# Fuzzy-Based Kernelized Clustering Algorithms for Handling Big Data Using Apache Spark

**Chapter** · January 2021

DOI: 10.1007/978-981-15-8603-3_37

**6 authors**, including:

Preeti Jha
Indian Institute of Technology Indore
**12** PUBLICATIONS **34** CITATIONS
SEE PROFILE

Neha Bharill
Indian Institute of Information Technology Dharwad
**35** PUBLICATIONS **1,105** CITATIONS
SEE PROFILE

Milind Ratnaparkhe
ICAR Indian Institute of Soybean Research
**85** PUBLICATIONS **1,905** CITATIONS
SEE PROFILE

ScienceDirect
www.sciencedirect.com

00 (2022) 1–14

# Apache Spark Based Kernelized Fuzzy Clustering Framework for Single Nucleotide Polymorphism Sequence Analysis

Preeti Jha[a,1,*], Aruna Tiwari[a], Neha Bharill[b], Milind Ratnaparkhe[c], Mukkamalla Mounika[a], Neha Nagendra[a]

[a] *Indian Institute of Technology Indore, 453552*
[b] *Mahindra University, Ecole Centrale School of Engineering , Hyderabad, 500043*
[c] *ICAR-Indian Institute of Soybean Research Indore, 452001*

## Abstract

This paper introduces a kernel based fuzzy clustering approach to deal with the non-linear separable problems by applying kernel Radial Basis Functions (RBF) which maps the input data space non-linearly into a high-dimensional feature space. Discovering clusters in the high-dimensional genomics data is extremely challenging for the bioinformatics researchers for genome analysis. To support the investigations in bioinformatics, explicitly on genomic clustering, we proposed high-dimensional kernelized fuzzy clustering algorithms based on Apache Spark framework for clustering of Single Nucleotide Polymorphism (SNP) sequences. The paper proposes the Kernelized Scalable Random Sampling with Iterative Optimization Fuzzy c-Means (KSRSIO-FCM) which inherently uses another proposed Kernelized Scalable Literal Fuzzy c-Means (KSLFCM) clustering algorithm. Both the approaches completely adapt the Apache Spark cluster framework by localized sub-clustering Resilient Distributed Dataset (RDD) method. Additionally, we are also proposing a preprocessing approach for generating numeric feature vectors for huge SNP sequences and making it a scalable preprocessing approach by executing it on an Apache Spark cluster, which is applied to real-world SNP datasets taken from open-internet repositories of two different plant species, i.e., soybean and rice. The comparison of the proposed scalable kernelized fuzzy clustering results with similar works shows the significant improvement of the proposed algorithm in terms of time and space complexity, Silhouette index, and Davies-Bouldin index. Exhaustive experiments are performed on various SNP datasets to show the effectiveness of proposed KSRSIO-FCM in comparison with proposed KSLFCM and other scalable clustering algorithms, i.e., SRSIO-FCM, and SLFCM.

*Keywords:*
High-dimensional, Non-linear, Apache Spark, SNP Sequences, Kernelized Fuzzy Clustering

## 1. Introduction

Clustering is an unsupervised learning method, which is used on similar data samples that can be grouped to form a subset of data (Saxena et al., 2017; Nasraoui and N'Cir, 2018). Since the bio-informatics field of genomics stepped into the clustering of the high-dimensional data, this raises the requirement of developing the machine learning algorithms to handle the huge genomics data (Hosseini and Kiani, 2019, 2018; Jiang et al., 2004). With ongoing advances in genomics, the clustering of high-dimensional Deoxyribonucleic Acid (DNA) sequences can be noticed all over the place (Castellanos-GarzóN et al., 2013). Yet, the developing requirement for storage and handling huge genome data are difficult challenges to biologists (Wong, 2016). Different clustering algorithms have been designed to cluster similar genes more accurately, as indicated by identical gene sequences (Kerr et al., 2008). Nonetheless, there is no single clustering algorithm that is best for each case (Xu and Wunsch, 2005). The Micro-array datasets are unpredictable and have inherent outliers or missing values (Moorthy et al., 2014). The clustering of soybean and rice genome data and their analyzed results will help in setting a strong foundation for handling and analysis of subsequent large scale genome re-sequencing efforts in the future. Large scale sequencing of genome data has generated huge genomic, protein, and omics data. Similarly, SNP data sets are also growing exponentially. SNPs are becoming the most popular type of marker in linkage and association studies to discover genes associated with various traits such as diseases and drought resistance. Therefore, faster methods for SNP clustering are required so that accurate

*Author correspondents.
*E-mail addresses:* phd1801201006@iiti.ac.in (Preeti Jha ), artiwari@iiti.ac.in (Aruna Tiwari), neha.bharill@mechyd.ac.in (Neha Bharill ), ratnaparkhe.milind@gmail.com (Milind Ratnaparkhe), mounikamukkamalla16@gmail.com (Mukkamalla Mounika ), nehanagendra02@gmail.com (Neha Nagendra )

data analysis can be done for the identification of genes associated with various traits. Recently, genomics like SNP datasets has become incredible both in volume and complexity (Zheng et al., 2012). Subsequently, scalable clustering algorithms are expected to deal with them (Bolón-Canedo et al., 2015; Bharill et al., 2016). The analytic breakthroughs brought an impressive and remarkable generation of biological data, which was a dream some years ago, which includes sequencing of Protein, DNA, and RNA (Zhao et al., 2015). To analyze huge biological data, there is a need for efficient data storage, searching, analysis, and feature extraction methods (Oussous et al., 2018; Veiga et al., 2016).

The fuzzy clustering method applies to the items that cannot be completely divided into two different groups, and thus it is profoundly applicable for the biological things that have a slow development relationship and can not be partitioned into two particular clusters. Fuzzy clustering is a soft clustering technique intended to recognize a data sample as a level of having a place with groups and in this manner empowers the fuzzy calculation strategy to allocate a data sample to more than one cluster and associate the data sample with a group of membership levels (Popescu et al., 2009). It is often in the SNP sequences that the number of dimensions of feature is a lot higher than the number of sequences. Applying a grouping model to high dimensional data sample prompts the following issue: the separation between data samples and clusters are practically equivalent. Accordingly, all the cluster become a solitary group at the centroid of all the data sample. To overcome this problem, fuzzy clustering is used with random sampling to handle huge data clustering. Thus, the fuzzy clustering algorithm accordingly divides the data into subsets such that each subset deal with a small size of data and it takes less number of iterations to converge faster and achieve good clustering quality results while dealing with huge data sizes (Di Nuovo and Catania, 2008).

Bezdek (Bezdek et al., 1984) proposed the first fuzzy clustering algorithm Fuzzy c-Means clustering (FCM), and since then, the fuzzy clustering method has come a long way. To minimize an objective function, the FCM adopts iterative optimization that uses a similarity measure on feature space. In the majority of cases, FCM is suitable for clustering of data that has linear distribution in feature space. For managing non-linear shape clusters (Zhao et al., 2018), the concept of kernel function is introduced (Chen and Kong, 2016). To handle the existing issues in fuzzy clustering, the fuzzy kernel c-means clustering (FKCM) algorithm was proposed by integrating FCM with a Mercer kernel function (Wu et al., 2003). The FKCM algorithm is suitable for the clustering of data that form the cluster having linear and non-linear data in feature space. Many research papers have worked on the clustering of nonlinear data Huang et al. (2011); Havens et al. (2012a); Liu and Xu (2008); Tsai and Lin (2011), which addresses the shortcomings of FCM transforming nonlinear data sample into high dimensional feature space by utilizing the kernel tricks. In examining past research work, the analyst exhibited that the transformation of kernel functions could improve the Euclidean distance measure standard clustering approach.

Presently, the random sampling plus extension Fuzzy c-Means (rseFCM) is an extension of the FCM algorithm employed for handling big data (Havens et al., 2012b). The issue of rseFCM is overlapping of the clusters. The overlapping of clusters has been overcome up to an extent by Random Sampling with Iterative Optimization Fuzzy c-Means (RSIO-FCM) (Bharill and Tiwari, 2014). Still, the increment in the number of iterations is an issue found in RSIO-FCM during the clustering. The SRSIO-FCM (Bharill et al., 2016) is the scalable version of RSIO-FCM, which overcome the drawbacks like slow convergence, the rise of number of iterations, and exceedingly deferred cluster centers of RSIO-FCM. For calculation of membership matrix and cluster centers for all subsets, the SRSIO-FCM utilizes the Scalable Literal Fuzzy c-Means (SLFCM) (Bharill et al., 2016) algorithm. The SRSIO-FCM can not cluster the non-linear separable data in the feature space. However, SRSIO-FCM algorithm does not cluster the data with non-linear separable data distribution. To handle challenges like clustering of non-linear separable data and to obtain a good quality of the clustering results, we are employing a fuzzy clustering technique to genome datasets taken from the bioinformatics domain. Hence, we propose kernel based fuzzy clustering algorithms for clustering of huge SNP data along with the scalable SNP preprocessing approach.

In this paper, we extended the SRSIO-FCM (Bharill et al., 2016) algorithm to propose KSRSIO-FCM, which inherently uses the proposed KSLFCM. KSLFCM algorithm is a kernelized version of SLFCM (Bharill et al., 2016) which utilizes the kernel Radial Basis Function (RBF) to optimize the clustering quality. The proposed scalable algorithms are executed on the Apache Spark clusters to tackle the problem related to fuzzy clustering for dealing with huge SNP data. We also propose a scalable SNP preprocessing approach to extract feature vectors from SNP sequences, which can be used as an input to the KSRSIO-FCM algorithm. KSRSIO-FCM distributes the data into various partitions, and KSLFCM is performed sequentially on each subset. Our proposed algorithm deal with both linear and non-linear data by applying RBF functions, which map the input data space nonlinearly into a high-dimensional feature space.

This paper is broadly structured as follows: Section 2 illustrates a few fuzzy clustering methodologies, and a preprocessing method. Section 3 discusses Apache Spark's working. The various steps involved in the proposed KSRSIO-FCM and KSLFCM, proposed scalable SNP preprocessing approach, and the complexity analysis: run time complexity and space complexity of these two algorithms, are provided in section 4. Section 5 has the results derived from experiments conducted on several SNP datasets are reported in terms of the Silhouette index, Davies-Bouldin index, and the performance of our method using illustrative examples are discussed. Section 6 presents our conclusions.

## 2. Methods

This section discusses a few methodologies which form the basis for our proposed algorithms and various steps involved in

Table 1: Main Math Symbols

| Notation | Description |
|---|---|
| $X$ | set of data samples |
| $x_i$ | $i^{th}$ data sample |
| $M$ | membership matrix |
| $m_{ij}$ | membership degree of a data sample $x_i$ to cluster $v_j$ |
| $I$ | new membership knowledge |
| $I'$ | updated membership knowledge |
| $V$ | set of initial cluster centers |
| $V'$ | set of final cluster centers |
| $v'_j$ | updated $j^{th}$ cluster center |
| $v_j$ | initial $j^{th}$ cluster center |
| $s$ | total number of data samples |
| $c$ | number of clusters |
| $R^d$ | $d$ dimensions of $R$ feature space |
| $p$ | fuzzification parameter |
| $n$ | number of partition |
| $w$ | number of workers |

the preprocessing of SNP sequences.

