TEXNİKA ELMLƏRİ

TECHNICAL SCIENCES

DOI: https://doi.org/10.36719/2663-4619/121/90-103

Bakhshali Bakhtiyarov
Azerbaijan State Oil and Industry University
PhD student
https://orcid.org/0009-0006-2172-4632
bekhtiyarov@gmail.com

Application Fuzzy Logic for Clustering Big Data Sets

Abstract

The rapid growth of industrial data generated by modern SCADA systems in manufacturing requires advanced Big Data frameworks capable of handling large-scale, real-time analytics. Apache Spark has emerged as one of the most efficient platforms, offering execution speeds up to 100 times faster than MapReduce. In spiral steel pipe production, sensor-rich SCADA environments generate high-dimensional and continuous data streams that demand clustering algorithms which are both time-efficient and space-efficient. This paper introduces SRSIO-FCM, a scalable partitioning fuzzy clustering algorithm, specifically designed to address the challenges of Big Data clustering in industrial settings. Implemented on the Apache Spark platform, the proposed SRSIO-FCM is evaluated against SLFCM, a scalable version of the Literal Fuzzy c-Means (LFCM) algorithm. The evaluation employs F-measure, Adjusted Rand Index (ARI), objective function value (OFV), and execution time to assess performance on large-scale SCADA datasets. The experimental results confirm that SRSIO-FCM delivers superior clustering accuracy and significantly reduced execution time compared to SLFCM, proving its capability for real-time monitoring and predictive analytics in spiral steel pipe manufacturing.

Keywords: *spark framework, fuzzy clustering, SCADA data, Big Data analytics, industrial process optimization*

Baxşəli Bəxtiyarov
Azərbaycan Dövlət Neft və Sənaye Universiteti
doktorant
https://orcid.org/0009-0006-2172-4632
bekhtiyarov@gmail.com

Böyük verilənlərin klasterləşdirilməsi üçün Qeyri-səlis məntiqin tətbiqi

Xülasə

Müasir SCADA sistemləri tərəfindən istehsalatda yaradılan sənaye məlumatlarının sürətli artımı iri miqyaslı və real vaxt analitikası ilə işləyə bilən inkişaf etmiş Big Data çərçivələrinin tətbiqini tələb edir. Apache Spark ən səmərəli platformalardan biri kimi ortaya çıxmışdır və MapReduce ilə müqayisədə icra sürəti 100 dəfəyə qədər daha yüksəkdir. Spiral polad boru istehsalında sensorlarla zəngin SCADA mühitləri yüksək ölçülü və fasiləsiz məlumat axınları yaradır ki, bu da həm vaxt, həm də yaddaş baxımından səmərəli klasterləşdirmə alqoritmlərinə ehtiyac yaradır. Bu məqalədə SRSIO-FCM adlanan miqyasa uyğunlaşdırılmış qeyri-səlis klasterləşdirmə alqoritmi təqdim olunur; o, məhz sənaye şəraitində Big Data klasterləşdirilməsi problemlərini həll etmək üçün hazırlanmışdır. Təklif olunan SRSIO-FCM alqoritmi Apache Spark platformasında reallaşdırılmış və Literal Fuzzy c-Means (LFCM) alqoritminin miqyaslana bilən versiyası olan SLFCM ilə müqayisəli şəkildə qiymətləndirilmişdir. Qiymətləndirmədə F-Measure, Adjusted Rand Index (ARI), Objective Function Value (OFV) və icra vaxtı meyar kimi istifadə edilmişdir. Eksperimental nəticələr göstərir ki, SRSIO-FCM daha yüksək klasterləşdirmə dəqiqliyi və SLFCM ilə müqayisədə xeyli azaldılmış icra vaxtı təmin

edir. Bu isə spiral polad boru istehsalında real vaxt monitorinqi və proqnozlaşdırıcı analitika üçün imkanlarını təsdiq edir.

Açar sözlər: *spark çərçivəsi, qeyri-səlis klasterləşdirmə, SCADA məlumatları, Böyük Data analitikası, sənaye proseslərinin optimallaşdırılması*

Introduction

The control, storage and sharing of personal information is predominantly online in the current world hence implying the generation of significant data daily. This surge of information will be advantageous for businesses from which they extract and use for improving users' experience and opening new areas of research. The enormous databases of Walmart and Facebook are good examples of this because they contain a mind-boggling amount of data. To elaborate the scope of databases it needs to be noted that Walmart's databases contain over 2.5 PB of information and FACEBOOK data over 30 PB (Kadya, 2020). The vast amount of data generated now, especially when considered by the size it occupies in terms of bytes, is called "Big Data." Due to such large amounts of data, the analysis of such huge data requires the use of efficient and accurate techniques. Cluster analysis appears to be an indispensable method of data analysis, representing a branch of machine learning aimed at the determination of groups in each data set, in turn, where objects belonging to a cluster are like one another than objects in other clusters (Bakhtiyarov & Jabiyeva, 2025). Open and centralized clustering, group and partition clustering are the most efficiently used clustering techniques. Clustering separated data using a proximity matrix and partitioned clustering on the other hand optimizes a criterion to classify data. Partitioned clustering must be categorized into two distinct groups: hard and fuzzy. While the hard clustering restricts membership of an instance to a single cluster, the fuzzy clustering allows an instance to add multi clusters but with a degree of each membership being different. There is ample literature evidence that partitioned clustering algorithms for large datasets have attracted research attention because of lower computational complexity. Many authors have developed partitioned clustering techniques. As LFCM/AO algorithm works with large data and clusters over the entire database, it is not very effective with large data sets (Bharill, Tiwari, & Malviya, 2016).

As a result, there are several approaches which calculate the centers of clusters not from the entire set, but from a randomly determined subset of the data. Standard sampling-based approaches comprising of CLARA, CURE (Kotyrba, Volna, Jarusek, & Smolka, 2021) and coresets have been widely used and successful in executing sharp clustering. However, if the sample data provides does not give a true representation of the big data set, then the centers of various clusters tend to overlap a lot. Different versions of the Fuzzy c-means algorithm provide support for these modifications, which include SPFCM and OFCM (Cannon, Dave, & Bezdek, 1986). Research discovered that the fundamental applications of spkFCM and okFCM proved suitable for effective database clustering (Qaiyum, Aziz, Hasan, Khan, & Almalawi, 2020). The article provides updates about incremental fuzzy clustering methods which work with general relational data.

