BAREKENG: Journal of Mathematics and Its Applications
June 2026 Volume 20 Issue 2 Page 1453–1472
P-ISSN: 1978-7227 E-ISSN: 2615-3017
https://doi.org/10.30598/barekengvol20iss2pp1453-1472

A GENETIC ALGORITHM–PARTICLE SWARM OPTIMIZATION OPTIMIZED DOFCM APPROACH TO ENHANCE CLUSTERING AND OUTLIER DETECTION

Sintia Afriyani [1], **Rohmatul Fajriyah** [2*]

[1,2]*Master Program in Statistics, Department of Statistics, Faculty of Mathematics and Natural Sciences, Universitas Islam Indonesia*
Jln. Kaliurang km. 14,5 Sleman, Yogyakarta, 55584, Indonesia

*Corresponding author's e-mail: * **966110101@uii.ac.id***

Article Info

Article History:
Received: 7th June 2025
Revised: 1st August 2025
Accepted: 22nd September 2025
Available online: 26th January 2026

Keywords:
Clustering;
Outlier;
DOFCM;
Genetic Algorithm;
Particle Swarm Optimization.

ABSTRACT

In the era of Industry 4.0, Big Data from the IoT demands advanced analysis techniques. Outlier detection is vital as anomalies may indicate sensor failures, fraud, or abnormal medical records. Fuzzy clustering methods such as DOFCM are often applied, yet their performance depends on accurate cluster center placement, which remains challenging. While several Fuzzy C-Means extensions address outlier sensitivity, most rely on single optimization strategies. The integration of PSO and GA into DOFCM has been rarely explored, making this study novel in evaluating how different evolutionary algorithms enhance clustering robustness and anomaly detection. This research introduces DOFCM-PSO and DOFCM-GA, tested on five benchmark datasets with outliers: Iris, Wine, Sonar, Diabetes, and Ionosphere. The Silhouette Coefficient (SC) was used as the evaluation metric. Results show that GA consistently outperforms PSO, with SC values improving by approximately 0.02–0.03 (equivalent to an increase of 8–12%) across datasets. For instance, the Iris dataset improved from 0.6029 (PSO) to 0.6291 (GA), while the Wine dataset increased from 0.2759 to 0.2958. In addition, evaluation of computational time and outlier detection further supports these findings. Although GA required slightly longer runtime than PSO, it substantially reduced the number of outliers while still achieving higher SC values. A similar pattern was observed in the Diabetes dataset, where GA decreased outliers from 20 to 7 with a modest SC improvement. These results indicate that PSO is more efficient in runtime, but GA provides more robust clustering by minimizing anomalies and producing better separation quality. Despite promising results, this study is limited by the relatively small dataset sizes and sensitivity to parameter settings, which may influence outcomes. Future work should apply the method to larger datasets and include additional clustering indices. Overall, DOFCM-GA can be considered a robust approach for fuzzy clustering in the presence of anomalies.

How to cite this article:

S. Afriyani and R. Fajriyah, "A GENETIC ALGORITHM–PARTICLE SWARM OPTIMIZATION OPTIMIZED DOFCM APPROACH TO ENHANCE CLUSTERING AND OUTLIER DETECTION", *BAREKENG: J. Math. & App.*, vol. 20, no. 2, pp. 1453-1472, Jun, 2026.

1. INTRODUCTION

In the era of Industry 4.0, advancements in information and communication technology have accelerated the growth of Big Data obtained from the Internet of Things (IoT) [1], [2]. Big Data encompasses large volumes, diverse types of data, and high velocity, necessitating sophisticated analytical techniques such as data mining to identify meaningful patterns [3]. One of the key techniques in data mining is outlier detection, which plays a crucial role in clustering and classification tasks [4]. Outliers, defined as data points that deviate significantly from the majority of observations, may occur in univariate or multivariate forms depending on the data dimensionality [5]-[7]. Their presence can distort analytical results and introduce uncertainty, making outlier detection critical in practical applications such as fraud detection, fault diagnosis, and medical analysis [8]-[10].

However, the presence of outliers often leads to clustering results that do not align with the true patterns [11]. Conventional methods are effective at detecting global outliers but perform poorly when handling local anomalies within clusters of varying densities [12]. Fuzzy clustering methods, particularly Density Oriented Fuzzy C-Means (DOFCM), have been developed to address this issue by partitioning data into groups based on membership levels [13]-[15]. Previous studies have shown that DOFCM generally outperforms other fuzzy techniques in handling outliers [16]-[18]. Nevertheless, the performance of this method is highly dependent on the accurate placement of cluster centers, which remains a challenging task [19].

Recent years have witnessed rapid advances in meta-heuristic optimization algorithms, which have been widely applied to clustering and outlier detection problems. Studies from 2022 to 2024 highlight that approaches based on Particle Swarm Optimization (PSO), Genetic Algorithm (GA), as well as newer algorithms such as Grey Wolf Optimizer (GWO) and Differential Evolution (DE), have significantly improved clustering quality [20]-[22]. However, the integration of meta-heuristic strategies into Density Oriented Fuzzy C-Means (DOFCM) remains underexplored. This study contributes to filling this gap by evaluating how PSO and GA can be incorporated into DOFCM to enhance clustering robustness and outlier detection.

While prior research has explored individual optimization approaches, there is still a lack of comparative studies that systematically evaluate the effectiveness of PSO and GA in optimizing DOFCM for outlier detection. The novelty of this research lies in the direct comparison between DOFCM optimized with PSO and GA for detecting outliers. Although both meta-heuristic algorithms are widely applied in optimization, studies that systematically assess their effectiveness in the context of DOFCM for handling outlier data remain limited. Therefore, this research is relevant to demonstrate which algorithm is more effective in generating representative cluster centers and improving anomaly detection quality.

This study aims to optimize the DOFCM method using PSO and GA for outlier detection, and to compare their effectiveness across multiple benchmark datasets using the Silhouette Coefficient as the primary evaluation metric. The remainder of this paper is organized as follows. Section 2 presents the methodology, including the DOFCM framework and optimization strategies using PSO and GA. Section 3 describes the experimental setup, datasets, and clustering evaluation, followed by the results and discussion. Finally, Section 4 concludes the study by summarizing the main findings, highlighting limitations, and providing directions for future research.

2. RESEARCH METHODS

This research employs DOFCM optimization for outlier detection. The PSO and the GA is applied to find the optimal configuration of cluster centers. Fig. 1 illustrates the flowchart of the research, which includes the application of DOFCM for outlier detection and optimizing the cluster centers by using the PSO and GA.

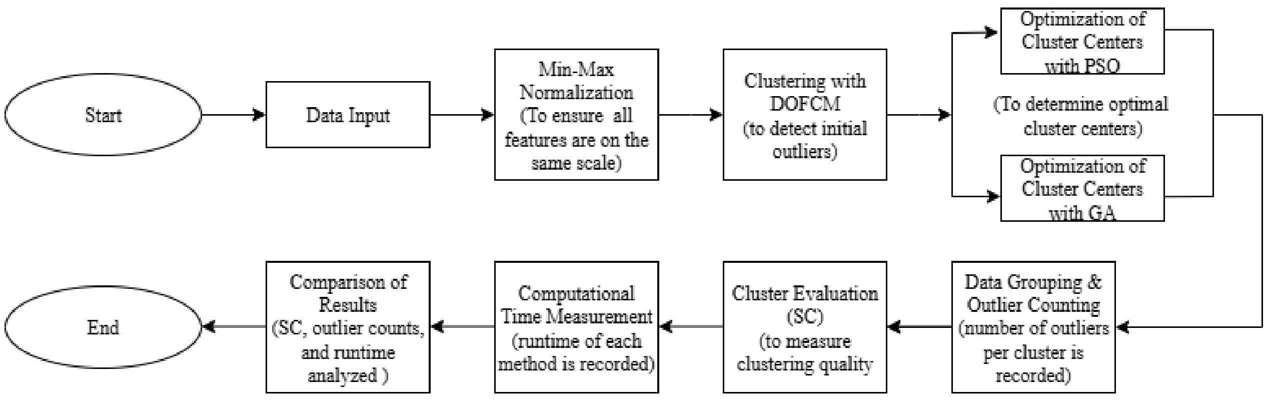
Start
Data Input
Min-Max Normalization (To ensure all features are on the same scale)
Clustering with DOFCM (to detect initial outliers)
Optimization of Cluster Centers with PSO
(To determine optimal cluster centers)
Optimization of Cluster Centers with GA
Data Grouping & Outlier Counting (number of outliers per cluster is recorded)
Cluster Evaluation (SC) (to measure clustering quality
Computational Time Measurement (runtime of each method is recorded)
Comparison of Results (SC, outlier counts, and runtime analyzed)
End

Figure 1. The Flowchart of the Research

Fig. 1 illustrates the overall research flowchart, starting from data input and preprocessing using Min-Max normalization to ensure that all features are on the same scale. The normalized data is then processed using the DOFCM algorithm for initial outlier detection. In this stage, the data are grouped into two clusters ($k = 2$), representing valid data and potential outliers, based on membership values in the DOFCM framework. This grouping enables the optimization algorithms PSO and GA to adjust cluster centers more effectively, thereby improving the separation of anomalies from the main data distribution. After optimization, data grouping and outlier counting are performed, followed by cluster evaluation using the Silhouette Coefficient (SC) to measure clustering quality. Additionally, the computational time of each method is recorded to analyze efficiency. Finally, the results are compared based on SC values, the number of detected outliers, and runtime, providing a comprehensive evaluation of the proposed approach.