## 2.1. FCM Algorithm

FCM is a widely used algorithm, in which each data sample belongs to a cluster with some degree of membership. It was originally developed by Bezdek in 1981 (Bezdek et al., 1984). In this algorithm, a dataset $X = \{x_1, x_2, x_3...x_s\}$ is an input that results the membership matrix $M$. Table 1 presents the description of the symbols that are used throughout the discussion in this paper. The FCM algorithm is built by minimizing the objective function, stated as follows:

$$J_p(M,\ V') = \sum_{i=1}^{s}\sum_{j=1}^{c} m_{ij}^{p}\|x_i - v'_j\|^2, \qquad p > 1 \tag{1}$$

KFCM, SLFCM, and SRSIO-FCM algorithms are discussed briefly in the subsequent subsection.

## 2.2. Kernel Based Fuzzy c-Means (KFCM)

KFCM algorithm uses a special technique named kernel trick (Havens et al., 2012a) to avoid some intensive computations. A kernel trick is an implicit non-linear map ($\phi$) from the input space $X$ to a high dimensional feature space $R$ (Li et al., 2017),

$$\phi : x \rightarrow \phi(x) \in R^d \tag{2}$$

where the data samples $\{x_1, x_2, ...., x_s\} \subseteq X$. In this algorithm, an input data space with lower dimension is mapped to potentially much higher-dimensional feature space ($R$) or inner product (Zaharia et al., 2012a). The inner product operation in the kernel space can be expressed by a mercer kernel represented as a function $K$ below:

$$K(x_i, v_j) = \phi(x_i)^T\phi(v_j) \tag{2a}$$

Where $x_i, v_j \in R^d$ such that $i = 1, .., s$ and $j = 1, ..., c$. Thus, using this mapping $\phi$, the kernelized version of FCM (Havens et al., 2012a) is represented as follows:

$$J_p(M,\ V') = \sum_{i=1}^{s}\sum_{j=1}^{c} m_{ij}^{p}\|\phi(x_i) - \phi(v'_j)\|^2, \qquad p > 1 \tag{3}$$

Like FCM, each data sample $x_i$ fulfills the constraint $\sum_{j=1}^{c} m_{ij} = 1$. By the kernel substitution, we have the following equation.

$$\begin{aligned}&\|\phi(x_i) - \phi(v_j)\|^2 = (\phi(x_i) - \phi(v_j))^T(\phi(x_i) - \phi(v_j))\\ &= \phi(x_i)^T\phi(x_i) - \phi(v_j)^T\phi(x_i) - \phi(x_i)^T\phi(v_j) + \phi(v_j)^T\phi(v_j)\\ &= K(x_i, x_i) + K(v_j, v_j) - 2K(x_i, v_j)\end{aligned} \tag{4}$$

In this manner, a new class of non-Euclidean distance measures in the original input space (additionally with a Euclidean distance in feature space) is achieved. Thus, different kernels will generate different measures for the original space. In this paper, $K(x_i, v_j)$ is the Radial Basis Function (RBF) kernel (Cai et al., 2007), which is a well-known kernel function and given as follows:

$$K(x_i, v_j) = \exp(-\|x_i - v_j\|^2/\sigma^2) \tag{4a}$$

where, $\sigma$ is denoted as the kernel parameter. The choice of kernel parameter is the most critical task (Cai et al., 2007). In this work, the kernel parameter is selected in the following way:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{s}(z - \bar{z})}{s - 1}} \tag{4b}$$

where, $z_i = \|x_i - \bar{x}\|$ and $\bar{z}$ is the average of all distances $z_i$. In case of RBF kernel (also called Gaussian Kernel), $K(x_i, x_i) = 1$ and $K(v_j, v_j) = 1$ (Cai et al., 2007). Thus Eq. (4) is simplified to

$$\|\phi(x_i) - \phi(v_j)\|^2 = 2(1 - K(x_i - v_j)) \tag{4c}$$

so, the Eq. (3) can be edited as:

$$J_p(M,\ V') = 2\sum_{i=1}^{s}\sum_{j=1}^{c} m_{ij}^{p}(1 - K(x_i, v'_j)) \tag{5}$$

where, $V' = \{v'_1, v'_2, ....v'_c\}$. The membership matrix $m_{ij}$ and the cluster center $v'_j$ are computed as follows:

$$m_{ij} = \frac{(1 - K(x_i, v_j))^{1/(p-1)}}{\sum\limits_{j=1}^{c}(1 - K(x_i, v_j))^{1/(p-1)}} \tag{6}$$

$$v'_j = \frac{\sum\limits_{i=1}^{s} m_{ij}^{p}K(x_i, v_j)x_i}{\sum\limits_{i=1}^{s} m_{ij}^{p}K(x_i, v_j)} \tag{7}$$

The kernel based fuzzy clustering algorithm explained in Algorithm 1. The Eq. (7) is updated until no relevant change in the values of cluster centers is recognized in Algorithm 1, and after that execution of the algorithm stops.

**Algorithm 1**: *Kernel based Fuzzy Clustering Algorithm*

```
Input: $X, c, p, \epsilon$
Output: $M, V^{'}$
1: Initialize the cluster center $V = \{v_1, v_2, ....v_c\}$ randomly.
2: Calculate the membership degree by using Eq. (6).
3: Calculate the set of updated cluster centers by using Eq. (7).
4: If $\| V' - V \| < \epsilon$ then stop.
5: Otherwise go to step 2.
```

### 2.3. Scalable Literal Fuzzy c-Means (SLFCM)

The SLFCM algorithm is implemented on Apache Spark clusters for handling big data. SLFCM (Bharill et al., 2016) algorithm computes membership degrees and cluster centers parallelly on each worker node. The membership degrees from each worker node are combined on the master node. Thereafter, the cluster center values are evaluated. This process repeats until no change in the values of cluster centers is acknowledged. Every information is saved as an array of features in Resilient Distributed Datasets (RDDs) (Zaharia et al., 2012b), which is an information structure to store items exactly in memory.

### 2.4. Scalable Random sampling with Iterative Optimization Fuzzy c-Means (SRSIO-FCM)

The SRSIO-FCM is implemented on Apache Spark clusters, distributes the data into various partitions (Bharill et al., 2016). The SRSIO-FCM uses SLFCM as an integral part of the algorithm to obtained cluster centers and the membership degrees of the first subset. The cluster centers attained from the first subset used as input to the second subset. Thereafter, the combined membership degrees obtained from the first and second subset are merged, and then the evaluated cluster centers are fed as input to the third subset. This process is executed a number of times for the clustering of all the subsequent subsets.

The kernelized version of these two approaches is explained in section 4, i.e., KSLFCM and KSRSIO-FCM algorithm.

### 2.5. Preprocessing method

A Single Nucleotide Polymorphism (SNP) is a DNA sequence variation occurring when a single nucleotide adenine ($A$), thymine ($T$), cytosine ($C$), or guanine ($G$) in the genome varies between members of a species or paired chromosome in an individual. The SNP is polymorphism which occurs within two DNA sequences, and the difference between single bases by addition, deletion, transversion, the change of the chromosome. A DNA sequence is formed from $A$, $T$, $G$, and $C$ nucleotides. For instance, an SNP may change the nucleotide $C$ with the nucleotide $T$ in a DNA sequence. Each SNP sequence is a lengthy collection of nucleotides, which can be challenging to work around, and hence, the aim is to reduce the length of sequences and extract useful features in the form of float values. We started with the approach given in the paper (Liu et al., 2006). In this paper, each DNA sequences is transformed into a 12-dimensional numeric feature vector. We used the similar approach on SNP data to extract the proper features. In this work, we build an Apache Spark cluster to convert SNP data into 12 features, in which 12 elements represent each nucleotide. All the sequences must be of the same length to achieve good results. The steps used for preprocessing of SNP data are stated as follows:

1. The first parameter in the feature extraction is the nucleotide $A$, $T$, $G$ and $C$ content from SNP sequence. The integers $l_A, l_T, l_G$, and $l_C$ are the total length of $A, T, G$, and $C$ respectively, and these four integers denote number of nucleotide $A, T, G$, and $C$ in the SNP sequence.
2. The second parameter chosen for the feature extraction is the sum of distances of each nucleotide base to the first nucleotide. The total distance $T_i$ is defined as follows:
$$T_i = \sum_{j=1}^{l_i} T_j \tag{8}$$
where, $i = A, T, G, C$; $t_j$ is the distance from the first nucleotide to the $j^{th}$ nucleotide of $i$ in the SNP sequence. $T_A, T_T, T_G$, and $T_C$ are the four feature vectors that denotes the total distances for $A, T, G$, and $C$ respectively.
3. The third parameter chosen for the feature vector extraction is the distribution of each nucleotide along the SNP sequence. The variance of distance for each nucleotide utilized to characterize the distribution is defined as follows:
$$D_i = \sum_{j=1}^{l_i} \frac{(t_j - \mu_i)^2}{l_i} \tag{9}$$
where, $i = A, T, G, C$; $t_j$ is the distance from the first nucleotide to the $j^{th}$ nucleotide of $i$ in the SNP sequence and $\mu_i = \frac{T_i}{l_i}$ So, the feature vector, which contains 12-dimensional data is given as follows:
$$\langle l_A, T_A, D_A, l_T, T_T, D_T, l_G, T_G, D_G, l_C, T_C, D_C \rangle$$

The propose KSRSIO-FCM algorithm takes the preprocessed 12-dimensional numeric feature vectors of SNP sequences as input and produces output in terms of cluster centers belong to each data sample. In section 4, we present a detailed explanation of the proposed KSRSIO-FCM and KSLFCM algorithms. Prior to introducing the proposed algorithms, we are giving the details of a well known framework for big data processing in section 3.

## 3. Apache Spark

The proposed approaches KSLFCM, KSRSIO-FCM, and scalable SNP preprocessing can be made scalable using Apache Spark clusters for handling huge SNP sequences. Apache Spark is a scalable in-memory computation framework for big data processing. It allows subsets of the dataset to be processed in parallel across a cluster. Apache Spark is a high-speed cluster computing system with efficient and straightforward development APIs which allows worker node to access dataset iteratively and execute efficiently. The in-memory cluster computing technique of spark increases the processing speed of an application by implementing a spark job on Hadoop framework to share a cluster and dataset while satisfying consistent levels of service and response. Apache Spark works with YARN in HADOOP to access data from spark engines (Borthakur et al., 2008). Spark builds with a stack of libraries, including SQL and Data Frames, MLlib, GraphX, and Spark Streaming. MLlib is a library that provides a machine learning algorithm for data science techniques.