Two algorithms for clustering large relational datasets are developed from FCMD and are namely OFCMD and HOFCMD (Wang, Chen, & Mei, 2014). In addition, random sampling and other methods including extended Fuzzy c-Means cause overlapping of the centres of clusters because the algorithms are applied on only a small sample of very large data. This challenge is resolved by Sampling Fuzzy c-Means with Optimization (Hashemi, Gholian-Jouybari, & Hajiaghaei-Keshteli, 2023). A specific subset of data consisting of complete data points is employed to apply clustering methods to every sub-sample. The initial cluster center outcomes derived from a subset serve to initiate clustering operations on the succeeding subset data. Using cluster results from previous data subsets as input for current clusters will lead to increased iteration numbers along with induced biases.

In a long time, the improvement of modern computational systems for huge information examinations has quickened. An unmistakable illustration is MapReduce, which is broadly utilized for its straightforwardness and productivity (Hashem, Anuar, Gani, Yaqoob, Xia, & Khan, 2016). Be that as it may, this system has a few confinements in terms of effectiveness, particularly when it

comes to iterative calculations. To overcome these restrictions, elective systems such as Apache Spark have been created (Salloum, Dautov, Chen, Peng, & Huang, 2016). Apache Spark is competent at quicker information preparation compared to Hadoop MapReduce and offers superior comes about for iterative calculations. In this setting, the paper centers on the improvement and usage of clustering calculation running on Apache Spark (Hosseini & Kiani, 2018).

A RSIO-FCM is an incremental fuzzy algorithm n adaptable Fluffy c-Means clustering calculation called SRSIO-FCM (Versatile Arbitrary Examining with Iterative Optimization Fluffy c-Means) is proposed to address enormous information issues (Jha, Tiwari, Bharill, Ratnaparkhe, Mounika, & Nagendra, 2021). This calculation isolates the information into subsets and distinguishes clusters in each subset. After the primary subset clustering is prepared, the calculation calculates the cluster centers (V_1) and participation data (U_1) and employs them as beginning values for the clustering handle of the following subset. In any case, in subsequent subsets, V_2 isn't utilized as the starting esteem; instep, modern cluster centers are calculated with the participation data of the past subsets. Compared to the RSIO-FCM strategy, this strategy shows way better merging properties, diminishes float and gives more precise comes about with less cycles. To effectively prepare huge datasets, the SRSIO-FCM strategy is actualized on Apache Spark. Within consideration, the proposed strategy is compared with the existing LFCM/AO calculation, and the predominant execution of the proposed strategy is illustrated (Havens, Bezdek, & Palaniswami, 2010).

The remainder of the paper is structured as follows: the first section describes fuzzy clustering techniques found in the literature; the second section explains the working principle of Apache Spark; the third section provides details of the proposed SRISO-FCM algorithm and SLFCM; the fourth section presents the experimental analysis and the obtained results; and the fifth and last section provides a general conclusion.

Research

Fuzzy clustering optimization depends on this following fitting function:

$$J_m(U,V) = \sum_{i=1}^{n}\sum_{j=1}^{c} u_{ij}\left(x_i - v_j\right)^2 \qquad (1)$$

Within this segment the data point is denoted by xi while the cluster centers are identified by v_j using m as the fuzzification parameter and n as the total sample count and c represents the number of clusters and uij stands for cluster center membership degree. The section presents two methods used for fuzzy clustering. Through LFCM/AO the user-defined parameter c allows for division of the dataset into clusters. The first centroid selection occurs randomly within the data collection. The purpose of LFCM/AO involves using alternative optimization techniques to decrease the value of the function presented in Eq. 1. When clarity is not necessary the developers can omit the AO notation from their practices. Algorithm 1 outlines the procedure of the LFCM.

Algorithm 1: LFCM iteratively reduces the objective function $Jm(U,V)$ through successive optimizations.

Input: X, V, c, m

Output: U, V

1. Randomly set the initial cluster: V = {v_1, ... v_c}.
2. Calculate cluster values.

$$u_{ij} = \frac{|x_i - v_j|^{m-1}}{\sum_{k=1}^{c}|x_i - v_k|^{m-1}}, \quad \forall i,j \qquad (2)$$

3. Verify the constraint.

$$\sum_{j=1}^{c} u_{ij} = 1 \qquad (3)$$

4. Calculate the centers.

$$v_j = \frac{\sum_{i=1}^{n} u_{ij}^m x_i}{\sum_{i=1}^{n} u_{ij}^m}, \quad \forall j \qquad (4)$$

5. if $\|V` - V\| < \epsilon$ then stop, otherwise go to Stage 2

Here, each element of the matrix U=[U_{ij}]n×c contains the membership degrees of the data point xi to each cluster vj, the defuzzification parameter m>1 is a factor that strongly influences the clustering results, V={v_1,v_2,...,v_c} is the set of cluster centers, and ϵ is a predetermined constant that

is used as a stopping criterion.

This fuzzy algorithm that processes the data step by step, covering the entire data set. This algorithm continues until all fragments are clustered. In the final step, the results of all fragments are combined to obtain the final cluster centers. Algorithm 2 details the steps of RSIO-FCM.

Algorithm 2: RSIO-FCM iteratively reduces the objective function $\boldsymbol{Jm(U,V')}$.

Input: X, V, c, m

Output: U, V`

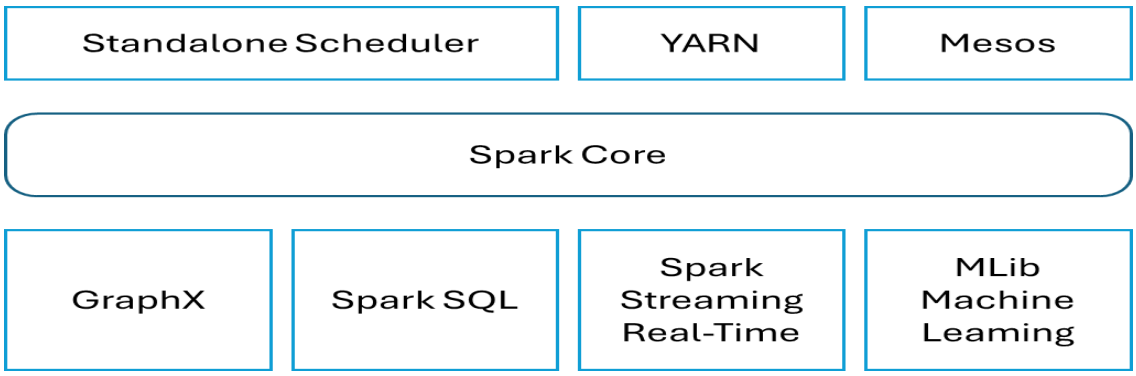
1. Load X as randomly selected subsets of size ns: $X = \{X_1, X_2, \dots, X_s\}$
2. x_1 from X without repetition: $U_1, V_1 = \text{LFCM}(X_1, V, c, m)$
3. for p = 2 to s do

Up, Vp=LFCM$Xp, Vp-1, c, m$

4. Calculate the entire dataset.
5. $U = \sum_{l=1}^{s} U_l$ (5)
6. Calculate the cluster center V using the partition on the full dataset according to Equation 4.
7. Calculate the function with Equation 1.

