EURASIAN JOURNAL OF MATHEMATICAL AND COMPUTER APPLICATIONS
ISSN 2306–6172
Volume 13, Issue 3 (2025) 36 – 49

LEAST SQUARE-BASED DIFFERENTIAL EVOLUTION ALGORITHM FOR n-DIMENSIONAL DATA CLUSTERING PROBLEM

Latief M.A., Pandiya R. [1] , Putri A.L.R.

Abstract The K-Means algorithm is commonly used for data clustering due to its simplicity and effective implementation. However, it has certain drawbacks. One of the main issues with K-Means is the random process involved in selecting the initial centroid, which can lead to varying results. To address this issue, many researchers have developed methodologies to enhance the performance of K-Means, aiming to achieve the optimal centroid. Data clustering is essentially a global optimization problem, making the objective function crucial. The traditional K-Means algorithm uses the Euclidean distance function as its objective function. However, this metric can sometimes result in non-separable clusters. To mitigate this effect, least squares terms are introduced. The objective function, dependent on the centroids, needs to be minimized using an appropriate method. This paper explores the impact of using metaheuristic approaches (such as differential evolution), a combination of differential evolution and K-Means, and the conventional K-Means algorithm on the formulated objective function to determine the optimal centroid. Computational performance data, such as silhouette score and CPU time, are collected during the computational phase. The results show that the combination of differential evolution and K-Means is more efficient based on these indicators. Additionally, numerical experiments are performed to solve the clustering problem.

Key words: global optimization, clustering, differential evolution, unsupervised machine learning.

AMS Mathematics Subject Classification: 49M37, 65K10, 90C30, 68T09.

DOI: 10.32523/2306-6172-2025-13-3-36-49.

1 Introduction

Clustering data is a data mining technique utilized to organize data points by identifying similarities within a dataset. As stated in other definitions [1, 2, 3, 4], clustering involves categorizing unlabeled data into clusters or groups based on similarities. Clustering plays a crucial role in organizing unlabeled data into labeled categories, even in the absence of evident pattern similarities. This approach in data mining can be applied to areas such as recommender systems, image segmentation, and customer segmentation. A variety of established methods exist to suit various data types and applications, including partitional clustering, hierarchical clustering, fuzzy clustering, distribution model-based clustering, and density-based clustering [5, 6].

K-Means is a popular non hierarchical clustering algorithm due to its computational speed and intuitive nature by minimizing its objective function in its simplest form [7]. K-Means is often used because the algorithm is simple and easy to implement. The way K-Means works starts by selecting k initial points as the initial centroid of the existing points. Then at each iteration, each point will be calculated its distance to the centroid using Euclidean Distance. The class grouping of each point is determined based on the closest distance from the cluster center. However, the distance between data points to the centroid calculated by Euclidean distance triggers a poor selection of centroid or cluster center and is stuck at the local minimum solution so as not to get the most optimal solution from the data [8, 9].

[1]Corresponding Author.

Moreover, the authors in [10] explained that the initial cluster centroid in K-Means is determined randomly causing unexpected convergence problems. Therefore, centroid selection for K-Means algorithm becomes an important issue to be investigated in order to improve the performance of K-Means algorithm. One of the many methods that can be used to overcome the shortcomings of the K-Means algorithm is the optimization approach. The optimization strategy is generally employed by transforming the clustering issue into an objective function, where observed data is integrated, and the variable to be identified is the centroid. This optimization strategy is categorized into two types: the metaheuristic optimization approach and the deterministic optimization approach. Deterministic methods often necessitate numerous assumptions for their application, unlike the metaheuristic approach, which is generally more adaptable. As a result, this study will utilize the metaheuristic approach. A metaheuristic algorithm is an advanced search strategy designed to solve optimization problems efficiently by exploring the solution space through specialized methods [11]. A particularly effective algorithm within metaheuristic optimization is the Differential Evolution algorithm.

Differential Evolution was also used in previous studies to optimize an algorithm. The Authors in [12] applied Differential Evolution to the LightGBM algorithm to obtain the optimal combination of parameter structure and model performance. The conclusion in the study resulted that the DE algorithm (able to find the optimal optimization quickly, which is only about 15 generations of optimization. Both previous studies only applied the algorithm to the data and did not modify the algorithm such as in [13]. The modifications made in the study aimed to improve the performance of the Differential Evolution algorithm by utilizing the Fuzzy C-Means clustering approach in numerical optimization. As a result, the integration of fuzzy clustering and adaptive strategies in the DE algorithm can produce better and more efficient solutions in numerical optimization. The authors in [14] used a different type of Differential Evolution method, namely Multi-Objective Differential Evolution (MODE) used to improve the feature selection process in emotion classification. The results obtained in the study stated that MODE was able to achieve higher emotion recognition accuracy by using a smaller number of features.

This research builds on previous studies by applying the Differential Evolution (DE) technique to establish the initial centroids of the K-Means algorithm. Traditionally, initial centroids are chosen randomly, which can sometimes lead the K-Means algorithm to converge to a local optimum rather than a global one. To address this, a more systematic approach, such as DE, can be employed to select initial centroids. The issue of selecting initial centroids for the K-Means algorithm can be effectively tackled using metaheuristic optimization methods like Differential Evolution. Hence, this study explores the process of determining initial centroids via the DE algorithm, which can then be used as starting points in the K-Means algorithm, thereby enhancing its ability to reach a global optimum and boosting its overall efficiency. By enhancing the K-Means algorithm through Differential Evolution, this study aims to develop a clustering algorithm that demonstrates more dependable outcomes and enables more precise clustering. The steps undertaken in this research are as follows:

1. Define the objective function, utilizing least squares.
2. Apply Differential Evolution to optimize the objective function established in stage 1.
3. Utilize the output of stage 2, specifically the initial centroid, as the initial center for the K-Means algorithm.

