Computing and Informatics, Vol. 44, 2025, 1178–1201, doi: 10.31577/cai_2025_5_1178

EFFICIENT DISTRIBUTED CLUSTERING WITH CUCKOO SEARCH ALGORITHM AND GPU ACCELERATION FOR BIG DATA ANALYSIS

Hadjir Zemmouri, Said Labed, Akram Kout

MISC Laboratory, University of Constantine 2 Abdelhamid Mehri Ali Mendjeli, 25000, Constantine, Algeria
e-mail: {zemmouri.hadjir, said.labed, akram.kout}@univ-constantine2.dz

El-Bay Bourennane

ImVia Laboratory, University of Bourgogne 9, Avenue Alain Savary, 21078 Dijon, France
e-mail: ebourenn@u-bourgogne.fr

Abstract. Clustering analysis is a crucial method in data mining, aimed at identifying clusters of data objects in the attribute space. Distributed clustering has gained prominence due to the emergence of Big Data. The rapid growth of data, particularly with the advent of technologies, such as the Internet of Things and 5G, has resulted in numerous challenges for data analysis and processing. Traditional clustering methods, such as K-means and hierarchical clustering, struggle with extensive datasets designed for smaller to moderately sized datasets. Meta-heuristic techniques have garnered significant attention among the various distributed clustering algorithms due to their ability to deliver high-quality solutions across a wide range of optimization problems. In this study, we proposed a new Cuckoo search (CS) clustering algorithm for distributed clustering to address the challenges of Big Data clustering. First, the CS clustering algorithm is executed on each local site, utilizing GPU acceleration for efficient local data clustering. Second, on a global scale, representative data from each site are aggregated and processed worldwide, with centroids iteratively updated to generate the final clustering result. We have significantly enhanced the processing efficiency by minimizing transmission costs and eliminating the need for inter-node communication. Furthermore, our approach

demonstrates adaptability in handling large datasets with competitive execution times through the utilization of parallel processing and distributed computing. Our approach demonstrates both efficiency and scalability across wide range of datasets, highlighting its potential for various applications.

Keywords: Clustering, Big Data, Cuckoo search algorithm, distributed computing, GPU

Mathematics Subject Classification 2010: 68Txx

1 INTRODUCTION

In the era of information technology, the generation, gathering, and storage of data in different sites have experienced unprecedented exponential growth, resulting in the advent of massive datasets [1]. Clustering, one of the most important tasks in data mining and machine learning, plays a pivotal role in uncovering meaningful patterns, structures, and insights within these extensive datasets. The intrinsic complexity and sheer volume of Big Data have resulted in numerous challenges for data analysis and processing in terms of scalability, computational efficiency, and memory requirements [2]. Consequently, management of these data in a centralized manner becomes infeasible due to privacy and transmission costs [3]. The development of novel algorithms and techniques capable of effectively handling these huge amounts of data becomes crucial. Traditional clustering techniques, such as K-means and hierarchical clustering, are frequently unable to cope with extensive datasets because they were designed for relatively small to moderately sized datasets. Consequently, distributed clustering algorithms have received increasing interest [2]. These algorithms take advantage of distributed computing systems to enhance the scalability and speed up the clustering. Distributed clustering techniques distribute data across multiple computational nodes, facilitating parallel processing and accelerating the analysis of large datasets [4]. Metaheuristic techniques have attracted significant interest among the plethora of distributed clustering algorithms because of their capacity to provide high-quality solutions in a wide range of optimization problems [5]. Bio-inspired algorithm-based clustering approaches regard clustering as an optimization problem (i.e., to find a set of points as cluster centers to optimize a certain similarity measure) [6]. Bio-inspired algorithms, such as particle swarm optimization (PSO) [7], have been introduced as an effective approach to address complex data clustering problems [6]. The Cuckoo search (CS) algorithm (CSA), inspired by the breeding behavior of cuckoo birds, stands out as a metaheuristic algorithm renowned for its efficiency in addressing complex optimization problems [8]. The simplicity, adaptability, and escape from local optima exhibited by CSA make it well-suited for clustering tasks. However, CSA's applicability must be expanded to a distributed computing environment to properly utilize its capa-

bilities for large data clustering. This work introduces a novel approach, referred to as the CS clustering algorithm (CSCA) for distributed clustering (CSCADC), specifically designed to address the particular difficulties associated with massive data clustering in distributed computing settings. The primary objective of this work is to provide a scalable and efficient method for leveraging the parallel processing power of distributed systems to cluster large datasets. Overall, the goal of this research is to contribute to the existing knowledge in the field of distributed clustering for large-scale data by introducing a novel algorithmic framework that leverages the principles of the CSA while addressing the requirements of distributed computing environments.

The findings of this study are expected to enhance our comprehension of scalable clustering methods for big data, thereby accelerating the speed and efficiency of clustering algorithms. The remainder of this paper is structured as follows: Section 2 provides an overview of some related work. Section 3 introduces a brief look about scaling platforms of Big Data. Section 4 outlines the methodology used. Section 5 presents the proposed approach. Section 6 presents findings that illustrate the benefits of our approach. Lastly, we will draw conclusions from this paper. Section 7 discusses future directions.

2 RELATED WORK

The exponential growth of data in the modern digital age has brought about significant challenges in data analysis, particularly when dealing with large and complex datasets, commonly known as "Big Data" [1]. Clustering is a key technique in data mining and machine learning that aims to discover inherent structures and patterns in data [9]. Clustering algorithms have been used in various fields, including web analysis, image processing, marketing, medical diagnostics, data science, and Internet of Things (IoT) [1]. Accordingly, clustering algorithms must be improved. A clustering problem is an NP-hard problem because of its complexity [10]. Consequently, traditional clustering algorithms typically require an extended period of time to find an approximate solution and commonly encounter difficulties in handling the scale and complexity of Big Data [11]. Hence, research efforts are aimed at discovering efficient clustering solutions capable of handling massive data clustering within a reasonable time. Given that clustering finds applications across various domains, it has received significant attention from researchers who are making notable advancements in the field. The K-means algorithm [12] is the most well-known and commonly used clustering technique due to its simplicity and high efficiency [6]. A drawback of the popular K-means algorithm is its requirement to predefine the number of centroids. Furthermore, the effectiveness of the K-means algorithm diminishes as the complexity of the dataset increases. Accordingly, numerous improved K-means algorithms have been developed in the literature to deal with large datasets. Cuomo et al. [13] introduced three distinct parallel implementations of the K-means clustering algorithm, specifically designed to manage extensive

datasets. These implementations leverage graphics processing units (GPUs), a parallel architecture utilized to minimize the execution time of K-means clustering. The optimization addresses space limitations on GPUs and minimizes host-device data transfer time.