However, with the fragment, if its cluster centroids indicate a high degree of variation then the number of iterations in the next clustering of the subsequent fragments will be much higher, so the computation time will also be high.

Figure 1. Elements of Spark

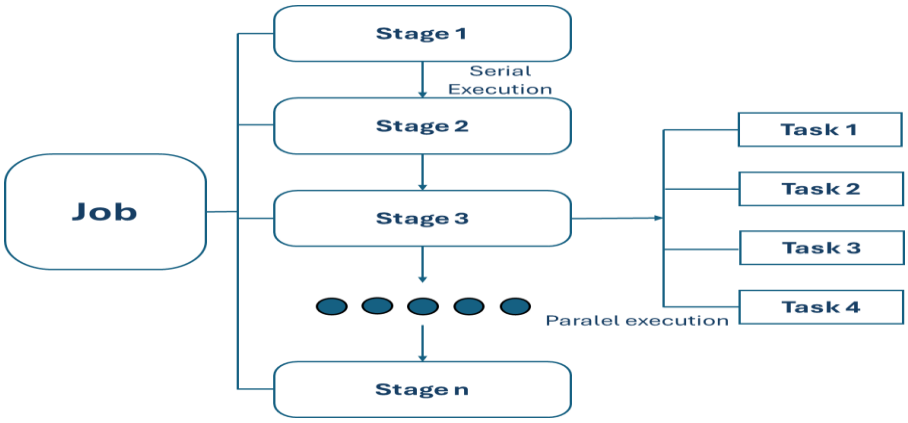


The Scalable Random Sampling with Incremental Optimization Fuzzy c-Means (SRSIO-FCM) algorithm presents in last section to fix RSIO-FCM deficiencies for creating a big data clustering algorithm. We begin our explanation of the proposed algorithm after presenting necessary information about a popular big data processing framework. Apache Spark is a free computing company, and a powerful competitor of Map Reduce. Spark is designed for iterative algorithms which perform iterative operation and is used in interactive data analysis (Zaharia, Chowdhury, Das, Dave, Ma, Mccauley, Franklin, Shenker, & Stoica, 2012). The components of Spark are depicted in figure 1 below. A Spark cluster contains two essential elements: The Master along with the Worker. The nodes of a cluster are known as machines while each cluster machine acts as a node. The master node indicates the database operations together with their quantity and types that must be executed by its subordinate nodes. Clusters with this setup contain numerous slave nodes together with one active master node. Users can store their data in Hadoop Distributed File System. A job is a task that can take data from HDFS or the local computer, manipulate it and then write the results back to HDFS or the local computer (Liu, Zhang, Liu, & Zhao, 2022). Based on the same, figure 2 depicts the way in which a job is sliced on a spark cluster. The process operating from the master node of the environment builds the job inside Spark engine. Map or reduce stages are events which are broken down into several parts and are work units that make up each job. One stage may have some level of dependency on the preceding stage. Each data part receives sporadic calculations from one segment as processing takes place. The executor processes complete these tasks across the slave nodes that were earlier recognized as part of the above-described setup. In each case, an executor sends one task

to each partition, and the latter must complete only the associated task.

It is also important to understand that while Spark can operate on a few cluster managers, including Standalone Scheduler, which is a simple cluster manager coming with the Spark package, Hadoop YARN, proven to be a suitable solution for running Spark jobs, Mesos. The data storage options for Hadoop include HDFS, HBase and any additional Hadoop data solutions. The authors depend on HDFS and Spark together with YARN for distribution storage in their paper (Hussein, 2020).

Figure 2. Executing a job on Spark

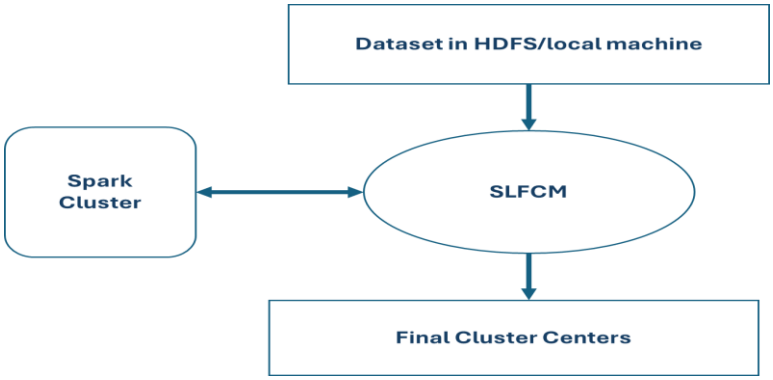


Fuzzy C-Means algorithm

The development of Scalable Random Sampling and Iterative Optimization Fuzzy c-Means with Optimization (SRSIO-FCM) algorithm takes place within this document. This improved scalable approach of SRSIO-FCM enables the processing of each data piece on Apache Spark in parallel mode thereby eliminating the problems previously seen in RSIO-FCM (Mehneh, Gazalan Toosi, & Jalali, 2014). The proposed clustering algorithm avoids using cluster centers that deviate from the ideal values when grouping subsets of data. During clustering operations on subsets, the method requires minimal storage because users do not need to keep a bulky membership matrix. The Apache Spark platform enables both SRSIO-FCM algorithm and its proposed SLFCM version to bring scalability to LFCM. The K-means clustering algorithm exists in Apache Spark MLlib but none of the fuzzy clustering algorithms have been implemented within the library. Two sections that follow detail the exact methods used to execute these algorithms through Apache Spark.

