International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Incremental Fuzzy Clustering Algorithm For Large Datasets

Ani Davis K. [1], Dr. Raj Mathew [2]

Submitted:14/03/2024 **Revised**: 29/04/2024 **Accepted**: 06/05/2024

Abstract: Clustering streaming data poses unique challenges that are different from traditional batch processing. Conventional clustering methods struggle with the high volume, speed, and diversity of data due to limitations in memory, computing power, and processing time. The challenges of clustering large datasets include limited storage capacity, the requirement to process the data in a single pass, and the concept drift of the data. A novel method, incremental fuzzy double clustering (IFDC) has been proposed to tackle these challenges. IFDC is an innovative version of Fuzzy c-Means (FCM) and incremental clustering. It divides the data into groups based on memory capacity and clusters them. Relevant samples from each group are selected using stratified sampling and k-Medoid methods and then transferred to the next group. This process carries the essence of the data from the beginning to the end. The newly reached dataset can be easily merged with the last block of data and clustered, instead of clustering the entire dataset as it arrives. The performance of IFDC was evaluated using Silhouette, Davies-Bouldin, and Calinski Harabasz Indexes, and the results demonstrate that IFDC outperforms traditional techniques such as FCM and k-means by successfully overcoming the challenges of clustering large data. The benefits of IFDC include improved efficiency and reduced clustering time. It efficiently manages large streaming datasets by continuously accommodating new data and utilizing sampling methods, thereby enhancing accuracy, reducing execution time, and eliminating the need for complete re-clustering.

Keywords*: Incremental Fuzzy c-Means Double clustering, Fuzzy c-Means, Silhouette, Davies-Bouldin, Calinski Harabasz*

1. INTRODUCTION

The encroachment of technology leads to the generation of a massive amount of data daily. This data contains essential information. The large data sometimes refers streaming data, is a continuous flow of generated and processed in real-time or near real-time. Unlike traditional batch processing, where data is collected and processed in discrete chunks or batches, large data is processed as it is produced, allowing for immediate analysis and response. The large or streaming data is typically generated from various sources such as IoT devices, sensors, social media platforms, financial transactions, and weblogs. These sources consistently produce a steady stream of data, which can be substantial in terms of volume, velocity, and variety. In today's real-world applications, data streams are widely utilized. Data clustering is an effective method for analyzing and extracting valuable information from large datasets. However, clustering huge datasets presents significant challenges in data analysis and machine learning. This is because it involves partitioning a massive amount of data into meaningful groups or clusters based on their similarities. This process aims to uncover inherent structures, patterns, or relationships within the dataset, even when dealing with large volumes of data. Clustering essentially involves dividing a set of data objects or populations into homogeneous groups. Traditional clustering techniques may struggle to handle these challenges due to limitations in memory, processing power, and time constraints.

With the rise of Big Data, which has led to an exponential growth in dataset size, clustering algorithms need to be scalable, efficient, and capable of handling the computational complexity associated with such vast amounts of data. Cluster analysis of streaming data poses unique challenges due to data streams' dynamic and continuous nature. Unlike batch data, huge data cannot be stored entirely in memory, requiring the clustering process to be performed in a single pass. Conducting an additional scan is often unfeasible. Consequently, algorithms must process the data using limited memory, which can influence the quality of clustering results.

The vast data often exhibits concept drift, meaning that the underlying data distribution may change over time. Additionally, large data can contain outliers and noisy data points that can significantly impact the accuracy of the clustering process. Identifying and handling these anomalies in real-time presents a considerable challenge. Clustering algorithms must be able to adapt to these changes to maintain accurate clusters. Moreover, selecting initial cluster centers for data stream clustering can be challenging. Traditional methods that rely on all data points for initialization are not suitable for streaming data.

Several clustering methods, such as FCM clustering, have been developed to handle large amounts of data. FCM is a highly effective data technique that allows for the

[1*] *Vimala College, Thrissur, Kerala 680009, India*
E-mail: anidavisk@gmail.com
[2] *St Thomas College, Palai, Kerala 686574, India*
E-mail:mathewrajm@gmail.com

classification of objects into multiple clusters by assessing their similarities. Unlike traditional clustering algorithms, FCM offers a soft assignment of data points to clusters, allowing for partial membership. This results in a more nuanced representation of complex data patterns. However, traditional static clustering algorithms are not appropriate for dynamic datasets, especially when data is entered in batches. This is because FCM, the clustering algorithm, requires the entire data to be in memory at once, which becomes unreasonably time-consuming for datasets that are too large for the main memory. This slows down the clustering process and reduces its performance. Additionally, the results of FCM can be influenced by the initial placement of cluster centers, leading to different cluster structures with different initializations. FCM is computationally demanding, especially for large datasets or when there are a high number of clusters. Ensuring the robustness of FCM in the presence of data variations or concept drift is a challenge, especially for real-time applications. Moreover, outliers can have a significant impact on FCM results, so it is important to identify and handle them appropriately.

The majority of existing fuzzy clustering methods are primarily intended for small, static datasets. However, with the continuous expansion of streaming data sources, the size of these datasets can become enormous at different intervals. Consequently, it becomes challenging to store the entirety of the data in memory all at once. An effective streaming algorithm should possess the capability to adapt to changes in the data, while still extracting valuable information from the complete dataset. Since it is assumed that loading all the data into memory is not feasible, non-incremental algorithms for expediting FCM or hard c-means are typically not applicable to clustering very large datasets.

An innovative approach known as Incremental Fuzzy c-Means Double clustering (IFDC), which is based on incremental style FCM, is proposed to address the challenges of clustering a data stream. Incremental clustering is a data analysis technique that allows for the clustering of data points as they are acquired, rather than clustering all of them at once. This approach eliminates the need to load the entire dataset into memory simultaneously. The utilization of memory is optimized through the use of incremental clustering. The characteristics of each chunk are passed on to the next chunk by incorporating selected elements. IFDC can easily incorporate new data sets into the existing clusters, rather than repeatedly clustering the entire dataset as new data arrives. This approach improves performance and reduces the workload. The IFDC method is efficient in terms of memory storage and effectively handles large data.

2. RELATED WORKS

Fuzzy logic has been successfully applied in various fields, including control systems, decision-making, pattern recognition, image processing, natural language processing, and expert systems[1].

Fuzziness was incorporated into the ISODATA algorithm to improve the detection of compact, well-separated clusters. This was initiated by Dunn who introduced a fuzzy version of the ISODATA process[2]. J.C. Bezdek [3] has made significant contributions to the field of fuzzy clustering. He focuses on developing and applying fuzzy clustering algorithms that use fuzzy objective functions to partition data into meaningful groups. Bezdek[4] transmitted FCM into the FORTRAN-IV code. A S Bozkire and E A Sezer[5] designed the Fuzzy Clustering Analysis Tool (FUAT) to perform clustering algorithms that incorporate the concept of fuzziness. FUAT also analyses, investigates, and visualises the clusters produced using the FCM method.

Fuzzy clustering has applications in many fields, including control systems, decision-making, data mining, and artificial intelligence. Agbonifo and O Catherine[6] determine students' learning inclinations by applying FCM clustering in the Honey and Mumford learning fashion. H. Izakian and A Abraham[7] propose a crossover between Fuzzy Clustering (FCM) and Fuzzy Particle Swarm Optimization (FPSO), aiming to enhance clustering by leveraging swarm algorithms' exploration and exploitation capabilities. Telmo M. Silva Filho[8] introduced two particle swarm optimization methods, FCM-IDPSO and FCM2-IDPSO. They are the result of combining FCM with a recent version of PSO and enhanced self-adaptive particle swarm optimization (IDPSO). Mahmoudi et al [9] proposed a fuzzy clustering approach to analyse and compare the spread rate of COVID-19 and the population size in high-risk countries using Pearson correlation. Tao Lei et al. [10] improved the FCM algorithm based on morphological reconstruction and membership filtering called Fast and Robust FCM (FRFCM). Thomas Bonis and Steve Oudot [11] presented an advanced FCM for the mode-seeking framework. Mode-finding algorithm determines a density function's modes or local maxima, and FCM clusters the dataset.