Clustering using evolutionary algorithms is a compelling approach to data analysis, particularly in scenarios where traditional clustering methods may face limitations [1]. Evolutionary algorithms, inspired by the process of natural selection, utilize the principles of population-based optimization and genetic variation to address complex data clustering problems [5]. Evolutionary algorithms view clustering as an optimization problem to discover high-quality clustering solutions based on a fitness function. The adaptability, ability to explore complex solution spaces, and versatility of evolutionary algorithms make them valuable tools in the data analysis toolkit for addressing a wide range of clustering challenges [14]. Reference [15] presented the challenge of clustering large datasets in a distributed environment and proposed a novel two-phase algorithm. This approach combines a parallel genetic algorithm (GA) with Mahalanobis distance in the first phase and refines the output using K-means in the second phase. Benmounah et al. [16] addressed big data clustering challenges by introducing a decentralized distributed solution using swarm intelligence algorithms within the MapReduce framework. The suggested algorithm combines a migration strategy with three well-known algorithms, namely, ant colony optimization [17], artificial bee colony [18], and PSO [7], to determine the optimal partition that improves the clustering quality. Reference [6] explored evolutionary computation for distributed clustering in IoT. It introduced distributed PSO using distributed computing to address the performance challenges in scenarios with partial data at each site. The method involves local data clustering using PSO at each site and integration of local results using K-means at the global site. Reference [19] introduced an enhanced PSO algorithm, multistart pattern reduction-enhanced PSO (MPREPSO), designed to improve the clustering efficiency with big data. The method aims to minimize the computational time through the compression of static patterns by incorporating pattern reduction and multistart operators and enhance the diversity in the population to avoid local optima. Reference [20] proposed the distributed-parallel PSO with K-means (D-PPSOK) clustering algorithm, which is designed to handle big data analysis by using data sampling techniques. The D-PPSOK algorithm is based on the idea that a smaller subset of the data can hold the same information as the whole dataset, allowing for an efficient processing of big data. Moreover, the D-PPSOK algorithm has been shown to be particularly effective in applications, such as document clustering. Reference [21] developed a highly efficient and optimal data clustering scheme using the FSF-Sparse FCM + PWO-based MRF for high-dimensional data. The FSF-Sparse FCM is designed by integrating SFO, fractional concept, and Sparse FCM. The MRF has two functions: mapper and reducer. FSF-Sparse FCM computes the cluster centroids in the mapper phase, generating intermediate data. PWO refines the data in the reducer phase. Han [22] proposed a big data clustering algorithm based on group intelligence utilizing the K-means clustering mining algorithm. This approach involves analyzing the data

storage capacity of the cloud computing Hadoop framework and the MapReduce computing model. The traditional K-means clustering algorithm of data mining is improved by adopting a group intelligence algorithm to address the local optimization issue. The improved ant colony clustering algorithm has been applied to the location of sports facilities, achieving good results. Reference [23] proposed an efficient and flexible distributed clustering framework. Tarkhaneh et al. [24] introduced HCSPSO, a hybrid algorithm for data clustering that combines CS, PSO, and K-means. The proposed algorithm addresses the high functional evaluation issue in standard CS utilizing PSO and K-means within CS, along with Mantegna Lévy distribution for faster convergence and local search. Tekieh and Beheshti [25] proposed a distributed fuzzy clustering approach using MapReduce to handle large-scale data with improved privacy-preserving methods based on Grasshopper optimization algorithm.

Clustering using evolutionary algorithms has been proven successfully addressing numerous big data problems [10]. However, the challenges related to computation time and transmission cost can significantly affect the performance of any system. Considering the above-mentioned issues, this work aims to address the slow computational speeds associated with clustering techniques, especially when dealing with large datasets, by proposing a new distributed approach based on the CS algorithm, aiming to accelerate the clustering process and provide efficient solutions.

3 SCALING STRATEGIES FOR BIG DATA CLUSTERING PLATFORMS

Big Data clustering, a crucial component of data analytics, relies on robust platforms that can efficiently handle the immense volume and complexity of large datasets [19]. Among the various strategies utilized, horizontal and vertical scaling stand out as key paradigms in optimizing the performance of clustering algorithms [11] as illustrated in Figure 1.

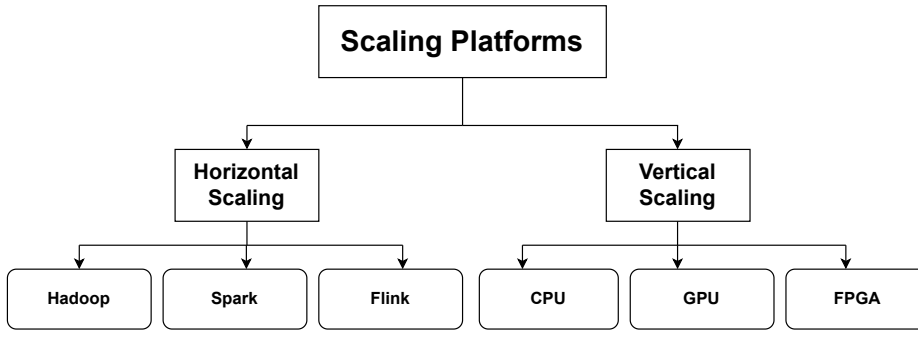

Figure 1. Scaling platforms for Big Data

Horizontal scaling, also known as scaling out, involves distributing computational tasks across multiple nodes or machines. This approach enables parallel processing and faster analysis by allowing simultaneous data processing across a cluster of interconnected machines. Technologies such as Apache Hadoop [26], Spark [27], and Flink [28] exemplify horizontal scaling in distributed computing environments, providing scalable solutions for clustering large datasets.

Vertical scaling, or scaling up, enhances the computational capacity of an individual machine by adding resources, such as CPU, RAM, and storage. This approach focuses on augmenting the capabilities of a single machine to accommodate larger datasets and complex computations. This technique is suitable for scenarios where the dataset partitioning is challenging or when specific algorithms require substantial computational resources [11].

Horizontal and vertical scaling are crucial for Big Data clustering infrastructure. The selection between these strategies depends on dataset characteristics, clustering algorithm, and available resources [29]. The flexibility of these strategies is vital in creating scalable platforms to address large dataset clustering challenges in Big Data analytics [11].

4 METHODOLOGY

In this section, we first introduce the notations used throughout this work, the clustering task, and the validation indexes for clustering. Thereafter, we present the concept of the CS clustering algorithm.

4.1 Clustering Task

Let $X = \{X_1, X_2, \ldots, X_n\}$ be a dataset with n data objects where X_i $(1 < i < n)$ be a data object described by m attributes $X_{i1}, X_{i2}, \ldots, X_{im}$. Clustering involves identifying inherent patterns, structures, or similarities within a dataset without having any prior knowledge of the group labels. The main objective of clustering is to group the objects with the most similarities into one cluster $P = \{C_1, C_2, \ldots, C_K\}$. The constraints mentioned here are applied to clustering: $\forall j : C_j \ (1 < j < K) \neq \emptyset$; $\forall i \neq j : C_i \cap C_j = \emptyset$, and $\sum_{i=1}^{K} C_i = X$. Each object in the dataset is defined by a set of attributes or measurements. Distance measuring techniques can be employed to determine the similarity or dissimilarity of data. Several measures of similarity are provided in the literature, such as Euclidean distance, Manhattan distance [30], Minkowski distance [30]. The Euclidean distance is used in this research to calculate the similarity between two points in the multidimensional space. Euclidean distance is the most popular distance calculated by Equation (1) where X_{is} is the value of

the attribute number s of the object number i (x_i).

$$\text{Distance}(X_i, X_j) = \sqrt{\sum_{s=1}^{m}(X_{is} - X_{js})^2}. \tag{1}$$

4.2 Validation Indexes for Clustering

Clustering evaluation relies on performance metrics, ensuring high similarity within clusters and dissimilarity between them. Various metrics are utilized to assess cluster separation and cohesion, providing valuable insights into clustering quality [2]. Below are some performance metrics that will be used later in this work, alongside others.