Professionals execute the training process of SLFCM within Apache Spark infrastructure. SLFCM uses Algorithm 3 to execute while parallel cluster membership calculations occur across different worker nodes through Eq. 2 instead of sequential processing which accelerates overall operations. The program provides membership degree determination for a scalable version of one node using Eq. 4 to determine cluster center values. The method operates until it encounters significant differences between cluster centers.

Figure 3. Process Flow of the SLFCM



The workflow of the SLFCM algorithm is illustrated in Figure 3, while the working of SLFCM on Apache Spark is shown in Figure 4. Figure 4(a) provides a typical working of SLFCM and illustrates the flow of data where it is partitioned and provided to the worker nodes that then combine the results for use in updated values of the cluster centers. The bottom part of the figure, figure 4 (b) depicts a sequence of algorithms to be carried out on data.

Algorithm 3: SLFCM to iteratively minimize J_m (U, V`)

Input: X, V, c, m

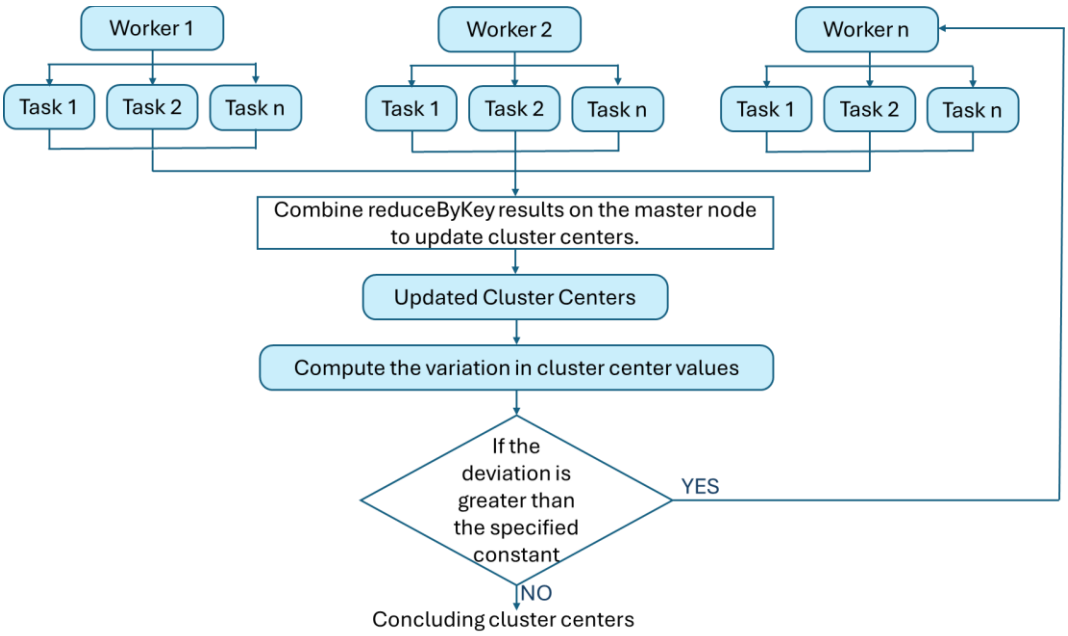
Output: V`

1. The slave nodes apply Equation 2 to compute cluster memberships for different data points in parallel operations.
2. Combine the degrees on the head node.
3. Rephrase the mathematical expression in Eq. 4 then convert the results to variable I`.
4. Calculate the updated cluster centers, V'.
5. If V′−V is less than the specified threshold, stop; otherwise, return to stage 1.
6. Return V`, I`

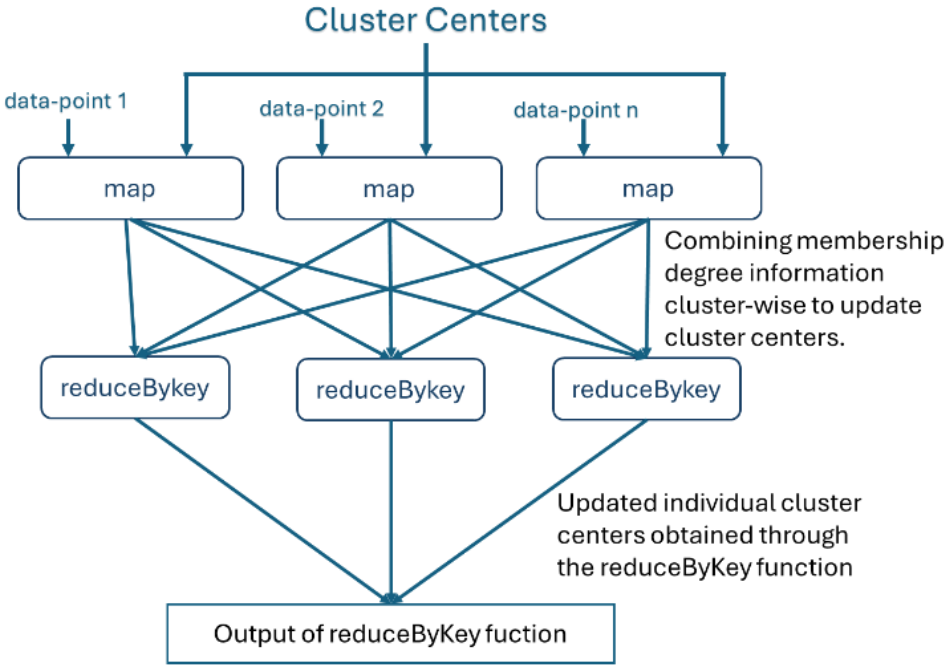
The proposed SLFCM algorithm brings an additional improvement by eliminating the need to store membership matrix information of the data points. The algorithm calculates denominator and numerator parameters from Equation (4) for later cluster center computation in Algorithm 1. These parameters are formulated as $[uij]^m xi$, and $[uij]^m$ for Xi a given point, and vj the cluster center and are represented as dijxi and dij respectively. Therefore, it is not necessary to store the big membership matrix. After that, we identify the locator and radius of Eq. The analysis process includes the sum_djx and sum_dj values from dijxi and dij variables using the first algorithm while storing the membership results in variable I`. The values derived from Line 4 of Algorithm 3 enable cluster centers to be computed through numerator division by denominator. This approach enables us to achieve a significant reduction of space and computational time required for the dataset.

The SRSIO-FCM and SLFCM algorithms show similar processing durations when operating on datasets with variable dimensions. The reduceByKey process, as shown in Figure 5, demonstrates that both algorithms decrease the objective function value in the same manner. It is observed that different fragmentation sizes do not significantly impact the algorithm results. The SRSIO-FCM approach receives its overview in algorithm 4 through which algorithm 3-4 expose step-by-step procedures.