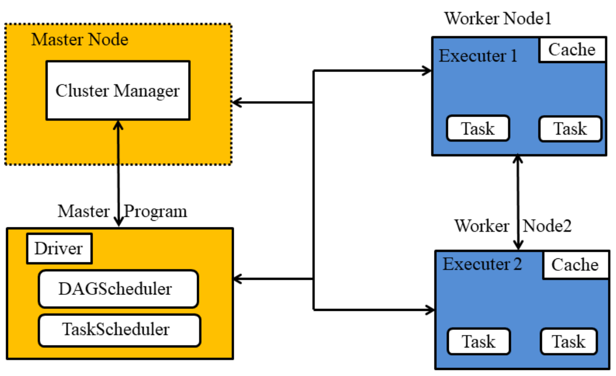


Figure 1: Architecture overview of Spark.

Figure 1 shows the overview of Apache Spark clusters. Apache Spark cluster consists of one master node and the number of worker nodes. The master node is known as a driver which is used for task scheduling. Spark hierarchically initiates a scheduling process with jobs, steps, and tasks. The step is a subset of tasks partitioned from collective jobs, which is used to match map and reduce phase. Apache Spark comprises DAG Scheduler and Task Scheduler. The DAG Scheduler calculates a directed acyclic graph (DAG) of steps for a job. It also keeps track of the record of RDD with each step outputs, whereas the Task Schedule submits tasks from each step to the cluster. Apache Spark provides different cluster modes to the user for the execution of their Apache Spark program by allowing the master node to connect to a cluster manager, standalone in our case. There is an executor created for each program for each worker node. The executor is responsible for running the tasks and caching the data in memory or disk (Tang et al., 2018).

## 4. Proposed method

In this paper, we proposed a kernelized version of SRSIO-FCM and SLFCM (Bharill et al., 2016) clustering algorithms termed as KSRSIO-FCM and KSLFCM, respectively. We also proposed a scalable SNP preprocessing approach to extract feature vectors from huge SNP sequences. The three contributory approaches overcome big data handling issues. The working of all the three algorithms is explained in the following subsections.

### *4.1. Kernelized Scalable Literal Fuzzy c-Means (KSLFCM)*

The concept of the kernel trick is introduced to handle non-linear relation. The RBF function is used to incorporate kernel trick to KSLFCM algorithm. The kernel RBF is characterized in Eq. (4a), which maps the input data space non-linearly into a high dimensional feature space (Jha et al., 2020). The KSLFCM algorithm is implemented on the Apache Spark clusters. The data samples and cluster center values are used to calculate membership degree using Eq. (6).

**Algorithm 2**: *KSLFCM to Iteratively Minimize $J_p(M, V')$*

```
Input: $X, c, p, \epsilon$
Output: $I', V'$
1: Initialize the cluster center $V = \{v_1, v_2, ....v_c\}$ randomly.
2: Calculate membership knowledge by using Eq. (6).
   $I' = X.Map(V).ReduceByKey()$
3: Calculate cluster centers given in Eq. (7).
4: If $\| V' - V \| < \epsilon$ then stop, otherwise continue with step 2.
5: Return $I', V'$.
```

In Algorithm 2, the membership degree of each data sample is evaluated parallelly on different worker nodes. In Line 2 of Algorithm 2, the Map and ReduceByKey functions are used to obtain the parallel evaluation of the membership degree of each data sample. Thereafter, the cluster center values are updated using membership degrees of each data samples, in Line 3 of Algorithm 2. The membership degree of entire data samples is fused and preserved as an updated membership knowledge $I'$, which is used to update the cluster center $v'_j$ using Eq. (7). Furthermore, in Line 4 of Algorithm 2, the difference between the previous cluster center and updated cluster center is calculated. Algorithm 2, repeat this process until no change in the values of cluster centers is identified. After that, the sequential computation of the entire iterations is done since the updated cluster centers are fed as input to the next iteration.

In Algorithm 2, Eq. (7) is used to calculate the cluster centers ($V'$) for each sample $x_i$ and cluster center $v_j$ by using the membership matrix $M$. The parameter present in the numerator and denominator of Eq. (7), i.e., $m_{ij}^p K(x_i, v_j)x_i$, and $m_{ij}^p K(x_i, v_j)$ is being calculated independently, which saves the lot of space by not storing huge membership matrices. Thereafter, the aggregation of all the values of $m_{ij}^p K(x_i, v_j)x_i$ and $m_{ij}^p K(x_i, v_j)$ which is represented as $sum\_d_jx$ and $sum\_d_j$ of the data samples corresponding to the cluster center $v_j$ is used to calculate the numerator and denominator of Eq. (7). Then the obtained output is stored as updated membership knowledge in a variable $I'$. Hence, the enormous amount of space and computational time are saved (Kolen and Hutcheson, 2002). Figure 2 demonstrates the methodology of space optimization by not saving membership degrees of the subsets.

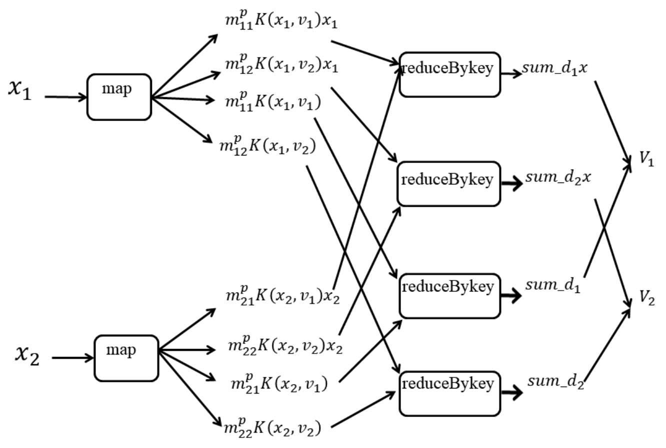


Figure 2: Demonstrates the methodology of space optimization.

### *4.1.1. Map Function*

The dataset is partitioned across all the worker nodes, and each worker computes the allocated task. The cluster center values and data sample are used to evaluate the membership degree of a point. So, the membership degrees calculation of two data samples is independent of each other. Thus, each worker node computes this operation independently and fuse the obtained results on the master node.

In Algorithm 3, the Map function estimates the membership degree of each data sample corresponding to each cluster center and returns each results independently. Hence, the output of each Map function depends on the number of clusters.These results are stored in the form of RDDs (Zaharia et al., 2012b). Algorithm 3, shows the set of tasks performed during a Map function to get $m_{ij}^pK(x_i,v_j)x_i$, and $m_{ij}^pK(x_i,v_j)$ for each data sample $x_i$ with respect to cluster center $v_j$.

**Algorithm 3**: *Map(x,V)*

```
Input: $x_i, V$
Output: $< j, < m_{ij}^pK(x_i,v_j)x_i, m_{ij}^pK(x_i,v_j) >>$
1: for each $v_j$ in $V$ do
2:   $j$ represents the index of cluster center $v$.
3:   $m_{ij}^pK(x_i,v_j) = m_{ij}^p$(membership degree of $x_i$ concerning $v_j$,
     $K(x_i,v_j)$ (kernelized value for $i^{th}$ data sample of $x$ in the
     $j^{th}$ cluster center of $v$).
4:   $m_{ij}^pK(x_i,v_j)x_i = m_{ij}^pK(x_i,v_j) * x_i$
5:   yield $< j, < m_{ij}^pK(x_i,v_j)x_i, m_{ij}^pK(x_i,v_j) >>$
6: end for
```

### *4.1.2. ReduceByKey Function*

The Map function results in many key-value pairs with the same key value. ReduceByKey function performs operations on these key-value pairs having the same value for a key. To explain things better, an operation that is performed on two such key-value is described here. Spark is used here to perform the same operations on all the key-value pairs.

**Algorithm 4**: *ReduceByKey($r_1, r_2$)*

```
Input: $r_1, r_2$ such that $r_1 =< j, < (m_{ij}^pK(x_i,v_j)x_i)_{r1}, (m_{ij}^pK(x_i,v_j))_{r1} >>$
       $r_2 =< j, < (m_{ij}^pK(x_i,v_j)x_i)_{r2}, (m_{ij}^pK(x_i,v_j))_{r2} >>$
Output:$< j, < sum\_d_jx, sum\_d_j >>$
1: $sum\_d_jx = (m_{ij}^pK(x_i,v_j)x_i)_{r1} + (m_{ij}^pK(x_i,v_j)x_i)_{r2}$
2: $sum\_d_j = (m_{ij}^pK(x_i,v_j))_{r1} + (m_{ij}^pK(x_i,v_j))_{r2}$
3: return:$< j, < sum\_d_jx, sum\_d_j >>$
```

The calculation of numerator and denominator in Line 3 of Algorithm 2 is performed to update the cluster centers in Line 3 of Algorithm 2. Since we have evaluated, $m_{ij}^pK(x_i,v_j)x_i$, and $m_{ij}^pK(x_i,v_j)$ for each data sample $x_i$ corresponding to each cluster center $v_j$. At the time of Map evaluations, all these values are added, and this operation is done by the ReduceByKey function, which is represented in Algorithm 4. The output obtained for each cluster center $v_j$ as $sum\_d_jx$ and $sum\_d_j$, respectively using the numerator and denominator of Eq. (7).

The parameters $r_1$ and $r_2$ in Algorithm 4 are the result of two Map functions, regarding cluster center $v_j$, $(m_{ij}^pK(x_i,v_j)x_i)_{r1}$ and $(m_{ij}^pK(x_i,v_j)x_i)_{r2}$ signifies the estimation of $m_{ij}^pK(x_i,v_j)x_i$ comparing to Map function yields $r_1$ and $r_2$ respectively, $(m_{ij}^pK(x_i,v_j))_{r1}$ and $(m_{ij}^pK(x_i,v_j))_{r2}$ signifies the estimation of $m_{ij}^pK(x_i,v_j)$ relating to Map function yields $r_1$ and $r_2$ respectively. The results obtained from the ReduceByKey function are used for computation of values of the new cluster center using Eq. (7) on the master node.

### *4.2. Kernelized Scalable Random Sampling with Iterative Optimization Fuzzy c-Means (KSRSIO-FCM)*

The proposed KSRSIO-FCM partitions the data into various worker nodes, and then clustering is performed on each worker node. The steps of KSRSIO-FCM is explained in Algorithm 5.