- The distances between clusters, specifically the distances between their centroids, are referred to as **inter-cluster distances**. In this context, the goal is to maximize the separation between clusters, and this objective is quantified as Equation (2)

$$\text{Inter-cluster} = \min \|C_i - C_j\|^2 \tag{2}$$

- **Intra-cluster** distances pertain to the distances between individual data vectors within a given cluster. The primary objective is to minimize these intra-cluster distances. This optimization goal is mathematically represented as Equation (3).

$$\text{Intra-cluster} = \frac{1}{n}\sum_{j=1}^{K} \|X_j - C_j\|^2. \tag{3}$$

- **Dunn index** [31]: This index compares the minimum inter-cluster distance relative to the maximum intra-cluster distance. A higher Dunn index means better cluster separation. Dunn index is calculated using Equation (4)

$$\text{IndexDN} = \frac{\min(\text{Inter-cluster})}{\max(\text{Intra-cluster})}. \tag{4}$$

- **Silhouette coefficient**: This metric is utilized to evaluate cluster compactness and separation by averaging dissimilarities within clusters and between neighboring clusters and calculated using Equation (5).

$$\text{Silhouette}(X_i) = \frac{b(X_i) - a(X_i)}{\max\{b(X_i), a(X_i)\}}, \tag{5}$$

where $a(X_i)$ is the average distance from the i^{th} data point to the other data points in the same cluster, and $b(X_i)$ is the smallest average distance from the i^{th} data point to data points in a different cluster, minimized over clusters. The overall silhouette coefficient for the entire dataset is the average of the individual silhouette coefficients for each data point.

- **Sum of squared errors (SSE)**: SSE is a metric used in centroid-based clustering algorithms to assess performance.This metric calculates the sum of squared distances between each data point and the centroid of its assigned cluster. The objective is to minimize this metric during clustering, as defined in Equation (6).

$$\text{SSE} = \sum_{k=1}^{K} \sum_{i=1}^{n} (X_{ij}, z_j), \tag{6}$$

where z_j is the mean (centroid) of the j^{th} feature across all data points in the assigned cluster.

4.3 CSA

The CSA, introduced by [8] in 2009, draws inspiration from the brood parasitism behavior of certain cuckoo bird species. In this analogy, cuckoo birds represent solutions to optimization problems. Meanwhile, host bird nests symbolize the search space. The algorithm iteratively lays eggs in randomly selected nests, selecting the best nests (with the highest quality) for the next iteration based on fitness values. CSA has been recognized for its simplicity, ease of implementation, and ability to handle continuous and discrete optimization problems. This mechanism utilizes Lévy flights and random walks to effectively balance exploitation and exploration. Moreover, CSA requires fewer parameters compared with other bio-inspired algorithms. The algorithm's process is summarized in Algorithm 1.

Algorithm 1 CSA

```
Objective function f(x) = (x_1, ..., x_d)^T;
Initialize a population of n hosts, x_i (i = 1, 2, ..., n);
while (the stop criterion is not met) do
    Randomly get a cuckoo x_k by Lévy flights;
    Evaluate its fitness (quality) F_k;
    Randomly choose a nest among n (let us say j) ;
    if (F_k > F_j) then
        Replace j by the new solution generated;
    end if
    Abandon a fraction p_α of worst nests and build new ones by Lévy flights;
    Keep the best solutions;
    Rank the solutions and find the current best;
end while
```

4.4 CSCA

The CSCA is based on the concept of brood parasitism observed in cuckoo birds [32]. This algorithm aims to group a set of input samples (data points) into clusters

with similar features in an unsupervised way. In the context of optimization and clustering, each cuckoo represents a potential solution or cluster configuration, while each egg laid by a cuckoo corresponds to a new potential solution. The ability of CSCA to balance exploration and exploitation makes it a promising tool for addressing complex and high-dimensional data analysis tasks [9]. The following is a high-level overview of the CSCA.

1. Randomly initialize a population of candidate solutions (cuckoo nests).
2. Assign each data point to the cluster with the lowest distance (or highest similarity).
3. While the stopping criterion is not met:
   - Randomly generate a new solution (cuckoo egg).
   - Replace a randomly selected solution in the population with the new solution if the new one has better fitness.
   - Abandon a fraction of the worst solutions in the population, and replace them with new ones (cuckoo eggs) randomly generated.
   - Reassign each data point to the cluster with the lowest distance (or highest similarity) based on the current clustering solution.
4. Return the best solution found.

5 PROPOSED METHOD

This section introduces the CSCADC. In the proposed algorithm, the system comprises a central global site and N local sites, each denoted as L_i; $\forall i \in \{0, 1, \ldots, N\}$. Here, L_0 represents the global site, while the remaining are local sites. The distributed clustering methodology involves the partitioning of the dataset into distinct subsets independently processed across distributed nodes. Consequently, each node L_i manages a portion of the dataset denoted as z_i, where $Z = z_1 \cup z_2, \ldots, \cup z_N$. This data distribution is instrumental in reducing time and spatial complexity, thereby significantly decreasing the computational time required for clustering extensive datasets in a parallel way. The parameter N, representing the number of local sites, is adaptable based on user preferences and machine availability. Each node is responsible for clustering its own data subset, iteratively transmitting the outcomes to the global site using the CSCA. We assume a predefined number of clusters centroids denoted as K in the clustering process $P = \{C_1, C_2, \ldots, C_K\}$. Subsequently, each node handles the clustering of its respective data subset to achieve K predefined clusters. The iterative transmission of the intermediate result (matrix T_i) from each node to the global site ensures convergence and facilitates the generation of the final clustering result, thereby enhancing the performance of our approach by minimizing transmission cost.

The T matrix comprises $(K * m + 1)$ columns, where m is the number of features and y_i is the value of the feature number i. In this matrix, each row signifies

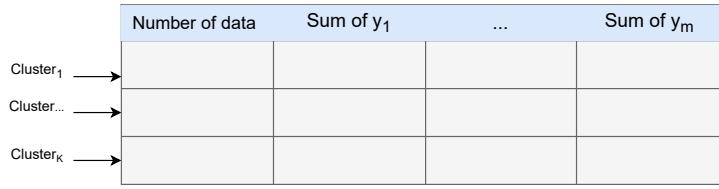

Figure 2. Matrix T representation

the count of the data objects assigned to each cluster and the summation of their attributes. The structural layout of the matrix T is illustrated in Figure 2. A visual representation of the iterative process is demonstrated by Figure 3.

