Iranian Journal of Numerical Analysis and Optimization
Vol. 15, No. 2, 2025, pp 475–507
https://doi.org/10.22067/ijnao.2025.89754.1508
https://ijnao.um.ac.ir/

Research Article

Unite and conquer approach for data clustering based on particle swarm optimization and moth flame optimization

E. Mosavi, S.A. Shahzadeh Fazeli*, , E. Abbasi and F. Kaveh-Yazdy

Abstract

Data clustering is a widely used technique in various domains to group data objects according to their similarity. Clustering molecules is a useful

*Corresponding author

Received 10 September 2024; revised 28 December 2024; accepted 15 January 2025

Esra Mosavi
Department of Computer Science, Yazd University, Yazd, Iran.
Parallel Processing Laboratory, Yazd University, Yazd, Iran. e-mail: esramosavi@stu.yazd.ac.ir

Seyed Abolfazl Shahzadeh Fazeli
Department of Computer Science, Yazd University, Yazd, Iran.
Parallel Processing Laboratory, Yazd University, Yazd, Iran.e-mail: fazeli@yazd.ac.ir

Elham Abbasi
Department of Computer Science, Yazd University, Yazd, Iran. e-mail: e.abbasi@yazd.ac.ir

Fatemeh Kaveh-Yazdy
Researcher at Oncober, Basel, Switzerland. e-mail: f.kaveh@oncober.ch

How to cite this article
Mosavi, E., Shahzadeh Fazeli, S.A., Abbasi, E. and Kaveh-Yazdy, F., Unite and conquer approach for data clustering based on particle swarm optimization and moth flame optimization. *Iran. J. Numer. Anal. Optim.*, 2025; 15(2): 475-507. https://doi.org/10.22067/ijnao.2025.89754.1508

process where you can easily subdivide and manipulate and large datasets to group compounds into smaller clusters with similar properties. To discover new molecules with optimal properties and desired biological activity, can be used by comparing molecules and their similarities. A prominent clustering technique is the k-means algorithm, which assigns data objects to the nearest cluster center. However, this algorithm relies on the initial selection of the cluster centers, which can affect its convergence and quality.

To address this issue, metaheuristic algorithms have been proposed as a type of approximate optimization algorithm capable of identifying almost optimal solutions. In this paper, a new meta-heuristic approach is proposed by combining two algorithms of particle swarm optimization (PSO) and moth flame optimization (MFO), following that, it is used to improve data clustering. The efficiency of the proposed approach is evaluated utilizing benchmark functions F1-F23. Its efficiency is evaluated with PSO and MFO algorithms on different datasets. Our experiential results show that the suggested approach exceeds the PSO and MFO algorithms with respect to speed of convergence and clustering quality.

AMS subject classifications (2020): 68T10; 62H30.

Keywords: Data clustering; k-means clustering; Metaheuristic optimization; Particle swarm optimization; Moth flame optimization.

1 Introduction

Clustering methods have become increasingly popular in various fields of science and engineering, making it an important field for data mining study. Using this method, objects are grouped into clusters where they have the least amount of distance between each object and the cluster's center. Ideally, objects within a cluster should be as similar as possible and distinct from those in other clusters. As a powerful tool in chemistry, clustering helps to analyze complex data and make decisions in various processes. Some of these applications such as analysis of chemical compounds, medicinal chemistry, analysis of spectra, and design of new materials can be mentioned. Clustering can be used to group chemical compounds with similar properties, to identify chemical compounds with similar biological properties, or to analyze spectral data by identifying compounds. It can also design new materials with desirable properties through the grouping of compounds [44]. Several algorithms exist for clustering. One of the most popular clustering techniques with a wide range of applications is the k-means algorithm [20, 27]. Extensive research has been conducted on k-means clustering, leading to various extensions, and it has been applied in various fundamental domains [4, 26, 29, 47]. However, this algorithm's performance depends heavily on the selection of the centers, making it susceptible to getting stuck in a local optimum. One way to overcome this problem is using optimization algorithms for clustering. Optimization involves finding the best possible solutions to a specific problem. Mathematical optimization techniques are deterministic but unable to avoid local optima. However, random optimization algorithms (metaheuristics) are much more efficient at avoiding local optimization by randomly exploring the search domain. While these algorithms can capture a certain pattern, their aim is to find an approximate solution that is close to the exact solution. Several approximate approaches are known as global search approaches referred to as metaheuristics. These methods have been designed to overcome the limitations of traditional techniques. Metaheuristic algorithms utilize the intelligence observed in natural phenomena to find solutions that are either near-global or global. Over the past few decades, numerous metaheuristic algorithms have been presented, drawing inspiration from natural processes, collective behavior, art, physics, and mathematical principles. Such as whale optimization algorithm (WOA) [31], particle swarm optimization (PSO) [11], grey wolf optimizer (GWO) [32], moth flame optimization (MFO) [30], cuckoo search (CS) [15], ant colony optimization [10], social spider optimization [25], mountain gazelle optimizer [1], dwarf mongoose optimization [2]. These studies showed that meta-heuristic algorithms can solve complex optimization problems with high accuracy and reasonable execution time. These algorithms have several advantages, such as simplicity of implementation, suitable accuracy and reasonable execution time. In clustering, which is considered as a problem of optimization with the aim of minimizing the sum of intra-cluster distances, choosing the appropriate metaheuristic algorithm is very important because the search domain is large and the goal of distance calculation is nonlinear [5, 6, 23, 28, 33, 40, 41].

The efficacy of MFO in producing a quick convergence rate and population diversity in clustering problems has been proven [42]. The PSO includes a built-in guidance strategy that helps solutions learn from better ones, improving their own solutions [21]. This article proposes a data clustering technique based on a combination of MFO and PSO called MFO_PSO_CLUST. The potential to avoid local optima and quickly converge to the global optimal solution is one of the main algorithm's features. Our main purpose is to cluster data using MFO_PSO_CLUST, aiming for more thorough search space coverage and improved accuracy compared to previously used methods. In brief, we will describe the new findings and contributions presented in this paper.

- The proposed approach MFO_PSO is a combination of PSO and MFO algorithms.
- Six datasets are utilized for evaluating the proposed approach: Iris, Wine, Banknote_authentication, Vowel, Glass, and Zoo.
- The proposed method's solution quality is evaluated against the MFO and PSO algorithms.
- The 23 benchmark functions are used to undertake tests and evaluations.
- MFO_PSO_CLUST method is presented for data clustering and evaluated in terms of FMI, NMI, and *Silhouette_score* criteria.
- The performance of the proposed approach is justified based on convergence curves and experimental results.

The sections of the article are organized as follows: Section 2 examines previous literature, while Section 3 introduces the main terms and definitions used in the paper. Section 4 presents the proposed approach based on MFO and PSO algorithms and utilizes it for the k-means clustering method. Section 5 includes experimental analysis using benchmark functions and the different datasets, demonstrating that the suggested approach is effective. The conclusion is discussed in Section 6.