**Algorithm 5**: *KSRSIO-FCM to Iteratively Minimize $J_p(M,V')$*

```
Input: $X, c, p, \epsilon$; $X = \{x_1, x_2, ....x_s\}$, $X$ denotes an array of the data samples.
Output: $I', V'$
1: Divide $X$ into $n$ subsets; $X = \{X_1, X_2, .......X_n\}$.
2: Select $X_1$ from $X$ randomly without replacement.
3: $I', V' = KSLFCM(X_1, c, p, \epsilon)$
4: for $j = 2$ to $n$ do
    4.1: $I, V' = KSLFCM(X_j, V', c, p, \epsilon)$
    4.2: Merge all the processed partitions.
         $I' = I' \cup I$
    4.3: Calculate updated cluster center $v_j'$ using Eq. (7)
    end for
5: Calculate the objective function using Eq. (5).
6: Return $I', V'$
```

The entire dataset is divided into various subset randomly by KSRSIO-FCM algorithm. Data samples in a single partition are different from other partitions (subsets). At first, the cluster centers are initialized randomly. The cluster centers ($V$) and membership knowledge ($I$) for the first partition $X_1$ are calculated by KSRSIO-FCM using KSLFCM algorithm. The output obtained after clustering of first partition (subset) i.e., $V'$ is used as an input to the second partition $X_2$ for clustering. Again, the cluster centers and membership knowledge is calculated by KSRSIO-FCM using KSLFCM algorithm. The results of clustering performed on second partition (subset) are updated membership knowledge and cluster centers $I'$ and $V'$, respectively After that, KSRSIO-FCM fuse the membership knowledge of all the processed partitions (subsets). So, it aggregates the membership knowledge of all the processed partitions, i.e., it fuses $I'$ and $I$, and then the updated cluster centers are computed using Eq. (7).

The KSRSIO-FCM and KSLFCM algorithms take 12-dimensional numeric feature vectors extracted after preprocessing of the huge SNP sequences as input and then both the algorithms cluster huge SNP sequences at high speed with high accuracy. The preprocessing approach for huge SNP sequences is explained in section 4.3. Figure 3 summarizes the KSRSIO-FCM diagrammatically. It shows how KSRSIO-FCM takes 12-dimensional numeric feature vectors as input and then huge SNP sequences are partitioned randomly across various subsets.

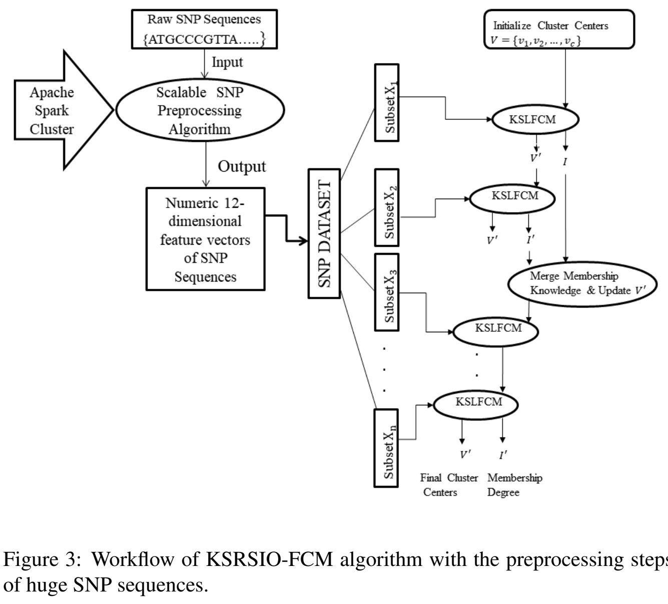

Raw SNP Sequences {ATGCCCGTTA.....}
Input
Apache Spark Cluster
Scalable SNP Preprocessing Algorithm
Output
Numeric 12-dimensional feature vectors of SNP Sequences
SNP DATASET
Subset $X_1$
Subset $X_2$
Subset $X_3$
Subset $X_n$
Initialize Cluster Centers $V = \{v_1, v_2, ..., v_c\}$
KSLFCM
$V'$
$I$
$I'$
Merge Membership Knowledge & Update $V'$
Final Cluster Centers
Membership Degree

Figure 3: Workflow of KSRSIO-FCM algorithm with the preprocessing steps of huge SNP sequences.

## 4.3. Scalable SNP preprocessing approach

The proposed scalable SNP preprocessing approach is being implemented using Apache Spark framework to represent each SNP sequence in terms of 12-dimensional numeric feature vector. The proposed scalable SNP preprocessing approach extracts features of SNP sequences in three sets of numerical parameter: the first parameter counts the number of nucleotides $A, T, G$, and $C$ in the sequence, the second parameter is the total distances of each nucleotide base to the first nucleotide, and the third parameter is the variance of distance for each nucleic base (Liu et al., 2006). Each set of a numerical parameter is not sufficient to denote a specific SNP sequence. So, the combination of all of the three sets of a numerical parameter, which contains 12-dimensional numeric feature vectors is used to characterize similarities between SNP sequences. After this, extracted 12-dimensional numeric feature vectors are passed as an input to proposed KSRSIO-FCM and KSLFCM algorithms.

| Sequences | Positions | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Sequence1 | $G$ | $A$ | $A$ | $T$ | $G$ | $C$ | $T$ | $G$ | $G$ |
| Sequence2 | $T$ | $G$ | $C$ | $T$ | $G$ | $T$ | $T$ | $A$ | $A$ |
| Sequence3 | $T$ | $A$ | $C$ | $T$ | $G$ | $A$ | $T$ | $C$ | $G$ |
| Sequence4 | $G$ | $C$ | $G$ | $A$ | $T$ | $A$ | $T$ | $G$ | $T$ |
| Sequence5 | $T$ | $T$ | $A$ | $C$ | $G$ | $G$ | $A$ | $T$ | $G$ |

Table 2: Example of SNP sequences

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 5 | 0.3 | 4 | 23 | 9.7 | 2 | 11 | 2.3 | 1 | 6 | 0 |
| 2 | 2 | 17 | 0.3 | 2 | 7 | 2.3 | 4 | 18 | 1.6 | 1 | 3 | 0 |
| 3 | 2 | 8 | 4 | 2 | 14 | 4 | 3 | 12 | 6 | 2 | 11 | 6.3 |
| 4 | 2 | 10 | 7.3 | 3 | 12 | 6.5 | 3 | 21 | 6.7 | 1 | 2 | 0 |
| 5 | 2 | 10 | 4 | 3 | 20 | 2.9 | 3 | 11 | 9.6 | 1 | 4 | 0 |

Table 3: Preprocessing result of SNP sequences present in Table 2

### 4.3.1. Calculation of length of sequence

At the very first step of scalable SNP preprocessing approach, the length of sequence of each nucleotide is calculated. The total numbers of $A, T, G$, and $C$ named as the length of the sequence i.e., $l_A, l_T, l_G$, and $l_C$, respectively. Here an illustration is presented by considering an example of five sequences shown in Table 2, and the output obtained after preprocessing of these five sequences are shown in Table 3, where the first column is the sequence number, and the other columns represents 12-dimensional feature vectors $\langle l_A, T_A, D_A, l_G, T_G, D_G, l_T, T_T, D_T, l_C, T_C, D_C \rangle$. The total number of $A, T, G$, and $C$ i.e., $l_A, l_T, l_G$, and $l_C$ present in sequence1 $< G\ A\ A\ T\ G\ C\ T\ G\ G >$ are $2, 2, 4$, and 1, respectively. Likewise, the value of $l_A, l_T, l_G$, and $l_C$ for other four sequences are shown in Table 3.

### 4.3.2. Total distances of each nucleotide base to the first nucleotide

The second numerical parameter i.e., the total distances of each nucleotide base to the first nucleotide $T_i$ is calculated using Eq. (8). Add the position values corresponding to each nucleotide as shown in Table 2, nucleotide $G$ appears at 1, 5, 8, and 9. So, the value of $T_G$ is calculated as follows:

$$T_G = 1 + 5 + 8 + 9 = 23$$

Likewise, the value of $T_A, T_T$, and $T_C$ for sequence1 and the total distances of each nucleotide base to the first nucleotide for all other four sequences are shown in Table 3.

#### 4.3.3. Variance of distance for each nucleic base

The third numerical parameter is the variance of distance for each nucleic base $D_i$ is calculated using Eq. (9). The first step is to compute $\mu_i$ using $\mu_i = \frac{T_i}{l_i}$. The value of $\mu_G$ in sequence1 is calculated as follows:

$$\mu_G = \frac{T_G}{l_G} = \frac{23}{4} = 5.75$$

The second step computes the variance of distance for each nucleic base i.e., $D_i$, for an example the value of $D_G$ in sequence1 is 9.67, calculations of $D_G$ is as follows:

$$D_G = \frac{[(1-5.75)^2+(5-5.75)^2+(8-5.75)^2+(9-5.75)^2]}{4} = 9.67$$

Hence, the result obtained from sequence1, which contains 12-dimensional feature vectors is as follows:

$$\langle 2, 5, 0.25, 4, 23, 9.67, 2, 11, 2.25, 1, 6, 0 \rangle$$

The results of all other four sequences are shown Table 3.

The significant characteristics of the proposed scalable SNP preprocessing algorithm is that it takes raw SNP sequences as input and produce 12-dimensional numeric feature vectors as an output in much less time using the Apache Spark framework. The proposed KSRSIO-FCM and KSLFCM algorithms are applied on the preprocessed SNP datasets for clustering of huge SNP sequences. The experimental results applied to various SNP datasets are present in section 5. Before presenting the experimental results, the complexity analysis of KSRSIO-FCM and KSLFCM algorithms are discussed in detail in the following subsection.

### 4.4. Complexity analysis of KSRSIO-FCM and KSLFCM

In the past, kernel-based clustering algorithms with big data is used by many researchers, but the complexity of such an algorithm is quadratic (Havens et al., 2012a). In contrast to that, the complexity of both of our algorithms is linear in terms of the input data sample.

The complexity analysis of the kernelized algorithms in terms of variables is illustrated in Table 4. In this section, we use the following convention, $X$ depicts the dataset $X = \{x_1, ..., x_s\}$ with $s$ number of data samples in the high dimensional space $d$, $c$ is the number of clusters, $w$ is the number of worker node with $n$ number of partitions represented as $X = \{X_1, X_2, \ldots. X_n\}$ such that each subset consists of $s/n$ data samples. Here, $t$ is the number of iteration required for termination, but the value of $t$ may change, thus for clarity, $t$ is the maximum number of iteration for a single execution of both the algorithms. The complexity analysis is done as follows: Firstly, the expense of membership degree is calculated during the Map stage where each Map task calculates the membership degree of one data sample as to $c$ cluster centers. As we have $d$ dimensional data sample, the time and space complexity of each membership degree is $O(d)$ and $O(d)$, respectively, also each Map task takes $O(cd)$ time and $O(cd)$ space. The ReduceByKey task linearly adds the estimations of all the Map operations correspond to one cluster on each slave node and joins the subsequent values on the master node. We have assumed that each worker node $w$ has an equal distribution of job to work upon. The reduce function takes $O(cwd)$ time and $O(cd)$ space.