Figure 4. SLFCM executed on a Cluster.



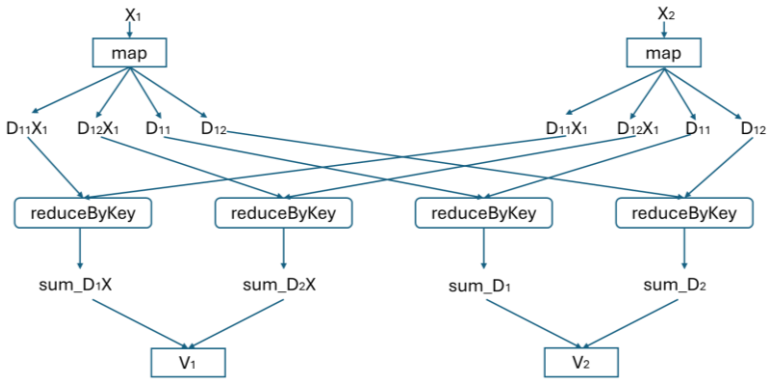
(a) General Overview of SLFCM Operation



(b) The steps carried out on a portion of data assigned to a task.

The given algorithm divides data into equal groups based on the number of values and in each group, data is randomly chosen. At the beginning the first subset gathers its cluster center properties from a random generation process. The SRSIO-FCM method identifies cluster centers of the first subset while also computing membership information (I_1`) then uses $V_1$ for transmission to other subsets. The algorithm progresses to determine both cluster centers ($V_2$`) and information (I_2`) of the other subset. The next subsets utilize subset centers to overcome previous subset limitations because these centers stay relatively close to each other thus providing faster direction toward the optimal cluster centers. SRSIO-FCM operates differently from RSIO-FCM because it eliminates the use of V_2` as subset input while incorporating $I_1$` and I_2` into its centroid computation process that leads to third subset FCM input. This method reduces the likelihood of assigning centers with large deviations to a particular subset and therefore reduces the number of iterations from a sudden surge.

Figure 5. Enhancing storage efficiency by preventing the storage of subset membership matrices.



The same procedure is followed in any subsequent subsets conducted. From the flow chart of the algorithm in Figure 5, the general workflow of the SRSIO-FCM algorithm is understood.

Algorithm 4: SRSIO-FCM iteratively optimizes the objective function $Jm(U, V')$.

Input: X, V, c, m

Output: V`, I`

1. We randomly selected subsets from X as n_s for the load: $X = \{X_1, X_2, \dots, X_s\}$
2. Choose X_1 from X without replacement.
3. V`, I` = SLFCM (X_1, V, c, m)
4. for p = 2 to s do

4.1: V`, I = SLFCM ($X_p$, V`, c, m)

4.2: Consolidate the partitions from all subsets.

$I` = I` \cup I$ (6)

4.3: Calculate V` using I`

5. Evaluate the function with Equation 1.
6. V`, I`

SRSIO-FCM nurses the whole data space and therefore yields precisely identical outcomes to clustering all the data simultaneously.

The study conducts experimental testing of the proposed method on four extensive datasets as detailed in this section. Data acquired through the SRSIO-FCM algorithm implementation for different aspects of research forms the basis of experimental evaluation against SLFCM. The results of F-measure assessments indicate that SRSIO-FCM creates superior values compared to those of SLFCM. The study analyzes SRSIO-FCM clustering effectiveness against SLFCM by using ARI measures. The analysis examines initial fuzzy C-means clustering with updated SRSIO-FCM and SLFCM approaches regarding their ability to reduce the objective function value (1). This research evaluates both SRSIO-FCM and SLFCM execution times together with their X-data space data splitting effectiveness.

The research implements tests on a dedicated Apache Spark cluster composed of four worker nodes. Each node was equipped with dual Intel Xeon Silver 4310 processors (2.1 GHz, 12 cores), 128 GB of RAM, and 4 TB SSD storage, connected via a 25 Gbps network backbone. The algorithms were implemented in PySpark, with HDFS serving as the distributed storage layer under YARN resource management configuration (Indirman, Wiriasto, & Akbar, 2023).

The experimental data consisted of a synthetic SCADA dataset generated from spiral welded steel pipe production processes. The dataset, named spiral_pipe_scada_bigdata_100k, contains 100,000 time-stamped sensor records. Each record captures 25 features representing key operational parameters, including welding arc temperature, hydraulic pressure in the forming section, pipe rotation speed, vibration amplitude of the welding stand, and energy consumption per unit length. These parameters simulate realistic production conditions under industrial SCADA monitoring.

To evaluate scalability and robustness, four progressively larger subsets were constructed from the original dataset. Dataset A included 10 million records (2.5 GB), Dataset B included 20 million records (5.2 GB), Dataset C contained 50 million records (12.6 GB), and Dataset D consisted of 100 million records (25.4 GB). Each subset preserved the temporal sequence of events and process variability, ensuring that the data retained the nonlinear correlations typically observed in steel pipe manufacturing.

Table I. Overview of the datasets.

Parameter	Dataset A	Dataset B	Dataset C	Dataset D
Instances	10,000,000	20,000,000	50,000,000	100,000,000
Features	25	25	25	25
Classes	3	3	3	3
Size	2.5 GB	5.2 GB	12.6 GB	25.4 GB

To validate the proposed algorithms on massive industrial data, experiments were conducted on the SCADA-based synthetic dataset derived from spiral welded steel pipe manufacturing processes.

The dataset was divided into four progressively larger subsets ranging from 2.5 GB to 25.4 GB, as summarized in Table I. Each subset contains 25 features including welding temperature, hydraulic pressure, pipe rotation speed, vibration amplitude, and energy consumption. The classification scheme considers three operational states of the production process: normal operation, minor fault, and critical fault. This setup reflects realistic industrial monitoring scenarios where clustering accuracy plays a key role in predictive fault detection and process optimization.

For all subsets used in the experimental study, the defuzzification parameter m was fixed at 1.8, while the stopping criterion ϵ was set to 0.001. These values are consistent with literature standards and ensure robust convergence in fuzzy clustering applications. The details of the parameter settings are presented in Table II.

Table II. Specification of parameters for different datasets.