The results of this research are intended to offer insight and solutions to scholars in the clustering domain, contributing to the creation of models that yield accurate clustering decisions.

The structure of this paper is as follows: Initially, the first section provides an introduction

to the discussed issue. Following this, the second section outlines the assumptions constraining the initial centroid's determination. Section 3 elaborates on the Differential Evolution algorithm. Subsequently, sections 4 and 5 offer an analysis of the outcomes derived from computational experiments and their comparisons. Finally, the research findings are summarized in the concluding section.

2 Problem formulation

In this section, we will examine the presumptions regarding the function's limitations in determining the initial partition. Consider a set $Q = \{q_i \in R^n : i = 1, \ldots, m\} \subset R^n$ divided into k mutually exclusive clusters $\pi_1, \ldots, \pi_k$, where $1 \leq m$, as follows:

$$\bigcup_{i=1}^{k} \pi_i = Q, \quad \pi_i \cap \pi_j = \emptyset, \quad i \neq j, \quad |x_j| \geq 1, \tag{1}$$

where $j = 1, \ldots, k$, the partition of Eq.(1) will be represented as $\prod Q = \pi_1, \ldots, \pi_k$, and each $\pi_1, \ldots, \pi_k$ subset will be termed a cluster in $\mathbb{R}^{\ltimes}$. If $d : \mathbb{R}^{\ltimes} \times \mathbb{R}^{\ltimes} \to [0, +\infty)$ is a distance-like function, then by minimizing the distance condition for each cluster $\pi_j \in \prod$, we can determine its center c_j, which is defined by

$$c_j = c(\pi_j) := \underset{x \in C_j}{\operatorname{argmin}} \sum_{a_i \in \pi_j} d(x, a_i). \tag{2}$$

In (2) C_j is $\operatorname{conv}(\pi_j)$. Let $f : P(Q, k) \to [0, +\infty)$ be defined as an objective function over the set of all partitions $P(Q, k)$ of Q containing k clusters,

$$f\left(\prod\right) = \sum_{j=1}^{k} \sum_{a_i \in \pi_j} d(c_j, a_i), \tag{3}$$

so then optimal partition can defined as

$$f\left(\overset{*}{\prod}\right) = \min_{\prod \in P(Q,k)} f\left(\prod\right).$$

Alternatively, for a specific set of centers $c_1, \ldots, c_k \in \mathbb{R}^n$, by applying the minimum distance criterion, from (3) we can define the partition $\prod = \{\pi_1, \ldots, \pi_k\}$ of the set Q as follows:

$$\pi_j = \{a \in Q \mid d(c_j, a) \leq d(c_s, a), \forall s = 1, \ldots, k\}, \tag{4}$$

where $j = 1, \ldots, k$. To guarantee that each member of set Q is included in only one cluster, we can thus transform the challenge of determining the optimal partition of set Q into the subsequent optimization problem.

$$\min_{c_1, \ldots, c_k \in \mathbb{R}^n} G(c_1, \ldots, c_k), \qquad G(c_1, \ldots, c_k) = \sum_{i=1}^{m} \min_{j=1,\ldots,k} d(c_j, a_i), \qquad G : \mathbb{R}^{kn} \to \mathbb{R}_+, \tag{5}$$

and $\mathbb{R}_+$ represents the set of all vectors in $\mathbb{R}^n$ that have nonnegative components. Generally, this functional is not differentiable and can possess multiple local minima. Accordingly, the problem stated in (5) necessitates optimization, and in this research, least squares will be employed to perform the optimization for the problem in (5). Equation (5) has a functional problem that is not differentiable and may have some local minimum problems. To solve these optimization problems, Kogan [15] and Teboulle [16] used the Least Squares sense for the distance-like function as follows:

$$d(x, y) = \|x - y\|^2, \quad x, y \in \mathbb{R}^n. \tag{6}$$

Accordingly, this research employs (6) as an objective function to make Differential Evolution suitable for solving clustering problem.

3 The implementation of DE to the clustering problem

Differential Evolution (DE) algorithm was introduced by Storn and Price in 1995 as a meta-heuristic approach to solving continuous optimization problems. While it borrows several concepts from the classical genetic algorithm (GA), DE is more straightforward to implement [17]. The first formal publication on DE appeared as a technical report in 1995. Since then, DE has shown its effectiveness in multiple contests, such as the IEEE International Contest on Evolutionary Optimization (ICEO) in 1996 and 1997. The fundamental idea of DE is to use variations among individuals within the population to find solutions. Though DE utilizes mutation and crossover operations, it focuses on geometric arguments during its search. DE follows the same computational steps as the Evolutionary Algorithm (EA) but distinguishes itself by employing unique parameter vectors to explore the objective function space. Like other population-based algorithms, DE generates new candidate solutions by perturbing existing ones. It modifies the current generation vector using the difference calculated between two randomly selected population vectors. To form a trial vector, DE, in its simplest form, adds a scaled difference of one random vector to another randomly selected population vector [17]. In the selection phase, trial vectors are pitted against population vectors with the same index. After evaluating all trial vectors, the winning vector from these crossover competitions is preserved for the next generation in the evolutionary cycle. The steps of the Differential Evolution algorithm are:

1. Vector Initialization

   The initialization process begins with a randomly generated population of real-valued dimension parameter vectors. Each vector in this population, termed a genome or chromosome, represents a potential solution to the multi-dimensional optimization problem. Differential Evolution (DE) aims to locate the global optimum in a D-dimensional continuous hyperspace. To determine the initial members of the population vector, the following equation (7) is applied:

   $$x_{i,j}(0) = x_{\min,j} + rand_{i,j}(0,1).(x_{\max,j} - x_{\min,j}), \tag{7}$$

   where $x_{\min}$ and $x_{\max}$ denote the lower and upper bounds of the data or problem, respectively. Once all components of the target vector are obtained, each vector is represented as:

   $$\vec{X}_i(t) = [x_{1,1}(t), x_{1,2}(t), ..., x_{i,D}(t)]^T, \qquad i = 1, 2, \ldots, NP.$$