2 Literature review

Several studies have been conducted in the field of data clustering; some of which are mentioned below. Saida, Nadjet, and Omar [36] proposed a data clustering with the CS optimization algorithm while Esmin, Coelho, and Matwin [13] proposed using the PSO as a globalized search algorithm for faster clustering convergence. However, complexity remains a significant challenge with this approach. Nasiri and Khiyabani [34] introduced a clustering method with the WOA, inspired by humpback whales' foraging behavior, and a technique for improving PSO using Renyi entropy clustering was presented by Emre Comak [9]. It sorts particles based on the entropy parameter to achieve the best outcomes. But when used to bigger datasets, this method becomes too complex and is not flexible enough to adjust to changing weights for the Renyi entropy parameter. Therefore, it is only appropriate for smaller datasets.

Qin-Hu Zhang et al. [46] presented the multivariant optimization algorithm, designed for global search to find optimal solutions. This method maintains the best cluster center values, offering stability and accuracy in addressing clustering challenges. However, its performance becomes more complex with increased dimensional size. Kumar and Sahoo [22] introduced a hybrid metaheuristic algorithm, MCSS-PSO, which combines MCSS and PSO algorithms. This approach improves exploration capabilities and ensures convergence at the global minimum instead of local minima. Indeed, its efficiency and robustness are primarily applicable to partition-based clustering methods. Jadhav and Gomathi [17] presented the KEGWO method, which efficiently calculates cluster centroids, while being excellent at identifying ideal centroids. However, this method may not be suitable for complex clustering. Jadhav and Gomathi [18] proposed the WGC algorithm to determine the appropriate center for the clustering algorithm. The WGC algorithm employs the new fitness function and hybrids the WOA and WEGWO. The WEGWO algorithm is a combination of the EGWO and the WOA algorithms, where the weight-based position update added to the GWO creates the EGWO.

Alswaitti, Albughdadi, and Isa [7] proposed a variance-based differential evolution algorithm for data clustering that improves both quality and convergence speed and includes an optional crossover, and an approach to improve clustering problems using the bio-inspired cuttlefish algorithm, aiming to find the best cluster centers to minimize clustering metrics was presented in [12]. Singh [39] presented WOA approach for data clustering because of its capacity to maintain search space diversity and its positive rate of convergence, and Singh et al. [42] proposed a novel method for solving data clustering problems that is based on the MFO. In addition, Kumar and Kumar [21] presented a fuzzy clustering technique aimed at enhancing the convergence performance of clustering methods. It enhances fuzzy c-means by creating a new measure that can withstand noisy surroundings. The PSO method is applied to solve the fuzzy c-means initialization problem. Preliminaries will be introduced, in the next section.

3 Preliminaries

Clustering is an unsupervised learning technique that helps organize and comprehend big data sets by grouping data into homogeneous clusters based on patterns and similarities. The main purpose of this classification is to put similar data in a cluster, while the data in different clusters have less similarity so that more appropriate information can be obtained from them [3, 8, 19]. There are different types of clustering. One of the most famous algorithms in clustering and classification is the k-means. The first step of this technique is to choose k points at random to serve as the initial cluster centers. Then, in an iterative loop, it finds the closest center for each object and assigns the object to that group. After, all objects are assigned to an appropriate group, the center of each group is recomputed. Until the centers converge, this process is repeated [14]. The k-means algorithm is indicated in Algorithm 1.

The primary challenge faced here is the identification of the best centroid. Many optimization algorithms are utilized for this purpose. Next, two types of these algorithms will be described.

Algorithm 1 k-means algorithm

Initialize k centroids randomly
while cluster assignments change **do**
  for each data point x_i **do**
    Assign x_i to the closest centroid c_j
  end for
  for each centroid c_j **do**
    Update the position of c_j to the mean position of all data points assigned to it
  end for
end while

3.1 Particle swarm optimization

The PSO is an approach for meta-heuristic optimization that is utilized for solving optimization problems. The algorithm starts by generating a population of particles, each particle representing a potential solution to the optimization problem. Each particle p_i with a specified position and a velocity, moves in the search space for the optimal solution. For each particle, the best position is the location with the best fitness value that the particle has yet encountered, and the global best position is the position with the best fitness value that any particle might find in the population.

The following equations are utilized to update the particle's position and velocity:

$$q_i(t+1) = wq_i(t) + c_1r_1(pbest_i - P_i(t)) + c_2r_2(gbest - P_i(t)), \tag{1}$$

$$P_i(t+1) = P_i(t) + q_i(t+1), \tag{2}$$

where $q_i(t+1)$ is the velocity of particle p_i at iteration $t+1$, $P_i(t)$ is the position of particle p_i at iteration t, $pbest_i$ is the personal best position of particle p_i, $gbest$ is the global best position of the population, w is a constant referred to as the inertia weight, c_1 and c_2 are acceleration constants, and r_1 and r_2 are random numbers in the range $[0,1]$ [11, 38]. The PSO algorithm is shown in Algorithm 2.

Algorithm 2 PSO algorithm

Initialize the swarm of particles S with n particles
Initialize the best position of each particle $pbest_i$ to its initial position
Initialize the best position of the swarm $gbest$ to the best position of any particle in S
Initialize the velocity of each particle p_i, q_i to 0
while stopping criterion not met **do**
    for each particle p_i in S **do**
        Update the particle p_i's velocity with (1)
        Update the particle p_i's position with (2)
        if fitness of the new position is an improvement over the fitness of $pbest_i$ **then**
            Update $pbest_i$ to the new position
        end if
        if fitness of the new position is an improvement over the fitness of $gbest$ **then**
            Update $gbest$ to the new position
        end if
    end for
end while

The simplicity and efficiency of PSO have made it a popular choice for optimization problems in many fields of research and industry.

3.2 Moth flame optimization

The MFO algorithm is a metaheuristic optimization algorithm inspired by the behavior of moths around a flame. The algorithm was presented by Mirjalili [30] in 2015. The foundational idea of MFO is to simulate the behavior of moths around a flame to optimize a given function. The flame represents the optimal solution, and the moths move towards the flame while also keeping exploration and exploitation in balance. The MFO algorithm consists of three phases: initialization, moth movement, and flame updating. In the first phase, the population of moths is randomly generated. It is shown in a matrix size of $n \times d$,

$$m_i = \vec{l} + \beta.(\vec{u} - \vec{l}). \tag{3}$$

The variable m_i represents the ith solution, $(m_{i,1} m_{i,2} \ldots m_{i,d})$, where i ranges from 1 to n, the population's size. The value of β is a randomly generated number between 0 and 1. The variable $\vec{l}$ represent the lower bound and the variable $\vec{u}$ represent upper bound for the problem being considered. The operator "." represents a point-by-point multiplication of the vectors. Each moth is evaluated using fitness value and included in a matrix of size $n \times 1$ called OM. Flame is another component in the MFO algorithm. They are also included in a matrix of size $n \times d$ called F. Every flame's fitness is assessed and added to an $n \times 1$ matrix called OF. In the moth movement phase, each moth searches for the flame by following a certain set of rules, which are based on the distance between the flame and the moth.

$$L(M_i, F_j) = D_i e^{bt} \cos(2\pi t) + F_j, \tag{4}$$

where D_i represents the distance from the ith moth to the jth flame, t is a random number in the range $[-1, 1]$, and for defining the shape of the logarithmic spiral of a moth, has been used b constant. Here, the calculation of D_i is as follows:

$$D_i = |F_j - M_i|, \tag{5}$$

where M_i denotes the ith moth and F_j denotes the jth flame. By updating their positions at n different locations in the search space, the moths can reduce the exploitation of the most promising solutions. Thus, to mitigate this effect, using the following formula, the number of flames reduces throughout the course of iterations:

$$\text{flameno} = \text{round}\left(F_j - It \times \frac{(F_j - 1)}{Maxit}\right), \tag{6}$$

where It represents the current iteration, $Maxit$ denotes the most number of iterations, and F_j corresponds to the maximum number of flames [37]. Until the termination condition is satisfied, the process continues. Finally, in the flame updating phase, the position of the flame is updated based on the position of the moths [30, 42]. The MFO algorithm is shown in Algorithm 3.