Zhang Siqing et al. [12] introduced a clustering protocol based on fuzzy logic for Multi-hop wireless sensor networks (FLCMN). The FLCMN extends the life of wireless sensor networks (WSNs) and lessens energy consumption. O. M. Saad [13] enhanced the Earthquake Early Warning System (EEWS) with the help of FPCM

algorithm. It recognises the arrival time of the earthquake and sounds an alarm as a warning. Paweł Karczmarek et al. [14] improved the Isolation Forest algorithm by incorporating FCM, which calculates anomaly scores by calculating membership grades of elements and forming Isolation Forest nodes into clusters. Mahmoud Salah [15] utilized FCM clustering to partition point cloud data into clusters based on spatial characteristics, a technique widely used in point cloud processing and filtering tasks. Jian-Ping Mei [16] introduces Hyperspherical FCM (HFCM) and Fuzzy Co-clustering. These novel approaches offer a fresh perspective on managing high-dimensional data by enhancing scalability, which addresses the challenges associated with clustering sizable document datasets.

Incremental clustering is an efficient technique for handling large volumes of data. It dynamically updates clustering models as new data becomes available, without reprocessing the entire dataset. This makes it well-suited for streaming or large-scale data, allowing the model to adapt to changes in data distribution over time. Incremental clustering is a scalable and memory-optimized approach that can process data in smaller batches, making it an effective solution for working with enormous volumes of data. Runhai Jiao et al. [17] proposed two methods to improve incremental kernel fuzzy clustering effectiveness: optimizing the initial cluster center based on distance and incremental clustering characteristics, and using multiple passing points. Arnaud Ribert et.al [18] presented a new algorithm that uses hierarchical clustering to handle time incremental data. It modifies the hierarchical representation of data rather than recomputing the entire tree when new patterns need to be taken into consideration. F.Can [19] developed the Cover-Coefficient-based Incremental Clustering Methodology (C^2ICM), an economical and versatile algorithm for updating and removing old documents from a large number of documents. Fazli Can et al [20] conducted experiments on the MARIAN database, implementing the C^2ICM method for incremental clustering, addressing C^3M's shortcomings and saving time and money compared to C^3M.

W Zhao et al [21] introduced an incremental anomaly detection technique based on the Gaussian Mixture Model (GMM) to identify typical patterns and exceptional cases in digital flight data. The model updates clusters using new data, retaining model parameters and addressing challenges in flight operations. Prodip Hore et al. [22] have developed a Single Pass FCM(SPFCM) algorithm for large data sets, offering efficient, comparable data partitions and improved clustering speed compared to traditional algorithms. P. Hore et al [23] introduced an online fuzzy clustering algorithm capable of clustering both streaming data and significantly large datasets, addressing the challenge of the unavailability of the entire dataset and the difficulty in determining partitions. Jian-Ping Mei et al [24] developed a methodology incorporating incremental fuzzy clustering techniques to improve document categorization accuracy and efficiency in web and text mining tasks.

Mahmoud Al-Ayyoub's study [25] introduces a GPU-powered breast cancer detection system using SPFCM clustering algorithm, enhancing accuracy and efficiency through advanced clustering techniques. Yangyang Li et al. [26] introduced a new approach to the SPFCM algorithm, incorporating density peaks to improve accuracy and efficiency. The algorithm reorders samples and assigns weights based on density peaks, but requires significant time for computation. Mitchell D. Woodbright [27] initiated the Unsupervised Incremental Clustering Algorithm (UIClust), an incremental clustering technique that detects concept drift, improving the accuracy and adaptability of clustering processes. Sirisup Laohakiat and Vera Saing[28] introduced Fuzzy Incremental Density-based Clustering (FIDC). This incremental density-based method enhances accuracy and adaptability by combining incremental and fuzzy local clustering techniques for large datasets. L. Wang, P. Xu, and Q. Ma [29] presented a method for incrementally clustering time series data using fuzzy clustering. This approach involves two stages: offline and online. In offline, a fuzzy clustering validity evaluation index determines the optimal number of clusters. Online, the algorithm updates existing clusters dynamically. Preeti Jha [30] designed the Scalable Incremental Fuzzy Consensus Clustering (SIFCC) algorithm for big data frameworks, enhancing scalability by handling large-scale data efficiently through an incremental approach using Apache Spark cluster framework.

Existing fuzzy clustering methods are designed for small, static datasets. However, with the expansion of streaming data sources, the dataset size can become enormous, making it challenging to store the entire data in memory. Non-incremental techniques are not frequently applicable to clustering streaming data sets since they assume that all of the data can be put into memory. Although SPFCM and OFCM algorithms are inadequate when addressing streaming data. The present study investigates the challenges of the abovementioned algorithms and proposes to address the shortcomings by introducing a novel algorithm called the Incremental Fuzzy c-Means Double Clustering (IFDC) method. IFDC allows for the clustering of data points as they are acquired, eliminating the need to load the entire dataset into memory simultaneously. IFDC optimizes memory usage, adapts to

changes in data, and seamlessly integrates new data sets into existing clusters. This enhances performance and efficiently handles large data sets. It effectively manages streaming datasets by consistently accommodating new data and utilizing sampling techniques. As a result, accuracy is improved, execution time is minimized, and the need for complete re-clustering is eliminated.

3. BACKGROUND ALGORITHMS

The proposed system is a novel approach called IFDC, designed to address the issues of clustering streaming or large data. The outline of the main background algorithms used in the proposed system is as follows:

3.1 FUZZY C-MEANS CLUSTERING (FCM)

FCM, a variant of the c-means clustering method, is the most widely utilized fuzzy-based clustering algorithm[4]. A dataset may be divided into clusters using the FCM clustering technique, in which each data point has a particular degree of membership in each cluster. In contrast to more traditional methods like k-means or hierarchical clustering, the fuzzy clustering algorithm provides a more nuanced and adaptable method of data grouping. The objective function of FCM is minimized as

$$J(U,V) = \sum_{j=1}^{n} \sum_{i=1}^{c} u_{ij}^{m} ||x_j - v_i||^2$$

Where $V=\{v_1, v_2, v_3, \ldots..v_c\}$ is c cluster centers, $U=\{u_{ij}\}_{n\times c}$ is the membership matrix, and $X=\{x_1, x_2, x_3, \cdots x_n\}$ is the data set with n *data points*. The u_{ij} is the membership of x_j in class i, $\|x_j - v_i\|^2$ Euclidean distance between the data point x_j and the cluster center v_i. FCM is attempting to find the best U and V values to minimize the objective function. Algorithm 1 shows the FCM algorithm[31]. The input parameters are the dataset $X=x_j$, $j=1,2,..n$, c number of clusters, m the degree of fuzzification i.e. $m > 1$, ε the target value.

Algorithm 1: FCM

1. State the number of the cluster *k.*
2. Randomly initialize the cluster center *V*
3. Compute the membership value u_{ij} using the cluster centers
4. Update the cluster center *V*', using the new membership values
5. Check the difference between the old and new cluster centers ie. *V'-V*
6. Repeat steps 3, 4 and 5 until $V'-V \leq$ target value or a maximum number of iterations is reached.