2.1 Dataset Description

In this study, five benchmark datasets with diverse characteristics were used to evaluate the proposed method. The Iris dataset [23] contains measurements of sepal length, sepal width, petal length, and petal width for three Iris species (Setosa, Versicolor, and Virginica). The Wine dataset [24] consists of 13 chemical attributes, such as alcohol content, flavonoids, and color intensity, used to classify three wine cultivars. The Sonar dataset [25] records sonar signals reflected from underwater objects, labeled as rock or mine. The Diabetes dataset [26] is the Pima Indians Diabetes, which includes patient medical records with features such as glucose concentration and BMI, labeled as diabetic or non-diabetic. Finally, the Ionosphere dataset [27] represents radar signals for detecting ionospheric structures, categorized into good and bad signals. A summary of the dataset characteristics, including the number of features, number of classes, total records, and percentage of outliers, is presented in Table 1. Outliers were identified using the DOFCM framework, where data points with membership values below the threshold (α) were classified as anomalies.

The percentage of outliers was calculated using Eq. (1).

$$P_{outlier} = \frac{N_{outlier}}{N_{total}} x100\%, \tag{1}$$

where $N_{outlier}$ is the number of data points identified as outliers, and N_{total} is the total number of records in the dataset.

Table 1. Characteristics of Benchmark Datasets

Dataset	Features	Classes	Records	%Outliers
Iris	4	3	150	2.00 %
Wine	13	3	178	2.81 %
Sonar	60	2	208	4.81 %
Diabetes	8	2	768	2.60 %
Ionosphere	34	2	351	1.99 %

From Table 1, it can be observed that the proportion of outliers varies across datasets. The Sonar dataset has the highest percentage of outliers (4.81%), indicating a higher degree of noise or anomalous patterns. In contrast, the Ionosphere dataset shows the lowest percentage of outliers (1.99%), suggesting more homogeneous data. The Iris, Wine, and Diabetes datasets exhibit moderate levels of anomalies, ranging from 2.00% to 2.81%. These variations highlight the importance of applying robust clustering methods such as DOFCM, where optimization with PSO and GA is expected to improve the ability to separate valid data from anomalous points across datasets with different characteristics.

2.2 Data Pre-Processing

Big data, characterized by the 5Vs (volume, velocity, variety, veracity, and value), refers to large datasets that can be analyzed to gain valuable insights. This data can be structured, semi-structured, or unstructured [28]. Data mining, on the other hand, is the process that utilizes pattern recognition techniques from mathematics and statistics to discover patterns, relationships, and trends within big data. By analyzing stored data, data mining facilitates better decision-making and the utilization of previously hidden information within large datasets [29]. In handling such large volumes of data, it is essential to apply data normalization to ensure that each feature contributes equally to the analysis. One commonly used method is Min-Max normalization, which scales the data to a fixed range, typically between 0 and 1. This approach helps improve the performance and accuracy of clustering algorithms by eliminating the bias caused by differing value ranges [22]. The formula for performing min-max normalization is given in Eq. (2).

$$x'_{i,j} = \frac{x_{i,j} - min\left(x_{:,j}\right)}{max\left(x_{:,j}\right) - min\left(x_{:,j}\right)}, \tag{2}$$

where, $x_{i,j}$ represents the original value of the data element at row i and column j, while $x'_{i,j}$ is the normalized value. The notation $x_{:,j}$ refers to all values in the column j, so $min\left(x_{:,j}\right)$ and $max\left(x_{:,j}\right)$ indicate the minimum and maximum values in that column, respectively. This normalization process is performed per column (feature), ensuring that each feature has a consistent scale and does not dominate others during data analysis or model training, such as in clustering or machine learning algorithms.

2.3 Clustering and Outlier Detection

Clustering is a technique used to group data into homogeneous subsets, where data points with similar characteristics are grouped together in the same cluster [23]. The main objective is to maximize similarity within clusters and minimize similarity between clusters [24]. However, not all data points fit neatly into clusters; some exhibit behaviors that significantly deviate from the rest. These anomalous data points are known as outliers, which may arise due to noise, variability, or truly rare events. Outlier detection is therefore a critical step in data analysis, as these data points can distort clustering results or reveal meaningful exceptions. Outliers are typically identified using statistical or distance-based approaches that evaluate how far a point deviates from the structure of established clusters [33].To address the challenge of grouping data while accounting for uncertainty and data fuzziness, various clustering techniques have been developed. One of the most widely used methods in this category is Fuzzy C-Means (FCM), which allows data points to belong to multiple clusters with varying degrees of membership. A dataset is denoted by X, where $X = \{x_1, x_2, x_3, \dots, x_n\}$ represents n points in an M dimensional space. The centroid of cluster k is denoted v_k, d_{ik} is the distance between x_i and v_k and $'c'$ is the number of clusters present in the dataset [34]. One of the most widely used clustering techniques and the basis for many extended methods is Fuzzy C-Means (FCM). FCM assumes that the number of clusters ‘c’ is known a priori and minimizes the objective function (J_{FCM}) as follows:

$$J_{FCM}(U,V) = \sum_{k=1}^{c} \sum_{i=1}^{n} u_{ki}^{m} d_{ki}^{2}, \tag{3}$$

where $d_{ki} = \|x_i - v_k\|$ and u_{ki} represents the membership of x_i in cluster $'k'$ which satisfies

$$\sum_{k=1}^{c} u_{ki} = 1, i = 1,2, \dots, n. \tag{4}$$

Known as the fuzzifier (fuzziness index), and any norm $\|\cdot\|$ can be used to calculate d_{ki} (using Euclidean distance). Minimization of J_{FCM} is performed using an iterative fixed-point scheme known as the alternating optimization technique. The conditions for local extrema of Eqs. (3) and (4) are derived using Lagrange multipliers. This minimization process is carried out by repeating two main steps, update the membership u_{ki} is implemented by using Eq. (5).

$$u_{ki} = \frac{1}{\sum_{j=1}^{c} \left(\frac{d_{ki}}{d_{ij}}\right)^{\frac{2}{m-1}}}, \tag{5}$$

where $1 \leq k \leq c$ and $1 \leq i \leq n$. Eq. (5) calculates the membership degree u_{ki} of a data point x_i with respect to cluster k. This value depends on the distance between x_i and the cluster center v_k, normalized by the sum of distance ratios to all cluster centers. The parameter $m > 1$ is a fuzzifier that controls the level of fuzziness in the clustering process. The notation d_{ki} refers to the distance between x_i and cluster center v_k, while d_{ji} denotes the distance from x_i to cluster center v_j. Here, c represents the total number of clusters, with index ranges $1 \leq j, k \leq c$, and $1 \leq i \leq n$, where n is the number of data points. The closer the data point x_i is to the centroid v_k the higher the value of u_{ki}, update the centroid v_k by using Eq. (6).

$$v_k = \frac{\sum_{i=1}^{n}(u_{ki}^{m} x_i)}{\sum_{i=1}^{n}(u_{ki}^{m})}. \tag{6}$$

Eq. (6) is used to update the position of the centroid v_k of cluster k, taking into account the contributions of all data points within the cluster based on their membership degrees. The centroid is updated as the weighted average of all data points in the cluster, where the weights are the membership degrees raised to the power of m. This ensures that the centroid moves toward the center of mass of the data within the cluster, thereby improving clustering accuracy.

2.4 Density Oriented Fuzzy C-Means (DOFCM)

Although Fuzzy C-Means (FCM) is effective in forming clusters based on membership degrees, it has limitations in handling outliers since all data points are considered to contribute equally to the formation of cluster centroids. To address this issue, several extensions have been proposed, including DOFCM, which specifically incorporates density information to improve outlier detection.

DOFCM modifies the membership function of FCM by considering the density of surrounding points [19], [35]. The algorithm forms $'n + 1'$ clusters, where n represent valid clusters, and one outlier cluster is assigned to the outlier. DOFCM uses neighboring membership to assess the density of objects within a certain radius, where distances between points are measured using Euclidean distance.

$$d(x_i, x_j) = \sqrt{\sum_{m=1}^{p} (x_{im} - x_{jm})^2}. \tag{7}$$

The neighborhood membership of a point i in the dataset X is defined as:

$$M^{i}_{neighbourhood} = \frac{\eta^{i}_{neighbourhood}}{\eta_{max}}. \tag{8}$$

Let q be the neighborhood surrounding point i, then q will satisfy:

$$\{q \in X \mid dist(i, q) \leq r_{neighbourhood}\}. \tag{9}$$

The neighborhood membership of each point in the dataset X is calculated, and the threshold $'\alpha'$ is determined based on the density of points. A point is considered an outlier if its neighborhood membership is less than α.

$$M^{i}_{neighbourhood} = \begin{cases} < \alpha, & outlier \\ \geq \alpha, & non\ outlier \end{cases}. \tag{10}$$

The value of α is chosen from the range of neighborhood membership values and is typically close to zero. A data point with membership below α is classified as an outlier. Since α depends on the dataset density, it becomes a critical parameter for outlier identification.

To illustrate the difference, consider a simple two-dimensional dataset with two compact clusters and one distant point. As shown in Fig. 2, FCM treats all points equally, causing the outlier to influence the cluster centroids. In contrast, DOFCM assigns a very low neighborhood membership to the outlier, isolating it into a separate cluster without affecting the main centroids. This demonstrates how DOFCM enhances clustering robustness by effectively identifying outliers.