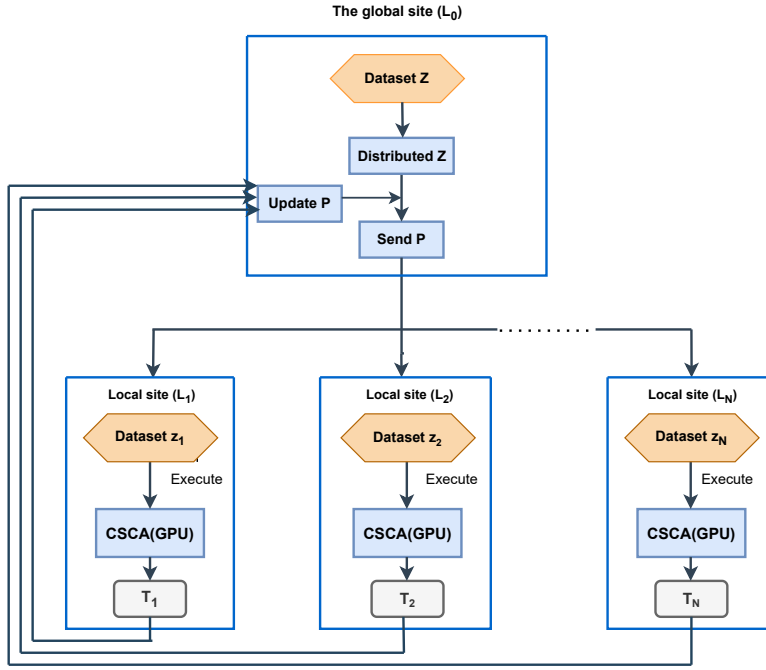

Figure 3. Flowchart of the CSCADC

5.1 Local Process

The primary objective at each node is to independently cluster the local dataset and transmit the results to the global site. This process necessitates the iterative application of the CSCA at each site in a parallel fashion. In the CS process, an egg in a nest serves as a candidate solution. Each nest accommodates a single egg, and

every cuckoo lays one egg at a time. Accordingly, a cuckoo and its nest collectively represent a candidate solution in our algorithm, where the terms nest, cuckoo, and egg are interchangeably used. In centroid-based clustering algorithms, the final results hinge on the identified cluster centers. The clustering task is comparable to searching for optimal cluster centers, and the set of these centers constitutes a candidate solution. Suppose a dataset has K clusters, the set of cluster centers $x_i = \{x_{i1}, x_{i2}, \ldots, x_{iK}\}$ represents the candidate solution where $1 < i < population_{size}$).

The clustering outcome from each node is represented as matrix T_i. The clustering results are compactly stored in a matrix format instead of sending all data to optimize communication costs. The local data clustering independently operates at each node without necessitating inter-node communication, contributing to the efficiency of the processing. The implementation of this iterative procedure leverages GPUs, a parallel architecture embedded in each node, thereby mitigating the execution time of the CSCA for handling extensive datasets. The objective function f is defined as $f(x)$; $x = \min \mathrm{SSE}(x_1, \ldots, x_{population_{size}})$ where n is the number of nests defined as Equation (6). The core steps of the local processing are elucidated in Algorithm 2.

Algorithm 2 Local process algorithm

Require: Local dataset z, list P
Ensure: Matrix T of the clustering result, result of clustering

Randomly initialize the population of n host nests containing one egg. A solution $\mathbf{x}_i$ includes K randomly selected cluster centroids (corresponding to K clusters)
Update the first nest generated by the received P list
Evaluate the population and pick up the best nest $\mathbf{x}_{\text{best}}$ from the population using Equation (6)
while the stop criterion is not met **do**
    Get a cuckoo $\mathbf{x}_i$ in a random manner using Equation (7)
    Randomly pick up a nest $\mathbf{x}_j$ from the population
    if $(F_i < F_j)$ **then**
        Replace the nest $\mathbf{x}_j$ with the new nest $\mathbf{x}_i$
    end if
    Abandon a fraction (p_{roa}) of the worse nests, and generate new ones using Equation (7).
    Evaluate the fitness of the new nests and update the best nest $\mathbf{x}_{\text{best}}$
    Rank the solutions
end while
Calculate matrix T

The generation of a new set of clusters, constituting the updated solution set, is accomplished by employing the current best solution from the ongoing iteration, as detailed as Equation (7). step denotes the step length of a Lévy flight, conforming to a Lévy distribution. Mantega's algorithm is utilized to derive the step length of the Lévy flight, following the suggestion of Yang [33]. In Mantega's algorithm, the

calculation of the step is performed using Equation (8), wherein the fixed value of β is set to 1.5.

$$x_{\text{new}} = x + \text{randn} \times 0.01 \times \text{step} \times (x - x_{\text{best}}), \tag{7}$$

where

$$\text{Step} = \frac{u}{|v|^{1/\beta}}, \tag{8}$$

and

$$u = v * \sigma, \tag{9}$$

where v is a matrix of the normally distributed random numbers with the same dimension of a solution as follows: $v \sim N(0, \sigma_v^2), \sigma_v = 1$, where:

$$\sigma = \left(\frac{\Gamma(1+\beta) \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \beta 2^{\frac{(\beta-1)}{2}}} \right)^{\frac{1}{\beta}}, \tag{10}$$

where Γ is the gamma function as follows:

$$\Gamma(1+\beta) = \int_0^{+\infty} t^{\beta} \, \mathrm{e}^{-t} \, \mathrm{d}t. \tag{11}$$

5.2 Global Process

Upon distributing the complete dataset across all nodes, we randomly select K centroids from the initial dataset and disseminate them to all sites. Subsequently, the CSCA is independently executed on each node in an iterative manner, and the centroids sent to the local sites are iteratively updated using the received matrices T. The update of each centroid is achieved by calculating the mean of its assigned objects using Equation (12).

$$C_j = \frac{1}{n} \sum_{i=1}^{n} O_i, \tag{12}$$

where n is the number of data point that belongs to cluster C_j and O_i is the i^{th} data point of cluster C_j.

Consequently, the global node is responsible for aggregating the results obtained from the local sites, facilitating the update of centroids, and ultimately delivering the global clustering result. When these centroids no longer change or the maximum number of iterations is reached, the algorithm is stopped. Algorithm 3 outlines the fundamental steps of the global process.

Algorithm 3 Global process

Require: Dataset, K number of clusters, N number of local sites
Ensure: Final clustering result
1: Distribute the data among N nodes
2: Randomly pick K centroids from the initial dataset and send them to all sites
3: **while** (the stop criterion is not met) **do**
4: Execute the CSCA on each node
5: Get matrices $T_1, T_2, \ldots, T_N$.
6: Update the centroids using $T_1, T_2, \ldots, T_N$ as Equation (12) and send them to all sites.
7: **end while**

5.3 Reliability and Fault Tolerance

Ensuring the reliability of distributed clustering algorithms is critical, especially in large-scale and decentralized environments. To address potential reliability issues, especially concerning the failure of the global site, we have integrated a Backup Central Nodes Algorithm within our distributed clustering approach, as elucidated in Algorithm 4.

This algorithm maintains a primary global node (PGN) and one or more backup nodes (local nodes) that take over when the primary fails. The global node ID is initiated as PRIMARY_NODE (node_id == PRIMARY_NODE).

Algorithm 4 Backup Central Nodes Algorithm

1: **Response_time** = 1 s
2: **if** (node_id == PRIMARY_NODE) **then**
3: **while** True **do**
4: Send heartbeat (signal) to all backup nodes
5: Wait(Response_time)
6: **end while**
7: **else**
8: **while** True **do**
9: **if** (not receive_heartbeat_from_PGN) **then**
10: Consider the global node failed
11: new_PGN = the backup node with the highest priority (Node ID, Availability, etc.)
12: The new_PGN broadcasts its status to all nodes, informing them of the change
13: **end if**
14: **end while**
15: **end if**

When the original PGN recovers, it can either remain a backup node or regain its role as PGN if it is needed. By incorporating this fault-tolerant mechanism, our

approach is robust against single points of failure, enhancing both reliability and resilience in distributed environments.