The number of clusters is selected and randomly initialized. In step 3, the membership value u_{ij} of each data point is evaluated using the cluster centers, where u_{ij} represents the degree to which data point *i* belongs to cluster *j*. Then in step 4, update the cluster centres based on the newly calculated membership value. The sets of current and previous cluster centres are compared and evaluate the differences. FCM algorithm is stopped when the difference value *(V'-V)* reaches the target value or a maximum number of iterations is reached. The convergence speed of the FCM is influenced by the initial value and may fall into local optimization in the case of a huge number of clusters. The steps to compute the degree of membership and the new centres of each cluster are repeated until the algorithm terminates.

3.2 INCREMENTAL CLUSTERING

The time series data sets are quite large and it is not possible to accommodate in the main memory. One way to handle this is to move the time series to the main memory and store the whole data matrix in a secondary memory. This approach stores just the cluster structure in the main memory to get around the space constraint and updates it piecemeal. Incremental clustering allows for the processing of large datasets by dividing them into smaller subsets, which are clustered separately[32]. This method avoids the need to load the entire dataset into memory at once, making it feasible to analyze massive data incrementally. Rather than analyzing the full dataset at once, the clustering algorithm works on a subset of the data at a time. A separate clustering technique, such as K-means or FCM, is applied to each data segment. Following that, the cluster centers are merged or amalgamated and clustered once again. The most recent cluster center is used to group all of the data points. Iterative repetition of the technique is necessary if the dataset is huge. The clustering procedure should be restarted whenever new data is introduced to the dataset. Although these techniques are challenging to use, they somewhat address the issue of grouping enormous datasets.

4. PROPOSED SYSTEM

Clustering large datasets and data that were too large to fit in the main memory took longer with the FCM approach. Additionally, it is challenging to mine useful information from the whole data collection while accepting flowing input and responding to data changes. For FCM clustering, all of the data must be in the main memory at all times. OFCM clustering partially solves these issues. In the event of a huge dataset, further computations are required, and the procedure must be repeated. The issue of limited memory storage can be resolved by the IFDC system, which can manage flowing data. Additionally, it minimizes the number of calculations and algorithm iterations.

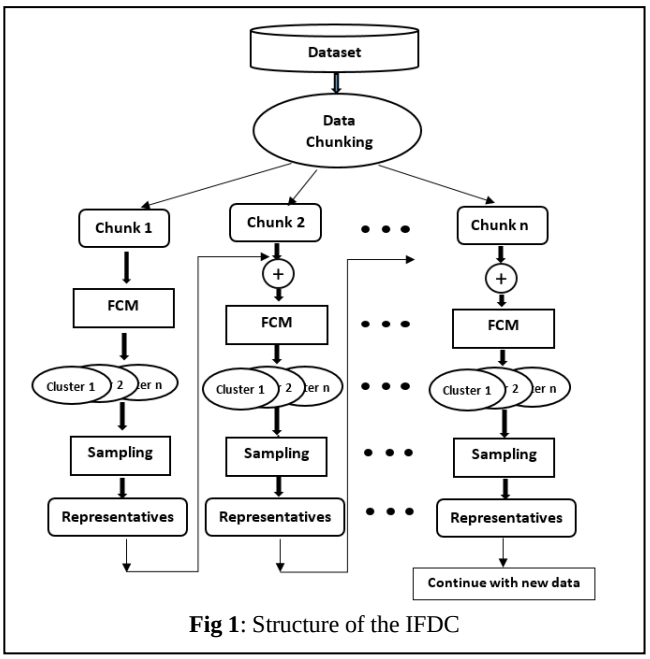

Fig 1: Structure of the IFDC

Algorithm 2 Incremental FCM Double Clustering

Input: X, c, m, ε

Output: U, V'

Load $X = X_1, X_2, X_3 ... X_s$, where $X_i = x_{(i-1)n+1}$ to x_{in}

1: $X_{new} = \Phi$
2: $reps = \Phi$
3: **for** $i = 1$ *to* s **do**
4: Load X_i
5: **if** $reps \neq None$ **then**
6: $X_i = X_i \cup reps$
7: **end if**
8: $ClusterLabels = FCM(X_i, c, m, \varepsilon)$
9: $reps = GetReps(X_i, r_s, ClusterLabels)$
10: **end for**
11: return C

function: GetReps(X, r_s, ClusterLabels)

1: $reps = \phi$
2: $cluster = \phi$
3: **for** $i = 1$ *to* $length(X)$ **do**
4: **if** $ClusterLabels$ *not in* $cluster$ **then**
5: $cluster_{label} = \phi$
6: **end if**
7: $cluster_{label} = cluster_{label} \cup x_i$
8: **end for**
9: **for** $j = 1$ *to* c **do**
10: $reps = Sampling(r_s, cluster_j)$
11: **end for**
12: return reps

function: Sampling (r_s, cluster$_i$)

1: $k = max(1, r_s * length(cluster_i))$
2: $clustercenters = KMedoids(cluster = k)$
3: return $clustercenters$

4.1 Incremental FCM Double Clustering Algorithm

The IFDC algorithm's structure is shown in Fig. 1. The entire dataset is split up into smaller groups, called chunks, chunks 1, chunk 2,... chunk n. The first chunk is divided into *n* number of clusters via the FCM method. Representatives from each cluster will be selected using

the sampling procedure, and they will be combined into chunk 2. This procedure keeps on till chunk n. Since the information of one subset spreads to the next, it is simple to cluster datasets based on this knowledge when a fresh stream of data is added. The large data is simply handled by the IFDC. To manage the subsets, the IFDC clustering only needs a small amount of memory space at a time. As a result, more RAM will be used, and massive data can be clustered.

The IFDC system is designed to handle large datasets efficiently and incrementally. The algorithm for this system is outlined in Algorithm 2. The IFDC algorithm takes the input parameters: X the dataset, c the number of clusters, m the fuzzification factor, n the size of the data subset. The large dataset of N items is divided into s smaller subsets, X_1, X_2, ..., X_s, where each subset X_i contains data points from $x_{(i-1)n+1}$ *to* x_{in}. ie. X_1 contains n_1 items. The final subset's dimensions could vary somewhat. This divide-and-conquer approach allows the system to handle very large datasets effectively.

The first subset X_1 is loaded into memory and clustered using the FCM algorithm, which returns the *ClusterLabels*. The *ClusterLabels*, sampling rate r_s, and

the current subset X_i are used as input to the *GetReps*() function. The *GetReps*() function considers each cluster and extracts representative data points as *reps* using the *Sampling()* function. The *Sampling()* function determines the number of representatives *k* based on the sampling rate r_s and the size of the cluster. The k-medoids algorithm is used to find the cluster centers, which are returned *reps*. The representative data points from the current subset are added to the next subset, which helps to increase the speed of clustering and allows for faster convergence.

The k-medoids algorithm used in the IFDC system reduces the cost and improves the efficiency of the clustering process. It ensures an equal contribution of data points from every cluster, and each member of the group has an equal chance of being chosen as a representative using the sampling method. By iterating through the subsets and adding representative data points from one subset to the next, the IFDC system achieves faster convergence and increased clustering speed. This knowledge transmission between subsets is a key feature of the algorithm. The algorithm iterates through each subset.