The KSLFCM runs Map and ReduceByKey task on the whole dataset. Every slave node works parallelly on the data samples by $(s/w)$. Thus, the map stage takes an aggregate of $O(scd/w)$ time and $O(scd)$ space for every iteration on all slave nodes. If we assume KSLFCM executes for $t$ iterations, then the total time taken for the Map stage is $O(scdt/w)$, and the total space complexity for the Map stage of KSLFCM comes out to be $O(scd)$ because the Map results have been held in memory just for the term of one iteration of KSLFCM. The output given by every slave node is $c$, and these outputs get accumulated on the master node and are added. This process takes $O(sd/w)$ time and $O(sd/w)$ space. Along with these lines, the total time complexity for the ReduceByKey stage is $O(scdt/w)$, and space complexity is $O(cd/w)$. Accordingly, the time complexity of KSLFCM is $O(scdt/w)$ and space complexity is $O(scd)$) where $s >> w, c$.

KSRSIO-FCM works differently by dividing whole dataset $X$ into $n$ equivalent subsets with the final goal as $X = \{X_1, X_2, \ldots. X_n\}$, where size of each subset is $(s/n)$. It basically executes KSLFCM on each of these subsets in a sequential manner. $O(scdt/nw)$ and $O(scd/n)$ are time and space complexity respectively for performing KSLFCM over every subset. As this algorithm handles every subset in a steady progression, and data corresponding to one subset is not held in the memory while executing the next subset, so the time complexity is $O(scdt/w)$ and the space complexity remains $O(scd/n)$.

As given in Table 4, the KSLFCM and KSRSIO-FCM share a similar time complexity. Apparently, it may appear that both the approaches are attaining a similar run-time, but this is not the case because the KSRSIO-FCM has partitioned the whole dataset into different subsets and then performed clustering over each subset. Due to this, clustering performed by KSRSIO-FCM on each subset is achieved by less number of iterations ($t$) for each subset. Hence, KSRSIO-FCM has lesser run-time as the clustering is performed on a small chunk of data in each subset in comparison with KSLFCM that performs clustering on the whole data.

Table 4: Complexity Analysis of Kernelized Algorithm.

| **Algorithm** | **Time Complexity** | **Space Complexity** |
|---|---|---|
| KSRSIO-FCM | $O(scdt/w)$ | $O(scd/n)$ |
| KSLFCM | $O(scdt/w)$ | $O(scd)$ |

## 5. Experimental results

In the experiments, we analyze the performance of KSRSIO-FCM in comparison with KSLFCM, SRSIO-FCM, and SLFCM using the Silhouette index and Davies-Bouldin index on an Apache Spark clusters.

### *5.1. Experimental environment*

The experimental evaluation is performed on Apache Spark clusters. The Apache spark cluster consists of one master and five worker nodes. The master node has 32 GB RAM, 8 cores, and 3TB storage. Each worker node has 16 GB RAM, 8 cores, and 1TB storage. The Hadoop Distributed File System (HDFS) is used for storage of data on the worker nodes. The server machine is used for preprocessing of SNP data with the following configuration: Total number of cores: 32, Total memory: 187 GB. Total disk: 12 TB.

### *5.2. Dataset description*

We analyze the exhibition of KSRSIO-FCM, KSLFCM, SRSIO-FCM, and SLFCM algorithm on following SNP datasets. The detailed characterization of the datasets utilized for the experimental analysis is presented in Table 5.

Table 5: Description of SNP Datasets.

| **Parameters** | **Datasets** | | |
|---|---|---|---|
| | SNP-seek rice | MAGIC-rice | 248Entries rice |
| **#sample** | 252 | 16932 | 248 |
| **#features** | 12 | 12 | 12 |
| **size** | 17.1 MB | 1.05 GB | 30.8 MB |

#### *5.2.1. SNP-seek rice*

The SNP-seek rice data contains rice chromosomes (ch1-12); we have merged all the rice chromosomes from ch1-12 into a single file to perform clustering on a huge SNP dataset. The SNP-seek rice data are discussed in detail as follows: (Mansueto et al., 2017; International, 2005). The subsequent size of the dataset is 17.1 MB.

#### *5.2.2. MAGIC-rice*

The MAGIC-rice dataset consists of SNP sequences; the MAGIC rice data for 1,411 Samples are divided into 12 files (for each chromosome). To perform clustering on a huge SNP dataset, we have merged all the chromosomes from ch1-12 and formed the MAGIC-rice dataset. Bandillo et al. (2013) discussed the detailed descriptions of the population.

#### *5.2.3. 248Entries rice*

The 248Entries rice data contains 248 data samples. (Dilla-Ermita et al., 2017) discussed the details of 248Entries rice data. The subsequent size of the dataset is 30.8 MB.

### *5.3. Parameter specification*

In the experimental analysis, the fuzzification parameter value $p = 1.75$ and stopping criteria value $\epsilon = 0.01$ is used for huge SNP datasets. However, these values achieve better results for most of the datasets as shown in the study performed by Schwämmle and Jensen (2010).

### *5.4. Performance evaluation*

#### *5.4.1. Silhouette index (SI)*

This measure is useful for the validation of consistency within clusters of genome data. SI (Bolshakova and Azuaje, 2003) is a measure of how similar a data sample to its own cluster compared with other clusters. Thus SI is characterized as:

$$S(i) = \frac{a_2(i) - a_1(i)}{max[a_1(i), a_2(i)]} \tag{10}$$

Where $a_1(i)$ is the average distance between $i^{th}$ sample from all other data samples within the same cluster, $a_2(i)$ is the lowest average distance of $i^{th}$ sample to all the data samples in any other cluster, of which $i$ is not a member. The Silhouette value is bounded in a range -1 to 1. A negative value indicates poor clustering, and the positive value indicates good clustering quality.

#### *5.4.2. Davies-Bouldin index (DBI)*

The DBI (Coelho et al., 2012) is used for evaluating the performance of clustering. It consolidates a single record in two measures, one identified with the scattering of individual clusters and the other to the partition between various clusters.

$$DBI = \frac{1}{c}\sum\nolimits_{i=1}^{c} \max\nolimits_{j \neq i} \left[ \frac{\text{diam}(C_i) + \text{diam}(C_j)}{\text{d}(C_i, C_j)} \right] \tag{11}$$

Where $d(C_i, C_j)$ correlates to the distance between the center of clusters $C_i$ and $C_j$, diam $(C_i)$ is the maximum distance between all the data samples of cluster $C_i$, and $c$ is the number of clusters. The DBI is not limited inside a given range and thus the lower DBI indicates good clustering quality.

### *5.5. Results and discussion*

In this section, we discuss the effectiveness of KSRSIO-FCM in comparison with KSLFCM evaluated on SNP data in terms of SI and DBI.

#### *5.5.1. Illustrative example*

To show the effectiveness of this algorithm, we present the diagrammatic representation which shows how the algorithm creates clusters out of the SNP data of soybean 31 sequences. These soybean 31 sequences of SNP data contains 6,289,747 SNPs (Lee et al., 2014). A soybean dataset consisting of 31 sequences are used, which includes samples of two categories, i.e., wild and cultivated (Lam et al., 2010). The comparison between the four algorithms is depicted in Figure 4 and Figure 5, which clearly shows how the results of KSRSIO-FCM are better than those of KSLFCM, SRSIO-FCM, and SLFCM, respectively. From Figure 4, we can infer that the clusters formed by KSRSIO-FCM, KSLFCM, SRSIO-FCM, and SLFCM are well separated into five respective groups (i.e., different colors represent different clusters). But, the clusters formed by KSLFCM, SRSIO-FCM, and SLFCM creates three overlapped clusters consists of wild and cultivated data samples. The number of data samples of wild and cultivated categories are perfectly cluster by KSLFCM, SRSIO-FCM, and SLFCM are 9,

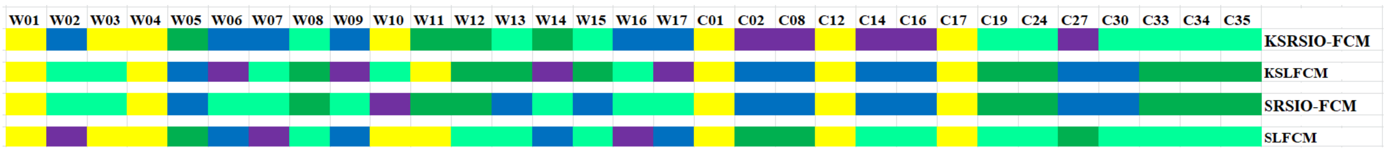


Figure 4: Cluster formation of soybean 31 sequences for KSRSIO-FCM, KSLFCM, SRSIO-FCM, and SLFCM with the number of clusters = 5

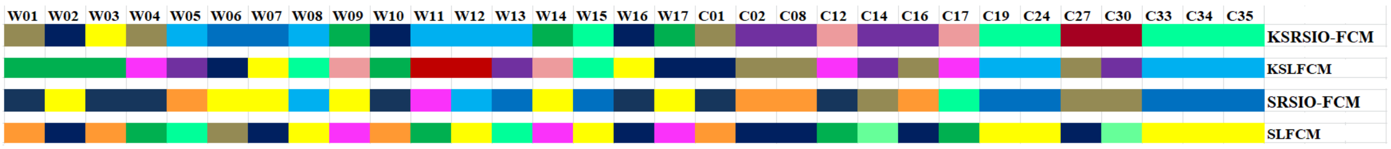
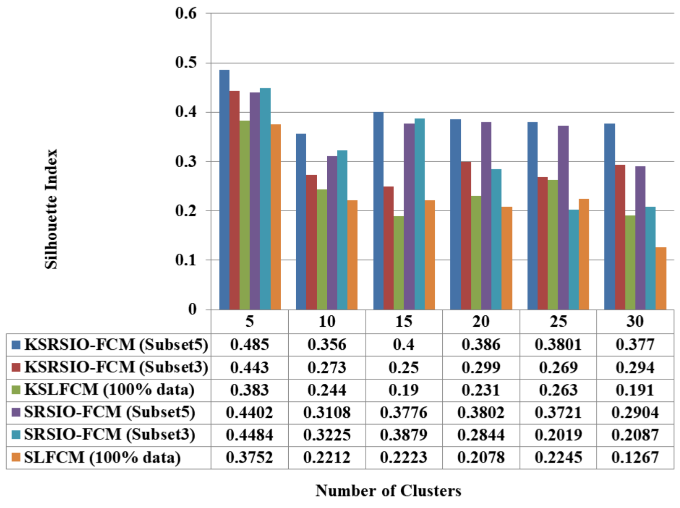


Figure 5: Cluster formation of soybean 31 sequences for KSRSIO-FCM, KSLFCM, SRSIO-FCM, and SLFCM with the number of clusters = 10

9, and 7, respectively. Whereas, KSRSIO-FCM only creates two overlapped clusters and 15 data Samples of wild and cultivated categories are perfectly clustered. Likewise, from Figure 5, we can infer that the KSRSIO-FCM formed total ten clusters. Out of ten clusters, two clusters are overlapping and 8 clusters are perfectly formed consist of 22 data samples of wild and cultivated categories. Whereas, KSLFCM generates three overlapped clusters out of total 10 clusters and SRSIO-FCM forms total 8 cluster with 3 overlapped clusters, and SLFCM forms total 7 clusters from which 5 clusters are overlapped consist of wild and cultivated data samples. Hence, we can conclude that the KSRSIO-FCM outperforms KSLFCM, SRSIO-FCM, and SLFCM, respectively. The effectiveness of our proposed algorithm can be proved by testing it on huge SNP data. For that purpose, we have tested our proposed algorithms on a huge SNP dataset in the subsequent subsection.