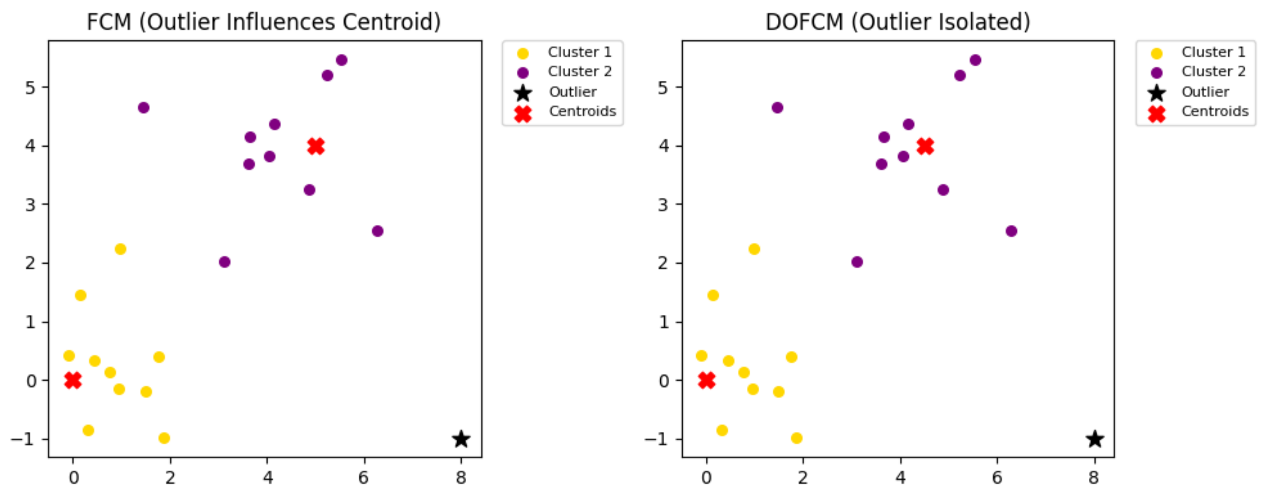

Figure 2. Comparison of Clustering Results Between FCM and DOFCM

Fig. 2 illustrates the difference between FCM and DOFCM in handling outliers. In the case of FCM, the outlier is treated as part of the cluster, which shifts the centroid position and reduces clustering accuracy. In contrast, DOFCM assigns a low neighborhood membership value to the outlier, isolating it into a separate cluster. This prevents the outlier from influencing the main cluster centroids and demonstrates the robustness of DOFCM in identifying anomalies more effectively.

2.5 Particle Swarm Optimization (PSO)

PSO is an optimization technique inspired by the social behavior of organisms such as flocks of birds or schools of fish [36], [37]. This algorithm is used to find optimal solutions in large and complex search spaces [38]. The process of PSO is as follows:

1. Initializing particles

$$x_i(0) = x_{min,d} + r_d\left(x_{max,d} - x_{min,d}\right),$$
$$\forall d = 1, \dots, n_x, \forall i = 1, \dots n_s. \tag{11}$$

Eq. (11) initializes the initial position of particles in the search space of the PSO algorithm. The initial position $x_i(0)$ of the i particle is determined randomly within the range bounded by $x_{\max,d}$ and $x_{\max,d}$ where r_d is a random number between 0 and 1.

2. Calculating the objective function $\left(f(x)\right)$

Complex optimization problems can be addressed using the PSO algorithm, where the objective function value is used to evaluate the fitness of individuals. To maximize the function h, the fitness is calculated as $f(x) = h$, while for minimization, the fitness is computed as $f(x) = \frac{1}{h}$, as stated in Eq. (12).

$$V_{id}(0) = 0. \tag{12}$$

The initial position of the optimal individual $(pBest_i)$ can be initialized with the particle's position at time $t = 0$. Eq. (13) indicates that the initial velocity $V_{id}(0)$ of the particle in each dimension is considered to be zero.

$$pBest_{i(0)} = x_i(0). \tag{13}$$

3. Finding the Local Best (pBest) and Global Best (gBest)

This version maintains formal and clear language appropriate for scientific writing. If you have specific calculations or formulas for $pBest$ can provide them for a more detailed explanation.

$$pBest_i(t+1) = \begin{cases} pBest_i(t), & \text{If } f\left(x_{id}(t+1)\right) \geq f\left(pBest_i(t)\right), \\ x_{id}(t+1), & \text{If } f\left(x_{id}(t+1)\right) < f\left(pBest_i(t)\right). \end{cases} \tag{14}$$

Eq. (14) states that the $pBest$ of a particle in the next iteration $pBest$ remains the same as the current $pBest$ if the objective function value $pBest$ at the new position is not better than the current $pBest$. However, if the value of $pBest$ at the new position is better, then $pBest$ is updated to this new position.

$$f\left(gBest_i(t)\right) = \min\left\{f\left(x_{id}(t)\right), \ldots, f\left(x_{nd}(t+m)\right)\right\}. \tag{15}$$

Eq. (15) defines $pBest$ as the minimum value of f among all particles in the population at the current iteration, representing the best position discovered by any particle in the population. This ensures that PSO continuously directs particles towards the best solution found so far.

4. Updating Speed and Position

The velocity (v_{id}) and position (x_{id}) of a particle can be calculated using the obtained $pBest$ and $gBest$. The following Equation describes the change in velocity (v_{id}).

$$v_{id}^{(t+1)} = w * v_{id}^t + c_1^t * r_1 * \left(pBest_i^t - x_{id}^t\right) + c_2^t * r_2 * \left(gBest^t - x_{id}^t\right). \tag{16}$$

Eq. (16) illustrates how the velocity of a particle (v) is updated by taking into account the previous inertia (w), the cognitive component (c_1) that directs the particle towards its own best position $(pBest)$, and the social component (c_2) that guides the particle towards the global best position $(gBest)$.

$$x_{id}^{t+1} = x_{id}^t + V_{id}^{t+1}. \tag{17}$$

Eq. (17) describes how a particle's position (x) is updated by adding the new velocity to the current position.

$$\begin{gathered} \text{if } v_{id}^{t+1} > v_d^{\max}, \text{so } v_{id}^{t+1} = v_{id}^{\max} = v_d^{\max}, \\ \text{if } v_{id}^{t+1} < -v_d^{\max}, \text{so } v_{id}^{t+1} = -v_d^{\max}. \end{gathered} \tag{18}$$

Eq. (18) ensures that the particle's velocity does not exceed the predefined maximum value (v^{max}) by capping the velocity at (v^{max}) if the calculated velocity surpasses this limit.

$$v_d^{\max} = \mathrm{g} * \left(x_d^{\max} - x_d^{\min}\right), \mathrm{g} \in (0,1). \tag{19}$$

Eq. (19) defines the maximum particle velocity (v^{max}) based on the particle's position range and a scaling factor (g) that lies within the interval $(0,1)$.

5. Stochastic Injection

Early convergence often occurs in small search spaces because particles quickly reach the global best position $(gBest)$ causing the population to lose diversity. To address this, a random injection system is employed, where the position of particle n is reinitialized at every iteration interval g. If the values of $(pBest)$ and fitness remain the same, the particle is randomly injected to maintain diversity and increase the chances of finding a better solution. The values of n and g are determined based on prior experiments.

6. Stopping Criteria

Criteria are used to terminate the iterations and obtain the best solution. To avoid prematurely converging on a suboptimal solution, the stopping condition must ensure that PSO does not halt too early. In this optimization problem, the stopping condition is defined as reaching the maximum number of iterations.

In this study, the PSO parameters were set as follows: population size (number of particles) = 5, dimensions = 4 (based on the number of features in the Iris dataset used as illustration), maximum iteration = 100, cognitive coefficient (c_1) = 2, social coefficient (c_2) = 2, and inertia weight (ω) = 0.5. These parameter settings were adopted from previous studies that demonstrated stable convergence under similar configurations.

2.6 Genetic Algorithm (GA)

Genetic Algorithm mimics the natural evolution process based on Darwin's theory of natural selection, utilizing selection, mutation, and crossover to find optimal solutions in a large search space [39], [40]. The GA process begins with the creation of an initial population, which is generated randomly to form a diverse set of potential solutions. Chromosome representation is a crucial step, as it defines how each solution is encoded for processing within the algorithm. Once encoded, the population is initialized with a predefined size to prepare for the evolutionary process. The reproduction stage involves generating offspring through crossover and mutation operations. In the crossover step, two parent chromosomes are randomly selected and combined using an advanced intermediate crossover technique to produce offspring that inherit traits from both parents. The crossover operation applies the intermediate method, as shown in Eqs. (20) and Eq. (21) .

$$C_1 = P_1 + \alpha(P_2 - P_1), \tag{20}$$

$$C_2 = P_2 + \alpha(P_1 - P_2). \tag{21}$$

The values are randomly selected within the interval $[-0.25, 1.25]$. With a crossover ratio of 0.2, two offspring will be produced, calculated as $0.2 \times 10 = 2$ offspring. Consequently, the crossover process can be performed twice. In Eq. (20) C_1 is generated by adding the product of α and the difference between P_2 and P_1 to P_1. In Eq. (21) C_2 is similarly generated, but in this case, α is multiplied by the difference between P_1 and P_2.

Mutation is a process in which a chromosome changes from its parent chromosome, resulting in characteristics of the offspring chromosome differing from those of the parent chromosome.

$$x_i' = x_i' + r(\max{}_i = \min{}_j). \tag{22}$$

If the mutation rate is 0.2, then there will be 2 offspring, calculated as $0.2 \times 10 = 2$. Each mutation process will produce one offspring.

Fitness Calculation involves selecting the best individuals from the population for the next generation, a process known as selection. This study uses the elitist selection method, which retains the individuals with the highest fitness. For the Genetic Algorithm, the parameters used were: population size = 10, number of generations = 5, crossover rate = 0.2, and mutation rate = 0.2. These settings were chosen to balance exploration and exploitation during the optimization process, preventing premature convergence while maintaining diversity in the population.