The first subset involves fetching and storing n_1 items in memory out of a total of *N* items. The FCM technique is used to cluster the n_1 objects into *c* partitions. r_1 items as reps were chosen as representatives from these *c* clusters using the k-medoid algorithm and a sampling technique. The first chunk of data is clustered using FCM, the memory is cleaned, and the subsequent n_2 items are put into memory in the subsequent subset, adding to the r_1 items that were chosen in the preceding stage. There will be $n_2 + r_1$ items in the memory for clustering.

The FCM technique is used to cluster the data, and r_2 items are chosen for the following step. Similarly, the r_2 items from the second level are added to the n_3 things that are retrieved in the third subset. $n_3 + r_2$ will be the total number of elements for clustering at the third level. Similarly, following the m^{th} subset, the memory for clustering will include just $n_m + r_{m-1}$ items. As a result, just a portion of the data is loaded into the memory depending on memory availability, as opposed to loading *N* items. Every subset's significance increases from the start to the finish, allowing for the same clustering of newly received data.

5. PERFORMANCE EVALUATION

The performance evaluation involves a systematic assessment of the algorithm's accuracy, speed, scalability, and robustness. It determines the effectiveness and efficiency of IFDC in solving a particular problem. By analyzing these metrics, it is easy to identify the strengths and weaknesses of the algorithm, compare it with other existing methods, and make informed decisions about its practical applicability. It also conducts a comprehensive performance evaluation of IFDC, utilizing a range of datasets and performance indicators to ensure a thorough and objective assessment.

5.1 DATESET & EVALUATION CRITERIA

The following two-dimensional datasets, IRIS, Diamond, and Codon Datasets, are utilized in this study. These datasets have been sourced from the UCI machine learning repository. The IRIS dataset encompasses 150 samples and 4 features, which provide measurements for the length and width of sepals and petals of three distinct species of the iris flower. The Diamond dataset comprises observations regarding the patterns and behaviours of various diamond types, encompassing both continuous and categorical features. It encompasses a total of 53,941 samples with 10 features. The Codon dataset contains 13,028 data rows, each consisting of 65 attributes. This dataset elucidates the codon usage frequencies observed in the genomic coding DNA of a diverse sample of organisms.

The performance evaluation of the IFDC system is based on the distances between clusters and the distances between data points within a cluster, as this algorithm operates as an unsupervised clustering method. To measure the reliability of clustering, three clustering metrics are utilized: the Silhouette Index (SI), Davies-Bouldin Index (DB), and Calinski-Harabasz Index (CH). The Silhouette Index is computed by considering each

sample's mean intra-cluster distance and the mean inter-cluster distance [33][29]. Let N represent the total number of data points in the dataset, N_c denote the number of clusters, C_i represents the i^{th} cluster, n_i indicates the number of objects in C_i, and V_i symbolises cluster C_i's centre. The Silhouette Index is defined as

$$SI = \frac{1}{Nc}\sum_{i}\left\{\frac{1}{n_i}\sum_{x\in C_i}\frac{b(x)-a(x)}{max[b(x),a(x)]}\right\}$$

Where a, represents the average distance between an object and all other objects in the same cluster, and b, represents the mean distance between an object and the nearest cluster that it does not belong to. It is crucial to highlight that the Silhouette coefficient can only be defined when the number of labels falls within the range of $2 <= n_{labels} <= n_{samples} - 1$. The Silhouette coefficient ranges from 1 to -1, with 1 indicating well-defined clusters, negative values indicating incorrect cluster assignments, and values close to zero suggesting overlapping clusters.

The Davies-Bouldin index (DB)[34][35] is used to assess the average ratio of within-cluster distances to between-cluster distances. The better cluster is indicated by values that are closer to zero. The mathematical formula for the Davies-Bouldin index is as follows:

$$DB = \frac{1}{Nc}\sum_{i} max_{j,j\neq i}\left\{\left[\frac{1}{n_i}\sum_{x\in C_i} d(x,v_i)+\frac{1}{n_j}\sum_{x\in C_j} d(x,v_j)\right]/d(v_i,v_j)\right\}$$

where the distance between x and v_i is denoted by $d(x, v_i)$. Values that are closer to 0 imply a better cluster.

The Calinski Harabasz index is a ratio that compares the within-cluster dispersion to the between-cluster dispersion [29]. This ratio is used to assess the quality of the clustering.

For CH, the equation is:

$$CH = \frac{\sum n_i d(v_i,v_j)/(N_c-1)}{\sum_i \sum_{x\in C_i} d(x,v_i)/(N-N_c)}$$

Where N is total number of points in a dataset, N_c is the number of clusters, $d(x,v_i)$ is the distance between x and v_i, n_i number of objects in C_i, C_i the i^{th} cluster, v_i is the centre of cluster C_i, $d(v_i, v_j)$ is the distance between the centers v_i and v_j .

5.2 RESULTS

The main objective of this study is to investigate the clustering ability of the IFDC algorithm. To assess the clustering optimization performance of IFDC, the algorithm was tested on three datasets: IRIS, Diamond, and Codon.The results of applying the IFDC algorithm to the three datasets are presented in the following tables and figures. In this experiment, the sample rates of 25%, 50%, and 75% are considered.

Table 1 contains results from an analysis of clustering algorithms on the IRIS dataset, evaluated with the CH, DB and SI indices across different sampling rates of 25%, 50%, 75% and numbers of clusters. The experiment evaluates clustering performance for clusters of 2, 3, 4, 5, and 6. The average values of the evaluation measurements are given in the table.

Lower values of the DB index at higher sampling rates indicate better clustering performance with more data. From table 1, it is inferred that better DB values occur for all the clusters except 2 at the sampling rate of 0.5. The higher sampling rates generally improve cluster quality and the increasing cluster numbers can reduce performance, as evidenced by lower DB values.

Table 1:DB, SI and CH values at sampling rates 25%,50% and 75% of IRIS

IRIS Dataset						
DB Value						
Clusters		**2**	**3**	**4**	**5**	**6**
Sampling rate	**25**	0.709246	0.84248	0.9868	1.134478	1.02757
	50	0.530241	**0.79597**	**0.90932**	**0.98965**	**0.94418**
	75	**0.526976**	0.80149	0.91786	1.010647	0.98154
SI Value						
Clusters		**2**	**3**	**4**	**5**	**6**
Sampling rate	**25**	0.503529	0.38814	0.32199	0.256179	0.29881
	50	0.601448	0.44612	0.38121	**0.371213**	**0.34435**

	75	**0.612414**	**0.46768**	**0.41128**	0.358394	0.32286
CH Value						
Clusters		**2**	**3**	**4**	**5**	**6**
Sampling rate	**25**	95.2483	99.25	83.3217	70.8782	80.6906
	50	191.1109	203.605	179.083	168.7035	160.854
	75	**295.084**	**296.54**	**264.995**	**260.2495**	**215.033**

The higher SI values represent better-defined clusters. The peak SI values were recorded at a sampling rate of 0.75 for clusters 2, 3, and 4, while for the remaining two clusters, the highest SI values were achieved at a sampling rate of 0.5. It shows an improvement in clustering quality with a higher sampling rate. The value decreases as the number of clusters increases. The higher sampling rates consistently yield higher SI values, indicating better clustering performance with more data.

The CH value generally decreases as clusters increase, suggesting diminishing returns for adding more clusters beyond a certain point. The table effectively shows how clustering performance, measured by the Calinski-Harabasz index, varies with different sampling rates and cluster counts. The higher sampling rate ie 0.75 leads to higher CH values, indicating better clustering performance. However, increasing the number of clusters tends to lower the CH value, implying a trade-off in cluster quality. The highest cluster homogeneity CH values are observed across all clusters at a sampling rate of 0.75.