2. Mutation

   In biological terms, mutation is a change in a chromosome's gene characteristics. In Differential Evolution, mutation creates a donor vector to modify each population member at every generation. To generate a donor vector for each target vector (population member), three different parameters, randomly selected from the current population, are used. These are mutually exclusive integer indices drawn randomly from the range $[1, NP]$ and distinct from the base vector index i. The difference between these indices is scaled by F and added to X_{r_1}. The result of this process is the donor vector. The mutation process is expressed by:

   $$\vec{V}_i(t) = \vec{X}_{r_1^i}(t) + F.(\vec{X}_{r_2^i}(t) - \vec{X}_{r_3^i}(t)). \tag{8}$$

3. Crossover

   To enhance the potential diversity of the population, crossover operations are implemented following the generation of donor vectors via mutation. Differential Evolution (DE) algorithms can adopt two crossover types: exponential and binomial. During this

process, the donor vector swaps its components with the target vector to create the trial vector. In exponential crossover, an integer n is initially chosen at random from the set $[0, D-1]$. This integer serves as the starting point on the target vector for the crossover, marking where the components' exchange with the donor vector begins. Additionally, another integer L is selected from within the interval $[1, D]$, indicating the number of components the donor vector contributes to the target. Following the selection of n and L, the crossover begins

$$u_{i,j}(t) = \begin{cases} v_{i,j}(t), j = \langle n \rangle_D, ..., \langle n + L - 1 \rangle_D \\ x_{i,j}(t), j \in [0, D-1] \end{cases}, \tag{9}$$

where the square brackets D denote a modulo function with modulus D. The integer L is taken from $[1, 2, \ldots, D]$. Therefore, the probability , is called the crossover rate and appears as a control parameter of DE just like F. For each donor vector, a new set of n and L must be randomly chosen.

4. Selection

The final phase of a DE iteration is selection, which determines whether the target vector $X(t)_i$ or the trial vector $U(t)_i$ will persist to the next generation. The choice of retaining the original $X(t)_i$ in the population or substituting it with $U(t)_i$ at the following time step $t + 1$ relies completely on the principle of survival of the fittest. If the trial vector produces a better fitness value, it will replace the target vector in the next time step.

In this context, a better fitness value refers to a lower objective function value for minimization problems, and a higher objective function value for maximization problems. The selection process can be expressed as

$$\vec{X}_i(t+1) = \begin{cases} \vec{U}_i(t) \text{if} f(\vec{U}_i(t)) \leq f(\vec{X}_i(t)) \\ \vec{X}_i(t) \text{if} f(\vec{U}_i(t)) > f(\vec{X}_i(t)) \end{cases}, \tag{10}$$

where $f(X)$ is the objective function aimed at minimization. Due to the binary nature of the selection process, meaning either the target vector or its offspring survives, the population size stays constant across many generations. Consequently, the fitness of the population members either increases over successive generations or stays the same, but it does not decline.

Subsequently, the DE algorithm will be incorporated with the objective function (6) using the pseudo code outlined below:

Least Square-Based Differential Evolution Algorithm

Set Initial Parameters:
Specify `NP` (Size of Population)
Specify `MaxIter` (Max Number of Iterations)
Specify `F` (Scaling Factor, with a range of [0, 2])
Specify `CR` (Crossover Rate, within the range [0, 1])
Identify `k` (Count of Clusters)
Define `m` (Total Data Points) and `n` (Number of Dimensions)
Formulate Objective Function G:

$$G(c_1, \ldots, c_k) = \sum_{i=1}^{m} \min_j \|c_j - a_i\|^2,$$

where a_i represents a data point and c_j stands for a centroid vector in an n-dimensional space.

Create Initial Population:
for each member p of the population (for $p = 1, \ldots, \texttt{NP}$) **do**
Randomly generate k centroids $c_1, \ldots, c_k$ as vectors in n-dimensional space.
Ensure all elements of the centroid vectors are nonnegative.
end for
Evaluate the Initial Population:
for each entity p within the population **do**
Compute $G(c_1, \ldots, c_k)$ for entity p.
Assign the fitness score of entity p as the computed G value.
end for
Evolutionary Process:
for iteration = 1 to `MaxIter` **do**
for every individual i in the population **do**
Mutation:
Randomly choose three distinct individuals r_1, r_2, and r_3 from the population.
Create mutant vector V where $V = r_1 + F \cdot (r_2 - r_3)$.
Ensure each component of V is nonnegative (enforce limits if required).
Crossover:
Create a trial vector U.
for each dimension j within the vector **do**
if rand$(0, 1) <$ `CR` **then**
Assign $V[j]$ to $U[j]$.
else
Assign individual$[i][j]$ to $U[j]$.
end if
end for
Selection:
Compute $G(U)$ for the trial vector U.
if $G(U)$ is lower than the fitness of individual$[i]$ **then**
Substitute individual$[i]$ with U within the population.
end if
end for
end for
Termination:
if `MaxIter` is attained **then**
Terminate the process.
end if
Output:
Identify and return the individual in the population with the smallest objective function value as the optimal solution.
This solution denotes the best-fitting cluster centroids.

4 Computational experiments

In this research, numerous experiments were executed to evaluate the effectiveness of the proposed method. Therefore, this section will elaborate on the experimental procedures and evaluation to derive conclusions. This experiment was conducted using a Jupyter notebook on a Windows 10 system with an Intel(R) Core i7 processor at 2.6 GHz and 8 GB of RAM.