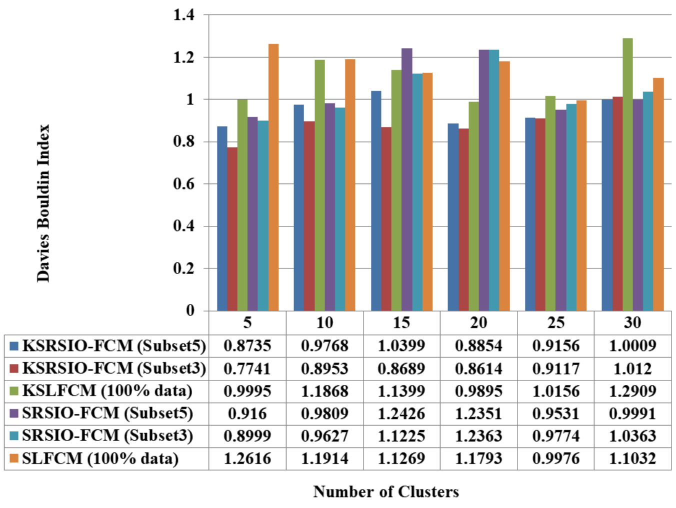


| | 5 | 10 | 15 | 20 | 25 | 30 |
|---|---|---|---|---|---|---|
| 🟦 KSRSIO-FCM (Subset5) | 0.485 | 0.356 | 0.4 | 0.386 | 0.3801 | 0.377 |
| 🟥 KSRSIO-FCM (Subset3) | 0.443 | 0.273 | 0.25 | 0.299 | 0.269 | 0.294 |
| 🟩 KSLFCM (100% data) | 0.383 | 0.244 | 0.19 | 0.231 | 0.263 | 0.191 |
| 🟪 SRSIO-FCM (Subset5) | 0.4402 | 0.3108 | 0.3776 | 0.3802 | 0.3721 | 0.2904 |
| 🟦 SRSIO-FCM (Subset3) | 0.4484 | 0.3225 | 0.3879 | 0.2844 | 0.2019 | 0.2087 |
| 🟧 SLFCM (100% data) | 0.3752 | 0.2212 | 0.2223 | 0.2078 | 0.2245 | 0.1267 |

Figure 6: Silhouette index of SNP-seek rice dataset

### 5.5.2. *Clustering performance on huge SNP datasets*

The section presents the discussion of the effectiveness of KSRSIO-FCM in comparison with KSLFCM, SRSIO-FCM, and SLFCM evaluated on three SNP datasets as per the estimates, such as SI and DBI, respectively. The estimation



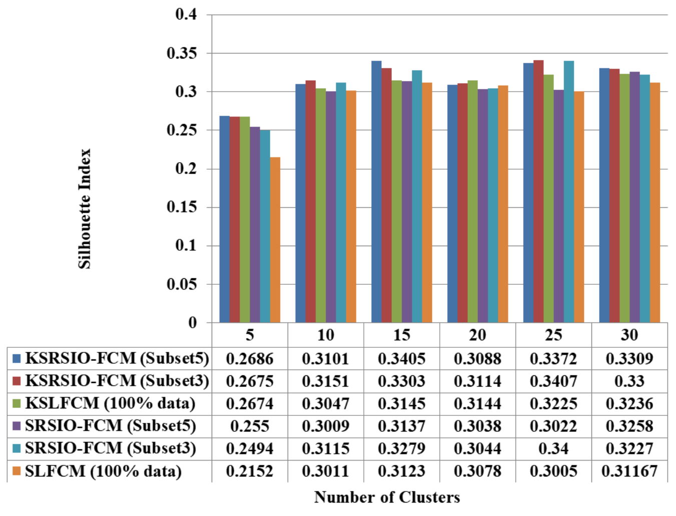
| | 5 | 10 | 15 | 20 | 25 | 30 |
|---|---|---|---|---|---|---|
| 🟦 KSRSIO-FCM (Subset5) | 0.8735 | 0.9768 | 1.0399 | 0.8854 | 0.9156 | 1.0009 |
| 🟥 KSRSIO-FCM (Subset3) | 0.7741 | 0.8953 | 0.8689 | 0.8614 | 0.9117 | 1.012 |
| 🟩 KSLFCM (100% data) | 0.9995 | 1.1868 | 1.1399 | 0.9895 | 1.0156 | 1.2909 |
| 🟪 SRSIO-FCM (Subset5) | 0.916 | 0.9809 | 1.2426 | 1.2351 | 0.9531 | 0.9991 |
| 🟦 SRSIO-FCM (Subset3) | 0.8999 | 0.9627 | 1.1225 | 1.2363 | 0.9774 | 1.0363 |
| 🟧 SLFCM (100% data) | 1.2616 | 1.1914 | 1.1269 | 1.1793 | 0.9976 | 1.1032 |

Figure 7: Davies Bouldin index of SNP-seek rice dataset

of SI and DBI for KSRSIO-FCM, on various subsets of SNP datasets in comparison with KSLFCM, SRSIO-FCM, and SLFCM are shown in figures.

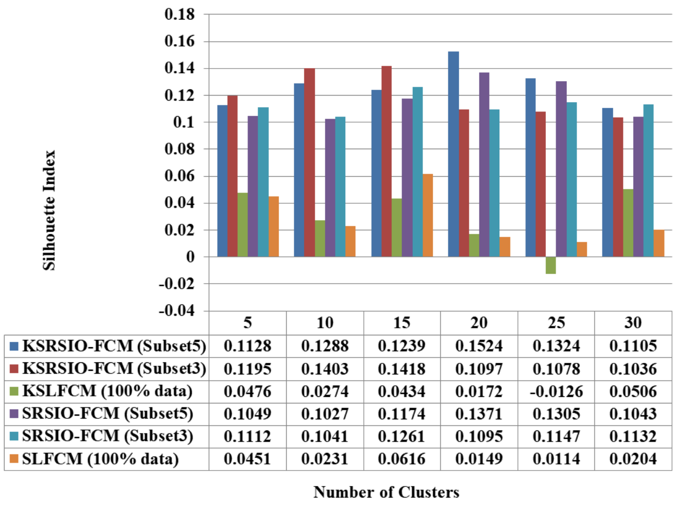
We perform clustering with the number of subsets 5 and 3, where the subset means entire data is data divided into chunks. The clustering is performed on the number of clusters 5, 10, 15, 20, 25, and 30, respectively. In this section, the subset5 and subset3 depict the number of subsets equal to 5 and 3, respectively. Likewise, the cluster5, cluster10, cluster15, cluster20, cluster25, and cluster30 depicts the number of clusters equal to 5, 10, 15, 20, 25, and 30, respectively. The subset5 and subset3 partitions the data using KSRSIO-FCM and SRSIO-FCM algorithms, the KSLFCM and SLFCM (100% data) performs the clustering on whole data. We have compared the performance of kernelized scalable algorithms, i.e., KSRSIO-FCM with SRSIO-FCM, KSLFCM, and SLFCM, thereafter, KSLFCM with SLFCM.

*Clustering performances on the SNP-seek rice dataset.* Figure 6 highlights the results of the SNP-seek rice dataset in terms of SI. SI demonstrates the quality of clustering. Subsequently, a superior clustering would bring about higher SI. Observing the



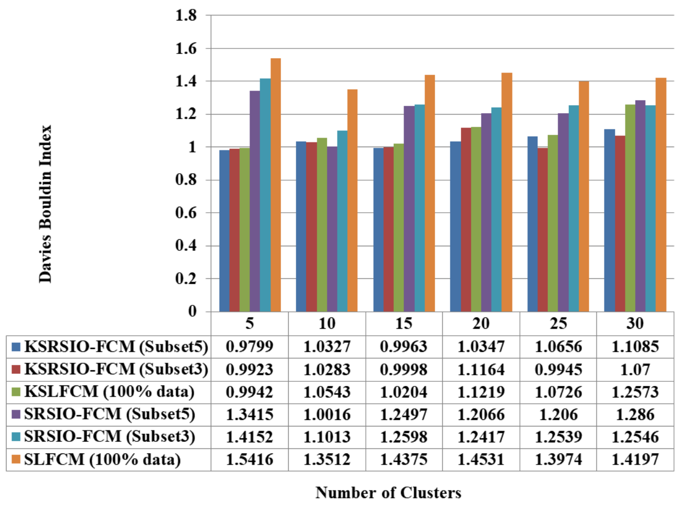
| | 5 | 10 | 15 | 20 | 25 | 30 |
|---|---|---|---|---|---|---|
| 🟦 KSRSIO-FCM (Subset5) | 0.2686 | 0.3101 | 0.3405 | 0.3088 | 0.3372 | 0.3309 |
| 🟥 KSRSIO-FCM (Subset3) | 0.2675 | 0.3151 | 0.3303 | 0.3114 | 0.3407 | 0.33 |
| 🟩 KSLFCM (100% data) | 0.2674 | 0.3047 | 0.3145 | 0.3144 | 0.3225 | 0.3236 |
| 🟪 SRSIO-FCM (Subset5) | 0.255 | 0.3009 | 0.3137 | 0.3038 | 0.3022 | 0.3258 |
| 🟦 SRSIO-FCM (Subset3) | 0.2494 | 0.3115 | 0.3279 | 0.3044 | 0.34 | 0.3227 |
| 🟧 SLFCM (100% data) | 0.2152 | 0.3011 | 0.3123 | 0.3078 | 0.3005 | 0.31167 |