Table 2 illustrates the performance of different clustering configurations of the dataset Diamond. The metrics are analyzed across various cluster numbers, including 5, 7, 9, 10, and 12. Different sampling rates of 25%, 50%, and 75% are also considered. The choice of cluster number is based on the number of options available in different features. For instance, a feature like 'cut' has 5 options, so cluster 5 is considered.

DB values show how different configurations perform in terms of cluster separation, with lower values being preferable. The lowest DB is detected for clusters 5,10 and 12 at the rate 0.5, for cluster 7 at the rate of 0.25 and for cluster 9 at the rate of 0.75. The lowest DB value occurred at the sampling rate of 0.5. The higher SI values indicate better clustering. The highest SI value occurs for clusters 5 and 10 at the rate of 0.75 and for clusters 7,9 and 12 at 0.25. The better SI value is detected at the rate of 0.25. CH values vary significantly across different sampling rates and cluster numbers. The better CH value is observed at different sampling rates for the different clusters. The highest CH values are observed for clusters 5 and 9 at a sampling rate of 0.25, for clusters 10 and 12 at a sampling rate of 0.75, and for cluster 7 at a sampling rate of 0.5. The highest CH value is detected at the sampling rate of 0.75. It explains that the better DB value for clusters 5, 10, and 12 at the sampling rate is 0.5, and for other clusters at the rate of 0.75. The good SI value is obtained at sample rates of 0.5 and 0.75. The biggest CH value occurred at different sampling rates for other clusters.

Table 2: DB, SI and CH values at sampling rate 25%,50% and 75% of Diamond

DIAMOND Dataset						
DB Value						
Clusters		**5**	**7**	**9**	**10**	**12**
Sampling rate	**25**	0.374925	**0.471**	0.45625	0.439072	0.44904
	50	**0.37273**	0.43814	0.4378	**0.37991**	**0.4109**
	75	0.406356	0.42786	**0.4183**	0.450351	0.43512
SI Value						
Clusters		**5**	**7**	**9**	**10**	**12**

Sampling rate	**25**	0.699423	**0.598**	**0.5786**	0.613647	**0.6095**
	50	0.715096	0.60799	0.63204	0.691537	0.62092
	75	**0.67506**	0.65422	0.63976	**0.6121**	0.61058
CH Value						
Clusters		**5**	**7**	**9**	**10**	**12**
Sampling rate	**25**	**336357**	132844	**456955**	211509.1	261580
	50	249950.6	203560	245157	**368012**	**416925**
	75	195373	**298110**	375260	322169.7	401276

This study effectively demonstrates the impact of sampling rates and cluster numbers on clustering performance across three metrics. By analyzing these metrics, it can determine the highest value of CH obtained at the rate of 0.75, better DB at the rate of 0.5 and SI at the rate of 0.25.

Table 3 provides the three indices DB, SI and CH values on the CODON dataset at different sampling rates. The results are analyzed across various numbers of clusters (4, 6, 8, 10) and sampling rates (25, 50, 75).

The DB index evaluates cluster separation and compactness. The lowest DB values are observed with a sampling rate of 25 suggesting the best compactness and separation under these conditions. The SI index assesses the quality of clusters based on cohesion and separation. Higher sampling rates improve SI values, indicating better cohesion and separation with more data. The sampling rate of 0.75 offers the best clusters. The CH index measures the separation between clusters. The highest CH value is obtained at the sampling rate of 0.25. The CH index suggests that more clusters are better defined in smaller samples. It illustrates the values of different criteria on the dataset Codon. The smallest DB and the highest CH value have occurred at the sampling rate of 0.25. The sampling rate of 0.75 gives a better value of SI. It can be inferred that better results occur at a higher sampling rate.

Table 3: DB, SI and CH values at sampling rates 25% ,50% and 75% of Codon

CODON Dataset					
DB Value					
Clusters		**4**	**6**	**8**	**10**
Sampling rate	**25**	**0.274633**	**0.2871426**	**0.2955118**	**0.359897**
	50	0.338189	0.3099534	0.3272583	0.402313
	75	0.339609	0.3903017	0.4080424	0.452782
SI Value					
Clusters		**4**	**6**	**8**	**10**
Sampling rate	**25**	0.967591	0.9384476	0.9207127	0.861704
	50	0.971292	0.9408224	0.9213075	0.879243
	75	**0.97209**	**0.9447104**	**0.9367976**	**0.905392**
CH Value					
Clusters		**4**	**6**	**8**	**10**
Sampling rate	**25**	**115710.6**	**280781.1**	**249304.63**	**658381.6**
	50	26790.16	101967.21	186975.80	192611.4
	75	24697.26	51682.155	65256.20	107526.9

The implementation of IFDC algorithm on three datasets IRIS, Diamond and Codon and the corresponding graphs are demonstrated. Figure 2, 3 and 4 illustrates the values of three evaluation criteria DB, SI, and CH of each incremental block of IRIS. The whole data is divided into blocks of size 50. i.e. 3 blocks, and added in the incremental order. The horizontal axis represents the number of blocks. It goes from 1 to 3. The DB, SI, and CH are shown on the vertical axis. Every block has been clustered and use the validity measurements to evaluate it. The IRIS dataset case study considers clusters 2, 3, 5, and 6 at the sampling rate of 0.75.

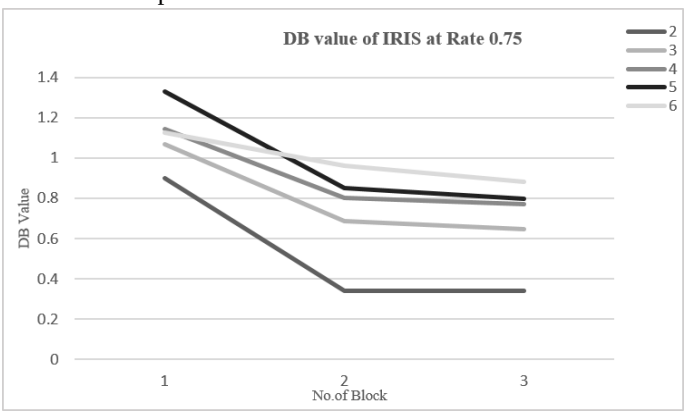

Fig 2: DB score of IRIS at Incremental blocks

Figure 2 indicates that the best DB values are associated with cluster 2. It suggests that cluster 2 is the most compact and well-defined structure among all the clusters in the dataset. SI values are explained in Figure 3. It is clear from the graph that cluster 2 has the best SI value. Figure 4 shows the CH value and it recommends the cluster 3. From the three validity measures its recommended cluster option is 2 or 3 for the IRIS dataset.