The first experiment aims to test the computation of the proposed method, the results of

which can be seen in Tab. 1. Tab. 1 uses the following notations:

1. α is the number of row
2. β is the number of dimension
3. λ is the number of clusters
4. sb is the Silhoeuette Score of DE+KMeans
5. tb is the running time of DE+KMeans

Computational experiments were performed on datasets containing 1,000, 5,000, and 10,000 samples. These datasets originate from numerical experiments produced by the data randomization process with values ranging between 0 and 100. Each dataset is available in 2 and 3 dimensions respectively, as highlighted in Tab. 1. The computational trials involved a predetermined set of clusters: 2, 3, and 5. This study reveals that simply increasing the number of clusters does not necessarily enhance the silhouette score for any specific data size. It suggests that merely adding clusters does not invariably improve cluster separation quality. In general, the best silhouette scores are achieved with fewer clusters (either 2 or 3), particularly in 3D datasets. For instance, in a dataset of 1000 with 3 dimensions, 2 clusters yield a relatively high score of 0.355713785. Conversely, when the number of clusters increases to 5, the silhouette score typically decreases, indicating reduced separation quality between clusters.

The execution time is also affected, rising with larger datasets and more clusters. For example, processing 10,000 data points with 3 dimensions and 5 clusters takes 0.56808567 seconds. As expected, both larger datasets and more clusters lead to higher execution times due to greater computational demands. Overall, DE+KMeans is more effective on low-dimensional (2D) data with fewer clusters, especially with smaller datasets. While the silhouette score slightly diminishes as the number of clusters grows, the execution time increases notably with both data size and cluster count.

Table 1: General Computational Results

α	β	λ	sb	tb
1000	2	2	0.351712	0.006982
		3	0.388482	0.008193
		5	0.378432	0.005983
	3	2	0.252825	0.005984
		3	0.25381	0.007968
		5	0.272624	0.008977
5000	2	2	0.355714	0.011971
		3	0.378233	0.103799
		5	0.378041	0.010355
	3	2	0.249015	0.005971
		3	0.245607	0.006982
		5	0.269963	0.009524
10000	2	2	0.35602	0.008976
		3	0.382222	0.052598
		5	0.382611	0.054601
	3	2	0.248241	0.319514
		3	0.242721	0.088114
		5	0.265742	0.568086

5 Comparison

During the comparison testing phase, two tests will be conducted: one on n-dimensional data and another on one-dimensional data. The n-dimensional test aims to assess the reliability of the DE+KMeans algorithm in data clustering. This test utilizes the same scenario and dataset as in the computation experiment phase, but in this comparison, the DE+KMeans algorithm is evaluated against two other algorithms: DE and KMeans. The one-dimensional data test seeks to assess the stability of the DE+KMeans algorithm by applying it to 100 one-dimensional datasets and comparing the results with those obtained from DE and KMeans. As in the computational test, this test also looks at how much silhouette score is obtained and how much time it takes to produce a cluster. Comparison result can be seen in the Tabs. 2 and 3. Tabs. 2 and 3 uses the following spesific notations:

1. α is the number of row
2. N is the i-th trial
3. β is the number of dimension
4. λ is the number of clusters
5. s is the Silhoeuette Score
6. t is the running time
7. a, b, c are Differential Evolution, DE+KMeans, and KMeans

According to the Silhouette Score evaluation across 18 tests in the Tab. 2, the DE + KMeans algorithm outperformed the KMeans algorithm 13 times, underperformed 3 times, and matched scores twice. Conversely, the DE algorithm alone did not outperform the DE+KMeans combination and KMeans but had a score that was comparatively close. In addition to evaluating the Silhouette Score, the study also examines the run time of the methods throughout the clustering process. Consequently, the hybrid algorithm combining DE and KMeans outperformed in almost all tests performed. Based on the computational testing outcomes, it is evident that the DE+KMeans combination is more effective for clustering than using KMeans or DE alone.

Across 100 trials comparing the two algorithms, it was observed that DE+KMeans outperformed both KMeans without optimization and DE. The silhouette score and runtime comparison results as shown in Tab. 3. The tables illustrate that out of 100 trials, DE+KMeans performs superiorly in terms of silhouette score and runtime. Additionally, we also graphically represent the distribution scores and runtime to compare DE+KMeans with KMeans. Figs. 1-2 illustrate that the distribution of DE+KMeans scores is generally superior. Additionally, DE+KMeans outperforms in computational time, as depicted in Fig. 2. Consequently, at this point of comparison, DE+KMeans effectively identifies the initial centroid better than the individual DE and KMeans algorithms.

6 Compactness and Homogenity Cluster Evaluation

This section presents a comparative analysis of the K-Means algorithm and the DE-KMeans algorithm to evaluate the degree of cluster compactness achieved by each technique. The experimental data differs from the previous test, yet it still utilizes random data for compactness test. The experimental data differs from the previous test, yet it still utilizes random data. The variation lies in how the data is generated, specifically by ensuring the formation of distinct groupings. The test involved crafting four distinct scenarios based on varying standard deviation values, which influence the spacing between clusters. The standard deviation values

Table 2: Comparison Results

α	β	λ	s			t		
			a	b	c	a	b	c
1000	2	2	0.334115	0.351712	0.351712	34.73574	0.006982	0.061423
		3	0.28449	0.388482	0.38845	57.87727	0.008193	0.103635
		5	0.274968	0.378432	0.378611	16.82297	0.005983	0.094745
	3	2	0.1632	0.252825	0.252972	38.93418	0.005984	0.089445
		3	0.136284	0.25381	0.253699	54.58894	0.007968	0.087197
		5	0.190135	0.272624	0.271217	82.2534	0.008977	0.089081
5000	2	2	0.269084	0.355714	0.355714	181.0725	0.011971	0.056023
		3	0.363331	0.378233	0.378232	265.3607	0.103799	0.053845
		5	0.215638	0.378041	0.375089	411.6037	0.010355	0.091681
	3	2	0.197189	0.249015	0.249006	182.6966	0.005971	0.057052
		3	0.221292	0.245607	0.245574	273.0665	0.006982	0.095544
		5	0.179124	0.269963	0.268296	421.3279	0.009524	0.118408
10000	2	2	0.332426	0.35602	0.356004	359.6633	0.008976	0.062538
		3	0.310461	0.382222	0.382211	485.6237	0.052598	0.086382
		5	0.291638	0.382611	0.379986	833.4744	0.054601	0.207403
	3	2	0.225036	0.248241	0.248226	420.1548	0.319514	0.086279
		3	0.229446	0.242721	0.24266	569.6182	0.088114	0.212933
		5	0.225157	0.265742	0.269555	889.8035	0.568086	0.187169

applied were 1.5, 2.0, 2.5, and 3.0. Throughout this experiment, two clusters, two features, and 1000 samples were maintained. Data distribution is illustrated in Fig. 3.