2.7 Evaluation Method

The Silhouette Coefficient (SC) method is used to evaluate how well data is clustered within clusters. The SC ranges from -1 to 1, with higher values indicating that objects are closer to their own cluster [41]. A value around 0 suggests that the objects are between two clusters, while values close to -1 indicate that the objects are far from their own cluster. The formula for calculating the SC is given as follows:

$$SC = \frac{b - a}{\max(a, b)} \,. \tag{23}$$

In this formula, a represents the average distance between a data point and all other points within the same cluster, also known as the intra-cluster distance. Meanwhile, b denotes the average distance between the data point and all points in the nearest neighboring cluster, referred to as the nearest-cluster distance. These two values are used to assess how appropriately a data point has been clustered, with higher Silhouette Coefficient values indicating better clustering performance. This Eq. (23) evaluates how well a data point fits within its assigned cluster compared to other clusters. An SC value close to 1 indicates that the data point is well matched to its own cluster and poorly matched to neighboring clusters.

At the computational level, the baseline DOFCM algorithm has a time complexity of $O(n \cdot k \cdot t)$, where n is the number of data points, k the number of clusters, and t the number of iterations. When optimized with PSO, the complexity increases to approximately $O(p \cdot d \cdot t)$, with p denoting the number of particles and d the data dimensionality, since each particle evaluates the objective function in every iteration. In contrast, DOFCM-GA has a complexity of about $O(g \cdot pop \cdot d)$, where g is the number of generations and pop is the population size. In practice, PSO tends to be computationally lighter for low-dimensional data and small populations, while GA often yields more stable solutions at the expense of slightly higher runtimeBold using proper case letters.

3. RESULTS AND DISCUSSION

This section presents the experimental results and discussion of the proposed methods. The evaluation focuses on comparing the clustering performance of DOFCM, DOFCM-PSO, and DOFCM-GA across five benchmark datasets. The SC is used as the primary metric to assess clustering quality in terms of compactness and separation. In addition, visualization of the clustering results is provided to highlight differences in how

PSO and GA optimize cluster centers. The findings are then analyzed to identify which optimization technique delivers more robust performance in handling outliers

3.1 Data Normalization

Data normalization is performed by scaling the original data into a range of [0, 1] using the min–max normalization method, as defined in Eq. (2). A small sample of the normalized data is shown in Table 2 for illustration.

Table 2. **Data Normalization in Data Iris**

No.	x_1	x_2	x_3	x_4	y
1	0.2222	0.6250	0.0678	0.0416	*Iris Setosa*
2	0.1667	0.4167	0.0678	0.0416	*Iris Setosa*
3	0.1111	0.5000	0.0508	0.0416	*Iris Setosa*
4	0.2778	0.5833	0.0847	0.0416	*Iris Setosa*
5	0.3333	0.4167	0.1186	0.0416	*Iris Setosa*

Table 2 shows an example (5 rows) of the Iris dataset normalized using the min–max method of Eq. (1), which scales the attributes (x_1–x_4) into the range [0,1] to ensure uniform feature contributions. Only the Irish data is shown here because it is simple, low-dimensional, easy to visualize, and often used as a benchmark. Normalization is crucial in clustering to prevent larger-scale features from dominating the distance calculation

3.2 Data Implementation in the Density Oriented Fuzzy C-Means (DOFCM)

DOFCM is an algorithm used to identify data considered as outliers based on the local density of nearest neighbors using the FCM method. The steps involved in the process are as follows:

1. Initial Centroid Initialization

Initialize Centroid Positions Randomly initialize the centroid positions (x_j) using the number of clusters $(k = 2)$ and the fuzziness parameter $(m = 2)$, as follows:

$$x_1 = [0.5000, 0.3333, 0.6271, 0.4583],$$

$$x_2 = [0.3888, 0.7500, 0.1186, 0.0833].$$

2. Calculating Distance.

For each data point x_i the Euclidean distance to the cluster centroids is calculated using Eq. (7). Based on this Equation, the distance between data point 1 and centroid 1 is 0.7873, while the distance between data point 1 and centroid 2 is 0.1060. Similarly, the distance between data point 150 and centroid 1 is 0.2127, and the distance to centroid 2 is 0.9075. This calculation is performed for all pairs of data points.

3. Update membership values

After calculating the distances between data points and cluster centroids, the membership value u_{ki} for each data point x_i in cluster k is updated using Eq. (5). For example, for data point 1, the distance ratio $\frac{d(x_i,x_1)}{d(x_1,x_1)}$ is 0.7873 and $\frac{d(x_i,x_2)}{d(x_1,x_2)}$ is 0.1016, resulting in a membership value $U_{i,1}$ of 0.0061. Conversely, for data point 150, the distance ratio $\frac{d(x_i,x_1)}{d(x_1,x_1)}$ is 0.9075 and $\frac{d(x_i,x_2)}{d(x_1,x_2)}$ is 0.2127, resulting in a membership value $U_{i,1}$ of 0.0206. This calculation is performed for all data points within the cluster.

4. Update Centroid

The next step involves updating the centroid positions v_k using Eq. (6). The updated centroids are obtained as follows:

$$v_1 = [0.5659, 0.3778, 0.6798, 0.6781],$$

$$v_2 = [0.2027, 0.5731, 0.1009, 0.0825].$$

5. Calculating Density

To calculate the density of a point x_i in the dataset X, the first step is to compute the Euclidean distance between each pair of points using Eq. (7). After calculating the Euclidean distances for all point pairs in the dataset, the number of other points within the neighborhood radius $r_{neighborhood} = 0.5$ can be computed using Eq. (9). Here, i is an indicator function that takes the value 1 if the condition in parentheses is met, and 0 otherwise. With $r_{neighborhood} = 0.5$, the results of the density calculation are presented in Table 3.

Table 3. Results of Density Calculation

Point ID	Density Value
1	49
2	48
…	…
150	88

Table 3 presents the density values of selected data points. A density threshold (α) is applied, where points with values below α are classified as outliers, while those above α are considered valid members of a cluster, it can be observed that data points with lower density values indicate a higher likelihood of being anomalies. By applying the threshold α, the algorithm effectively separates valid data from potential outliers, ensuring more compact and reliable clustering results. Next, the maximum density value $(\eta_{max} = 90)$, and the neighborhood membership value $\left(M^i_{neighborhood}\right)$is calculated using Eq. (8).

6. Outlier Identification.

The next step is to identify outliers using the threshold value $(\alpha = 0.5)$ and Eq. (10). To determine whether a data point x_i in the dataset is considered an outlier or not, the following procedure is used:

$$M^1_{neighborhood} > \alpha = 0.5444 > 0.5 \text{ (non-outlier)},$$

$$\vdots$$

$$M^{150}_{neighborhood} > \alpha = 0.9778 > 0.5 \text{ (non-outlier)}.$$

Using the threshold value $(\alpha = 0.5)$, 17 data points are identified as outliers in this dataset. The dataset, including the identified outliers, is visualized using a scatter plot in Fig. 3.

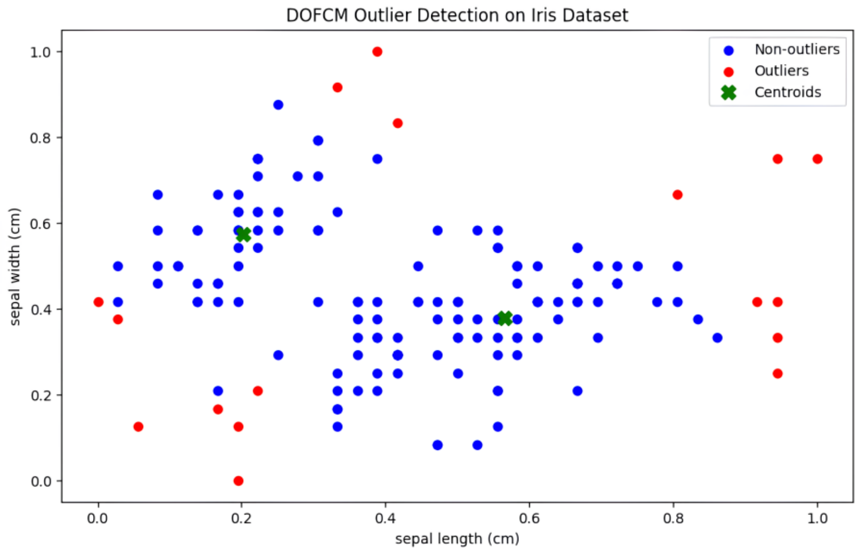

Figure 3. Visualization of Outlier Identification by DOFCM, Highlighting 17 Detected Outliers with Low Neighborhood Density that can Affect Clustering and Data Analysis

In Fig. 3, the blue dots represent data points considered as non-outliers by the DOFCM algorithm. These points have a sufficiently high neighborhood density, equal to or greater than the predefined threshold value α. In contrast, the red dots indicate 17 data points identified as outliers due to their low neighborhood density, falling below the threshold value α, and being far from most other data points. The green "X" marks represent the centroid positions of clusters found by the DOFCM algorithm, calculated as the average of all points within the cluster, taking into account the membership value of each point. Detecting and excluding such outliers is crucial for ensuring robust clustering performance and reliable interpretation of the dataset.

After outlier identification using DOFCM by analyzing the density of each data point, clustering quality evaluation is carried out using the SC. The process begins by reading and separating the dataset features, then standardizing them. The DOFCM algorithm is run through fuzzy membership initialization, centroid calculation, and updates until convergence. After the cluster is formed, the label is determined based on the largest membership, and then the SC is calculated. The results are in Table 8.

3.3 Implementation of the Particle Swarm Optimization (PSO)

Optimization of centroids based on the DOFCM method using the PSO algorithm aims to find the best cluster centers where there are outliers. The PSO processes include the initialization of particles and velocities, evaluating particles, searching for the best individual value (pBest), searching for the global best value (gBest), and updating particle positions and velocities. The detailed steps are as follows:

1. Initializing Particle Position and Velocity

The initial stage of the PSO process begins with setting the positions and velocities of the particles. Five sample data points from the independent variables of the Iris dataset are used to determine these initial positions. The position values vary across clusters, where Cluster 1 has a higher range (from 0.333 to 0.8083), while Cluster 2 has a lower range (from 0.1560 to 0.4402). In iteration 0, each particle is assigned an initial position across all dimensions, as shown in Table 4. The initial velocity (v_i) of all particles is set to zero in each dimension, indicating that there is no movement at the start of the process.