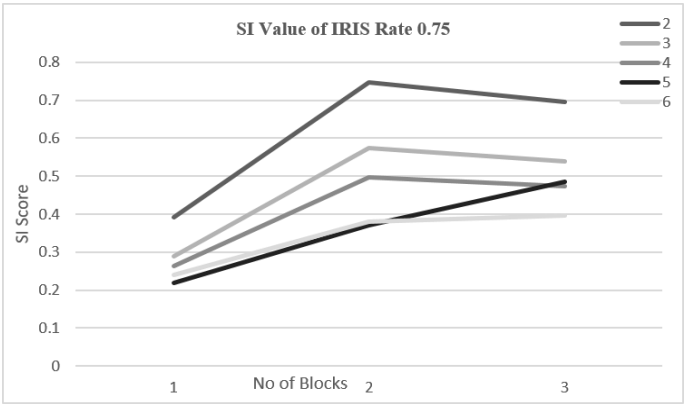

Fig 3: SI score of IRIS at Incremental blocks

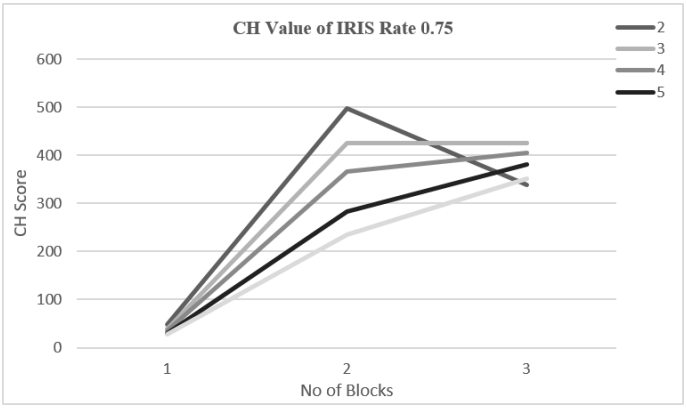

Fig 4: CH score of IRIS at Incremental blocks

Figures 5, 6 and 7 indicate the criteria values DB, SI, and CH of dataset Diamond for clusters 5,7,9,10,12. The dataset is divided into 6 blocks of size 10000. The size of the last block varies. From Table 2, it can be understood that the sampling rate of 0.5 is acceptable for DB and SI values. So the DB and SI values are calculated using the sampling rate of 0.5, and CH at 0.75. The x-axis represents the number of blocks, ranging from 1 to 6. The y-axis corresponds to the DB, SI, and CH values. Based on Figures 5, 6 and 7 it can be deduced that cluster 7 is the best.

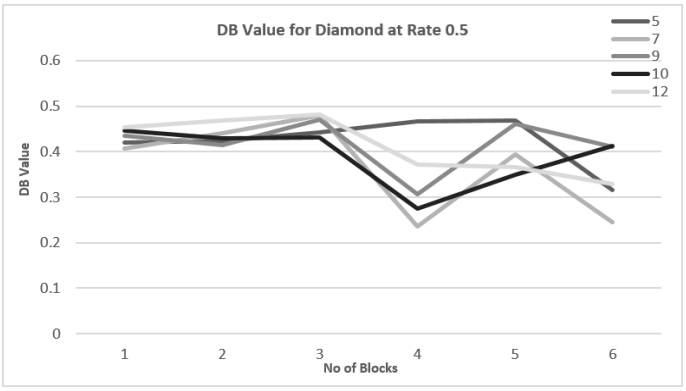

Fig 5: DB value of Diamond at Incremental blocks

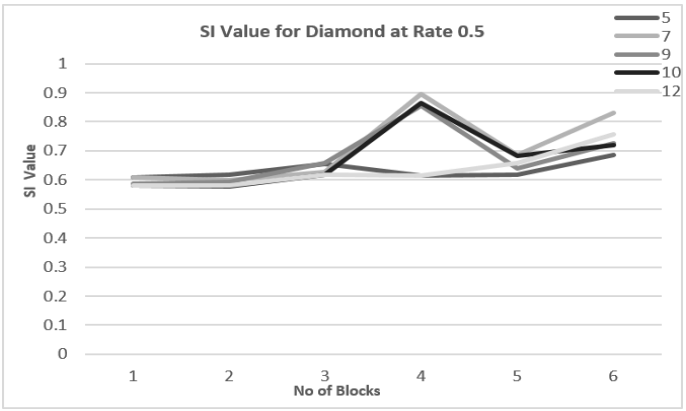

Fig 6: SI value of Diamond at Incremental blocks

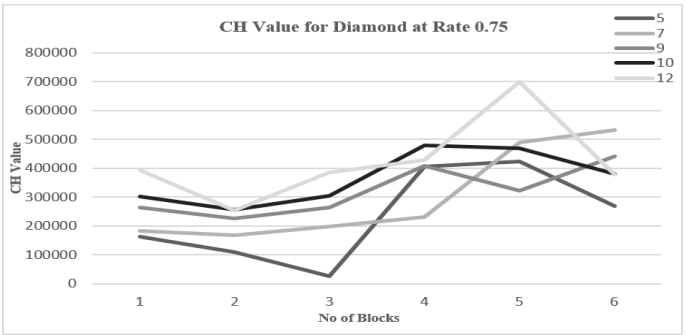

Fig 7:CH value of Diamond at Incremental blocks

Figures 8, 9 and 10 exhibit the values of dataset Codon for clusters 4, 6, 8, and 10. The whole dataset is divided into 14 blocks. There are 13 blocks with a size of 1000 and the last one with a different size. Based on Table 3, the preferable sampling rate for DB and CH is 0.25 and for SI it is 0.75. Figures 8, 9 and 10 visually represent the clustering performance for each cluster in the dataset Codon. The horizontal axis depicts the number of blocks from 1 to 14. The vertical axis represents the values for DB, SI and CH. According the graphs 8 and 9 it is understood that cluster 6 is the best one. The figure 10 depicts CH value and shows that cluster 8 is the best.

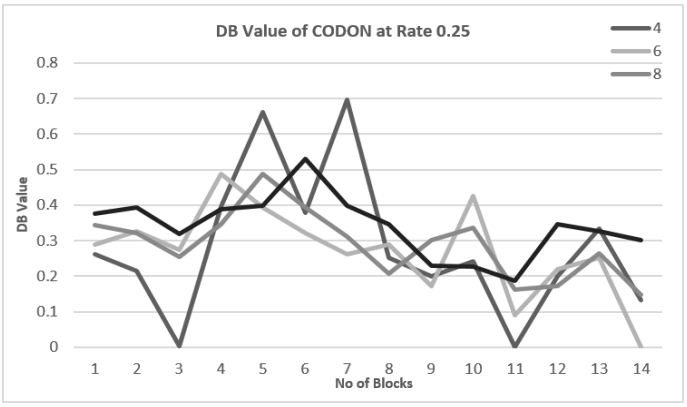

Fig 8:DB value of Codon at Incremental blocks

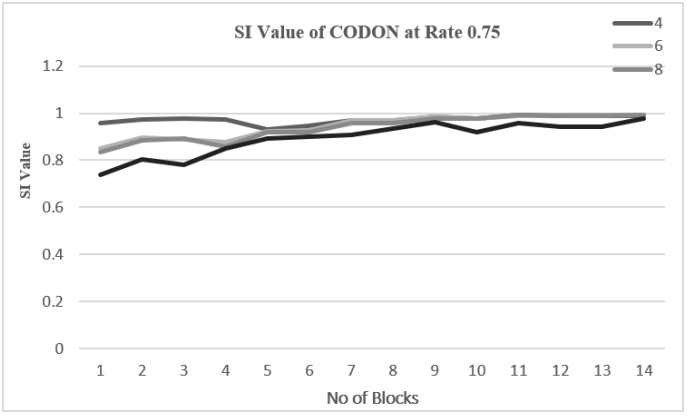

Fig 9:SI value of Codon at Incremental blocks

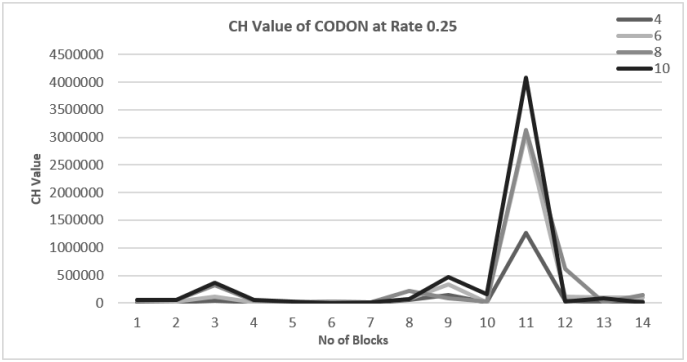

Fig 10:CH value of Codon at Incremental blocks

Figures 11, 12 and 13 illustrate the analytical study of the three clustering techniques i.e. IFDC, Simple FCM, and k-means. The analysis was done on three datasets IRIS, Diamond, and Codon using three validity measures. The promising two clusters from each dataset were considered for comparison study. According to Figures 2, 3, and 4, clusters 2 and 3 of the IRIS dataset are selected. From Figures 6, 7, and 8, clusters 9 and 12 of the Diamond dataset are chosen. Similarly, from Figures 9, 10, and 11, clusters 6 and 8 of the Codon dataset are considered. The sampling rate taken in this experiment is 0.75.