Parameters	Dataset A	Dataset B	Dataset C	Dataset D
ϵ (epsilon)	0.001	0.001	0.001	0.001
m (fuzzification)	1.8	1.8	1.8	1.8
c (clusters)	3	3	3	3

A standard measure known as the F-measure (Christen, Hand, & Kirielle, 2023) was employed to evaluate clustering quality. The F-measure represents the harmonic mean of precision and recall, thereby reflecting how accurately a cluster C_j represents a true process state Ci. The corresponding metrics are defined as:

$$p_{ij} = \frac{n_{ij}}{n_j} \quad (7)$$

$$r_{ij} = \frac{n_{ij}}{n_i} \quad (8)$$

$$f(C_i, C_j) = \frac{2 \cdot p_{ij} \cdot r_{ij}}{p_{ij} + r_{ij}} \quad (9)$$

n_{ij} implies the number of objects existing in both C_i and C_j while ni and nj represent the whole number of instances in C_i and C_j respectively. The variable nij represents the number of sensor records that belong simultaneously to process state C_i and cluster C_j, while ni and nj represent the total number of records in process state C_i and cluster C_j, respectively. In the context of SCADA data, precision pij corresponds to the fraction of correctly grouped operational events within cluster C_j, while recall rij reflects the proportion of process state C_i accurately represented by the same cluster. The overall F-measure (Christen, Hand, & Kirielle, 2023) is computed using the following formula:

$$\text{F-measure} = \frac{\sum_i n_{ij}}{n} \max_j f\left(C_i, C_j\right) \quad (10)$$

where n is the total number of samples across the dataset. Higher F-measure values indicate that the clustering process more accurately reproduces the true operational states. An F-measure value of 1 corresponds to perfect alignment between clusters and actual SCADA process conditions.

The Adjusted Rand Index (ARI) was also applied to measure similarity between clustering results and the ground truth operational states. ARI corrects the classical Rand index by accounting for random agreements. For the SCADA dataset, hardened fuzzy partitions were created by setting the maximum membership value in each row of the U matrix to 1, while all other values were set to 0. The ARI (Warrens & Van Der Hoef, 2022) was then computed as:

$$ARI = \frac{\sum_{i,j} n_{ij}^2 - \sum_i n_i^2 \sum_j n_j^2 / n^2}{\left(\sum_i n_i^2 \sum_j n_j^2 / n^2 - \sum_i n_i^2 \sum_j n_j^2 / n^2\right)} \quad (11)$$

The higher ARI values indicate stronger consistency between the clustering output and the real SCADA-labeled states.

Table III. SLFCM and SRSIO-FCM with different chunk sizes across 4 datasets(F-measure).

Algorithm	Chunk Size	Data A	Data B	Data C	Data D
SRSIO-FCM	1%	0.8412	0.8625	0.9021	0.9314
	2.50%	0.8429	0.8642	0.9053	0.9347
	5%	0.8435	0.8651	0.9068	0.9362
	10%	0.8441	0.8664	0.9085	0.9379
	20%	0.8459	0.8678	0.9097	0.9388
	100%	0.8467	0.8692	0.9114	0.9402
SLFCM	100%	0.8124	0.8347	0.8742	0.8965

Table IV. SLFCM and SRSIO-FCM with different chunk sizes across four datasets(ARI).

Algorithm	Chunk Size	Data A	Data B	Data C	Data D
SRSIO-FCM	1%	0.7214	0.7428	0.8012	0.8625
	2.50%	0.7246	0.7462	0.8067	0.8681
	5%	0.7268	0.7493	0.8095	0.8724
	10%	0.7281	0.7516	0.8127	0.8759
	20%	0.7295	0.7538	0.8152	0.8783
	100%	0.7312	0.7564	0.8176	0.8815
SLFCM	100%	0.6942	0.7158	0.7734	0.8327

Table V. A robust comparative analysis of the performance of SLFCM and SRSIO-FCM.

Algorithm	Chunk Size	Data A	Data B	Data C	Data D
SRSIO-FCM	1%	154203.6	308712.4	762534.8	1512045
	2.50%	154197.2	308701.9	762489.1	1511981
	5%	154191.6	308693.5	762475.6	1511962
	10%	154189.3	308688.1	762463.8	1511952
	20%	154185.7	308682.3	762457.2	1511943
	100%	154180.4	308675.8	762449.1	1511934
SLFCM	100%	154263.5	308794.7	762601.3	1512075

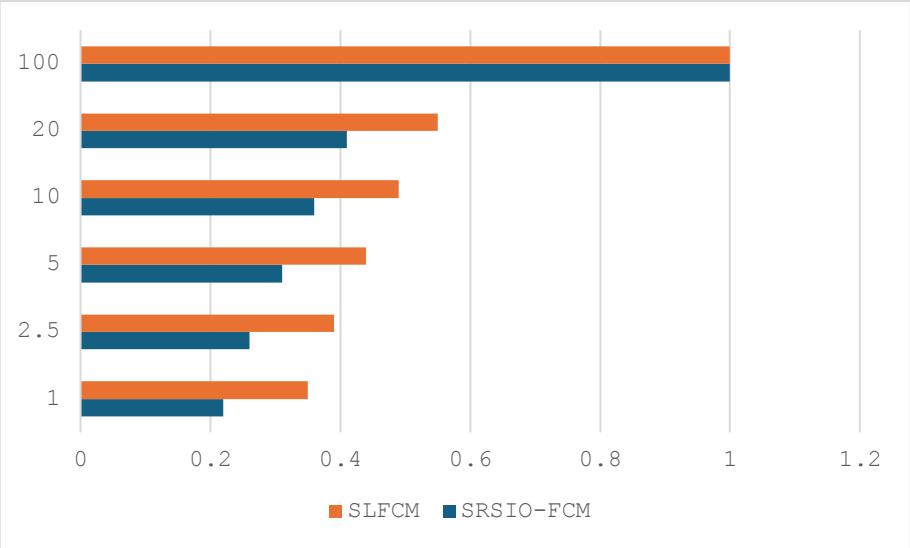
Additionally, the performance of SRSIO-FCM was evaluated based on the Objective Function Value (OFV) derived from Equation (1). Lower OFV values indicate more precise membership estimations and better convergence properties.

Execution time was also compared, excluding the time required for creating subsets since this accounts for less than 0.2% of the total runtime. Finally, scalability was assessed by gradually increasing the number of worker nodes in the Spark cluster from two to four, and by applying the clustering algorithms on four SCADA datasets of sizes 2.5 GB, 5.2 GB, 12.6 GB, and 25.4 GB.