6 EXPERIMENTS

In this section, we assess the effectiveness of the proposed approach through a comparative analysis with the K-means algorithm [12], DKmeans [34], iCSPM [35], and DPSO [6] utilizing real datasets. The evaluation of CSCADC is conducted based on performance metrics, including the sum of intra-cluster distances, inter-cluster distance, silhouette value, Dunn index, and SSE value.

6.1 Datasets

In this study, we assess the effectiveness of our proposed algorithm using standard datasets, sourced from the UCI dataset [36]. Additionally, we incorporate a supermarket consumer behavior dataset [37] in our evaluation. The datasets utilized in this investigation are presented in Table 1.

Dataset	Number of Samples	Number of Features	Number of Classes
Iris	150	4	3
Breast cancer	683	9	2
CMC	1 473	9	3
Yeast	1 485	8	5
Supermarket consumer behavior	2 019 501	12	5

Table 1. Properties of the datasets used in our experiments

The Iris dataset contains features of iris flowers, and it is commonly used for classification tasks. By contrast, the breast cancer dataset distinguishes between malignant and benign tumors. The CMC dataset records socio-economic factors and contraceptive choices of married women in Bangladesh. The yeast dataset provides gene expression levels under various conditions valuable for clustering algorithms in gene expression analysis. Finally, in the supermarket consumer behavior dataset, businesses may benefit from cluster marketing by identifying customers who share similar needs or respond similarly to particular marketing initiatives. All previous datasets are ready for use except the last one, which requires preprocessing before utilization.

6.2 Pre-Processing of the Supermarket Consumer Behavior

In the supermarket consumer behavior dataset, we attempt to group the clients into separate classes according to their shopping behaviors and preferences using unsupervised machine learning techniques. The supermarket dataset contains over

2 million purchase records and 12 variables, as shown in Table 1. After studying the correlation between variables [38], the following columns were removed because they have no effect: order_id, user_id, order_number, product_id, department_id, and department. After removing irrelevant columns and applying label encoding for categorical data and MinMax normalization [39], the dataset is prepared for analysis. The optimal number of clusters K is determined by the Elbow method [40] in this study. The dataset is tested with the default K-means clustering of MATLAB using a range of values for K (between one and nine). Figure 4 shows that the inertia score begins to drastically drop between four and five clusters, resulting in the selection of five clusters for further analysis.

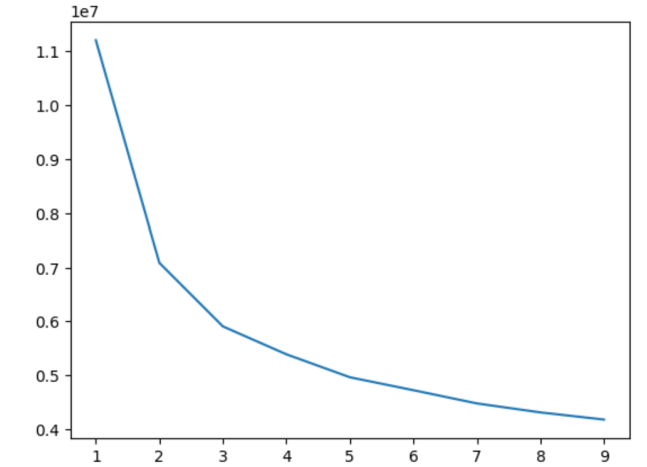

Figure 4. Inertia plot per K

6.3 Results and Discussions

The experiment involved running the algorithm 20 times with random initial solutions on each dataset. A population size of 50 and a maximum of 30 iterations were set for the CSA. Parameters, such as p_{roa} (discovery probability) = 0.7 and β (Lévy flight parameter) = 1.9, were optimized to enhance the algorithm's accuracy and convergence speed, and they were used during the local clustering phase at each node where CSCA (Algorithm 2, line 10) iteratively updates centroids before sending representative results to the global site. Proper tuning of p_{roa} and β improves clustering quality, convergence stability, and overall performance of the distributed framework. The number of functional evaluations served as a fair measure for comparing algorithms [1]. MATLAB's parallel computing framework was leveraged to exploit multi-core processors and distributed computing resources, with each worker representing a local site for executing the CSCA algorithm. The experiment was

conducted using MATLAB 2017b on a computer with Intel Core i9, 2.40 GHz, 32 GB RAM, and an NVIDIA GFORCE GTX 1080 GPU.

The study recorded results from multiple runs and selected the best values. Tables 2, 3, 4, 5 and 6 present the clustering results obtained from different algorithms including standard k-means, distributed k-means, CSCA, iCSPM, and distributed PSO clustering algorithms. CSCADC demonstrated significant improvements, particularly evident in the Iris dataset, where SSE was lower at 96.8544, and the intra-cluster values were minimized to 0.6457. In the Cancer dataset, CSCADC achieved an SSE value of 1.52e+5, outperforming other algorithms, along with superior index DN and intra-cluster values. The results from our approach consistently showed better performance metrics, including silhouette, SSE, index DN, and intra-cluster values, across various datasets, such as the supermarket behavior dataset and Yeast dataset.

The scalability of the proposed algorithm was also assessed, demonstrating superior efficiency compared with other algorithms, particularly evident in Figures 5 and 6. Meanwhile, smaller datasets, such as Iris and Cancer, exhibited better performance with standard k-means due to the lower communication overhead. CSCADC showcased enhanced efficiency as the dataset sizes increased, highlighting its suitability for handling massive datasets. The benefits of the parallel processing and distributed nature of CSCADC become more pronounced, offsetting the communication overhead (Figure 6). These observations underscore the adaptive nature of the proposed algorithm, demonstrating increased efficiency with the increase in the dataset size. This result highlights the algorithm's scalability and effectiveness, particularly in handling substantial datasets where the advantages of parallel processing can be fully realized. Figure 7 illustrates a significant decrease in execution time with the integration of GPUs at each local site. The utilization of GPUs demonstrates a clear trend of reducing execution time, aligning with the known benefits of GPU-accelerated computations in enhancing computational efficiency. This observation highlights the crucial role of GPU integration in expediting task execution within the experimental setup.

	Silhouette	SSE	IndexDN	Intra-class	Inter-class	Time
Kmeans	0.7357	97.2046	0.0988	0.6480	1.7972	0.85 s
DKmeans	0.7357	97.2046	0.0988	0.6480	1.7972	1.22 s
DPSO	0.6723	98.7511	0.0789	0.6853	1.2433	1.23 s
iCSPM	0.7257	97.9734	0.0988	0.6532	1.7178	1.23 s
CSCADC	0.7357	96.8548	0.0988	0.6457	1.7873	0.98 s

Table 2. Iris dataset results

Our approach integrates vertical and horizontal scaling, distributing data across machines and utilizing GPUs at each node. This hybrid framework optimizes efficiency by enhancing individual machine capabilities and facilitating parallel processing. GPU acceleration further boosts performance, demonstrating the effectiveness

	Silhouette	SSE	IndexDN	Intra-class	Inter-class	Time
Kmeans	0.8343	1.5264e+5	0,0173	2.6827e+2	1.3313e+3	0.77 s
DKmeans	0.8343	1.5264e+5	0,0173	2.6827e+2	1.3313e+3	1.12 s
DPSO	0.6699	2.7881e+5	0.0010	1.768e+2	2.9765e+2	1.10 s
iCSPM	0.8291	1.5248e+5	0.0156	2.6721e+2	1.2634e+03	1.53 s
CSCADC	0.8277	1.5200e+5	0.0196	2.6712e+2	1.3124e+03	0.87 s

Table 3. Cancer Breast dataset results

	Silhouette	SSE	IndexDN	Intra-class	Inter-class	Time
Kmeans	0.6420	5.5422e+3	0.0538	3.7625	9.4835	0.61 s
DKmeans	0.6420	5.5422e+3	0.0538	3.7625	9.4835	0.88 s
DPSO	0.5420	1.4324e+4	0.0456	3.7717	9.5623	0.42 s
iCSPM	0.6437	5.5452e+3	0.0538	3.7646	9.5950	1.12 s
CSCADC	0.6470	5.5420e+3	0.0538	3.7624	9.6749	0.32 s

Table 4. CMC dataset results

of our hybrid scaling strategy in achieving scalability and computational efficiency, as evidenced in previous results.