The vertical axis represents two criteria number of clusters and the validity measures. The horizontal axis represents the values of each validity measure. In essence, the graph suggests that the IFDC depicts better results while clustering large data.

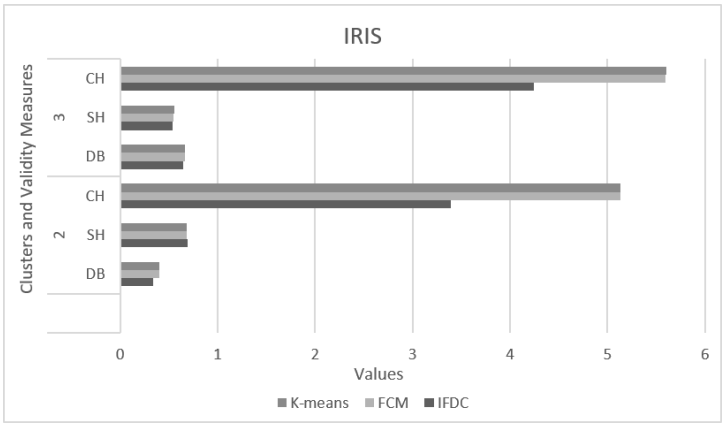

Fig 11: Comparison of IFDC, FCM and K-means on IRIS

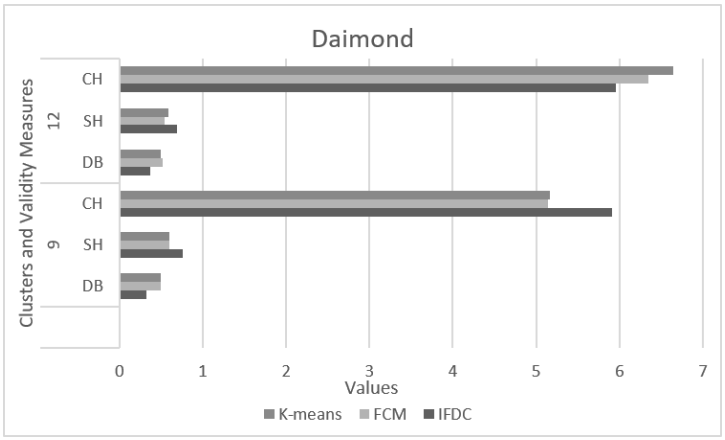

Fig 12: Comparison of IFDC, FCM and K-means on Diamond

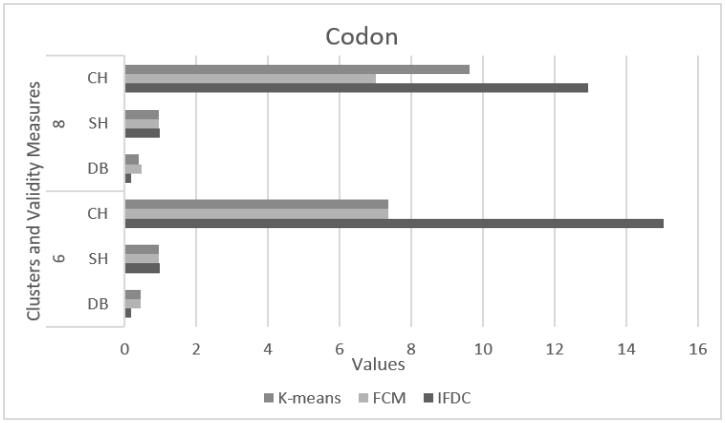

Fig 13: Comparison of IFDC, FCM and K-means on Codon

It can be inferred that IFDC clustering has the highest index values compared to the FCM and k-means clustering methods. From the values, it is deciphered that while using a small dataset the results of the IFDC are not always superior to that of the others. IFDC operates similarly to simple FCM and k-means when employing a small dataset as a single set such as the IRIS dataset. At the same time, it performs well in a large dataset. This is deciphered by the values of criteria measures of other datasets. IFDC segregates the complete data into small chunks according to the size of the memory and then processes it. It enhances memory consumption while reducing the problem of limited memory. IFDC also handles streaming data very efficiently by propagating the characteristics of each block to the next one.

6. CONCLUSION

The Incremental FCM Double Clustering (IFDC) algorithm offers a significant advancement in handling large datasets and streaming data by overcoming the limitations of traditional FCM clustering. IFDC segments the data into manageable chunks processes each chunk

individually, and uses a representative sampling technique to merge and propagate knowledge through subsequent chunks. This method reduces the need for extensive memory and repetitive calculations, enhancing efficiency and performance, particularly with large and dynamic datasets.

However, while IFDC addresses many challenges, some issues remain. Anomalies and the selection of initial cluster centers still pose difficulties, and further research is needed to refine these aspects. Overall, IFDC marks a notable improvement in data clustering, particularly for real-time applications and streaming data, providing a robust foundation for future advancements in the field. This study provided insights into the strengths, potential and weaknesses areas for improvement of the IFDC algorithm.

REFERENCE

[1] T. J. Ross, Fuzzy Logic with Engineering Applications. 2010.

[2] J. C. Dunn, "A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters," J. Cybern., vol. 3, no. 3, pp. 32–57, 1973, doi: 10.1080/01969727308546046.

[3] J. C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms. 1981.

[4] J. C. Bezdek, "FCM : THE FUZZY c-MEANS CLUSTERING ALGORITHM 1 ; yk E Y ~ l," vol. 10, no. 2, pp. 191–203, 1984.

[5] A. S. Bozkir and E. A. Sezer, "FUAT - A fuzzy clustering analysis tool," Expert Syst. Appl., vol. 40, no. 3, pp. 842–849, 2013, doi: 10.1016/j.eswa.2012.05.038.

[6] Agbonifo and O. Catherine, "Fuzzy C-Means Clustering Model for Identification of Students ' Learning Preferences in Online Environment," Int. J. Comput. Appl. Inf. Technol., vol. 4, no. I, pp. 15–21, 2013.

[7] H. Izakian and A. Abraham, "Fuzzy C-means and fuzzy swarm for fuzzy clustering problem," Expert Syst. Appl., vol. 38, no. 3, pp. 1835–1838, 2011, doi: 10.1016/j.eswa.2010.07.112.

[8] T. M. Silva Filho, B. A. Pimentel, R. M. C. R. Souza, and A. L. I. Oliveira, "Hybrid methods for fuzzy clustering based on fuzzy c-means and improved particle swarm optimization," Expert Syst. Appl., vol. 42, no. 17–18, pp. 6315–6328, 2015, doi: 10.1016/j.eswa.2015.04.032.

[9] M. R. Mahmoudi, D. Baleanu, Z. Mansor, B. A. Tuan, and K. H. Pho, "Fuzzy clustering method to compare the spread rate of Covid-19 in the high risks countries," Chaos, Solitons and Fractals, vol. 140, pp. 1–9, 2020, doi: 10.1016/j.chaos.2020.110230.