Table 4. Sample Data for PSO Process

x_1	x_2	x_3	x_4
0.2222	0.6250	0.0678	0.0417
0.1667	0.4167	0.0678	0.0417
…	…	…	…
0.1944	0.6667	0.0677	0.7916

The PSO process starts by initializing the positions and velocities of particles. A subset of sample data is shown in Table 4 to illustrate the initialization. Cluster 1 has higher position values (0.333–0.8083) compared to Cluster 2 (0.1560–0.4402). At iteration 0, each particle is assigned an initial position across all dimensions, with an initial velocity v_i set to zero. The next step is to determine the parameters to be used during the optimization process. The PSO parameters are presented in Table 5.

Tabel 5. PSO Parameters Used in the Experiment

Number of Particles	Number of Dimensions	Acceleration Coefficient (c_1)	Acceleration Coefficient (c_2)	Number of Clusters (k)
5	4	2	2	2

The PSO parameters were selected to balance solution quality and computational efficiency. A small particle size and iteration limit were chosen for faster convergence, while coefficients c_1 and c_2 = 2 are commonly used values that promote exploration and exploitation balance. At this stage, the initial positions of the particles are ready to be used in the iterative optimization process to update the cluster centers. The number of particles was set to five to balance computational efficiency with the ability to adequately explore the solution space. A small particle size reduces computation time, which is suitable for relatively low-dimensional datasets such as Iris, while still maintaining sufficient diversity among solutions. The number of dimensions was chosen as four, corresponding directly to the four independent variables of the Iris dataset (sepal length, sepal width, petal length, and petal width). The acceleration coefficients c_1 and c_2 were both set to 2, following common practice in PSO literature [38], to maintain a balance between exploration (searching new areas of the solution space) and exploitation (refining known good solutions). Finally, the constriction factor $k = 2$ was employed to stabilize the particle movements and avoid divergence, ensuring convergence toward optimal cluster centers. These parameter settings are consistent with standard PSO implementations and provide reliable convergence for clustering problems with outliers.

2. To calculate the Objective Function and Fitness Function

The next step is to find the objective value of the DOFCM method with the fuzzy parameter $(m) = 2$ using the following Equation.

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{C} u_{ij}^{m} \left\| x_i - c_j \right\|^2 . \tag{24}$$

The objective values represent the clustering quality obtained before the optimization process begins, where a lower value reflects a better initial clustering configuration. The fitness value of each particle is then calculated using Eq. (25):

$$f = \frac{1}{J_m + \varepsilon}, \tag{25}$$

where $\varepsilon = 1e - 8$. The combined objective and fitness values for each particle at iteration 0 are shown in Table 6. The objective values represent the clustering quality generated by the DOFCM method before optimization, where lower values indicate better initial clustering. The fitness values are computed directly from the corresponding objective values using Eq. (25), providing a normalized measure of each particle's clustering performance.

Table 6. Objective and Fitness Values at Iteration 0

Particle	Objective Value (unitless)	Fitness Value (unitless)
1	57.2611	0.0174
2	59.7840	0.0167
3	41.1348	0.0243
4	48.2075	0.0207
5	53.1815	0.0188

Table 6 presents the initial objective and fitness values for each particle at iteration 0. Particle 3 shows the lowest objective value (41.1348), suggesting the most favorable initial cluster arrangement. The corresponding fitness values provide a normalized measure of clustering quality and guide the Particle Swarm Optimization (PSO) process by highlighting particles with greater potential to influence the search toward optimal clustering solutions.

3. Initializing Local Best (pBest) and Global Best (gBest)

The pBest position values for each particle are calculated based on the initial particle values, and the pBest fitness values are determined using the fitness values of each particle. At iteration 0, the pBest position for each particle is calculated from the initial particle values, and the pBest fitness is computed using the fitness values of the particles. The results show that the particle x_2 in Cluster 1 has the best pBest value with a fitness of 0.0167, while particle x_3 has the highest pBest fitness of 0.0243, indicating variability in the optimality of particle positions across clusters. Subsequently, the gBest position is initialized from the pBest position with the highest strength, and the gBest strength starts with the maximum strength value. At iteration 0, the best gBest position for Cluster 1 is achieved with a fitness value of 0.0243 and a position of (0.3042, 0.5247, 0.4319, 0.2912). Meanwhile, Cluster 2 has a gBest position at (0.6118, 0.1394, 0.2921, 0.3663), but the gBest fitness value is not displayed, indicating that optimality focus is more prominent in Cluster 1 during this iteration.

4. Update Particle Velocity and Particle

After obtaining the pBest and gBest values for each particle, the next step is to update the velocity using the following Equation.

$$v_{ij}^{t+1} = \omega_t v_{ij}^{t} + c_1 r_{1j}^{t}\left(pBest_{ij}^{t} - x_{ij}^{t}\right) + c_2 r_2^{t}\left(gBest_{ij}^{t} - x_{ij}^{t}\right). \tag{26}$$

Given $\omega = 0.5$, $r_1 = 0.5$ and $r_2 = 0.5$, update the velocities of individuals v_1 through v_5 within two clusters c_1 and c_2 during the first iteration, with varying velocity updates in each dimension x_1 through x_4. Some individuals, such as v_3, experience no change in velocity (i.e., zero), while others, such as v_5, exhibit significant changes, particularly within specific clusters. Overall, this data illustrates the differences in velocity dynamics among individuals in the two clusters, reflecting their movement or positional changes within the multidimensional space.

5. Calculating Velocity Clamping.

In the process of constraining particle velocity, the newly computed velocity values are adjusted to ensure that the particles remain within the desired solution range, which in this case is the interval [-0.1, 0.1]. For example, in Iteration 1, the particle velocities for each cluster and variables x_1 through x_4 have been adjusted to fit within this interval. Velocity values exceeding the upper or lower bounds, respectively, as 0.1 or -0.0703, have been clipped to ensure compliance with the specified range, supporting better solution convergence.

6. Update Particle Position

To update the position of a particle at the next iteration, we use the Equation ($x_{ij}^{t+1} = x_{ij}(t) + v_{ij}^{t+1}$). In the first iteration, the particle's position is in cluster 1, with the initial position being [0.3042, 0.8507, 0.6319, 0.4986], and in cluster 2, the initial position is [0.2560, 0.1394, 0.1580, 0.7661]. Using the Equation provided and the given velocity data, the particle's position will be updated according to the changes calculated during that iteration.

7. Update the positions of pBest and gBest

By repeating the same computational steps, the process continues until it reaches the maximum number of iterations, which is set to 100. At the fifth iteration, the positions of pBest are obtained and presented in Table 7, respectively. The pBest positions are selected based on the best fitness value observed for each particle during the optimization process.

Tabel 7. pBest Iteration

Particle	**Cluster 1** (x_1, x_2, x_3, x_4)	**Cluster 2** (x_1, x_2, x_3, x_4)	**Fitness**
x_1	(0.0117, 0.4220, 0.2950, 0.4860)	(0.5772, 0.0437, 0.1230, 0.5586)	0.03386513
x_2	(0.3432, 0.7291, 0.6522, 0.8456)	(0.6924, 0.4299, 0.6729, 0.2753)	0.03386521
x_3	(0.3063, 0.7889, 0.4464, 0.7983)	(0.8224, 0.8575, 0.9166, 0.4309)	0.03386515
x_4	(0.3188, 0.5821, 0.3712, 0.6011)	(0.7056, 0.6884, 0.3746, 0.1668)	**0.03386525**
x_5	(0.4305, 0.1426, 0.8901, 0.3459)	(0.1545, 0.0255, 0.6458, 0.6369)	0.03386507

Based on Table 7, the best fitness value of 0.03386507 was achieved by particle x_5. Therefore, the pBest position of this particle is selected as the best cluster center for this iteration. Specifically, the pBest for Cluster 1 is located at $[0.4305, 0.1426, 0.8901, 0.3459]$, while for Cluster 2, it is at $[0.1545, 0.0255, 0.6458, 0.6369]$.

The results at iteration 5, the global best position (gBest) achieved a fitness value of 0.03386533, outperforming all individual personal best (pBest) fitness values. This gBest encodes two optimized cluster centers, each represented by four features: Cluster 1 is located at [0.5696, 0.4490, 0.6472, 0.4999], and Cluster 2 at [0.2028, 0.5368, 0.0422, 0.0336]. These final cluster centers will serve as the optimized solution in subsequent iterations of the DOFCM method, with the aim of enhancing cluster separation and improving the overall accuracy of the clustering results.

3.4 Implementation of Genetic Algorithm (GA)

In the DOFCM method, the GA is employed to optimize the cluster centers obtained from the initial clustering process. The GA performs a series of operations starting from random population generation, followed by fitness evaluation, selection, crossover, mutation, and elitism to iteratively improve the quality of clustering. Each step in the GA is designed to enhance the separation between clusters and reduce intra-cluster variance. The following subsections explain each step of the GA implementation in detail.

1. Random Population Generation

In the genetic algorithm, the formation of a new population is the initial stage where individuals or solutions are generated randomly. In this study, each individual consists of a number of genes calculated as the number of variables multiplied by the number of clusters. With cluster initialization $k = 2$, each individual has 8 genes. The initialization of individuals in the population involves assigning each individual to cluster center positions for each cluster. For example, for the Iris dataset, individual IND_1 has cluster centers for cluster 1 as left [0.3745, 0.9507, 0.7319, 0.5986] and for cluster 2 as [0.1560, 0.1559, 0.0580, 0.8661]. Other individuals, such as seperti IND_2, IND_3, IND_4, and IND_5, also have different cluster center positions.