Figure 8: Silhouette index of MAGIC-rice dataset



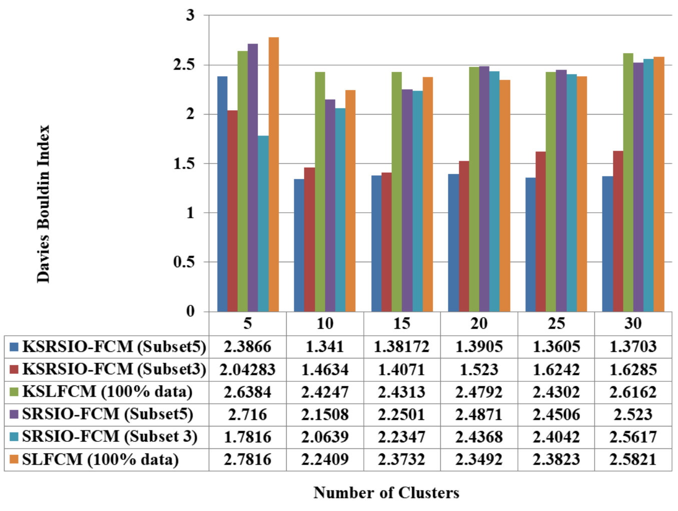
| | 5 | 10 | 15 | 20 | 25 | 30 |
|---|---|---|---|---|---|---|
| 🟦 KSRSIO-FCM (Subset5) | 0.1128 | 0.1288 | 0.1239 | 0.1524 | 0.1324 | 0.1105 |
| 🟥 KSRSIO-FCM (Subset3) | 0.1195 | 0.1403 | 0.1418 | 0.1097 | 0.1078 | 0.1036 |
| 🟩 KSLFCM (100% data) | 0.0476 | 0.0274 | 0.0434 | 0.0172 | -0.0126 | 0.0506 |
| 🟪 SRSIO-FCM (Subset5) | 0.1049 | 0.1027 | 0.1174 | 0.1371 | 0.1305 | 0.1043 |
| 🟦 SRSIO-FCM (Subset3) | 0.1112 | 0.1041 | 0.1261 | 0.1095 | 0.1147 | 0.1132 |
| 🟧 SLFCM (100% data) | 0.0451 | 0.0231 | 0.0616 | 0.0149 | 0.0114 | 0.0204 |

Figure 10: Silhouette index of 248Entries rice dataset



| | 5 | 10 | 15 | 20 | 25 | 30 |
|---|---|---|---|---|---|---|
| 🟦 KSRSIO-FCM (Subset5) | 0.9799 | 1.0327 | 0.9963 | 1.0347 | 1.0656 | 1.1085 |
| 🟥 KSRSIO-FCM (Subset3) | 0.9923 | 1.0283 | 0.9998 | 1.1164 | 0.9945 | 1.07 |
| 🟩 KSLFCM (100% data) | 0.9942 | 1.0543 | 1.0204 | 1.1219 | 1.0726 | 1.2573 |
| 🟪 SRSIO-FCM (Subset5) | 1.3415 | 1.0016 | 1.2497 | 1.2066 | 1.206 | 1.286 |
| 🟦 SRSIO-FCM (Subset3) | 1.4152 | 1.1013 | 1.2598 | 1.2417 | 1.2539 | 1.2546 |
| 🟧 SLFCM (100% data) | 1.5416 | 1.3512 | 1.4375 | 1.4531 | 1.3974 | 1.4197 |

Figure 9: Davies Bouldin index of MAGIC-rice dataset



| | 5 | 10 | 15 | 20 | 25 | 30 |
|---|---|---|---|---|---|---|
| 🟦 KSRSIO-FCM (Subset5) | 2.3866 | 1.341 | 1.38172 | 1.3905 | 1.3605 | 1.3703 |
| 🟥 KSRSIO-FCM (Subset3) | 2.04283 | 1.4634 | 1.4071 | 1.523 | 1.6242 | 1.6285 |
| 🟩 KSLFCM (100% data) | 2.6384 | 2.4247 | 2.4313 | 2.4792 | 2.4302 | 2.6162 |
| 🟪 SRSIO-FCM (Subset5) | 2.716 | 2.1508 | 2.2501 | 2.4871 | 2.4506 | 2.523 |
| 🟦 SRSIO-FCM (Subset 3) | 1.7816 | 2.0639 | 2.2347 | 2.4368 | 2.4042 | 2.5617 |
| 🟧 SLFCM (100% data) | 2.7816 | 2.2409 | 2.3732 | 2.3492 | 2.3823 | 2.5821 |

Figure 11: Davies Bouldin index of 248Entries rice dataset

values of SI, SRSIO-FCM has obtained a lower value, whereas the KSRSIO-FCM achieved a higher value for the subset5. Also, the KSRSIO-FCM achieved the highest value of SI for cluster 5 of subset5. Also, we analyzed that SI obtained by KSRSIO-FCM for subset3 is higher for cluster 20, 25, and 30 in comparison with SRSIO-FCM. The figure shows that the estimation of SI for KSLFCM is higher for all the clusters except cluster 15 in comparison with SLFCM for SNP-seek rice dataset. While the SI value is essentially lower for KSLFCM and SLFCM when compared with KSRSIO-FCM in most of the clusters. Along these lines, we can conclude that KSRSIO-FCM performs better than KSLFCM, SRSIO-FCM, and SLFCM in terms of SI values for the SNP-seek rice dataset.

Conversely to the SI, the DBI is not bounded within a given range. As a general rule, the lower the DBI value is, the better the clustering result will be. Figure 7, we have reported the results on the SNP-seek rice dataset in terms of DBI. The value achieved by KSRSIO-FCM is much better than SRSIO-FCM on almost all the clusters for subset5. Moreover, KSRSIO-FCM attained a very low value on all the clusters for subset3. Additionally, KSRSIO-FCM achieves the remarkable value of DBI for cluster 5 of subset3. On comparing KSLFCM with SLFCM, the DBI values for most of the clusters are lower for KSLFCM. Therefore, comparing DBI, we conclude that KSRSIO-FCM performs much better than KSLFCM, SRSIO-FCM, and SLFCM in terms of SI for subset5 and DBI for subset3. Overall, we can say that KSRSIO-FCM for subset3 or subset5 for each cluster performs better than KSLFCM. As depicted in the figure, the estimation of DBI for KSRSIO-FCM is significantly better than SRSIO-FCM and SLFCM. In this way, we can conclude that KSRSIO-FCM performs better than KSLFCM, SRSIO-FCM, and SLFCM in terms of DBI values for the SNP-seek rice dataset.

*Clustering performances on the MAGIC-rice dataset.* Figure 8 shows the results of the MAGIC-rice dataset in terms of SI. On

Table 6: Run-time analysis (in seconds) of KSRSIO-FCM and KSLFCM algorithms.

| **#clusters** | **Datasets** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | SNP-seek rice | | | MAGIC-rice | | | 248Entries rice | | |
| | **KSRSIO-FCM** | | **KSLFCM** | **KSRSIO-FCM** | | **KSLFCM** | **KSRSIO-FCM** | | **KSLFCM** |
| | Subset5 | Subset3 | 100% data | Subset5 | Subset3 | 100% data | Subset5 | Subset3 | 100% data |
| **5** | 80 | 76 | 96 | 120 | 120 | 300 | 74 | 56 | 76 |
| **10** | 72 | 70 | 120 | 900 | 720 | 960 | 68 | 76 | 80 |
| **15** | 220 | 190 | 310 | 960 | 1680 | 1980 | 207 | 160 | 260 |
| **20** | 240 | 200 | 340 | 1268 | 1140 | 1560 | 196 | 114 | 182 |
| **25** | 260 | 275 | 365 | 2460 | 2400 | 3615 | 154 | 178 | 192 |
| **30** | 300 | 310 | 390 | 2340 | 6300 | 6600 | 120 | 277 | 290 |

comparing SI, SRSIO-FCM has attained the lowest SI value on all the clusters compared to the SI values achieved on KSRSIO-FCM for subset5 and subset3. Besides this, the SI of KSRSIO-FCM achieves a higher value for cluster 15 of subset5. Furthermore, the figure shows that the estimation of SI for KSLFCM and SLFCM is close to various clusters of the MAGIC-rice dataset. While the SI value is essentially lower for KSLFCM and SLFCM when compared with KSRSIO-FCM. Along these lines, we can conclude that KSRSIO-FCM performs better than KSLFCM, SRSIO-FCM, and SLFCM in terms of SI values for the MAGIC-rice dataset.

Similarly, for DBI, as shown in Figure 9, SRSIO-FCM has obtained a higher value compared to KSRSIO-FCM on subset5 and subset3 for the various cluster. Additionally, KSRSIO-FCM achieves the remarkable value of DBI for cluster 5 of subset5. Therefore, we conclude that KSRSIO-FCM performs much better than SRSIO-FCM in comparison with SRSIO-FCM. Furthermore, the figure shows that the estimation of DBI for KSLFCM is significantly better than SLFCM. In this way, we can conclude that KSRSIO-FCM performs much better than KSLFCM, SRSIO-FCM, and SLFCM in terms of DBI values for the MAGIC-rice dataset.

*Clustering performances on the 248Entries rice dataset.* Figure 10 shows the results of the 248Entries rice dataset in terms of SI. On comparing SI, SRSIO-FCM has attained the lowest SI values on most of the clusters corresponded to the SI values achieved by KSRSIO-FCM for subset5 and subset3. Besides this, KSRSIO-FCM attains the higher value of SI for cluster 20 of subset5. The SI value of KSLFCM is better than SLFCM except for cluster 15. The estimation of SI for various subsets of KSRSIO-FCM is better than SLFCM. Furthermore, the figure shows that the SI values obtained by SRSIO-FCM and SLFCM are close for the 248Entries dataset. Additionally, the estimation of SI for different subsets of KSRSIO-FCM is better than KSLFCM. While speaking about the difference in SI values among various clusters, SI for KSLFCM (100% data) is comparatively smaller than SI values achieved with KSRSIO-FCM on subset3 and subset5 with varying number of clusters. Moreover, we observed that KSRSIO-FCM has attained positive SI for each cluster, whereas KSLFCM obtained negative SI values for cluster 25. Along these lines, we can finish up that KSRSIO-FCM performs better than KSLFCM, SRSIO-FCM, and SLFCM in terms of SI values for the 248Entries rice dataset.

Similarly, for DBI, as shown in Figure 11, SRSIO has obtained a higher value compared to KSRSIO-FCM on each cluster except cluster5 of subset3. While speaking about the difference in DBI values among various clusters of subset5 and subset3, DBI for KSLFCM (100% data) is comparatively higher than DBI values achieved with KSRSIO-FCM. Furthermore, KSRSIO-FCM achieves the remarkable value of DBI for subset5 and subset3. Moreover, KSLFCM has attained lower DBI values than SLFCM. Though the DBI is essentially lower for SLFCM for some of the clusters when compared with KSLFCM, the estimation of DBI for different subsets of KSRSIO-FCM is better than SLFCM. In this way, we can finish up that KSRSIO-FCM performs better than KSLFCM, SRSIO-FCM, and SLFCM in terms of DBI values for the 248Entries dataset.

Table 6 tabulates the run-time analysis of KSRSIO-FCM and KSLFCM algorithms. In any case, the run-time analysis of KSRSIO-FCM and KSLFCM would likewise rely upon the total number of nodes and their configuration. Here, the total number of nodes is 6 and cores are 52. According to the run-time analysis given in table 6, the KSRSIO-FCM takes less time in comparison with KSLFCM for computation.