[10] T. Lei, X. Jia, Y. Zhang, L. He, H. Meng, and A. K. Nandi, "Significantly Fast and Robust Fuzzy C-Means Clustering Algorithm Based on Morphological Reconstruction and Membership Filtering," IEEE Trans. Fuzzy Syst., vol. XXX, no. XXX, pp. 1–15, 2018, doi: 10.1109/TFUZZ.2018.2796074.

[11] T. Bonis and S. Oudot, "A fuzzy clustering algorithm for the mode-seeking framework," Pattern Recognit. Lett., vol. 102, pp. 37–43, 2018, doi: 10.1016/j.patrec.2017.11.019

[12] Z. Siqing, T. Yang, and Y. Feiyue, "ScienceDirect ScienceDirect Fuzzy Logic-Based Clustering Algorithm for Multi-hop Wireless Fuzzy Logic-Based Clustering Algorithm for Multi-hop Wireless Sensor Networks Sensor Networks," Procedia Comput. Sci., vol. 131, pp. 1095–1103, 2018, doi: 10.1016/j.procs.2018.04.270.

[13] O. M. Saad, A. Shalaby, L. Samy, and M. S. Sayed, "Automatic arrival time detection for earthquakes based on Modified Laplacian of Gaussian filter," Comput. Geosci., vol. 113, pp. 43–53, 2018, doi: 10.1016/j.cageo.2018.01.013.

[14] P. Karczmarek, A. Kiersztyn, W. Pedrycz, and D. Czerwiński, "Fuzzy C-Means-based Isolation Forest," Appl. Soft Comput., vol. 106, p. 107354, 2021, doi: 10.1016/j.asoc.2021.107354.

[15] M. Salah, "Filtering of remote sensing point clouds using fuzzy C-means clustering," Appl. Geomatics, vol. 12, no. 3, pp. 307–321, 2020, doi: 10.1007/s12518-020-00299-3.

[16] J. P. Mei, Y. Wang, L. Chen, and C. Miao, "Large Scale Document Categorization With Fuzzy Clustering," IEEE Trans. Fuzzy Syst., vol. 25, no. 5, pp. 1239–1251, 2017, doi: 10.1109/TFUZZ.2016.2604009.

[17] R. Jiao, S. Liu, W. Wen, and B. Lin, "Incremental kernel fuzzy c-means with optimizing cluster center initialization and delivery," Kybernetes, vol. 45, no. 8, pp. 1273–1291, 2016, doi: 10.1108/K-08-2015-0209.

[18] A. Ribert, A. Ennaji, Y. Lecourtier, P. S. I. F. Sciences, and U. De Rouen, "An Incremental Hierarchical Clustering," Interface, no. May, pp. 19–21, 1999, [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:An+incremental+hierarchical+clustering#3.

[19] R. J. Kuo, T. C. Lin, F. E. Zulvia, and C. Y. Tsai, "A hybrid metaheuristic and kernel intuitionistic fuzzy c-means algorithm for cluster analysis," Appl. Soft Comput. J., vol. 67, pp. 299–308, 2018, doi: 10.1016/j.asoc.2018.02.039.

[20] F. Can, "Incremental Clustering for Dynamic Information Processing," ACM Trans. Inf. Syst., vol. 11, no. 2, pp. 143–164, 1993, doi: 10.1145/130226.134466.

[21] W. Zhao, L. Li, S. Alam, and Y. Wang, "An incremental clustering method for anomaly detection in flight data," Transp. Res. Part C Emerg. Technol., vol. 132, no. September 2019, p. 103406,

2021, doi: 10.1016/j.trc.2021.103406.
[22] P. Hore, L. O. Hall, and D. B. Goldgof, “Single pass fuzzy c means,” IEEE Int. Conf. Fuzzy Syst., 2007, doi: 10.1109/FUZZY.2007.4295372.
[23] P. Hore, L. O. Hall, D. B. Goldgof, and W. Cheng, “Online fuzzy C means,” Annu. Conf. North Am. Fuzzy Inf. Process. Soc. - NAFIPS, pp. 1–5, 2008, doi: 10.1109/NAFIPS.2008.4531233.
[24] J. P. Mei, Y. Wang, L. Chen, and C. Miao, “Incremental fuzzy clustering for document categorization,” IEEE Int. Conf. Fuzzy Syst., pp. 1518–1525, 2014, doi: 10.1109/FUZZ-IEEE.2014.6891554.
[25] M. Al-Ayyoub, S. M. Alzu’Bi, Y. Jararweh, and M. A. Alsmirat, “A GPU-based breast cancer detection system using Single Pass Fuzzy C-Means clustering algorithm,” Int. Conf. Multimed. Comput. Syst. - Proceedings, vol. 0, pp. 650–654, 2017, doi: 10.1109/ICMCS.2016.7905595.
[26] Y. Li, Q. Wang, K. Ran, and L. Jiao, “Weighted Single-Pass Fuzzy c-Means Algorithm Based on Density Peaks,” IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON, vol. 2018-Octob, no. October, pp. 2214–2217, 2019, doi: 10.1109/TENCON.2018.8650348.
[27] M. D. Woodbright, M. A. Rahman, and M. Z. Islam, “A Novel Incremental Clustering Technique with Concept Drift Detection,” 2020, [Online]. Available: http://arxiv.org/abs/2003.13225.
[28] S. Laohakiat and V. Sa-ing, “An incremental density-based clustering framework using fuzzy local clustering,” Inf. Sci. (Ny)., vol. 547, pp. 404–426, 2021, doi: 10.1016/j.ins.2020.08.052.
[29] X. Wang and Y. Xu, “An improved index for clustering validation based on Silhouette index and Calinski-Harabasz index,” IOP Conf. Ser. Mater. Sci. Eng., vol. 569, no. 5, 2019, doi: 10.1088/1757-899X/569/5/052024.
[30] P. Jha, A. Tiwari, N. Bharill, M. Ratnaparkhe, N. Nagendra, and M. Mounika, “Scalable incremental fuzzy consensus clustering algorithm for handling big data,” Soft Comput., vol. 25, no. 13, pp. 8703–8719, 2021, doi: 10.1007/s00500-021-05733-1.
[31] A. D. Kochuveettil and R. Mathew, “A novel approach to fuzzy c-Means clustering using kernel function,” Intell. Decis. Technol., vol. 16, no. 4, pp. 643–651, 2022, doi: 10.3233/IDT-210091.
[32] J. C. . R. E. ;WILLIAM F. Bezdek, “FCM: THE FUZZY c-MEANS CLUSTERING ALGORITHM,” FCM Fuzzy c-Means Clust. Algorithm, vol. 10, no. 2, pp. 191–203, 1984.
[33] P. J. Rousseeuw, “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis,” J. Comput. Appl. Math., vol. 20, no. C, pp. 53–65, 1987, doi: 10.1016/0377-0427(87)90125-7.
[34] X. Gu, Q. Ni, and G. Tang, “A Novel Data-Driven Approach to Autonomous Fuzzy Clustering,” IEEE Trans. Fuzzy Syst., vol. 30, no. 6, pp. 2073–2085, 2022, doi: 10.1109/TFUZZ.2021.3074299.
[35] D. L. Davies and D. W. Bouldin, “A Cluster Separation Measure,” IEEE Trans. Pattern Anal. Mach. Intell., vol. PAMI-1, no. 2, pp. 224–227, 1979, doi: 10.1109/TPAMI.1979.4766909.