Following the clustering execution using both techniques, the results evaluated through the Silhouette Score and Within-Cluster Sum of Squares (WCSS) are presented in Tab. 4, where σ is the standard deviation, s is the Silhouette Score, w is the WCSS, b is DE+KMeans, and c is KMeans.

The findings illustrate that for data with standard deviations of 1.5 and 2.0, K-Means and DE+KMeans show no noticeable distinctions regarding the Silhouette Score and WCSS. This implies that both algorithms can generate similarly cohesive and distinct clusters when the natural separation is ample. Nonetheless, for data with a standard deviation of 2.5, DE+KMeans outperforms K-Means. The Silhouette Score for DE+KMeans reaches 0.580089926977142, compared to 0.580094211588442 for K-Means. Regarding WCSS, DE+KMeans achieves a lower value of 11755.2945687871, surpassing K-Means' result of

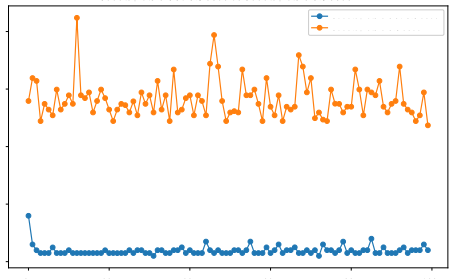

Figure 1: Distribution Score of DE+KMeans and KMeans

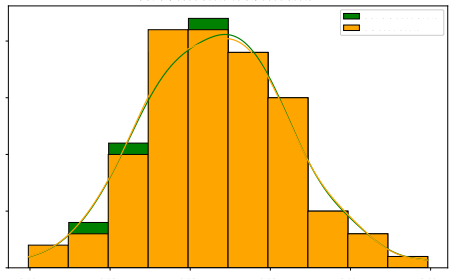

Figure 2: The Running Time of DE+KMeans and KMeans

Table 3: Comparison Results on 1-D Data

N	s			t		
	a	b	c	a	b	c
1	0.562647535	0.591013315	0.590744209	50.67484832	0.015947104	0.055851221
2	0.591528681	0.59509394	0.59497149	44.94694924	0.005983829	0.063828707
3	0.576266952	0.58982947	0.590287935	45.63507724	0.003963947	0.062832355
4	0.516336154	0.598402206	0.598594775	44.62163568	0.002994299	0.048870087
5	0.55093843	0.580999322	0.580999322	44.6436379	0.00299263	0.054853678
6	0.245503964	0.581162052	0.581025998	45.31455898	0.002990007	0.052857161
7	0.477903175	0.601897283	0.601897283	45.14308381	0.004948139	0.050863743
8	0.545836431	0.597939659	0.598192647	45.594769	0.002995968	0.059837103
9	0.360803454	0.597443895	0.597358166	44.89892912	0.002990723	0.052858114
10	0.362430404	0.589149295	0.589038765	44.62690616	0.00299263	0.054850101
11	0.542172516	0.585079535	0.585193538	44.66195011	0.003988266	0.05783391
12	0.467428361	0.58993992	0.5851764	45.27667046	0.002991915	0.054862022
13	0.452009518	0.601212079	0.601343148	44.65931439	0.002993584	0.084773064
14	0.557765548	0.610782663	0.610782663	44.79385519	0.002993345	0.057845354
15	0.587372983	0.587234848	0.587195208	44.91555309	0.002992153	0.056848526
16	0.466270768	0.584366205	0.58776882	44.62131548	0.002991676	0.058843374
17	0.583397218	0.587377619	0.587242246	46.39012909	0.002991676	0.051864147
18	0.576561991	0.593730477	0.593624592	45.01229858	0.002994776	0.055851221
19	0.389643248	0.593583632	0.593583632	44.95328712	0.002994061	0.059839249
20	0.361420221	0.590930595	0.591155689	45.25005817	0.003963709	0.056847811
21	0.598372823	0.602329413	0.602351796	45.13917112	0.002965689	0.052859306
22	0.47574888	0.619643898	0.619643898	45.10650849	0.002990246	0.048870325
23	0.571497947	0.590343256	0.590195573	45.19552851	0.002960443	0.052859068
24	0.463108942	0.597731301	0.597731301	46.38924837	0.002990007	0.054846048
25	0.564911061	0.595304426	0.594669747	46.49196863	0.00299263	0.054405689
26	0.566206606	0.574920507	0.574552543	44.98650217	0.003988504	0.051860332
27	0.542910938	0.596524534	0.596524534	45.20502782	0.002989769	0.055738688
28	0.551501743	0.581130709	0.579919917	45.06911278	0.003989697	0.050864935
29	0.573741367	0.591556946	0.591091464	44.49493027	0.003988028	0.058844566
30	0.550875294	0.607609919	0.607609919	44.76673889	0.002990007	0.054853201
31	0.227133838	0.599888453	0.599888453	44.77537417	0.00299263	0.057846069
32	0.575963683	0.594284532	0.594284532	44.9046638	0.001998663	0.051861763
33	0.376037384	0.569862661	0.569796137	44.8649683	0.003987789	0.062831879
34	0.576766132	0.603109293	0.603394226	44.96564746	0.003988743	0.052855492
35	0.501052612	0.586659707	0.58669072	45.09357333	0.00299263	0.057846546
36	0.379425606	0.585272658	0.585682074	44.90278339	0.002992153	0.048868656
37	0.558144805	0.591877465	0.590812984	44.87715316	0.003988743	0.066807032
38	0.486017167	0.577365515	0.577708682	45.44776201	0.00398922	0.05185318
39	0.350493722	0.594240698	0.594240698	44.66088033	0.00498724	0.0528512
40	0.479880983	0.597571134	0.597571134	44.7402308	0.002991676	0.056847334
41	0.577661915	0.60210753	0.602107777	44.9872849	0.003989935	0.057845354
42	0.44168964	0.593622055	0.593757617	45.76324034	0.002991438	0.050864935
43	0.426231418	0.581111839	0.58131986	45.54889703	0.002994299	0.057844639
44	0.46043052	0.59287171	0.59287171	45.83807135	0.002991915	0.055852175
45	0.471037273	0.577499311	0.577615009	45.29319501	0.006980658	0.050862789
46	0.538536299	0.596187612	0.596145131	45.92455196	0.003986359	0.068814754
47	0.443742905	0.603515858	0.603515858	45.49062872	0.002992392	0.078789711
48	0.583597958	0.583813789	0.584106536	45.88088465	0.003989697	0.067819357
49	0.358412047	0.600787546	0.600787546	45.21264648	0.002996683	0.055850506
50	0.405358012	0.592369596	0.591837386	45.89582253	0.002991915	0.048869133

