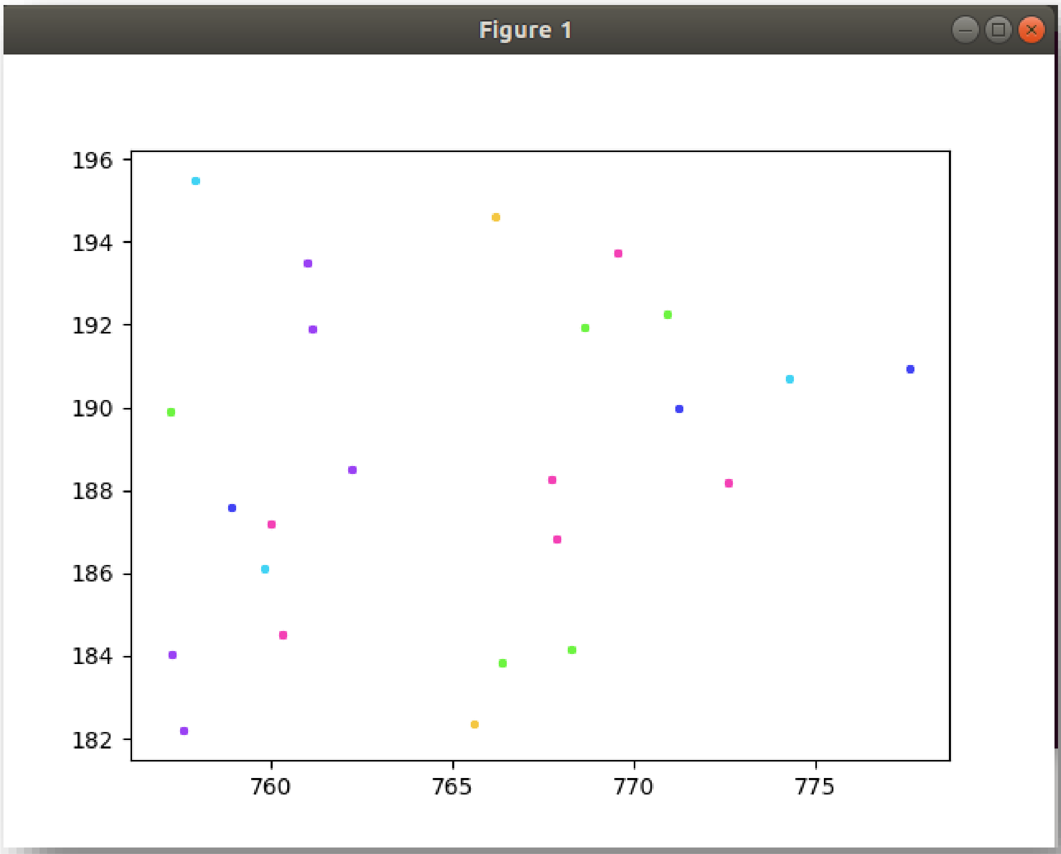


Figure III.41: Partial fusion.

Points of the same colour presents the same cluster.

- In the **figure III.41** we can notice that the clusters are dispersed in the batch field and not gathered.
  The clustering in the first step was based on the parameter itself, but when we launched the split soil module for the second time, the similarity measure changed which means that the clustering was done based on the distance between different centroids of different parameters, to give as a final result dispersed but good quality clusters.

The difference between partial and total fusion is:

- In total fusion, the merging between the clusters is hierarchical which automatically lead to elite and clear formed clusters that contains all parameter values homogenously. **(Figures III.37, 38, 39)**

- In partial fusion, the merging between clusters in the second phase was done arbitrary which lead to good clusters but not elite, because each cluster contains more than one parameter. **(Figure III.41)**

On the field, total fusion leads to homogenous and uniform planting because each elite cluster would be convenient to a specific harvesting.

The partial fusion in the other hand leads on the field to disparity and diversity.

III.5.4 Comparison

After showing the results of application of the parallel split soil module on each parameter, we devote this section to compare between both parallel and sequential implementations of the k-means algorithm.

This comparison would be based on the runtime of each algorithm on each parameter since the k-means is known to be a time consuming algorithm.

We present the following table that gives both runtimes of each parameter against each other.

Parameter	Cluster number	Sequential Runtime (s)	Parallel Runtime (s)
Soil depth	**2**	**0.17400**	**0.007673**
Clay	**3**	**0.26900**	**0.007956**
Fine Silt	**3**	**0.29400**	**0.007600**
Coarse Silt	**3**	**0.30100**	**0.007277**
Fine Sand	**3**	**0.19700**	**0.008353**
Coarse Sand	**4**	**0.23000**	**0.007583**
Salinity	**5**	**0.13500**	**0.007400**
Ph	**1**	**0.04400**	**0.006712**

Table III.1: Comparison table between sequential and parallel implementations.

Sequential average runtime	Parallel average runtime (s)
0.208875	0.0075342

Table III.2: Average runtimes of the 8 parameters.

The problem of time consuming in sequential k-means applications has always been known as a con, from what we notice in the tow tables above, we can clearly see that the sequential implementation is very heavy time consuming compared to the parallel one. Regardless of the fact that this database in only 30 values per parameter which means that applying the sequential k-means on larger database would take an enormous time.