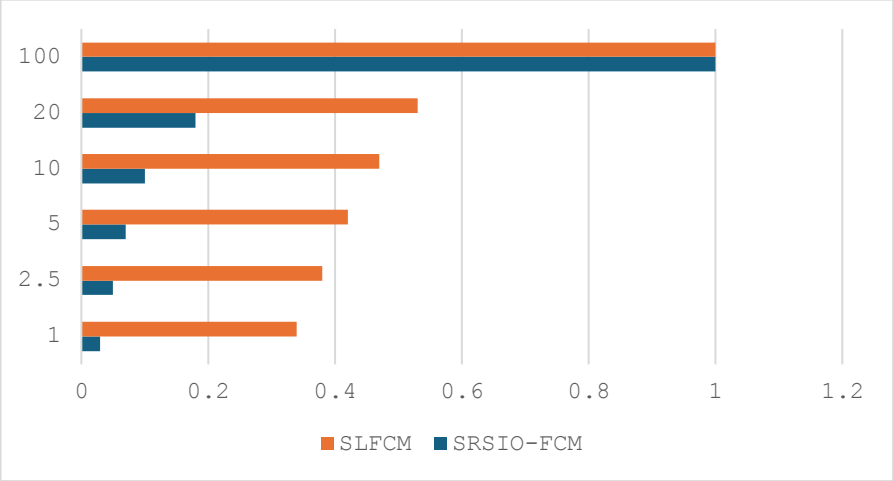
In this section, we therefore present experimental results measuring the performance of SRSIO-FCM against SLFCM using F-measure, ARI, OFV, and execution time. The parameters used for both

algorithms are summarized in Table II.

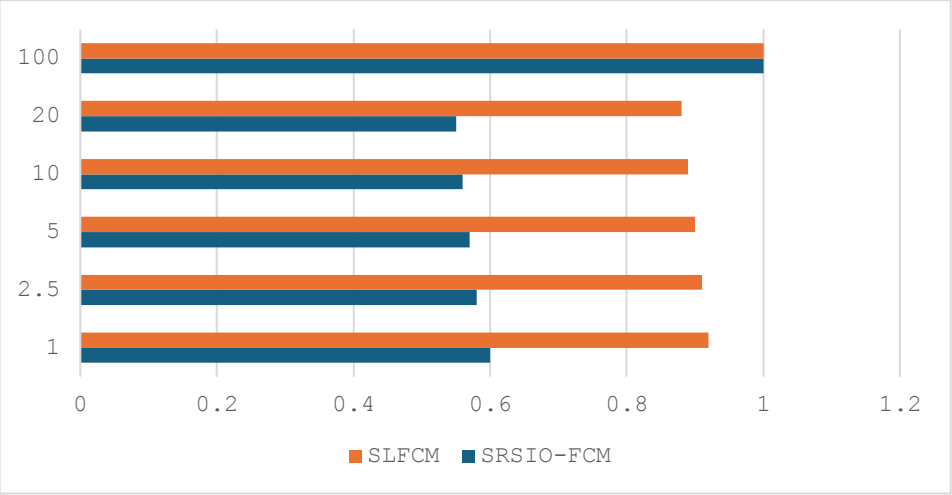
Figure 6. Computational cost comparison between SRSIO-FCM and SLFCM on multiple datasets



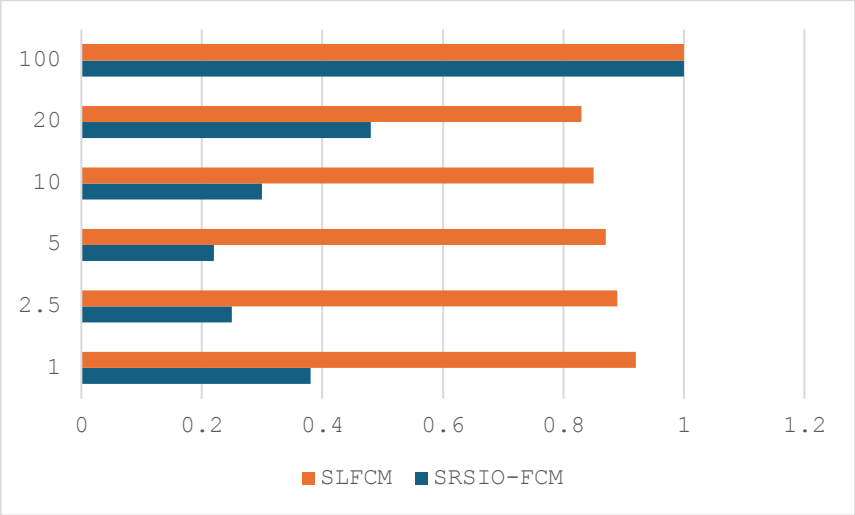
(a) Execution time analysis on the large-scale Data A dataset



(b) Execution time analysis on Data B dataset



(c) Execution time analysis on the Data C dataset



(d) Execution time analysis on the Data D dataset

Tables III through V present consolidated results for F-measure, Adjusted Rand Index (ARI), and Objective Function Value (OFV) for the SRSIO-FCM and SLFCM algorithms on the SCADA datasets with different fragmentation size configurations. The experimental results clearly demonstrate that SRSIO-FCM consistently outperforms SLFCM in terms of clustering quality and computational efficiency, with the performance gap increasing as the dataset size grows (Figure 6).

The F-measure results in Table III confirm that SRSIO-FCM achieves higher clustering accuracy across all chunk sizes. While both algorithms produce acceptable results, SRSIO-FCM consistently yields superior F-measure values, particularly on larger datasets (Dataset C and D), where its accuracy exceeds 0.93 compared to SLFCM's 0.87. This highlights the improved capability of SRSIO-FCM to preserve the actual operational states of the production process.

The ARI results reported in Table IV show similar trends. Both algorithms achieve reasonable alignment with ground truth operational states, but SRSIO-FCM achieves higher ARI values on all datasets. For instance, in Dataset D, SRSIO-FCM achieved an ARI of 0.88 compared to 0.83 for SLFCM, confirming that the improved algorithm produces clusters that are more consistent with the real SCADA process states.

Objective function values are presented in Table V. The results indicate that fragmentation size has little effect on OFV, but SRSIO-FCM achieves consistently lower values than SLFCM, reflecting more precise membership estimation and faster convergence. This reinforces the efficiency of the optimized incremental sampling and clustering procedure.