51	0.440976884	0.595292188	0.595292188	44.7676065	0.002991676	0.051860571
52	0.444608642	0.589050877	0.588221993	45.55388713	0.003990412	0.052322626
53	0.380479036	0.612065813	0.612106237	45.63908124	0.003994942	0.051865578
54	0.52504356	0.586231289	0.586040243	46.65978456	0.002991438	0.06682086
55	0.346505959	0.584476239	0.584476239	45.088902	0.003988266	0.0578444
56	0.580383921	0.589285905	0.589285905	45.34135342	0.006980896	0.057845831
57	0.498595475	0.58332306	0.582995696	44.8041048	0.002991915	0.059840202
58	0.42354484	0.594073552	0.594073552	44.96531701	0.002990723	0.054843426
59	0.411003938	0.607939198	0.607970154	45.25614023	0.002994299	0.048871279
60	0.529314726	0.603432703	0.603432703	45.07545829	0.004968643	0.063832283
61	0.433525384	0.60318488	0.60318488	45.38379765	0.002991676	0.053856373
62	0.420587812	0.594430388	0.594430388	45.20732188	0.003988981	0.050864697
63	0.570835629	0.587096756	0.587096756	44.97121239	0.005985737	0.057840824
64	0.400412741	0.600538944	0.600392634	44.89221859	0.00299263	0.048870564
65	0.59447097	0.596209814	0.596209814	45.32764578	0.003989697	0.053855181
66	0.457910067	0.594884629	0.594952989	45.05395436	0.003997564	0.052860498
67	0.559458686	0.593664675	0.59376192	45.8599298	0.004987717	0.053853989
68	0.582687235	0.587557856	0.587655837	45.34940267	0.002993584	0.071808815
69	0.286075542	0.596426521	0.596426521	44.99223542	0.002992868	0.067817688
70	0.568317647	0.58321179	0.582669411	46.12144852	0.003988504	0.058842897
71	0.572450108	0.602011383	0.602011383	45.36452413	0.002993345	0.063827515
72	0.448469458	0.587480454	0.587480454	44.73765969	0.003990412	0.049868345
73	0.444291661	0.590659527	0.590579558	45.09645247	0.001993895	0.051859856
74	0.469754831	0.59932601	0.599368883	44.36743379	0.005990982	0.049376011
75	0.471712553	0.587173232	0.586896836	45.69979525	0.003990173	0.048869371
76	0.288449513	0.597947005	0.597947005	44.4369328	0.003989935	0.05984354
77	0.529497761	0.59623602	0.59623602	45.31856871	0.002992392	0.054903507
78	0.427719862	0.600118686	0.600118686	45.27994823	0.003990412	0.054853916
79	0.549172241	0.594743419	0.594826985	45.94388771	0.006969213	0.05185914
80	0.561253965	0.584980565	0.584918191	45.32071781	0.002990961	0.053855896
81	0.492775611	0.586987964	0.587044194	45.21375847	0.003988028	0.053857565
82	0.571164131	0.585127793	0.585127793	45.78133059	0.002989769	0.066822052
83	0.263681648	0.589206984	0.588926977	45.12504625	0.002991915	0.059841394
84	0.405528982	0.585896037	0.585896037	45.74722314	0.003989935	0.050864935
85	0.419570575	0.59863157	0.598544731	45.17078328	0.003989458	0.059850454
86	0.399294899	0.606352404	0.606946152	45.26563311	0.007979155	0.058842182
87	0.569981316	0.577444712	0.577359869	45.81158018	0.002992392	0.057845592
88	0.527771528	0.593933543	0.593933543	47.38597918	0.00299263	0.062832594
89	0.398598003	0.599911628	0.599922238	60.74008417	0.00498724	0.05385828
90	0.560041951	0.597563376	0.597068771	57.76946115	0.002991676	0.051861525
91	0.270633347	0.611196046	0.610571521	54.19370484	0.00299263	0.05485487
92	0.558178365	0.60076038	0.600594863	54.69181418	0.00299263	0.055851698
93	0.545294761	0.580740296	0.580931122	54.84835124	0.003989458	0.067823172
94	0.579181764	0.582272358	0.582272358	54.49602509	0.004966497	0.054854393
95	0.456406483	0.58930752	0.589285421	55.34188485	0.002994061	0.052857876
96	0.465952704	0.590431781	0.590431781	52.76015997	0.003989697	0.05186224
97	0.522684964	0.590662324	0.590537309	48.89275622	0.003988028	0.048860312
98	0.582525384	0.607964417	0.607964417	48.52631283	0.003981352	0.050863504
99	0.506174904	0.598889681	0.598867952	49.28045106	0.00598526	0.058842659
100	0.506094496	0.606834916	0.6068935	53.48102164	0.00398922	0.047385216