On the other hand, in parallel implementation we succeeded to minimise the time consuming problem to the max as we can see above in **Table III.1 and Table III.2.**

III.6 Conclusion

This chapter is the last step into this work, we had sight on the actual conception of this application passing by the implementation of the split soil module to conclude with the results obtained through executing in HPC and comparing between sequential and parallel implementations of our application.

General Conclusion

General Conclusion

The development of computer science throughout the last two decades have resulted a total merging in different fields other than technologies as the agricultural field for example.

The union between agriculture and computer science came in to solve huge international issues that affects the world which is the case in our study where we used the best of machine learning methods in order to help with food security problems.
Our project had the objective to introduce the farmer to his field through splitting this last into groups of batch soils with homogenous parameters in order to perfect the exploitation of these batches and maximise the profitability as minimise the food loss.

In the first chapter of this thesis we introduced the international classification of soils in agriculture than detailed the most known and used clustering algorithms to give a general glance of the work to finally present a synthesis with previous works in the same domain.

The second chapter took advantage of using the HPC of University of Biskra to explain modern computing also as cluster-specific components and its interconnection technology.

The third and last chapter of this thesis was devoted to the application, where we have done a conception than followed it with the implementation of our parallel split soil model in the high process computer of IBN KHALDOUN to finally conclude with the experimental results on El Outaya database where we illustrated the results on plot graphs as we showed the efficacy of our parallel implementation against the sequential in terms of time.

Bibliography

[1] Parikh, S. J. & James, B. R. (2012) Soil: The Foundation of Agriculture. *Nature Education.* URL: https://www.nature.com/scitable/knowledge/library/soil-the-foundation-of-agriculture-84224268/. Accessed on: August 12/2020.

[2] ClassZone. URL: http://www.classzone.com/cz/index.htm. Accessed on: August 30/2020.

[3] FAO SOILS PORTAL. URL:http://www.fao.org/soils-portal/soil-survey/soil-classification/ru/. Accessed on: July 15/2020.

[4] AgFunderNews. AFN. URL: https://agfundernews.com/what-is-precision-agriculture.html. Accessed on March 11/2020.

[5] Nicole Rogers. Sustainable America. URL: https://sustainableamerica.org/blog/what-is-precision-agriculture. Accessed on February 24/2020.

[6] Mailloux, A., Dubé, A. & Tardif, L. (1964). Classement des sols selon leurs possibilités d'utilisation agricole. Cahiers de géographie du Québec, 8 (16), 231–249. URL : https://doi.org/10.7202/020501ar

[7] Google developer's site. Clustering in Machine Learning. URL: https://developers.google.com/machine-learning/clustering/overview. Accessed on March 11/2020.

[8] Mohammed El-Helly, Hoda Onsi, Ahmed Rafea, Salwa El-Gammal. Segmentation Technique for Detecting Leaf Spots in Cucumber Crop Using Fuzzy Clustering Algorithm. 2003. 14. URL: http://www.arc.sci.eg/NARIMS_upload/CLAESFILES/3914.pdf

[9] Georg Ruß, Rudolf Kruse and Martin Schneider. A CLUSTERING APPROACH FOR MANAGEMENT ZONE DELINEATION IN PRECISION AGRICULTURE. 2010. 14. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.712.3898&rep=rep1&type=pdf.

[10] By Joseph D. Sloan, High Performance Linux Clusters with OSCAR, Rocks, OpenMosix, and MPI. O'Reilly, 2004.

[11] Moreno Baricevic, Stefano Cozzini. Workshop on High Performance Computing (HPC) Architecture and Applications in the ICTP, 2013.

[12] Pascale Vicat-Blanc, Sébastien Soudan, Romaric Guillier, Brice Goglin. Computing Networks, from cluster to cloud computing. ISTE Ltd and John Wiley & Sons, Inc. 2011.

[13] Archana Singh, Avantika Yadav, Ajay Rana. K-means with three different Distance Metrics. International Journal of Computer Applications (0975 – 8887) Volume 67– No.10, April 2013

[14] Margaret Rouse. Search Enterprise Desktop .URL: https://searchenterprisedesktop.techtarget.com/definition/message-passing-interface-MPI.Accessed on: June 20/2020.

[15] Guru 99. What is C Programming Language? Basics, Introduction and History. URL: https://www.guru99.com/c-programming-language.html. Accessed on: June 21/2020.

[16] HPC IBN KHALDOUN.
URL: http://hpc.univ-Biskra.dz/index.php/presentation/hpc-ibnkhaldoun.
Accessed on June 25/2020.

[17] CRSTRA. Presentation.
URL: https://www.crstra.dz/en/presentation.php. Accessed on June 25/2020.

[18] DIS4ME. Desertification Indicator System For Mediterranean Europe. URL: https://esdac.jrc.ec.europa.eu/public_path/shared_folder/projects/DIS4ME/indicator_descriptions/soil_depth.htm. Accessed on: August 30/2020.

Bibliography

[19] Gerhard Lagaly, Werner Tufar , A. Minihan,A. Lovell. Silicates. 15 June 2000. URL : https://onlinelibrary.wiley.com/doi/abs/10.1002/14356007.a23_661.

[20] Morgan Stanley National Geographic. Encyclopaedia Entering: Silt. URL: https://www.nationalgeographic.org/encyclopedia/silt. Accessed on: July 1/2020.

[21] 123RF. URL: https://fr.123rf.com/sand-is-a-naturally-occurring-granular-material-composed-of-finely-divided-rock-and-mineral-particle.html. Accessed on June 30/2020.