Algorithm 3 MFO algorithm

Initialize population $M = \{M_{n,1}, M_{n,2}, \ldots, M_{n,d}\}$ randomly.
Update flameno using (6).
(OM), Evaluate the fitness of each moth $M_{n,i}$ using the objective function.
while stopping criterion not met **do**
  $F = sorted(M)$;
  $OF = sorted(OM)$;
  for each moth $M_{n,i}$ in M **do**
    Calculate D_i using (5) for the corresponding moth;
    Update Moth(i) using (4) for the corresponding moth;
  end for
  Increment the iteration counter t;
end while
$F = sorted(M)$;
Return the best moth $F(0)$;

While the original PSO and MFO algorithms have demonstrated commendable optimization performance, they still encounter certain limitations, such as becoming trapped in local optima and exhibiting slow convergence. Therefore, we have presented an algorithm, which is the result of hybridizing the PSO and MFO algorithms, as a solution to this problem. The next section provides an explanation for this.

4 Proposed method

Today, numerous metaheuristic algorithms exist. The no free lunch theorem (NFL) [45] explicates the reason for the existence of numerous methods for optimization. NFL is a basic concept in machine learning and optimization, stating that there is not a single method for solving all optimization or machine learning problems. This principle emphasizes that not a single algorithm suits every optimization or machine learning task. In simpler terms, one algorithm may excel at solving one type of problem but not be as effective for another type. In consideration of the theorem above as well as the MFO and PSO algorithms, we made the decision to present a novel approach to incorporate these two techniques. When compared to the PSO and MFO methods, this method expressed better convergence and increased efficiency. The operation of this proposed approach will be explained in the next section. The proposed approach begins with the MFO algorithm. After this process, the optimum result determined by the objective function value is compared with the best solution. If it is improved, then the best solution is updated accordingly. Following that, the MFO algorithm's top s results are chosen and incorporated as input to the PSO algorithm to some extent, ensuring that the best solutions are further improved. The value of s is selected at random within the range of 3 particles and $\frac{1}{4}$ of the initial population. The chosen results replace the random initial population in PSO. The best s results from the MFO are assigned at the start of the initial population list in PSO, instead of creating a completely random starting population. Then, randomly generated solutions are inserted into the remaining spaces. After that, PSO is carried out with this improved initial population. Now, it is the turn of the PSO algorithm to run. The output produced by PSO is evaluated, and once it completes, the best result from PSO is compared with the best solution found overall. The best solution is updated once again if an improvement is found. Choose the top s outcomes from the optimal solutions determined by PSO. Following that, these values are added back into the MFO algorithm. The MFO algorithm's initial population is established through the utilization of a similar approach to PSO. The cluster centers are the end result of this iterative process, which continues until the required number of repeats is obtained. In summary, each iteration of the algorithm starts with the previous one and uses the output of the previous step as the input for the subsequent one. By investigating the results and taking consideration of the objective function value, the algorithm identifies the best cluster centers for the data. Algorithm 4 presents the suggested algorithm, and Figure 1 illustrates the method used to improve k-means clustering.

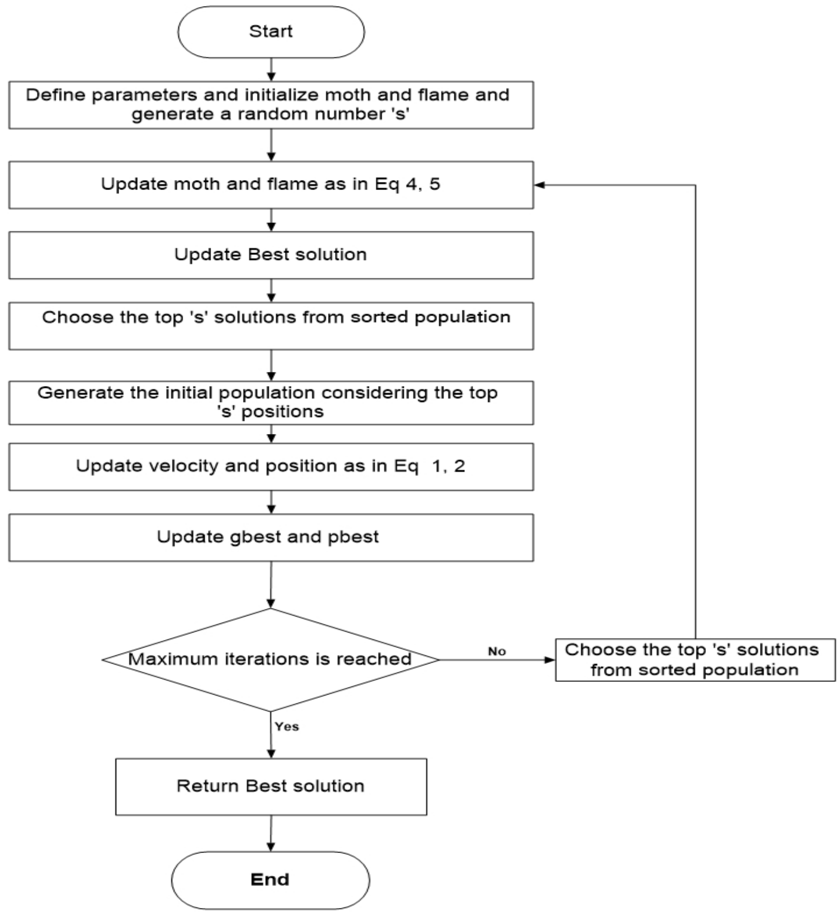

Figure 1: MFO_PSO flowchart

4.1 Clustering and the proposed method

To improve clustering, it is essential to find the most appropriate centroid of clusters that minimize intra-cluster distance and maximize the distance between clusters. It is possible to use optimization methods to identify the optimal cluster centers. As mentioned earlier, an optimization algorithm is a collection of algorithms designed to determine the best feasible solution or values for certain parameters. The aim of optimization while meeting certain constraints is to either reduce or increase an objective function. When employed for this purpose, rather than the global optimum, many optimization