2. Calculation of Fitness Value

In the early stages of a genetic algorithm, a new population is formed with individuals that are collections of random solutions. Each gene in an individual consists of multiple variables multiplied by the number of clusters, with an example of $k = 2$ resulting in 8 genes per individual. The distance between solutions is computed using the Euclidean distance formula, $d(x_i, y_j) = \sqrt{\sum_{b=1}^{p} (x_{ib} - y_{jb})^2}$. In this research, for a solution IND_1, the distance to data point 1 is $D1 = 0.9384$ and the distance to data point 2 is $D2 = 0.9509$. The minimum distance is calculated for each data point, resulting in a total minimum distance of 103.7533. The fitness value of IND_1 is calculated as follows $F = \frac{1}{D} = \frac{1}{103.7533} = 0.0096$.

3. Individual Selection Process

In the individual selection stage of a genetic algorithm, a roulette wheel is used to select individuals based on their fitness values. Each individual has a probability in the range of 0-1, calculated from their relative fitness value. In this study, IND_3 has a probability of 0.2428 with a cumulative range of 0.3415-0.5842, and IND_5 has a probability of 0.2047 with a cumulative range of 0.7954-1.0000. Selection is performed by generating a random number p and comparing it to the cumulative ranges. The selected individuals, such as IND_3 and IND_4 in this study, will serve as parents to produce new offspring.

4. Crossover Reproduction Process

Crossover is the initial step in genetic reproduction where two individuals (Parent 1 and Parent 2) are randomly selected to produce offspring. The number of offspring generated depends on the crossover rate (Cr) and the population size. In this study, the crossover rate was set to $Cr = 0.2$, which means that two offspring were produced because $Cr \times Population\ Size = 2$. A moderate crossover probability was chosen to balance exploration and exploitation, helping preserve high-quality gene segments while ensuring population diversity.

The crossover rate was set to 0.2 to balance exploration and exploitation [40]. This value is suitable for clustering problems, where maintaining stable cluster center information is important. A low crossover probability helps preserve high-quality gene segments while still ensuring diversity in the population, and moderate crossover rates support stable convergence and reduce the risk of premature stagnation [40]. Additionally, there are 3 crossover points chosen randomly. During the crossover process, genetic material from the parents is exchanged at predefined crossover points. To simplify illustration, one example of crossover at Gene Index 1 is presented in Table 8, where shading is used to highlight the exchanged segments between parents.

Table 8. Example of Crossover between Two Parents Illustrating how Gene Exchange Enhances Population Diversity

Gene Index	Parent 1	Parent 2	Offspring 1 (OF_1)	Offspring 2 (OF_2)
1*	0.3042	0.0650	0.3042	0.0650
2	0.5247	0.9488	0.9488	0.5247
3	0.4319	0.9656	0.4319	0.9656
4	0.2912	0.8083	0.8083	0.2912

From Table 8, it can be observed that Offspring 1 inherits the first segment from Parent 1 and the following segments from Parent 2, while Offspring 2 follows the opposite pattern. The asterisk (*) indicates the crossover point where genetic segments are exchanged, highlighting how genetic material is recombined between parents. This recombination clearly demonstrates how crossover generates new genetic combinations, thereby enhancing population diversity and reducing the risk of premature stagnation in the optimization process.

5. Gene Mutation Reproduction Process

In the gene mutation phase of a genetic algorithm, an individual is randomly selected for mutation, resulting in offspring based on the mutation rate (Mr) and population size. With an Mr of 0.2, two offspring are produced by mutating two genes. In the first iteration, IND_4 and IND_1 are chosen for mutation at gene positions 7 and 4, respectively. A random value r of 0.0901 is added to these genes, resulting in two new offspring with mutated genes.

6. Elitism Process

At this stage, the population size increased from 10 to 14 individuals after the reproduction process, which generated 4 new offspring. To maintain a consistent population size, the elitist selection method (elitism) was applied by preserving individuals with the best fitness values. Table 9 presents a summary of the fitness values of all individuals, including both the original and offspring, after reproduction.

Table 9. Population After Selection and Gene Reproduction

Individual	Chromosome	Fitness
$\boldsymbol{IND_3}$	$\begin{bmatrix} \mathbf{0.3042} & \mathbf{0.5247} & \mathbf{0.4319} & \mathbf{0.2912} \\ \mathbf{0.6118} & \mathbf{0.1394} & \mathbf{0.2921} & \mathbf{0.3663} \end{bmatrix}$	**1.472**
IND_5	$\begin{bmatrix} 0.0650 & 0.9488 & 0.9656 & 0.8083 \\ 0.3046 & 0.0976 & 0.6842 & 0.4401 \end{bmatrix}$	1.242
IND_4	$\begin{bmatrix} 0.4560 & 0.7851 & 0.1996 & 0.5142 \\ 0.5924 & 0.0464 & 0.6075 & 0.1705 \end{bmatrix}$	1.228
IND_1	$\begin{bmatrix} 0.3745 & 0.9507 & 0.1996 & 0.5142 \\ 0.5924 & 0.0464 & 0.6075 & 0.1705 \end{bmatrix}$	1.063
OFF_1	$\begin{bmatrix} 0.3042 & 0.9488 & 0.4319 & 0.2912 \\ 0.6118 & 0.1394 & 0.6842 & 0.4401 \end{bmatrix}$	1.258
OFF_2	$\begin{bmatrix} 0.0650 & 0.5247 & 0.9656 & 0.8083 \\ 0.3046 & 00976 & 0.2921 & 0.3663 \end{bmatrix}$	1.251
OFF_3	$\begin{bmatrix} 0.4560 & 0.7851 & 0.1996 & 0.5142 \\ 0.5924 & 0.0464 & 0.6741 & 0.1705 \end{bmatrix}$	1.228
OFF_4	$\begin{bmatrix} 0.3745 & 0.9507 & 0.7319 & 0.6791 \\ 0.1560 & 0.1559 & 0.0580 & 0.8661 \end{bmatrix}$	1.039

At the end of the optimization process using the Genetic Algorithm (GA), the best cluster centers were successfully obtained after five generations. Each individual in the population encodes two cluster centers, each represented by four features, forming a total of eight genes. The final solution was derived from the individual (IND_3) with the highest fitness value of 0.1472. The corresponding cluster centers obtained using the DOFCM method are as follows:

$$\begin{bmatrix} 0.3042 & 0.5247 & 0.4319 & 0.2912 \\ 0.6118 & 0.1394 & 0.2921 & 0.3663 \end{bmatrix}.$$

These cluster centers represent the best outcome from the optimization process and are used as the final solution for clustering in the DOFCM-GA framework.

3.5 Cluster Evaluation Process

The clustering results were evaluated using the SC method to assess clustering quality with the assistance of computational tools across five datasets. After identifying outliers using DOFCM through local density analysis, the clustering process proceeded with feature standardization and initialization of fuzzy memberships. Cluster centroids were computed and updated iteratively until convergence. The final cluster label was assigned based on the highest membership value.

The centroids resulting from DOFCM represent the spatial position of each cluster in the feature space. The distance to the centroid plays a significant role in determining fuzzy memberships, which ultimately affects the SC value. The combination of SC values and centroid positions highlights DOFCM's effectiveness in handling datasets with varying complexity and distribution characteristics.

To further demonstrate the effectiveness of the optimization-based clustering, we provide a comparative analysis of SC values obtained by DOFCM, DOFCM-PSO, and DOFCM-GA. Table 10 summarizes the results.

Table 10. SC Values of DOFCM, DOFCM-PSO, and DOFCM-GA

No	Dataset	DOFCM	DOFCM-PSO	DOFCM-GA
1	Iris	0.4574	0.6029	**0.6291**
2	Ionosphere	0.2678	0.3116	0.3136
3	Diabetes	0.1794	0.2423	0.2532
4	Sonar	0.1350	0.1512	0.1631
5	Wine	0.1304	0.2759	0.2958

The results in Table 10 show that both DOFCM-PSO and DOFCM-GA improve clustering quality compared to the baseline DOFCM. Among the two optimization methods, GA consistently achieves the highest SC values across all datasets, demonstrating its effectiveness in generating more compact and well-separated clusters. The most notable gain is observed in the Wine dataset, where GA achieves the best improvement over both DOFCM and PSO.

A particular case is found in the Iris dataset, where PSO performs relatively well. The SC value of DOFCM-PSO (0.6029) is close to the GA result (0.6291). This outcome can be attributed to the simplicity of the Iris dataset, which has only four features and well-separated natural classes. The lower dimensionality and clearer structure allow PSO to converge toward near-optimal cluster centers, leading to well-defined clusters despite its tendency toward premature convergence. In contrast, the Sonar, Ionosphere, and Diabetes datasets have higher dimensionality and noisier features, which make clustering more difficult. Under these conditions, PSO struggles to balance exploration and exploitation, often producing clusters that are less compact and overlapping, reflected in their relatively low SC values. GA, on the other hand, benefits from crossover and mutation operations that preserve population diversity, enabling it to avoid local optima and refine cluster boundaries more effectively.