7 CONCLUSIONS AND FUTURE WORK

In this study, we presented a novel approach, CSCADC, designed to address the challenges posed by Big Data clustering. A distinctive feature of CSCADC lies in its two-step process. First, CSCA is independently and iteratively executed on local data at each local site, effectively utilizing distributed computing to cluster local datasets. Second, the representative data from each local site is aggregated and processed at the global site, where the centroids are iteratively updated to generate the final clustering result. We have significantly enhanced the processing efficiency by the minimizing transmission costs and eliminating the need for inter-node communication. The backup central nodes algorithm ensure that the proposed algorithm is both resilient and robust, providing reliable performance even if the global node fails.

The algorithm showcases the adaptability to handle substantial datasets while maintaining competitive execution times using parallel processing and distributed computing capabilities. Our proposed algorithm is designed with scalability in mind.

	Silhouette	SSE	IndexDN	Intra-class	Inter-class	Time
Kmeans	0.3204	271.3357	0.0988	0.1828	0.1956	0.86 s
DKmeans	0.3190	270.1557	0.0988	0.1832	0.1913	1.23 s
DPSO	0.2978	268.2343	0.0789	0.1934	0.1765	2.93 s
iCSPM	0.3200	271.2343	0.0988	0.1832	0.1912	7.23 s
CSCADC	0.3191	271.3355	0.0988	0.1828	0.1962	0.23 s

Table 5. Yeast dataset results

	Silhouette	SSE	IndexDN	Intra-class	Inter-class	Time
Kmeans	0.7420	8.0695e+5	0.077	0.4258	0.6086	12.053 s
DKmeans	0.7422	8.0471e+5	0.078	0.4246	0.6164	6.228 s
DPSO	0.6345	1.7623e+6	0.062	0.6634	0.6321	7.63 s
iCSPM	0.7645	8.2748e+5	0.078	0.4266	0.6266	8.91 s
CSCADC	0.7650	8.1003e+5	0.078	0.4266	0.6362	3.90 s

Table 6. Supermarket behavior dataset results

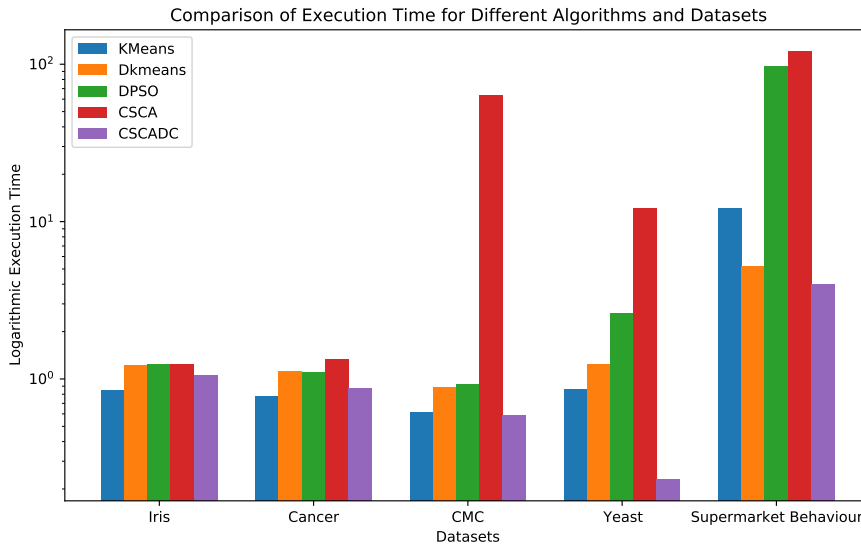

Figure 5. Comparison of the execution time for the different algorithms

It leverages distributed computing, which inherently supports the processing of larger datasets by spreading the workload across multiple machines. By combining vertical and horizontal scaling, our method efficiently utilizes computational resources. Vertical scaling makes a single machine better, while horizontal scaling enables parallel processing across multiple nodes. This flexibility makes the algorithm suitable to dataset from real world applications with higher complexity and data size. We also discussed that our proposition gives advantageous results and we shown it especially when the dataset was increased.

The results of the thorough evaluation of the diverse datasets indicated that CSCADC demonstrates remarkable efficiency and scalability, particularly with the increase in dataset sizes. The context-dependent nature of the algorithm's performance is evident, with competitiveness observed in scenarios involving larger datasets. However, in smaller datasets, traditional algorithms, such as k-means, may be outperformed due to reduced communication overhead. Future work may focus on further refining communication strategies and exploring the algorithm's

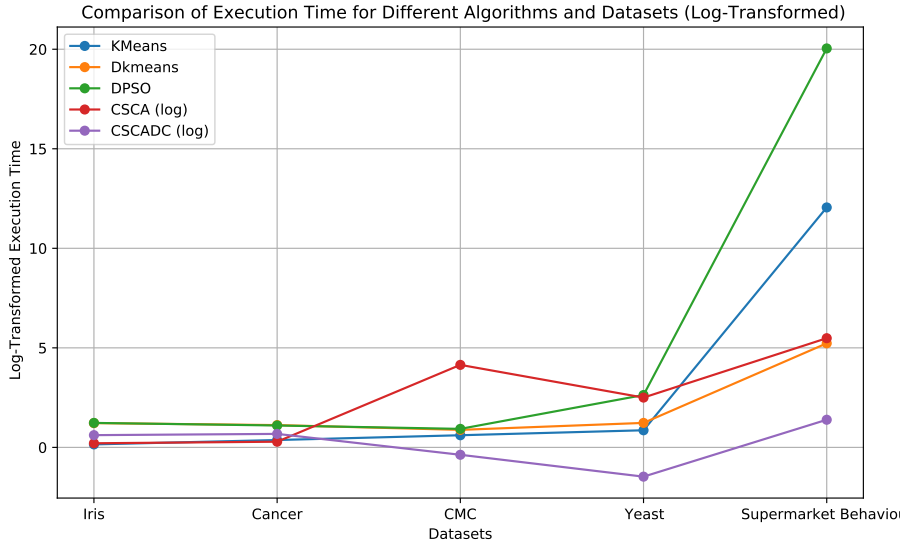

Figure 6. Comparison of the execution time for the different algorithms and datasets (log-transformed)