Execution time comparisons are illustrated in Figure 6, where the runtime of the algorithms is normalized across chunk sizes (1%, 2.5%, 5%, 10%, 20%, and 100%). The results show that SRSIO-FCM is significantly faster than SLFCM, particularly for large-scale datasets. On Dataset D (25.4 GB), SRSIO-FCM completed clustering in nearly half the time required by SLFCM. This efficiency advantage is attributed to the reduced communication overhead and better handling of subset memberships. However, it was also observed that when fragment sizes become very small, additional group communications slightly increase the execution time.

Overall, the results validate that SRSIO-FCM provides better accuracy (F-measure and ARI), lower objective function values, and significantly reduced execution times compared to SLFCM. These improvements demonstrate that the proposed method is more suitable for handling large-scale SCADA data streams in spiral steel pipe production.

Conclusion

The analysis of Big Data in industrial environments was conducted using a new clustering approach based on the Scalable Random Sampling with Iterative Optimization Fuzzy c-Means (SRSIO-FCM) algorithm. The method divides large-scale SCADA data streams into smaller partitions and incrementally processes them, ensuring higher clustering accuracy and reduced computational overhead. Unlike conventional subset-based clustering methods, which often suffer from drifting cluster centers and accumulation of errors, the proposed SRSIO-FCM algorithm leverages optimized incremental updates that stabilize convergence and improve efficiency.

Experimental evaluations on four SCADA-based datasets from spiral steel pipe production demonstrated the superiority of SRSIO-FCM compared to SLFCM. The results confirmed that SRSIO-FCM achieves higher F-measure and ARI values, lower objective function values, and significantly shorter execution times across all dataset sizes. In particular, the method achieved near-linear scalability when the number of cluster nodes increased, proving its suitability for large-scale parallel processing in Spark.

Overall, the findings validate that SRSIO-FCM is an effective and scalable solution for clustering massive industrial datasets. Its ability to deliver both accuracy and computational efficiency makes it a promising tool for real-time monitoring, predictive analytics, and process optimization in spiral steel pipe manufacturing environments.

References

1. Kadya, V. (2020). The sustained improvements in e-commerce business through big data and data analytics of Wal-Mart Company.
2. Bakhtiyarov, B., & Jabiyeva, A. (2025). A fuzzy logic-based approach to multiple evaluation factors in big data technology. In *International Conference on Intelligent and Fuzzy Systems*, Springer Nature Switzerland. 354–361.
3. Bharill, N., Tiwari, A., & Malviya, A. (2016). Fuzzy based clustering algorithms to handle big data with implementation on Apache Spark. In *2016 IEEE Second International Conference on Big Data Computing Service and Applications (BigDataService)*, IEEE. 95–104.
4. Kotyrba, M., Volna, E., Jarusek, R., & Smolka, P. (2021). The use of conventional clustering methods combined with SOM to increase efficiency. *Neural Computing and Applications, 33*(23), 16519–16531.
5. Cannon, R. L., Dave, J. V., & Bezdek, J. C. (1986). Efficient implementation of the fuzzy c-means clustering algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 2*(2), 248–255.
6. Qaiyum, S., Aziz, I., Hasan, M. H., Khan, A. I., & Almalawi, A. (2020). Incremental interval type-2 fuzzy clustering of data streams using single pass method. *Sensors, 20*(11), 3210.
7. Wang, Y., Chen, L., & Mei, J.-P. (2014). Incremental fuzzy clustering with multiple medoids for large data. *IEEE Transactions on Fuzzy Systems, 22*(6), 1557–1568.
8. Hashemi, S. E., Gholian-Jouybari, F., & Hajiaghaei-Keshteli, M. (2023). A fuzzy C-means algorithm for optimizing data clustering. *Expert Systems with Applications, 227*, 120377.
9. Hashem, I. A. T., Anuar, N. B., Gani, A., Yaqoob, I., Xia, F., & Khan, S. U. (2016). MapReduce: Review and open challenges. *Scientometrics, 109*(1), 389–422.
10. Salloum, S., Dautov, R., Chen, X., Peng, P. X., & Huang, J. Z. (2016). Big data analytics on Apache Spark. *International Journal of Data Science and Analytics, 1*(3), 145–164.
11. Hosseini, B., & Kiani, K. (2018). A robust distributed big data clustering-based on adaptive density partitioning using Apache Spark. *Symmetry, 10*(8), 342.
12. Jha, P., Tiwari, A., Bharill, N., Ratnaparkhe, M., Mounika, M., & Nagendra, N. (2021). Apache Spark based kernelized fuzzy clustering framework for single nucleotide polymorphism sequence analysis. *Computational Biology and Chemistry, 92*, 107454.
13. Havens, T. C., Bezdek, J. C., & Palaniswami, M. (2010). Incremental kernel fuzzy c-means. In *International Joint Conference on Computational Intelligence*, 3–18.

14. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., Mccauley, M., Franklin, M., Shenker, S., & Stoica, I. (2012). Fast and interactive analytics over Hadoop data with Spark. *USENIX ;login:*, *37*(4), 45–51.
15. Liu, Y., Zhang, X., Liu, B., & Zhao, X. (2022). The research and analysis of efficiency of hardware usage base on HDFS. *Cluster Computing*, *25*(5), 3719–3732.
16. Hussein, A. A. (2020). Using Hadoop technology to overcome big data problems by choosing proposed cost-efficient scheduler algorithm for heterogeneous Hadoop system (BD3). *Journal of Scientific Research and Reports*, *26*(9), 58–84.
17. Mehneh, S. F., Gazalan Toosi, J., & Jalali, M. (2014). An optimized approach for unbalanced big data categorizing using fuzzy clustering. In *2014 International Congress on Technology, Communication and Knowledge (ICTCK)*, 1–4. IEEE.
18. Indirman, M. D. C., Wiriasto, G. W., & Akbar, L. A. S. I. (2023). Distributed machine learning using HDFS and Apache Spark for big data challenges. In *E3S Web of Conferences* (Vol. 465, 02058. EDP Sciences.
19. Christen, P., Hand, D. J., & Kirielle, N. (2023). A review of the F-measure: Its history, properties, criticism, and alternatives. *ACM Computing Surveys*, *56*(3), 1–24.
20. Warrens, M. J., & Van Der Hoef, H. (2022). Understanding the adjusted Rand index and other partition comparison indices based on counting object pairs. *Journal of Classification*, *39*(3), 487–509.

Received: 12.05.2025
Accepted: 03.09.2025