## 6. Conclusion

In this paper, two approaches KSRSIO-FCM and KSLFCM have been proposed to cluster huge SNP sequences using Apache Spark cluster. Before using KSRSIO-FCM and KSLFCM, we preprocessed the huge SNP sequences using the proposed scalable SNP preprocessing algorithm which extracts 12-dimensional numeric feature vectors from huge SNP sequences using Apache Spark cluster. The preprocessed SNP sequences are further used as an input to KSRSIO-FCM and KSLFCM algorithms.

Our proposed KSRSIO-FCM and KSLFCM approaches are used to cluster SNP sequences efficiently at high speed and with high accuracy. The significant characteristics of the KSRSIO-FCM algorithm is that it takes preprocessed SNP sequences as input and produce output in much less time. The subsequent aspect is that it does not store the huge membership matrix. Subsequently, in this way, the performance of KSRSIO-FCM algorithm can be fundamentally improved by reducing space and run-time complexity. We directed the exact evaluation of KSRSIO-FCM on the different SNP datasets, which exhibited likely advantages for utilizing our methodology for clustering SNP sequences. In the future, the scalable kernelized fuzzy clustering approach can be applied for handling massive genome sequences for clustering.

## Acknowledgements

This research is financed by the Council of Scientific and Industrial Research (CSIR), Govt. of India (grant no. 22(0750)/17/EMR-II).

## Compliance with ethical standards

### *Conflict of interest*

The authors acknowledge that they have no conflict of interests.

### *Ethical approval*

This article does not contain any examinations with human members or animals performed by any of the authors.

### *Informed consent*

Informed consent was gotten from all individual members remembered for the investigation.

## References

Bandillo, N., Raghavan, C., Muyco, P.A., Sevilla, M.A.L., Lobina, I.T., Dilla-Ermita, C.J., Tung, C.W., McCouch, S., Thomson, M., Mauleon, R., et al., 2013. Multi-parent advanced generation inter-cross (magic) populations in rice: progress and potential for genetics research and breeding. Rice 6, 11.

Bezdek, J.C., Ehrlich, R., Full, W., 1984. Fcm: The fuzzy c-means clustering algorithm. Computers & Geosciences 10, 191–203.

Bharill, N., Tiwari, A., 2014. Handling big data with fuzzy based classification approach, in: Advance Trends in Soft Computing. Springer, pp. 219–227.

Bharill, N., Tiwari, A., Malviya, A., 2016. Fuzzy based scalable clustering algorithms for handling big data using apache spark. IEEE Transactions on Big Data 2, 339–352.

Bolón-Canedo, V., Sánchez-Maroño, N., Alonso-Betanzos, A., 2015. Recent advances and emerging challenges of feature selection in the context of big data. Knowledge-Based Systems 86, 33–45.

Bolshakova, N., Azuaje, F., 2003. Cluster validation techniques for genome expression data. Signal processing 83, 825–833.

Borthakur, D., et al., 2008. Hdfs architecture guide. Hadoop Apache Project 53, 2.

Cai, W., Chen, S., Zhang, D., 2007. Robust fuzzy relational classifier incorporating the soft class labels. Pattern Recognition Letters 28, 2250–2263.

Castellanos-GarzóN, J.A., GarcíA, C.A., Novais, P., DíAz, F., 2013. A visual analytics framework for cluster analysis of dna microarray data. Expert Systems with Applications 40, 758–774.

Chen, L., Kong, L., 2016. Fuzzy clustering in high-dimensional approximated feature space, in: 2016 International Conference on Fuzzy Theory and Its Applications (iFuzzy), IEEE. pp. 1–6.

Coelho, G.P., Barbante, C.C., Boccato, L., Attux, R.R., Oliveira, J.R., Von Zuben, F.J., 2012. Automatic feature selection for bci: an analysis using the davies-bouldin index and extreme learning machines, in: The 2012 international joint conference on neural networks (IJCNN), IEEE. pp. 1–8.

Di Nuovo, A.G., Catania, V., 2008. An evolutionary fuzzy c-means approach for clustering of bio-informatics databases, in: 2008 IEEE International Conference on Fuzzy Systems (IEEE World Congress on Computational Intelligence), IEEE. pp. 2077–2082.

Dilla-Ermita, C.J., Tandayu, E., Juanillas, V.M., Detras, J., Lozada, D.N., Dwiyanti, M.S., Cruz, C.V., Mbanjo, E.G.N., Ardales, E., Diaz, M.G., et al., 2017. Genome-wide association analysis tracks bacterial leaf blight resistance loci in rice diverse germplasm. Rice 10, 1–17.

Havens, T.C., Bezdek, J.C., Leckie, C., Hall, L.O., Palaniswami, M., 2012a. Fuzzy c-means algorithms for very large data. IEEE Transactions on Fuzzy Systems 20, 1130–1146.

Havens, T.C., Bezdek, J.C., Palaniswami, M., 2012b. Incremental kernel fuzzy c-means, in: Computational Intelligence. Springer, pp. 3–18.

Hosseini, B., Kiani, K., 2018. A robust distributed big data clustering-based on adaptive density partitioning using apache spark. Symmetry 10, 342.

Hosseini, B., Kiani, K., 2019. A big data driven distributed density based hesitant fuzzy clustering using apache spark with application to gene expression microarray. Engineering Applications of Artificial Intelligence 79, 100–113.

Huang, H.C., Chuang, Y.Y., Chen, C.S., 2011. Multiple kernel fuzzy clustering. IEEE Transactions on Fuzzy Systems 20, 120–134.

International, R.G.S.P., 2005. The map-based sequence of the rice genome. Nature 436, 793.

Jha, P., Tiwari, A., Bharill, N., Ratnaparkhe, M., Mounika, M., Nagendra, N., 2020. A novel scalable kernelized fuzzy clustering algorithms based on in-memory computation for handling big data. IEEE Transactions on Emerging Topics in Computational Intelligence .

Jiang, D., Tang, C., Zhang, A., 2004. Cluster analysis for gene expression data: a survey. IEEE Transactions on Knowledge & Data Engineering , 1370–1386.

Kerr, G., Ruskin, H.J., Crane, M., Doolan, P., 2008. Techniques for clustering gene expression data. Computers in biology and medicine 38, 283–293.

Kolen, J.F., Hutcheson, T., 2002. Reducing the time complexity of the fuzzy c-means algorithm. IEEE Transactions on Fuzzy Systems 10, 263–267.

Lam, H.M., Xu, X., Liu, X., Chen, W., Yang, G., Wong, F.L., Li, M.W., He, W., Qin, N., Wang, B., et al., 2010. Resequencing of 31 wild and cultivated soybean genomes identifies patterns of genetic diversity and selection. Nature genetics 42, 1053.

Lee, T.H., Guo, H., Wang, X., Kim, C., Paterson, A.H., 2014. Snphylo: a pipeline to construct a phylogenetic tree from huge snp data. BMC genomics 15, 162.

Li, T., Zhang, L., Lu, W., Hou, H., Liu, X., Pedrycz, W., Zhong, C., 2017. Interval kernel fuzzy c-means clustering of incomplete data. Neurocomputing 237, 316–331.

Liu, J., Xu, M., 2008. Kernelized fuzzy attribute c-means clustering algorithm. Fuzzy sets and systems 159, 2428–2445.

Liu, L., Ho, Y.k., Yau, S., 2006. Clustering dna sequences by feature vectors. Molecular phylogenetics and evolution 41, 64–69.

Mansueto, L., Fuentes, R.R., Borja, F.N., Detras, J., Abriol-Santos, J.M., Chebotarov, D., Sanciangco, M., Palis, K., Copetti, D., Poliakov, A., et al., 2017.

Rice snp-seek database update: new snps, indels, and queries. Nucleic acids research 45, D1075–D1081.

Moorthy, K., Saberi Mohamad, M., Deris, S., 2014. A review on missing value imputation algorithms for microarray gene expression data. Current Bioinformatics 9, 18–22.

Nasraoui, O., N'Cir, C.E.B., 2018. Clustering Methods for Big Data Analytics: Techniques, Toolboxes and Applications. Springer.

Oussous, A., Benjelloun, F.Z., Lahcen, A.A., Belfkih, S., 2018. Big data technologies: A survey. Journal of King Saud University-Computer and Information Sciences 30, 431–448.

Popescu, M., Bezdek, J.C., Keller, J.M., 2009. eccv: A new fuzzy cluster validity measure for large relational bioinformatics datasets, in: 2009 IEEE International Conference on Fuzzy Systems, IEEE. pp. 1003–1008.

Saxena, A., Prasad, M., Gupta, A., Bharill, N., Patel, O.P., Tiwari, A., Er, M.J., Ding, W., Lin, C.T., 2017. A review of clustering techniques and developments. Neurocomputing 267, 664–681.

Schwämmle, V., Jensen, O.N., 2010. A simple and fast method to determine the parameters for fuzzy c–means cluster analysis. Bioinformatics 26, 2841–2848.

Tang, S., He, B., Yu, C., Li, Y., Li, K., 2018. A survey on spark ecosystem for big data processing. arXiv preprint arXiv:1811.08834 .

Tsai, D.M., Lin, C.C., 2011. Fuzzy c-means based clustering for linearly and nonlinearly separable data. Pattern recognition 44, 1750–1760.

Veiga, J., Expósito, R.R., Pardo, X.C., Taboada, G.L., Tourifio, J., 2016. Performance evaluation of big data frameworks for large-scale data analytics, in: 2016 IEEE International Conference on Big Data (Big Data), IEEE. pp. 424–431.

Wong, K.C., 2016. Computational biology and bioinformatics: Gene regulation. CRC Press.

Wu, Z.d., Xie, W.x., Yu, J.p., 2003. Fuzzy c-means clustering algorithm based on kernel method, in: Proceedings Fifth International Conference on Computational Intelligence and Multimedia Applications. ICCIMA 2003, IEEE. pp. 49–54.

Xu, R., Wunsch, D.C., 2005. Survey of clustering algorithms .

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M.J., Shenker, S., Stoica, I., 2012a. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing, in: Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, USENIX Association. pp. 2–2.

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauly, M., Franklin, M.J., Shenker, S., Stoica, I., 2012b. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing, in: Presented as part of the 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12), pp. 15–28.

Zhao, G., Ling, C., Sun, D., 2015. Sparksw: scalable distributed computing system for large-scale biological sequence alignment, in: 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, IEEE. pp. 845–852.

Zhao, Y.P., Chen, L., Chen, C.P., 2018. Multiple kernel shadowed clustering in approximated feature space, in: International Conference on Data Mining and Big Data, Springer. pp. 265–275.

Zheng, X., Levine, D., Shen, J., Gogarten, S.M., Laurie, C., Weir, B.S., 2012. A high-performance computing toolset for relatedness and principal component analysis of snp data. Bioinformatics 28, 3326–3328.