Algorithm 4 MFO_PSO algorithm

```
Create a random number s and the population X.
while Stopping criteria not achieved do
    if iter = 1 then
        Use the MFO algorithm.
        if new solution's fitness is an improvement over the best_solution's fitness then
            Update best_solution
        end if
        Best_position is the top s solutions from sorted population (F);
    else
        {M_{n,1}, M_{n,2}, ..., M_{n,s}} = Best_position ;
        Initialize population {M_{n,s+1}, M_{n,s+2}, ..., M_{n,d}} randomly;
        M = {M_{n,1}, M_{n,2}, ..., M_{n,s}, M_{n,s+1}, ..., M_{n,d}} ;
        calculate the fitness of each moth M_{n,i}
        for each moth M_{n,i} in M do
            Calculate D_i using (5)
            Update Moth(i) using (4)
        end for
        if the new solution's fitness is superior to the best_solution's fitness then
            Update best_solution;
        end if
        {p_1, p_2, ..., p_s} = Best_position;
        Initialize population {p_{s+1}, p_{s+2}, ..., p_n} randomly;
        S = {p_1, p_2, ..., p_s, p_{s+1}, ..., p_n};
        for each particle p_i in S do
            Update the particle p_i's velocity with (1)
            Update the particle p_i's position with (2)
            if the new position's fitness is an improvement over the pbest_i's fitness then
                Update pbest_i;
            end if
            if the new position's fitness is an improvement over the gbest's fitness then
                Update gbest;
            end if
        end for
        Best_position = {pbest_1, pbest_2, ..., pbest_s};
        if new solution's fitness is an improvement over the best_solution's fitness then
            Update best_solution;
        end if
        iter=iter+1;
    end if
end while
```

approaches usually converge to a local optimum. Consequently, the necessity arises for an efficient optimization algorithm. Here, the focus is on utilizing a unite and conquer approach of optimization algorithms to identify centroids that minimize the distance from the remaining data points and achieve optimal performance. In this article, the proposed approach was utilized, which hybridizes the PSO and MFO algorithms, to improve clustering. The following explains the operation of the MFO_PSO_CLUST algorithm.

The goal of clustering is to minimize the intra-cluster distance while maximizing the inter-cluster distance. The objective function is expressed as:

$$Fit = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - c_i\|^2 \tag{7}$$

where x presents data points, c_i is cluster centroid, and C_i is data points in cluster i. The MFO_PSO algorithm aims to minimize Fit by optimizing the placement of centroids. MFO_PSO generates an initial population of candidate solutions, where each solution represents a set of centroids. The objective function Fit is computed for each solution to assess its clustering performance. Next, the MFO_PSO algorithm is executed, leading to the optimization of the centroids. The algorithm iterates until the maximum number of iterations is reached or minimal improvement in Fit is observed. The optimized centroids obtained from MFO_PSO are used to initialize k-means, ensuring high-quality clustering. Algorithm 5 displays the MFO_PSO_CLUST algorithm.

5 Results and discussion

Several widely used benchmark functions, each of which may display a distinct set of MFO_PSO_CLUST abilities, have been used to assess the efficiency of the MFO_PSO_CLUST algorithm. With only one global optimum point, the benchmark functions (F1-F7) show the optimization algorithms' exploitative powers. The definition of optimization techniques for exploration is done using standard functions (F8-F23). It is worth mentioning that the evaluation functions (F1-F23) are sourced from CEC 2005 special session [24, 43]. Tables 1, 2, and 3 display the functions properties and mathematical formulas. The MFO_PSO_CLUST algorithm was tested on a system comprising MATLAB R2022a and Windows 10 Enterprise 64-bit operating system. The hardware used included 4.00 GB of RAM and an Intel Core(TM) i5-3210M 2.50GHz CPU. Similar evaluations were conducted for MFO and PSO optimization algorithms. In these assessments, the

Algorithm 5 MFO_PSO_CLUST algorithm

Input: Data set $\{x_1, x_2, \ldots, x_n\}$, number of clusters k, maximum iterations T

Output: Optimized cluster centroids

Initialize population with candidate solutions (set of centroids) randomly

for each iteration $t = 1 \ldots T$ **do**

  for each solution in population **do**

    Evaluate the objective function Fit for the current centroid configuration

  end for

  Optimization Phase:

  for each moth in population **do**

    Update the position of the particle using the MFO_PSO algorithm towards optimal centroids

  end for

  if stopping criteria met (e.g., max iterations or minimal improvement in Fit) **then**

    Exit the loop

  end if

end for

Finalizing Clusters:

Use the optimized centroids obtained from MFO_PSO to initialize k-means

Run k-means with the optimized centroids to finalize clustering

MFO_PSO_CLUST algorithm was subjected to 30 populations, each with a maximum of 600 iterations. The results gained from 10 autonomous results were applied to compare the findings. The parameters of corresponding optimization methods were set using the settings available in the MFO [30] and PSO [11]. Details of the parameters that these optimization techniques used are presented in Table 4. In Table 2, the values of y_i and $Q(x_i, a, k, m)$ are computed as follows:

$$y_i = 1 + \frac{x_i + 1}{4},$$

$$Q(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & \text{if } x_i > a \\ 0 & \text{if } -a < x_i < a \\ k(-x_i - a)^m & \text{if } x_i < -a \end{cases}.$$

Table 1: Information on unimodal benchmark functions [24, 43].

No	Function	Dimensions	Range	Fmin
F1	$f(x) = \sum_{i=1}^{d} x_i^2$	30	$[-100, 100]^d$	0
F2	$f(x) = \sum_{i=1}^{d} \lvert x_i\rvert + \prod_{i=1}^{d} \lvert x_i\rvert$	30	$[-10, 10]^d$	0
F3	$f(x) = \sum_{i=1}^{d}(\sum_{j=1}^{i} x_j)^2$	30	$[-100, 100]^d$	0
F4	$f(x) = \max_i\{\lvert x_i\rvert\}, 1 \leq i \leq d$	30	$[-100, 100]^d$	0
F5	$f(x) = \sum_{i=1}^{d-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-30, 30]^d$	0
F6	$f(x) = \sum_{i=1}^{d}(\lvert x_i + 0.5\rvert)^2$	30	$[-100, 100]^d$	0
F7	$f(x) = \sum_{i=1}^{d} ix_{4i} + \text{random}[0, 1)$	30	$[-128, 128]^d$	0

5.1 Qualitative and quantitative MFO_PSO results

In this section, MFO_PSO quality was assessed using six common benchmark functions (F1, F3, F7, F8, F12, and F13). Three main factors were examined: convergence behavior, average population fitness, and changes in the first dimension of the first particle. During optimization, the diagram of convergence behavior tracked the cost of the best solution, while the average population fitness curve displayed how the cost for each search parameters changed during the optimization procedure. Moreover, the modifications made to the first particle were shown in the plots for its first dimension. The convergence diagrams for the best solution during optimization are shown in Figure 2. They demonstrate that MFO_PSO exhibits strong convergence capabilities with a fast decrease in cost.

Table 2: Information on multimodal benchmark functions [24, 43].