Overall, the comparison highlights that GA is superior for complex datasets such as Wine, Sonar, Ionosphere, and Diabetes, while PSO is only competitive in simpler datasets like Iris. This suggests that the choice of optimization strategy should be aligned with dataset characteristics to achieve optimal clustering performance. Moreover, GA demonstrates an advantage in producing denser and more compact clusters, as evidenced in the Iris and Wine datasets, where the separation between groups appears clearer. In contrast, PSO often yields clusters that are less compact, with greater overlap between boundaries, particularly visible in the Sonar and Ionosphere datasets that contain high-dimensional and noisy features. Despite these differences, both optimization strategies are able to capture the general data structure, though the presence of overlapping regions suggests that the intrinsic complexity of some datasets still limits cluster distinctiveness. This highlights that while GA provides better refinement in centroid positioning, dataset characteristics remain a critical factor influencing clustering quality

Although the SC value provides a general overview of the compactness and fragmentation of the clusters, the following Table 11 presents an analysis of the number of outliers in each cluster, providing additional insight into the robustness of the clustering method to anomalous data.

Tabel 11. Number of Outliers Detected in each Cluster of Dataset

Dataset	Cluster	Outliers DOFCM	Outliers DOFCM-PSO	Outliers DOFCM-GA
Iris	1	4	2	0
	2	2	1	0
	3	1	2	0
Wine	1	3	2	1
	2	2	1	0
	3	1	1	0
Sonar	1	10	4	2
	2	2	3	1
Diabetes	1	21	15	5
	2	0	5	2
Ionosphere	1	14	6	2
	2	0	7	2

From Table 11, it is evident that the number of outliers differs across datasets and optimization strategies. For example, in the Iris dataset, DOFCM identified several outliers in all clusters, while DOFCM-PSO slightly reduced this number, and DOFCM-GA successfully eliminated outliers in most clusters. In the Wine and Ionosphere datasets, GA consistently detected fewer outliers compared to PSO, reflecting its ability to generate more representative cluster centers. Meanwhile, the Sonar and Diabetes datasets, which contain more noise and higher complexity, still exhibited outliers under all methods, although GA managed to minimize their presence. These findings indicate that GA generally improves clustering robustness by reducing anomalous points across different datasets.

To further evaluate model performance, empirical runtime measurements were conducted. Table 12 presents the average computational time of DOFCM, DOFCM-PSO, and DOFCM-GA across the five benchmark datasets.

Tabel 12. **Average Computational Time of DOFCM, DOFCM-PSO, and DOFCM-GA**

Dataset	DOFCM (Second)	DOFCM-PSO (Second)	DOFCM-GA (Second)
Iris	0.42	0.88	1.12
Wine	0.65	1.34	1.81
Sonar	0.78	1.52	2.09
Diabetes	1.15	2.03	2.67
Ionosphere	0.89	1.76	2.21

As shown in Table 12, the baseline DOFCM algorithm requires the shortest computational time due to its simpler update rules. When optimization is introduced, PSO adds additional computation but remains more efficient than GA across all datasets. GA, while slightly more time-consuming, consistently provides more stable clustering results and higher Silhouette Coefficient values. This confirms the trade-off between runtime and clustering robustness, where GA sacrifices speed for improved cluster quality.

In this study, the GA demonstrated superior performance compared to PSO across most datasets. This difference can be explained theoretically through the distinct mechanisms of exploration and exploitation in both algorithms. PSO exhibits strong exploitation ability, as particles tend to follow the global and individual best positions, enabling rapid convergence. However, this property makes PSO prone to being trapped in local optima, particularly when dealing with high-dimensional datasets or complex data distributions.

In contrast, GA incorporates crossover and mutation operations that preserve population diversity. These mechanisms enhance its exploration ability, allowing GA to search a wider solution space and avoid premature stagnation. With a better balance between exploration and exploitation, GA is able to generate more representative cluster centers and achieve higher SC values compared to PSO.

4. CONCLUSION

This study contributes to advancing clustering and outlier detection by integrating meta-heuristic optimization strategies into the DOFCM method. The comparative analysis demonstrated that DOFCM optimized with GA consistently outperforms DOFCM-PSO, as evidenced by higher SC values across all five benchmark datasets. On average, GA improved SC values by approximately 0.02–0.03 points (e.g., Iris increased from 0.6029 with PSO to 0.6291 with GA, and Wine from 0.2759 to 0.2958). In addition, the comparative evaluation of computational time and the number of detected outliers further supports these findings. For example, GA required 2.21 seconds on the Ionosphere dataset compared to 1.76 seconds with PSO, yet it reduced outliers from 13 (PSO) to 2 (GA) while still achieving a higher SC value (0.3136 vs. 0.3116). A similar trend was observed in the Diabetes dataset, where GA decreased outliers from 20 to 7 with a modest SC improvement. This highlights the trade-off between runtime efficiency (PSO) and clustering robustness (GA). Nevertheless, this study has limitations. The datasets used are relatively small and may restrict the generalizability of results. Moreover, the performance of PSO and GA is sensitive to parameter settings, and the method has not been tested on real-time or streaming data, which is increasingly relevant in practical anomaly detection scenarios. For future research, it is recommended to apply this method to larger and more diverse real-world datasets (e.g., healthcare records, financial transactions, IoT sensor data) and to explore alternative optimizers such as GWO, DE, or hybrid GA–PSO approaches. Integration with deep learning–based clustering methods, such as autoencoder-assisted clustering, is also a promising direction for handling high-dimensional and large-scale datasets. In practical terms, the findings of this study can support practitioners in domains such as healthcare, finance, and IoT-based systems, where robust outlier detection is essential for identifying anomalies in patient data, detecting fraudulent transactions, or monitoring sensor reliability.

Author Contributions

Sintia Afriyani: Conceptualization, Methodology, Software, Formal Analysis, Investigation, Writing Original Draft, Visualization. Rohmatul Fajriyah: Supervision, Validation, Resources, Writing, Review, and Editing. All authors discussed the results and approved the final version of the manuscript.

Funding Statement

This research was funded by the Statistics Department, Universitas Islam Indonesia, Yogyakarta.

Acknowledgment

The author would like to express his deepest gratitude to the Statistics Department, Universitas Islam Indonesia, for providing financial support for this research. The author would also like to thank the support of the Data Science Laboratory of Universitas Islam Indonesia, where this research was conducted.

Declarations

The authors declare no conflicts of interest.

Declarations of Generative AI and AI-assisted Technologies

ChatGPT was used solely for language refinement, including grammar, spelling, and clarity. The scientific content, analysis, interpretation, and conclusions were developed entirely by the authors. All final text was reviewed and approved by the authors.

REFERENCES

[1] S. Munirathinam, *INDUSTRY 4 . 0 : INDUSTRIAL INTERNET OF THINGS (IIOT)*, 1st ed. Elsevier Inc., 2020.

[2] G. Aceto, V. Persico, and A. Pescapé, “INTEGRATION INDUSTRY 4 . 0 AND HEALTH : INTERNET OF THINGS, BIG DATA, AND CLOUD COMPUTING,” *J. Ind. Inf. Integr.*, vol. 18, no. February, p. 100129, 2020. doi: https://doi.org/10.1016/j.jii.2020.100129.

[3] M. Lavasani, R. Sotudeh-gharebagh, and R. Zarghami, “BIG DATA ANALYTICS OPPORTUNITIES FOR APPLICATIONS IN PROCESS ENGINEERING,” *Rev. Chem. Eng.*, vol. 39, no. 3, pp. 479–511, 2023. doi: https://doi.org/10.1515/revce-2020-0054.

[4] A. Boukerche, L. Zheng, and O. Alfandi, “OUTLIER DETECTION: METHODS, MODELS, AND CLASSIFICATION,” *ACM Comput. Surv.*, vol. 53, no. 3, 2020. doi: https://doi.org/10.1145/3381028.

[5] K. Yu, W. Shi, and N. Santoro, “DESIGNING A STREAMING ALGORITHM FOR OUTLIER DETECTION IN DATA MINING — AN INCREMENTAL APPROACH,” *Sensors*, vol. 20, no. 5, pp. 1–24, 2020. doi: https://doi.org/10.3390/s20051261.

[6] A. Abid, S. El, and A. Kachouri, “WIRELESS SENSOR NETWORKS,” *Computing*, vol. 103, no. 10, pp. 2275–2292, 2021. doi: https://doi.org/10.1007/s00607-021-00939-5.

[7] M. Boersma, K. Manoorkar, A. Palmigiano, and V. Universiteit, “OUTLIER DETECTION USING FLEXIBLE CATEGORISATION AND INTERROGATIVE AGENDAS,” *Decis. Support Syst.*, vol. 180, no. 5, pp. 1–28, 2024. doi: https://doi.org/10.1016/j.dss.2024.114196.

[8] W. Wang, X. Hu, and Y. Du, “ALGORITHM OPTIMIZATION AND ANOMALY DETECTION SIMULATION BASED ON EXTENDED JARVIS-PATRICK CLUSTERING AND OUTLIER DETECTION,” *Alexandria Eng. J.*, 2021. doi: https://doi.org/10.1016/j.aej.2021.08.009.

[9] P. Filzmoser and M. Gregorich, “MULTIVARIATE OUTLIER DETECTION IN APPLIED DATA ANALYSIS : GLOBAL, LOCAL, COMPOSITIONAL, AND CELLWISE OUTLIERS,” *Math. Geosci.*, vol. 52, no. 8, pp. 1049–1066, 2020. doi: https://doi.org/10.1007/s11004-020-09861-6.

[10] A. Conde, U. Mori, and J. A. Lozano, “A REVIEW ON OUTLIER/ANOMALY DETECTION IN TIME SERIES DATA,” 2020.

[11] M. Al Samara, I. Bennis, A. Abouaissa, and P. Lorenz, “A SURVEY OF OUTLIER DETECTION TECHNIQUES IN IOT : REVIEW AND CLASSIFICATION,” *J. Sens. Actuator Netw.*, vol. 11, no. 1, 2022. doi: https://doi.org/10.3390/jsan11010004.