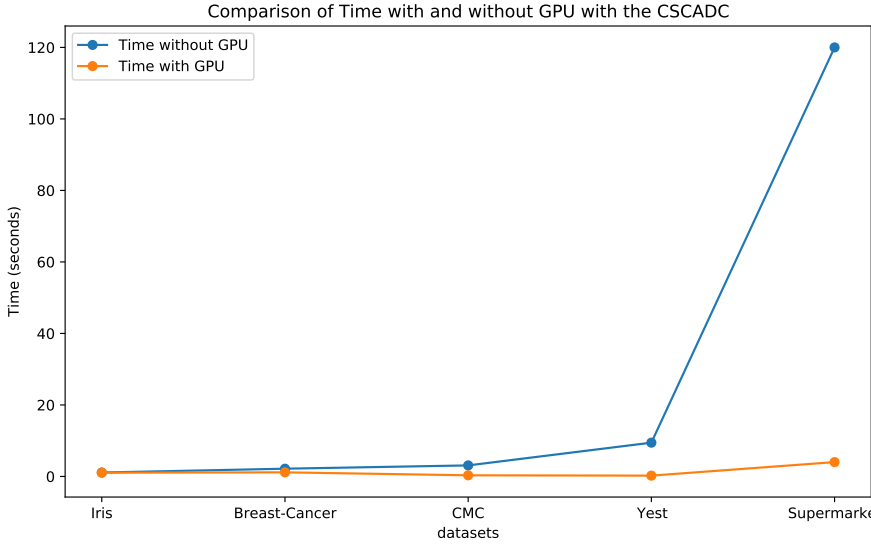

Figure 7. Run time with/without GPU for CSCADC

applicability in various domains and more extensive datasets. We may also consider addressing security aspects in our approach. Nonetheless, our approach has limitations; for instance, the number of clusters must be specified. In future work, we plan to incorporate dynamic clustering algorithms that automatically determine the number of clusters based on dataset characteristics if not provided. The use of our approach for online-stream data is also interesting. Addressing and overcoming this limitation are the key areas for improvement in future research. In conclusion, CSCADC presents a valuable contribution to the field of distributed clustering, offering a balanced approach between efficiency and scalability. This study opens avenues for continued exploration and application of the algorithm in diverse domains requiring effective solutions for clustering large-scale datasets.

REFERENCES

[1] Cheng, S.—Zhang, Q.—Qin, Q.: Big Data Analytics with Swarm Intelligence. Industrial Management & Data Systems, Vol. 116, 2016, No. 4, pp. 646–666, doi: 10.1108/IMDS-06-2015-0222.

[2] Kogan, J.: Introduction to Clustering Large and High-Dimensional Data. Cambridge University Press, 2007.

[3] Zemmouri, H.—Kout, A.—Labed, S.: Scalable Federated Learning for Massive Medical Image Classification: Tackling Noisy and Imbalanced Data. Engineering, Technology & Applied Science Research, Vol. 15, 2025, No. 5, pp. 26978–26984, doi: 10.48084/etasr.12040.

[4] Su, S.—Zhao, S.: A Hierarchical Hybrid of Genetic Algorithm and Particle Swarm Optimization for Distributed Clustering in Large-Scale Wireless Sensor Networks. Journal of Ambient Intelligence and Humanized Computing, 2017, doi: 10.1007/s12652-017-0619-9.

[5] Chen, J. X.—Gong, Y. J.—Chen, W. N.—Li, M.—Zhang, J.: Elastic Differential Evolution for Automatic Data Clustering. IEEE Transactions on Cybernetics, Vol. 51, 2021, No. 8, pp. 4134–4147, doi: 10.1109/TCYB.2019.2941707.

[6] Li, Z. X.—Guo, X. Q.—Chen, W. N.—Hu, X. M.: A Distributed Particle Swarm Optimization Algorithm for Distributed Clustering. Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '22), ACM, 2022, pp. 260–263, doi: 10.1145/3520304.3529016.

[7] Kennedy, J.—Eberhart, R.: Particle Swarm Optimization. Proceedings of ICNN'95 – International Conference on Neural Networks, IEEE, Vol. 4, 1995, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.

[8] Yang, X. S.—Deb, S.: Cuckoo Search via Lévy Flights. 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), 2009, pp. 210–214, doi: 10.1109/NABIC.2009.5393690.

[9] Ji, J.—Pang, W.—Li, Z.—He, F.—Feng, G.—Zhao, X.: Clustering Mixed Numeric and Categorical Data with Cuckoo Search. IEEE Access, Vol. 8, 2020, pp. 30988–31003, doi: 10.1109/ACCESS.2020.2973216.

[10] Gómez-Rubio, Á.—Soto, R.—Crawford, B.—Jaramillo, A.—Mancilla, D.—Castro, C.—Olivares, R.: Applying Parallel and Distributed Models on Bio-Inspired Algorithms via a Clustering Method. Mathematics, Vol. 10, 2022, No. 2, Art. No. 274, doi: 10.3390/math10020274.

[11] Zemmouri, H.—Labed, S.—Kout, A.: A Survey of Parallel Clustering Algorithms Based on Vertical Scaling Platforms for Big Data. 2022 4th International Conference on Pattern Analysis and Intelligent Systems (PAIS), 2022, pp. 1–8, doi: 10.1109/PAIS56586.2022.9946663.

[12] MacQueen, J.: Some Methods for Classification and Analysis of Multivariate Observations. In: Le Cam, L. M., Neyman, J. (Eds.): Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics. Vol. 5.1, 1967, pp. 281–297.

[13] Cuomo, S.—De Angelis, V.—Farina, G.—Marcellino, L.—Toraldo, G.: A GPU-Accelerated Parallel K-Means Algorithm. Computers & Electrical Engineering, Vol. 75, 2019, pp. 262–274, doi: 10.1016/j.compeleceng.2017.12.002.

[14] Rana, S.—Jasola, S.—Kumar, R.: A Review on Particle Swarm Optimization Algorithms and Their Applications to Data Clustering. Artificial Intelligence Review, Vol. 35, 2011, pp. 211–222, doi: 10.1007/s10462-010-9191-9.

[15] Sinha, A.—Jana, P. K.: A Hybrid MapReduce-Based k-Means Clustering Using Genetic Algorithm for Distributed Datasets. The Journal of Supercomputing, Vol. 74, 2018, No. 4, pp. 1562–1579, doi: 10.1007/s11227-017-2182-8.

[16] Benmounah, Z.—Meshoul, S.—Batouche, M.—Lio, P.: Parallel Swarm Intelligence Strategies for Large-Scale Clustering Based on MapReduce with Application to Epigenetics of Aging. Applied Soft Computing, Vol. 69, 2018, pp. 771–783, doi: 10.1016/j.asoc.2018.04.012.

[17] Dorigo, M.—Birattari, M.—Stutzle, T.: Ant Colony Optimization. IEEE Computational Intelligence Magazine, Vol. 1, 2006, No. 4, pp. 28–39, doi: 10.1109/MCI.2006.329691.

[18] Karaboga, D.—Basturk, B.: A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm. Journal of Global Optimization, Vol. 39, 2007, No. 3, pp. 459–471, doi: 10.1007/s10898-007-9149-x.

[19] Hashemi, S. E.—Tavana, M.—Bakhshi, M.: A New Particle Swarm Optimization Algorithm for Optimizing Big Data Clustering. SN Computer Science, Vol. 3, 2022, No. 4, Art. No. 311, doi: 10.1007/s42979-022-01208-8.