No	Function	Dimensions	Range	Fmin
F8	$f(x) = \sum_{i=1}^{d} \left(x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)\right)$	30	[-500, 500]d	$-418.9829 \times n$
F9	$f(x) = 10d + \sum_{i=1}^{d} \left[x_i^d - 10\cos(2\pi x_i)\right]$	30	[-5.12, 5.12]d	0
F10	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + e$	30	[-32, 32]d	0
F11	$f(x) = \frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[-600, 600]d	0
F12	$f(x) = \frac{\pi}{d}\left\{10\sin(\pi y_1) + \sum_{i=1}^{d-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right] + (y_d-1)^2\sum_{i=1}^{d} Q(x_i,10,100,4)\right\}$	30	[-50, 50]d	0
F13	$f(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{d}(x_i-1)^2\left[1+\sin^2(3\pi x_i+1)\right] + (x_d-1)^2\right\} + \sum_{i=1}^{d} Q(x_i,5,100,4)$	30	[-50, 50]d	0

Table 3: Information on benchmark functions [24, 43].

No	Function	Dimensions	Range	Fmin
F14	$f(x) = \left[\frac{1}{500} + \sum_{i=1}^{25} \frac{1}{i+\sum_{j=1}^{2}(x_j - a_{j,i})^6}\right]^{-1}$	30	$[-65, 65]^d$	-1
F15	$f(x) = \sum_{i=1}^{d} [a_i - \frac{x_1(b_{2i}+b_i x_2)}{b_i{}^2+b_i x_3+x_4}]^2$	30	$[-5, 5]^d$	0.00030
F16	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	30	$[-5, 5]^d$	-1.0316
F17	$f(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	30	$[-5, 5]^d$	0.398
F18	$f(x) = \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right]$			
	$\times \quad \left[30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$	30	$[-2, 2]^d$	3
F19	$f(x) = -\sum_{i=1}^{4} a_i \exp\left(-\sum_{j=1}^{3} b_{ij}(x_j - p_{ij})^2\right)$	30	$[1, 3]^d$	-3.86
F20	$f(x) = -\sum_{i=1}^{4} a_i \exp\left(-\sum_{j=1}^{6} b_{ij}(x_j - p_{ij})^2\right)$	30	$[0, 1]^d$	-3.32
F21	$f(x) = -\sum_{i=1}^{5} \left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	30	$[0, 10]^d$	-10.1532
F22	$f(x) = -\sum_{i=1}^{7} \left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	30	$[0, 10]^d$	-10.4028
F23	$f(x) = -\sum_{i=1}^{10} \left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	30	$[0, 10]^d$	-10.5363

Table 4: Parameters for comparing and evaluating optimization algorithms.

Algorithm	Parameter	Value
PSO	v_{Max}	0.1
	c_1	2
	c_2	2
MFO	Convergence constant a	[-2, -1]
	Logarithmic spiral constant b	1
MFO_PSO_CLUST	v_{Max}	0.1
	c_1	1.496
	c_2	1.496
	Convergence constant a	[-2, -1]
	Logarithmic spiral constant b	1

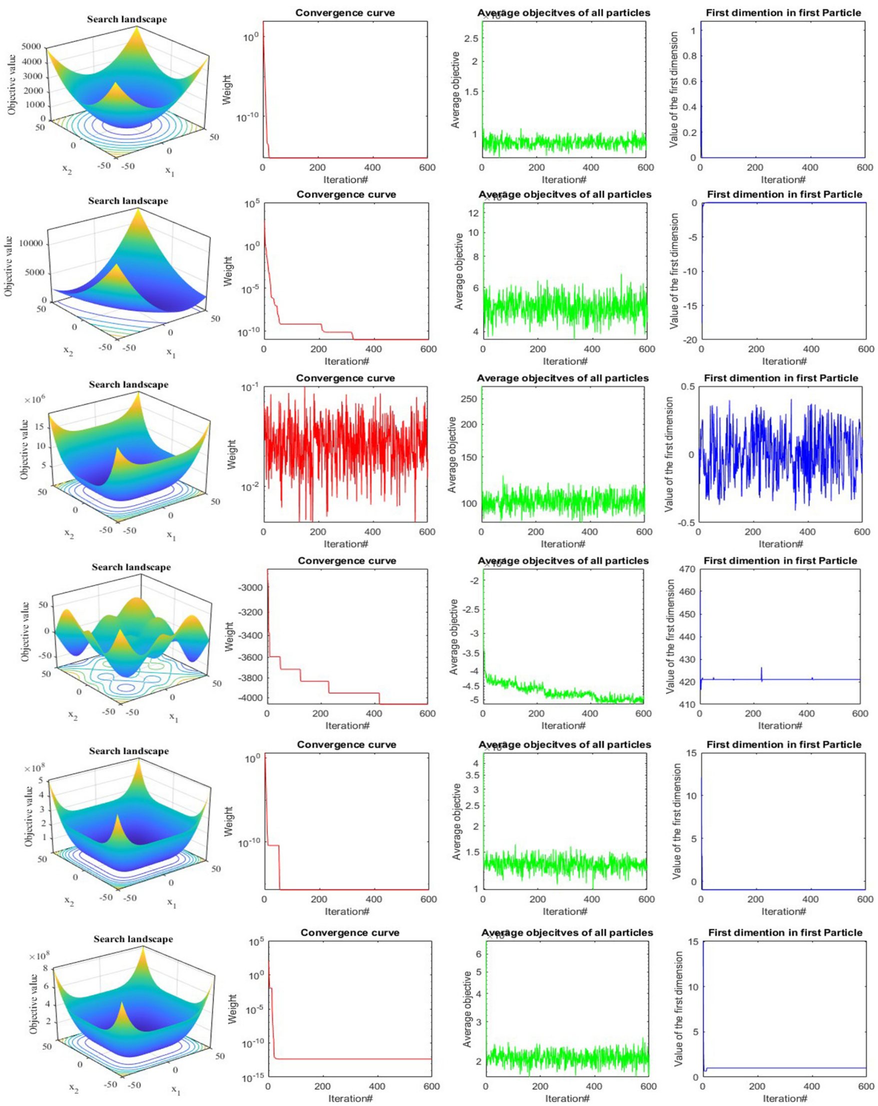

Figure 2: The qualitative results of F1, F3, F7, F8, F12, and F13 function using MFO_PSO algorithm

The quantitative efficiency of MFO_PSO was tested and assessed. The results achieved by MFO_PSO were evaluated with the results of two other optimization methods. The testing utilized 23 benchmark functions, including scalable F1-F23 functions. The tests included results from 10 independent iterations and included the AVG, Worst, and Best mean error criterion. The outcomes of scalability evaluations for the F1-F23 functions, the MFO_PSO algorithm, and associated algorithms are indicated in Table 5. MFO_PSO performed better than two other optimization algorithms (MFO, PSO) across various sizes and the majority of benchmark functions. This is highlighted in Table 5, where the results of MFO_PSO are in comparison to those of MFO and PSO algorithms for the benchmark functions F1-F23 with a dimension of 30.

5.2 Experiential results of the MFO_PSO_CLUST algorithm

This section includes multiple examples to show the efficiency of our proposed approach for improving k-means clustering by comparing it with other methods. We utilized this hybrid optimization method to Iris, Wine, Banknote_authentication, Vowel, Glass and Zoo datasets and evaluated its efficiency in finding optimal solutions.