[12] F. Ridzuan, W. Mohd, and N. Wan, “DIAGNOSTIC ANALYSIS FOR OUTLIER DETECTION IN BIG DATA ANALYTICS,” *Procedia Comput. Sci.*, vol. 197, pp. 685–692, 2022. doi: https://doi.org/10.1016/j.procs.2021.12.189.

[13] M. C. Massi, F. Ieva, and E. Lettieri, “DATA MINING APPLICATION TO HEALTHCARE FRAUD DETECTION : A TWO-STEP UNSUPERVISED CLUSTERING METHOD FOR OUTLIER DETECTION WITH ADMINISTRATIVE DATABASES,” vol. 9, pp. 1–11, 2020.

[14] N. H. M. M. Shrifan, M. F. Akbar, N. Ashidi, and M. Isa, “AN ADAPTIVE OUTLIER REMOVAL AIDED K-MEANS CLUSTERING ALGORITHM,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 8, pp. 6365–6376, 2022. doi: https://doi.org/10.1016/j.jksuci.2021.07.003.

[15] H. E. G. Lopes and M. de S. Gosling, “CLUSTER ANALYSIS IN PRACTICE : DEALING WITH OUTLIERS IN,” *Rev. Adm. Contemp. J. Contemp. Adm.*, vol. 25, no. 1, pp. 1–19, 2021. doi: https://doi.org/10.1590/1982-7849rac2021200081.

[16] M. Corain, P. Torino, and P. Torino, “DBSCOUT : A DENSITY-BASED METHOD FOR SCALABLE OUTLIER DETECTION IN VERY LARGE DATASETS,” *Publ. IEEE*, 2021. doi: https://doi.org/0.1109/ICDE51399.2021.00011.

[17] O. Alghushairy, R. Alsini, T. Soule, and X. Ma, “A REVIEW OF LOCAL OUTLIER FACTOR ALGORITHMS FOR OUTLIER DETECTION IN BIG DATA STREAMS,” *big data Cogn. Comput.*, vol. 5, no. 1, pp. 1–24, 2021. doi: https://doi.org/10.3390/bdcc5010001.

[18] A. Nowak-brzezi, “QUALITATIVE DATA CLUSTERING TO DETECT OUTLIERS,” *entropy*, vol. 23, no. 7, p. 869, 2021. doi: https://doi.org/10.3390/e23070869.

[19] H. Yadav, J. Singh, and A. Gosain, “EXPERIMENTAL ANALYSIS OF FUZZY CLUSTERING TECHNIQUES FOR OUTLIER DETECTION,” *Procedia Comput. Sci.*, vol. 218, no. 3, pp. 959–968, 2023. doi: https://doi.org/10.1016/j.procs.2023.01.076.

[20] W. Hyun, S. Sanghoun, and C. W. Ahn, “METAHEURISTIC-BASED TIME SERIES CLUSTERING FOR ANOMALY DETECTION IN MANUFACTURING INDUSTRY,” *Appl. Intell.*, vol. 53, no. 6, pp. 21723–21742, 2023. doi: https://doi.org/10.1007/s10489-023-04594-5.

[21] M. H. Sulaiman, Z. Mustaffa, M. Ahmed, Q. Li, L. Guo, and X. Wang, “GREY WOLF OPTIMIZER AND OTHER METAHEURISTIC OPTIMIZATION TECHNIQUES WITH IMAGE PROCESSING AS THEIR APPLICATIONS : A REVIEW,” *Mater. Sci. Eng.*, vol. 1136, no. 1, pp. 0–17, 2021. doi: https://doi.org/10.1088/1757-899X/1136/1/012053.

[22] R. P. Parouha and P. Verma, “A SYSTEMATIC OVERVIEW OF DEVELOPMENTS IN DIFFERENTIAL EVOLUTION AND PARTICLE SWARM OPTIMIZATION WITH THEIR ADVANCED SUGGESTION,” *Appl. Intell.*, vol. 52, no. 1, pp. 10448–10492, 2022. doi: https://doi.org/10.1007/s10489-021-02803-7.

[23] U. M. L. Repository, “IRIS DATASET,” *Kaggle*, 2016. https://www.kaggle.com/datasets/uciml/iris (accessed Sep. 22, 2024).

[24] U. I. M. L. Repository, “RED WINE QUALITY,” *Kaggle*, 2009. https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009 (accessed Sep. 22, 2024).

[25] M. H. Shawon, “SONAR DATA,” *Kaggle*, 2024. https://www.kaggle.com/datasets/mahmudulhaqueshawon/sonar-data (accessed Sep. 22, 2024).

[26] Mathchi, “DIABETES DATA SET,” *Kaggle*, 2020. https://www.kaggle.com/datasets/mathchi/diabetes-data-set (accessed Sep. 22, 2024).

[27] V. G. Sigillito, “IONOSPHERE,” *Kaggle*, 1989. https://www.kaggle.com/datasets/jamieleech/ionosphere (accessed Sep. 22, 2024).

[28] A. Karami and C. Igbokwe, “THE IMPACT OF BIG DATA CHARACTERISTICS ON CREDIT RISK ASSESSMENT,” *Int. J. Data Sci. Anal.*, vol. 2, no. March, pp. 1–13, 2025. doi: https://doi.org/10.1007/s41060-025-00753-8.

[29] G. P. Selvarajan, “HARNESSING AI-DRIVEN DATA MINING FOR PREDICTIVE INSIGHTS : A FRAMEWORK FOR ENHANCING DECISION- MAKING IN DYNAMIC DATA ENVIRONMENTS,” *Int. J. Creat. Res. Thoughts*, vol. 9, no. 2, pp. 5476–5486, 2021.

[30] M. Shantal, Z. Othman, and A. A. Bakar, “A NOVEL APPROACH FOR DATA FEATURE WEIGHTING USING CORRELATION COEFFICIENTS AND MIN–MAX NORMALIZATION,” *Symmetry (Basel).*, vol. 15, no. 12, 2023. doi: https://doi.org/10.3390/sym15122185.

[31] S. S. Nagari, L. Inayati, U. Airlangga, E. Java, M. V. Midwife, and E. Java, “IMPLEMENTATION OF CLUSTERING USING K-MEANS METHOD TO DETERMINE NUTRITIONAL STATUS,” *J. Biometrika dan Kependud.*, vol. 9, no. 1, pp. 62–68, 2020. doi: https://doi.org/10.20473/jbk.v9i1.2020.62.

[32] A. Shahraki, A. Taherkordi, Ø. Haugen, and F. Eliassen, “CLUSTERING OBJECTIVES IN WIRELESS SENSOR NETWORKS : A SURVEY AND RESEARCH DIRECTION ANALYSIS,” *Comput. Networks*, p. 107376, 2020. doi: https://doi.org/10.1016/j.comnet.2020.107376.

[33] C. Arimie, E. Biu, and M. Ijomah, “OUTLIER DETECTION AND EFFECTS ON MODELING,” *Open Access Libr. J.*, vol. 7, no. 9, pp. 1–30, 2020. doi: https://doi.org/10.4236/oalib.1106619.

[34] S. S. Kumar, S. T. Ahmed, Q. Xin, S. Sandeep, and M. Madheswaran, “UNSTRUCTURED ONCOLOGICAL IMAGE CLUSTER IDENTIFICATION USING IMPROVED UNSUPERVISED CLUSTERING TECHNIQUES,” *Comput. Mater. Contin.*, vol. 72, no. 1, pp. 281–299, 2022. doi: https://doi.org/10.32604/cmc.2022.023693.

[35] K. Bhalla and A. Gosain, “A NOVEL HYBRIDIZED FUZZY CLUSTERING TECHNIQUE FOR SEGMENTATION OF NOISY MAMMOGRAM IMAGES,” *J. Inf. Optim. Sci.*, vol. 46, no. 2, pp. 383–401, 2025. doi: https://doi.org/10.47974/JIOS-1922.

[36] T. M. Shami, A. A. El-saleh, and S. Member, “PARTICLE SWARM OPTIMIZATION : A COMPREHENSIVE SURVEY,” *IEEE Access*, vol. 99, no. 1, pp. 10031–10061, 2022. doi: https://dx.doi.org/10.1109/ACCESS.2022.3142859.

[37] A. G. Gad, “PARTICLE SWARM OPTIMIZATION ALGORITHM AND ITS APPLICATIONS: A SYSTEMATIC REVIEW,” *Arch. Comput. Methods Eng.*, vol. 29, no. 5, pp. 2531–2561, 2022. doi: https://doi.org/10.1007/s11831-021-09694-4.

[38] H. A. Alsattar and A. A. Z. B. B. Zaidan, “NOVEL META-HEURISTIC BALD EEAGLE SEARCH OPTIMISATION ALGORITHM,” *Artif. Intell. Rev.*, vol. 53, no. 7, pp. 2237–2264, 2019. doi: https://doi.org/10.1007/s10462-019-09732-5.

[39] T. Alam, S. Qamar, A. Dixit, and M. Benaida, “GENETIC ALGORITHM : REVIEWS , IMPLEMENTATIONS , AND APPLICATIONS,” *Int. J. Eng. Pedagog.*, vol. 10, no. 6, pp. 57–77, 2020. doi: https://doi.org/10.3991/ijep.v10i6.14567.

[40] S. Damodaran, *GENETIC ALGORITHMS (GAS); MIMICKING THE NATURAL SELECTION PROCESS TO ARRIVE AT OPTIMUM DESIGN SOLUTION (S)*, no. February. 2022.

[41] H. B. Tambunan, “ELECTRICAL PEAK LOAD CLUSTERING ANALYSIS USING K-MEANS ALGORITHM AND SILHOUETTE COEFFICIENT,” in *International Conference on Technology and Policy in Energy and Electric Power (ICT-PEP)*, 2020, pp. 258–262. doi: https://doi.org/10.1109/ICT-PEP50916.2020.9249773.