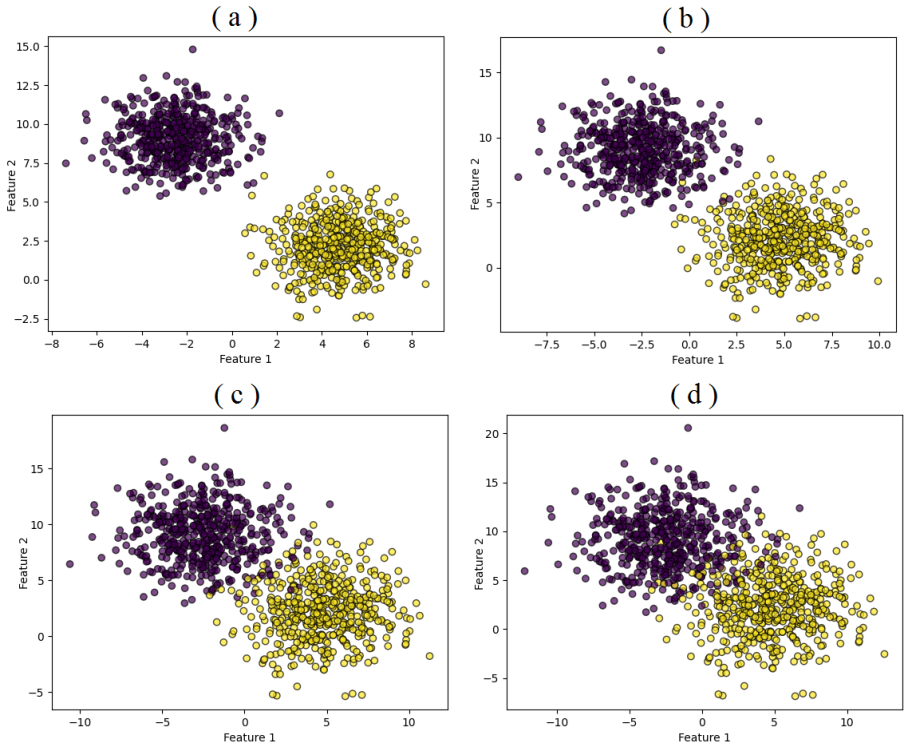

Figure 3: Data distribution: a) 1000 Rows with 2 Clusters and 1.5 Standard Deviation Configuration, b) 1000 Rows with 2 Clusters and 2.0 Standard Deviation Configuration, c) 1000 Rows with 2 Clusters and 2.5 Standard Deviation Configuration, d) 1000 Rows with 2 Clusters and 3.0 Standard Deviation Configuration.

11755.3715909048. A lower WCSS value suggests that DE+KMeans excels at clustering the data, forming clusters that are more compact compared to those of K-Means.

In data with a standard deviation of 3.0, a significant difference is observed in the WCSS metric. DE+KMeans obtained a value of 16294.2890125106, while K-Means had a slightly higher value, namely 16294.3952598849. This shows that in more dispersed data conditions, where the distance between clusters is getting smaller, DE+KMeans is still able to produce more compact clusters than K-Means. From the results of this experiment it can be concluded that under conditions of more naturally separated data, both algorithms give almost identical results. However, when the data become more difficult to cluster, such as at standard deviations of 2.5 and 3.0, DE+KMeans shows an advantage in maintaining cluster compactness, as indicated by the lower WCSS value. This advantage indicates that the DE+KMeans method is able to optimize the centroid position better than conventional K-Means, especially in more complex data conditions. The homogeneity test was carried out using the Breast Cancer Wisconsin (Diagnostic) dataset sourced from the UCI dataset repository. This dataset comprises two classes, Malignant and Benign, with 569 instances across 30 features. The purpose of the homogeneity test is to assess how well the clustering outcomes align with the original class labels. A Homogeneity Score of 1 means each cluster contains samples from only one class, whereas a score nearing 0 implies that classes are intermixed within clusters. The results indicate that the Homogeneity Score for both K-Means and K-Means optimized via Differential Evolution (DE + KMeans) remains the same at 0.61147 on the Breast Cancer Wisconsin (Diagnostic) dataset. This score suggests that the clustering results account for approximately 61.1% of the original label information. The similar Homogeneity Scores for K-Means and

DE+KMeans imply that Differential Evolution optimization does not enhance class separation quality. This might be due to factors like a naturally well-separated data distribution, making DE-based centroid initialization redundant. Additionally, the parameters F (0.7) and CR (0.9) may not be optimal, leading to ineffective exploration of the solution space. Furthermore, the dataset's class imbalance may hinder K-Means from achieving improved separation, thus maintaining the same Homogeneity Score. An alternative might involve adjusting the DE algorithm parameters.

Following the clustering execution using both techniques, the results evaluated through the Silhouette Score and Within-Cluster Sum of Squares (WCSS) are presented in Tab. 3. Where σ is the standard deviation, s is the Silhouette Score, w is the WCSS, b is DE+KMeans, and c is KMeans.