Table 6 evaluates the fitness values for different datasets, where the MFO_PSO_CLUST, MFO_CLUST, and PSO_CLUST algorithms were utilized. The fitness values, including the best, mean, and worst outcomes, were analyzed for each dataset with the corresponding number of clusters. Each algorithm was examined using 30 populations, with each run having a maximum of 500 iterations and 10 independent iterations for each dataset. The results reported in the table represent the average values obtained from 10 iterations.

For the Iris dataset, consistent fitness values 96.6556 were achieved by the MFO_PSO_CLUST algorithm across the best, mean, and worst cases, indicating stable performance. In contrast, a significantly higher mean fitness value 205.9479 was observed for MFO_CLUST, however, its variability (best: 195.5371, worst: 212.1155) suggested less stability. Average performance with slightly higher variability (mean: 135.4312, worst: 146.0318) was demonstrated by PSO_CLUST. In the Banknote_authentication dataset, the best fitness value (7200.6576) was achieved by the MFO_PSO_CLUST

Table 5: Benchmark function results (F1-F23) using 30 dimensions.

No.Function	Type	MFO_PSO	PSO	MFO
F1	Best	4.7922E-21	7.1071E-09	5.3715E-01
	Worst	6.6727E-11	7.3614E-05	2.0002E+04
	Mean	1.1744E-11	4.4590E-06	1.6722E+03
F2	Best	2.9181E-12	5.6941E-06	1.7467E-01
	Worst	2.3649E-06	2.0504E-02	8.0223E+01
	Mean	5.8226E-07	3.3272E-03	3.5329E+01
F3	Best	3.4763E-12	1.8171E+01	2.3463E+03
	Worst	8.0948E-09	3.4851E+03	4.7370E+04
	Mean	1.5474E-09	5.8934E+02	2.0003E+04
F4	Best	5.1595E-06	2.6406E-01	4.7078E+01
	Worst	2.9516E-05	2.1015E+00	8.5407E+01
	Mean	1.4508E-05	7.9578E+00	6.9113E+01
F5	Best	6.4368E-04	1.9874E+01	1.3794E+02
	Worst	7.6082E+00	1.0846E+02	8.0123E+07
	Mean	8.607E-01	4.6616E+01	2.6939E+06
F6	Best	1.0626E-20	6.7447E-09	7.4584E-01
	Worst	2.2514E-10	4.1088E-05	1.9801E+04
	Mean	4.1271E-11	2.9114E-06	2.6696E+03
F7	Best	1.6307E-02	4.0726E-02	4.0524E-02
	Worst	6.9200E-01	1.5602E-01	5.9234E+01
	Mean	3.5400E-02	9.5056E-02	2.7669E+02
F8	Best	-4.1898E+03	-3.315E+03	-1.0353E+04
	Worst	-3.8345E+03	-1.949E+03	-6.3081E+03
	Mean	-4.0359E+03	-2.590E+03	-8.5561E+03
F9	Best	0	1.9899E+01	1.0158E+02
	Worst	4.0956e-11	7.2632E+01	2.6097E+02
	Mean	7.0031e-12	3.8671E+01	1.5809E+02
F10	Best	2.1231e-08	3.7741E-06	1.4912E+00
	Worst	4.6272e-06	2.4083E+00	1.9964E+01
	Mean	1.4335e-06	3.1302E-01	1.5907E+01
F11	Best	2.4605E-02	6.3190E+01	6.3238E-01
	Worst	9.8345E-02	1.0377E+02	9.1184E+01
	Mean	4.9904E-02	8.3190E+01	3.9966E+01
F12	Best	2.2284e-22	9.3714E-11	6.5723E-01
	Worst	7.4927e-13	1.5674E+00	3.4177E+02
	Mean	1.8579e-13	2.4928E-01	2.1472E+01
F13	Best	1.5237e-17	2.9246E-11	2.9276E+00
	Worst	3.9970e-12	1.1006E-02	4.1582E+03
	Mean	8.6499e-13	2.5662E-03	1.7993E+02
F14	Best	9.9800E-01	9.9800E-01	9.9800E-01
	Worst	9.9800E-01	1.9926E+00	1.0763E+01
	Mean	9.9800E-01	1.3294E+00	2.3131E+00
F15	Best	3.0749e-04	3.0749E-04	0.0011604
	Worst	3.0749e-04	2.0363E-02	4.1582E+03
	Mean	3.0749e-04	1.2877E-03	1.7993E+02
F16	Best	-1.0316E+00	-1.0316E+00	-1.0316E+00
	Worst	-1.0316E+00	-1.0316E+00	-1.0316E+00
	Mean	-1.0316E+00	-1.0316E+00	-1.0316E+00
F17	Best	3.9789E-01	3.9789E-01	3.9789E-01
	Worst	3.9791E-01	3.9789E-01	3.9789E-01
	Mean	3.9789E-01	3.9789E-01	3.9789E-01
F18	Best	3.0000E+00	3.0000E+00	3.0000E+00
	Worst	3.0000E+00	3.0000E+00	3.0000E+00
	Mean	3.0000E+00	3.0000E+00	3.0000E+00
F19	Best	-3.8628E+00	-3.8628E+00	-3.8628E+00
	Worst	-3.8628E+00	-3.8549E+00	-3.8628E+00
	Mean	-3.8628E+00	-3.8620E+00	-3.8628E+00
F20	Best	-3.3220E+00	-3.3220E+00	-3.3220E+00
	Worst	-3.2031E+00	-2.9564E+00	-3.1376E+00
	Mean	-3.2150E+00	-3.2389E+00	-3.2324E+00
F21	Best	-1.0153E+01	-1.0153E+01	-1.0153E+01
	Worst	-5.1008E+00	-2.6305E+00	-2.6305E+00
	Mean	-8.6375E+00	-6.7321E+00	-5.8820E+00
F22	Best	-1.0403E+01	-1.0403E-01	-1.0403E-01
	Worst	-1.0403E+01	-2.7519E+00	-2.7519E+00
	Mean	-1.0403E+01	-6.7370E+00	-7.4010E+00
F23	Best	-1.0536E+01	-1.0536E+01	-1.0536E+01
	Worst	-1.0536E+01	-2.4217E+00	-2.4217E+00
	Mean	-1.0536E+01	-7.2984E+00	-8.0934E+00

algorithm, which outperformed MFO_CLUST 7674.9282 and PSO_CLUST (7348.2371). The mean fitness value for MFO_PSO_CLUST (7200.6908) indicated high consistency, whereas greater variation was observed for the other

algorithms. For the Wine dataset, the lowest best fitness value (16292.1846) was achieved by MFO_PSO_CLUST, outperforming both MFO_CLUST (8379
4.3558) and PSO_CLUST (16409.438). Robustness in MFO_PSO_CLUST was evidenced by its nearly identical mean and worst fitness values. In the Vowel dataset, MFO_PSO_CLUST achieved superior performance
(148967.2408) compared to MFO_CLUST (1036524.9394) and PSO_CLUST (177395.8541). Its lower mean fitness value (150320.9081) further highlighted its efficiency. For the Glass dataset, the best fitness value (210.4287) was achieved by MFO_PSO_CLUST, surpassing MFO_CLUST (939.6499) and PSO_CLUST (388.7199). The lower variability in its fitness values indicated superior stability compared to the other algorithms. In the Zoo dataset, MFO_PSO_CLUST demonstrated the best performance (best fitness value: 101.1195) in comparison to MFO_CLUST (222.2408) and PSO_CLUST (181.4975). Its consistent mean fitness value 107.1153 further reinforced its reliability. Across all datasets, MFO_PSO_CLUST consistently achieved better fitness values with minimal variation between the best, mean, and worst outcomes. This high consistency in its optimization process demonstrated the robustness and reliability of MFO_PSO_CLUST. In contrast, MFO_CLUST exhibited significant variability, suggesting susceptibility to suboptimal solutions in some iterations. Although PSO_CLUST displayed moderate performance, it failed to match the accuracy and stability of MFO_PSO_CLUST consistently. Altogether, MFO_PSO_CLUST demonstrated superior performance across all datasets by achieving better fitness values with a high degree of consistency and robustness. These results underscore its effectiveness in addressing diverse clustering challenges when compared to MFO_CLUST and PSO_CLUST. Table 6 presents additional detailed results.