[20] Dhas, C. S. G.—Yuvaraj, N.—Kousik, N. V.—Geleto, T. D.: Chapter 27 – D-PPSOK Clustering Algorithm with Data Sampling for Clustering Big Data Analysis. In: Johri, P., Anand, A., Vain, J., Singh, J., Quasim, M. (Eds.): System Assurances: Modeling and Management. Elsevier, Emerging Methodologies and Applications in Modelling, 2022, pp. 503–512, doi: 10.1016/B978-0-323-90240-3.00027-8.

[21] Kulkarni, O.—Jena, S.—Ravi Sankar, V.: MapReduce Framework Based Big Data Clustering Using Fractional Integrated Sparse Fuzzy C Means Algorithm. IET Image Processing, Vol. 14, 2020, No. 12, pp. 2719–2727, doi: 10.1049/iet-ipr.2019.0899.

[22] Han, S.: Analysis of the Clustering Effect of Sports Big Data Transmission Based

on Ant Colony Intelligent Algorithm. In: Chang, J. W., Yen, N., Hung, J. C. (Eds.): Frontier Computing (FC 2020). Springer Singapore, Lecture Notes in Electrical Engineering, Vol. 747, 2021, pp. 2165–2170, doi: 10.1007/978-981-16-0115-6_261.

[23] Bendechache, M.—Kechadi, M. T.—Le-Khac, N. A.: Efficient Large Scale Clustering Based on Data Partitioning. 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), 2016, pp. 612–621, doi: 10.1109/DSAA.2016.70.

[24] Tarkhaneh, O.—Isazadeh, A.—Khamnei, H. J.: A New Hybrid Strategy for Data Clustering Using Cuckoo Search Based on Mantegna Levy Distribution, PSO, and K-Means. International Journal of Computer Applications in Technology, Vol. 58, 2018, No. 2, pp. 137–149, doi: 10.1504/IJCAT.2018.094576.

[25] Tekieh, R.—Beheshti, Z.: A MapReduce-Based Big Data Clustering Using Swarm-Inspired Meta-Heuristic Algorithms. Scientia Iranica. Transaction D, Computer Science & Engineering & Electrical Engineering, Vol. 31, 2024, No. 10, doi: 10.24200/sci.2024.61178.7184.

[26] White, T.: Hadoop: The Definitive Guide, Third Edition. O'Reilly Media, Inc., 2012.

[27] Salloum, S.—Dautov, R.—Chen, X.—Peng, P. X.—Huang, J. Z.: Big Data Analytics on Apache Spark. International Journal of Data Science and Analytics, Vol. 1, 2016, No. 3, pp. 145–164, doi: 10.1007/s41060-016-0027-9.

[28] Carbone, P.—Katsifodimos, A.—Ewen, S.—Markl, V.—Haridi, S.—Tzoumas, K.: Apache Flink: Stream and Batch Processing in a Single Engine. The Bulletin of the Technical Committee on Data Engineering, Vol. 38, 2015, No. 4, pp. 28–38.

[29] Hadian, A.—Shahrivari, S.: High Performance Parallel k-Means Clustering for Disk-Resident Datasets on Multi-Core CPUs. The Journal of Supercomputing, Vol. 69, 2014, No. 2, pp. 845–863, doi: 10.1007/s11227-014-1185-y.

[30] Nishom, M.: Comparison of the Accuracy of Euclidean Distance, Minkowski Distance, and Manhattan Distance in the Chi-Square-Based K-Means Clustering Algorithm. Jurnal Informatika, Vol. 4, 2019, No. 1, pp. 20–24, doi: 10.30591/jpit.v4i1.1253 (in Indonesian).

[31] Dunn, J. C.: Well-Separated Clusters and Optimal Fuzzy Partitions. Journal of Cybernetics, Vol. 4, 1974, No. 1, pp. 95–104, doi: 10.1080/01969727408546059.

[32] Goel, S.—Sharma, A.—Bedi, P.: Cuckoo Search Clustering Algorithm: A Novel Strategy of Biomimicry. 2011 World Congress on Information and Communication Technologies, IEEE, 2011, pp. 916–921, doi: 10.1109/WICT.2011.6141370.

[33] Yang, X. S.—Deb, S.: Cuckoo Search via Lévy Flights. 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), IEEE, 2009, pp. 210–214, doi: 10.1109/NABIC.2009.5393690.

[34] Esteves, R. M.—Hacker, T.—Rong, C.: Competitive K-Means, a New Accurate and Distributed K-Means Algorithm for Large Datasets. 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, Vol. 1, 2013, pp. 17–24, doi: 10.1109/CloudCom.2013.89.

[35] Abed-Alguni, B. H.: Island-Based Cuckoo Search with Highly Disruptive Polyno-

mial Mutation. International Journal of Artificial Intelligence, Vol. 17, 2019, No. 1, pp. 57–82.

[36] Datasets – UCI Machine Learning Repository. doi: 10.24432/C56C76.

[37] Supermarket Dataset for Predictive Marketing 2023. `https://www.kaggle.com/datasets/hunter0007/ecommerce-dataset-for-predictive-marketing-2023` [accessed 2023-11-22].

[38] Kalé, N.—Jones, N.: Practical Analytics, 2nd Ed. Epistemy Press, 2020, `https://epistemypress.com/books/practical-analytics/`.

[39] Tzortzis, G.—Likas, A.: The MinMax K-Means Clustering Algorithm. Pattern Recognition, Vol. 47, 2014, No. 7, pp. 2505–2516, doi: 10.1016/j.patcog.2014.01.015.

[40] Liu, F.—Deng, Y.: Determine the Number of Unknown Targets in Open World Based on Elbow Method. IEEE Transactions on Fuzzy Systems, Vol. 29, 2021, No. 5, pp. 986–995, doi: 10.1109/TFUZZ.2020.2966182.

Hadjir Zemmouri is Ph.D. student at the University of Constantine 2, Algeria, in the MISC Laboratory. She obtained her Master's degree in 2019 in information and communication science and technology. Her research focuses on advanced machine learning techniques for Big Data analytics and classification. She is actively exploring innovative approaches to improve the efficiency and accuracy of predictive models on large-scale datasets.

Said Labed is Associate Professor at the University of Constantine 2-Abdelhamid Mehri, Algeria, Department of Fundamental Computer Science and Its Applications. He received his Ph.D. in computer science from the University of Constantine, in 2013. He is a member of the SCAL team (Soft Computing and Artificial Life) of MISC Laboratory (Modeling and Implementation of Complex Systems). His research interests include deep learning, cloud computing, Big Data, optimisation and artificial vision.

Akram Kout obtained his Ph.D. degree in 2017 from the University of Constantine 2, Algeria. He is currently Associate Professor at Ferhat Abbas University Setif 1 and a member of the MISC Laboratory (SCAL team). He has contributed numerous publications on mobile ad hoc networks, UAV and MANET network optimization, clustering techniques, graph coloring, and predictive modeling. He is actively involved in mentoring graduate students and leading collaborative research projects in AI, optimization, and intelligent network systems.

El-Bay Bourennane is Professor at the University of Bourgogne, France, affiliated with the Laboratory ImVia in Dijon. He holds his Ph.D. in electronics and has a longstanding academic career in signal processing, embedded systems, and hardware design. His research interests encompass a broad range of topics, including image and video processing, FPGA-based hardware implementations, dynamic reconfigurable systems, and machine learning. He has authored over 200 publications, reflecting his significant contributions to these fields.