Table 4: Result of Compactness Cluster Evaluation

σ	s		w	
	b	c	b	c
1.5	0.7379930396	0.7379930396	4376.0944133074	4376.0944133074
2	0.6547868066	0.6547868066	7683.3590752235	7683.3590752235
2.5	0.5800899270	0.5800942116	11755.2945687871	11755.3715909048
3	0.5241596100	0.5241596100	16294.2890125106	16294.3952598849

7 Conclusion

Optimization algorithms can address various issues, including algorithmic challenges. As in this study, metaheuristic optimization algorithms are employed to solve the global solution issue in the KMeans algorithm, which arises due to random initial centroid searches. These random searches often result in suboptimal centroids, leading to less effective clustering outcomes. The researchers utilized the Differential Evolution (DE) algorithm with a least squares objective function to optimize KMeans for determining the optimal initial centroids.

The results of the comparison between the DE, DE + KMeans, and KMeans algorithms demonstrate that the KMeans algorithm optimized with DE exhibits superior performance. Specifically, in the compactness test using Silhouette Score and Within-Cluster Sum of Squares (WCSS), DE+KMeans showed better clustering quality than KMeans, particularly when the data distribution became more complex (higher standard deviation). While both algorithms performed similarly when clusters were naturally well-separated (standard deviation 1.5 and 2.0), DE+KMeans achieved lower WCSS and comparable or better Silhouette Scores at standard deviations of 2.5 and 3.0. This indicates that DE+KMeans can produce more compact clusters in challenging clustering scenarios.

In addition to compactness, the Homogeneity Score was also analyzed to assess how well clusters contain only data points from a single class. The results indicate that, in some cases, the homogeneity of DE+KMeans and KMeans was identical, suggesting that the optimization process did not significantly improve the class purity within clusters. This could be due to the inherent structure of the dataset or the influence of the chosen optimization parameters. However, DE+KMeans still showed advantages in cluster compactness and overall clustering quality, making it a valuable optimization approach.

Thus, it is concluded that the DE algorithm effectively optimizes KMeans for finding optimal initial centroids and improving cluster compactness. However, while DE+KMeans enhances clustering in terms of WCSS and Silhouette Score, its impact on homogeneity varies depending on the dataset characteristics. This research is confined to numerical data and does

not encompass data with categorical characteristics or small data dimensions. Future research should explore the same method on categorical data and large data dimensions (big data) to further validate its effectiveness and investigate parameter tuning techniques to enhance both compactness and homogeneity in clustering results.

References

[1] Adeen N., Abdulazeez M., Zeebaree D., *Systematic review of unsupervised genomic clustering algorithms techniques for high dimensional datasets*, Technology Reports of Kansai University, 62 (2020), 355-374.

[2] Ur Rehman A., Brahim Belhaouari S., *Divide well to merge better: A novel clustering algorithm*, Pattern Recognition, 122 (2022), 108305.

[3] Xu X., Ding S., Wang Y., Wang L., Jia W., *A fast density peaks clustering algorithm with sparse search*, Information Sciences, 554 (2021), 61-83.

[4] Zhao J., Ding Y., Zhai Y., Jiang Y., Zhai Y., Hu M., *Explore unlabeled big data learning to online failure prediction in safety-aware cloud environment*, Journal of Parallel and Distributed Computing, vol. 153 (2021), pp. 53–63.

[5] Aljibawi M., Nazri M. Z. A., Sani N. S., *An enhanced mudi-stream algorithm for clustering data stream*, Journal of Theoretical and Applied Information Technology, 100 (2022), 3012-3021.

[6] Bataineh B., *Fast component density clustering in spatial databases: A novel algorithm*, Information, 13 (2022), 1-18.

[7] Melnykov V., Michael S., *Clustering large datasets by merging k-means solutions*, Journal of Classification, 37 (2020), 97-123.

[8] Pandiya R., Ahdika A., Khomsah S., Ramadhani R. D., *A new integral function algorithm for global optimization and its application to the data clustering problem*, MENDEL, 29 (2023), 162-168.

[9] Paul S., De S., Dey S., *A novel approach of data clustering using an improved particle swarm optimization based kB–means clustering algorithm*, IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), IEEE, 2020, 1-6.

[10] Ahmed M., Seraj R., Islam S. M. S., *The k-means algorithm: A comprehensive survey and performance evaluation*, Electronics, 9 (2020), 1295.

[11] Harifi S., Khalilian M., Mohammadzadeh J., Ebrahimnejad S., *Using metaheuristic algorithms to improve k-means clustering: A comparative study*, Revue d'Intelligence Artificielle, 34 (2020), 297-305.

[12] Pan Z., Fang S., Wang H., *Lightgbm technique and differential evolution algorithm based multi-objective optimization design of ds-apmm*, IEEE Transactions on Energy Conversion, 36 (2020), 441-455.

[13] Bilal, Pant M., Vig G., *Clustering based adaptive differential evolution for numerical optimization*, IEEE Congress on evolutionary computation (CEC), IEEE, 2020, 1-8.

[14] Yue L., Hu P., Chu S., Pan J., *English speech emotion classification based on multi-objective differential evolution*, Applied Sciences, 13 (2023), 12262.

[15] Kogan J., *Introduction to clustering large and high-dimensional data*, Cambridge University Press, 2007.

[16] Teboulle M., *A unified continuous optimization framework for center-based clustering methods*, Journal of Machine Learning Research, 8 (2007), 65-102.

[17] Das S., Abraham A., Konar A., *Metaheuristic clustering*, Springer Science & Business Media, 2009.

Muhammad Abdul Latief,
Department of Data Science, Telkom University,
Jl. DI. Panjaitan 128, Purwokerto, Indonesia,
Email: abdullatief@student.telkomuniversity.ac.id

Aina Latifa Riyana Putri,
Department of Data Science, Telkom University,
Jl. DI. Panjaitan 128, Purwokerto, Indonesia,
Email: ainaqp@telkomuniversity.ac.id

Ridwan Pandiya,
Department of Informatics, Telkom University,
Jl. DI. Panjaitan 128, Purwokerto, Indonesia,
Email: ridwanp@telkomuniversity.ac.id

Received 19.06.2024, Accepted 01.10.2024, Available online 30.09.2025.