Figure 3 illustrates the convergence behavior of the MFO_PSO_CLUST, MFO_CLUST, and PSO_CLUST algorithms across six datasets: Iris, Banknote_authentication, Wine, Zoo, Vowel, and Glass. These figures display the algorithm's progress towards the optimal solution across a defined number of iterations. For the Iris dataset, the MFO_PSO_CLUST algorithm converges rapidly to a stable solution with a significantly lower score

Table 6: Evaluation of the fitness values for different datasets.

Algorithm	Dataset	MFO_PSO_CLUST	MFO_CLUST	PSO_CLUST	No. cluster	No. iteration
best fit. val	Iris	96.6556	195.5371	119.7239	3	10
mean fit. val		96.6555	205.9479	126.0913		
worst fit. val		96.6555	212.1155	132.3762		
best fit. val	Banknote_	7200.6576	7674.9282	7348.2371	2	10
mean fit. val	authentication	7200.6908	7896.7872	7488.0422		
worst fit. val		7200.9767	8186.5658	7610.7892		
best fit. val	Wine	16292.1846	83794.3558	16409.438	3	10
mean fit. val		16292.2329	83794.6123	16469.1824		
worst fit. val		16292.6672	83794.9193	16520.0933		
best fit. val	Vowel	148967.2408	1036524.9394	177395.8541	6	10
mean fit. val		150320.9081	1036524.9394	185652.7939		
worst fit. val		160668.3641	1036524.9394	193752.0194		
best fit. val	Glass	210.4287	939.6499	388.7199	6	10
mean fit. val		226.0724	948.786	405.1631		
worst fit. val		246.6828	957.2347	423.1936		
best fit. val	Zoo	101.1195	202.2408	181.4975	7	10
mean fit. val		107.1153	212.7521	185.0522		
worst fit. val		112.0928	222.7578	189.0228		

compared to MFO_CLUST and PSO_CLUST, demonstrating superior efficiency and faster convergence. In contrast, MFO_CLUST exhibits a slower and less stable convergence process, while PSO_CLUST reaches a higher score, indicating suboptimal performance. In the Banknote_authentication dataset, MFO_PSO_CLUST achieves the best score earlier in the iterations, outperforming both MFO_CLUST and PSO_CLUST. The curves for MFO_CLUST and PSO_CLUST display higher variability, reflecting less consistent optimization performance. For the Wine dataset, the MFO_PSO_-CLUST algorithm demonstrates robust convergence, achieving a stable and optimal solution earlier than the other methods. MFO_CLUST and PSO_CLUST converge at higher scores, with MFO_CLUST showing slower convergence and higher variability throughout the iterations. The Zoo dataset showcases the significant advantage of MFO_PSO_CLUST, which converges rapidly to a stable and optimal solution. Both MFO_CLUST and PSO_CLUST exhibit prolonged and less effective convergence, with higher final scores, indicating poorer optimization. In the Vowel dataset, the performance trend remains consistent, with MFO_PSO_CLUST converging to the lowest score, followed by MFO_CLUST and PSO_CLUST. The gap between the algorithms final scores further emphasizes the superior performance of MFO_PSO_CLUST in minimizing the fitness function. Finally, for the Glass dataset, MFO_PSO_CLUST achieves the best score with minimal variability, outperforming the other algorithms. Both MFO_CLUST and PSO_CLUST converge at significantly higher scores, showcasing their limitations in handling this dataset.

The convergence curves across all datasets indicate the consistent superiority of MFO_PSO_CLUST in terms of faster convergence, lower fitness values, and greater stability, making it a highly robust and efficient algorithm for clustering tasks.

The efficiency evaluation of the proposed approach, utilizing the Iris, Wine, and Banknote_authentication datasets, is shown in Table 7. It is assessed in terms of the Normalized mutual information (NMI), Fowlkes-Mallows index (FMI), and $Silhouette\_score$ criteria [16, 35]. The table mentioned shows optimal efficiency ranges. For all criteria, A value that is larger and closer to 1 indicates higher efficiency.

Table 7: Assessing MFO_PSO_CLUST and k-means using evaluation criteria

	Dataset	MFO_PSO_CLUST	k-means	number of cluster	Note
NMI	Wine	0.42876	0.42411	3	The ranges [0,1].
	Iris	0.75821	0.74193	3	A higher index indicates
	Banknote_authentication	**0.030324**	**0.030324**	2	greater similarity.
	vowel	0.4948	0.48029	6	
	Glass	0.42693	0.4126	6	
	Zoo	0.7616	0.73226	7	
FMI	Wine	0.58387	0.5314	3	The ranges [0,1].
	Iris	0.80518	0.79821	3	A higher index indicates
	Banknote_authentication	**0.52769**	**0.52769**	2	greater similarity.
	vowel	0.47726	0.4635	6	
	Glass	**0.40502**	**0.40813**	6	
	Zoo	0.81281	0.74278	7	
Silhouette_score	Wine	0.7323	0.71578	3	The ranges [0,1].
	Iris	0.73546	0.73421	3	A higher index indicates
	Banknote_authentication	**0.63519**	**0.63519**	2	greater similarity.
	vowel	0.53741	0.53001	6	
	Glass	0.65475	0.60565	6	
	Zoo	0.6264	0.53085	7	

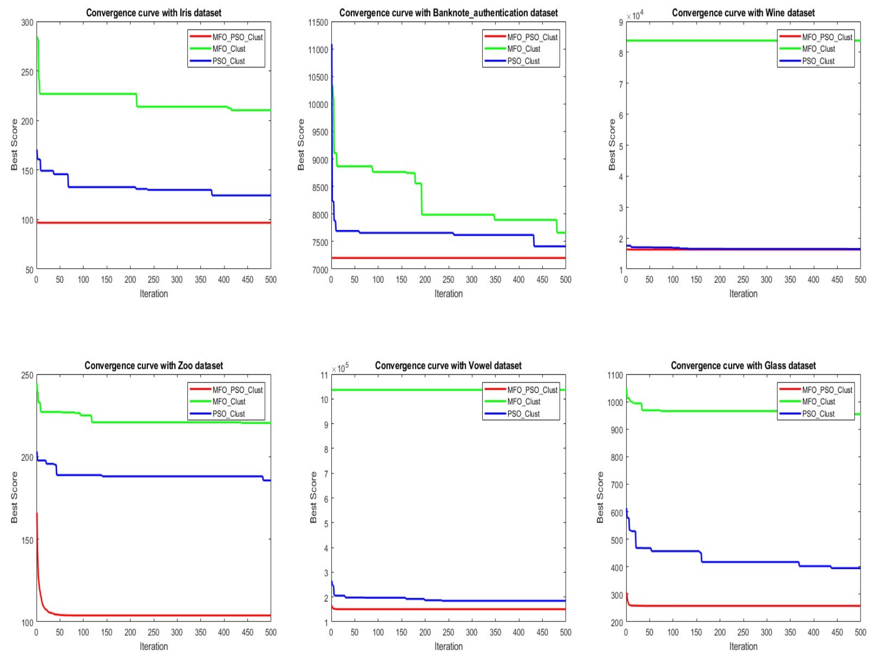

Figure 3: Convergence curve MFO_PSO_CLUST, MFO_CLUST and PSO_CLUST with different datasets

The NMI is a technique used for the analysis of clustering's performance against the true labeling. The results indicate that MFO_PSO_CLUST performs better than k-means in most datasets, such as Iris (0.75821 vs. 0.74193) and Zoo (0.7616 vs. 0.73226). However, for the Banknote_authentication dataset, both algorithms performed poorly, achieving an identical NMI score of 0.030324, which highlights a potential challenge in clustering this specific dataset. FMI metric assesses the balance between precision and recall in clustering. The MFO_PSO_CLUST algorithm demonstrated superior performance across most datasets, except for Glass, where k-means performed slightly better (0.40813 vs. 0.40502). Notably, for the Iris dataset, MFO_PSO_CLUST achieved a significantly higher FMI score (0.80518 vs. 0.79821), indicating a more balanced and accurate clustering. *Silhouette _score* measure evaluates the cohesion within clusters and the separation between clusters. MFO_PSO_CLUST consistently outperformed k-means in most datasets, including Wine (0.7323 vs. 0.71578) and Zoo (0.6264 vs.

0.53085). Based on these results, the new algorithm can generate tighter and better-separated clusters of data points as opposed to k-means clustering. The results indicate that MFO_PSO_CLUST generally performs better than k-means in terms of clustering quality, particularly in the NMI and *Silhouette_score* metrics. This demonstrates the effectiveness of the proposed algorithm in producing more coherent and meaningful clusters across various datasets. However, there are specific cases, such as the Glass dataset with the FMI metric, where k-means shows a slight advantage. The uniqueness of this dataset may be an explanation for this results in the clustering performance.

6 Conclusion

In this article, using the metaheuristic algorithm and their combination, we achieved better results for clustering. The advantages of several algorithms can be utilized to get better results by merging them. MFO and PSO methods were combined in the MFO_PSO algorithm. In the first part, MFO_PSO was tested with 23 benchmark functions and proved to be a robust algorithm for solving various problems. Its excellent performance in various tests suggests that it consistently finds superior solutions. Then, the MFO_PSO algorithm was applied to cluster data and determine the centroid with the lowest fitness function as the optimal center. Once the best centroid is identified by utilizing the MFO_PSO_CLUST algorithm, the data are clustered using it. Based on three datasets and three NMI, FMI, and *Silhouette_score* criteria, performance analysis demonstrated that the MFO_PSO_CLUST algorithm outperforms the MFO_CLUST and PSO_CLUST methods. The outcomes of this method can be further improved by adjusting the parameters.

Data Availability Statement

The datasets used in this paper are as follows:

- The Banknote_authentication dataset from https://archive.ics.uci.edu/ml/datasets/banknote+authentication.
- The Iris dataset from https://archive.ics.uci.edu/dataset/53/iris.
- The Vowel dataset from https://archive.ics.uci.edu/ml/datasets/Vowel.
- The Zoo dataset from https://archive.ics.uci.edu/dataset/111/zoo.
- The Glass dataset from https://archive.ics.uci.edu/dataset/42/glass+identification.
- The Wine dataset from https://archive.ics.uci.edu/dataset/109/wine.

Acknowledgements

Authors are grateful to there anonymous referees and editor for their constructive comments.

References

[1] Abdollahzadeh, B., Gharehchopogh, F.S., Khodadadi, N. and Mirjalili, S. *Mountain gazelle optimizer: a new nature-inspired metaheuristic algorithm for global optimization problems*, Advances in Engineering Software, 174 (2022), 103282.

[2] Agushaka, J.O., Ezugwu, A.E. and Abualigah, L. *Dwarf mongoose optimization algorithm*, Comput. Methods Appl. Mech. Engrg, 391 (2022), 114570.

[3] Ahmad, A. and Dey, L. *A k-mean clustering algorithm for mixed numeric and categorical data*, Data & Knowledge Engineering, 63 (2007), 503–27.

[4] Alhawarat, M. and Hegazi, M. *Revisiting K-Means and topic modeling, a comparison study to cluster arabic documents*, IEEE Access, 6 (2018), 42740–42749.

[5] Aljarah, I., Mafarja, M., Heidari, A.A., Faris, H. and Mirjalili, S. *Clustering analysis using a novel locality-informed grey wolf-inspired clustering approach*, Knowledge and Information Systems, 62 (2020), 507–539.

[6] Almasri, A., Alkhawaldeh, R.S. and Çelebi, E. *Clustering-Based EMT Model for Predicting Student Performance*, Arab. J. Sci. Eng, 45 (2020), 10067–10078.

[7] Alswaitti, M., Albughdadi, M. and Isa, N.A.M. *Variance-based differential evolution algorithm with an optional crossover for data clustering*, Appl. Soft Comput, 80 (2019), 1–17.

[8] Barbakh, W.A. Wu, Y. and Fyfe, C. *Review of clustering algorithms, In Non-Standard Parameter Adaptation for Exploratory Data Analysis*, Springer: Berlin/Heidelberg, Germany, (2009), 7–28.

[9] Çomak, E. *A modified particle swarm optimization algorithm using Renyi entropy-based clustering*, Neural Computing and Applications, 27 (5) (2016), 1381–1390.

[10] Dorigo, M., Bonabeau, E. and Theraulaz, G. *Ant algorithms and stigmergy*, Future Generation Computer System, 16 (2000), 851–871.

[11] Eberhart, R. and Kennedy, J. *Particle swarm optimization*, InProceedings of the IEEE international conference on neural networks, 4 (1995), 1942-1948.

[12] Eesa, A.S. and Orman, Z. *A new clustering method based on the bio-inspired cuttlefish optimization algorithm*, Expert Syst, 37 (2020), e12478.

[13] Esmin, A.A., Coelho, R.A. and Matwin, S. *A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data*, Artif. Intell. Rev, 44 (2015), 23–45.

[14] Gan, G., Ma, C. and Wu, J. *Data clustering: theory, algorithms, and applications*, SIAM, Society for Industrial and Applied Mathematics, (2007).

[15] Gandomi, A.H., Yang, X.S. and Alavi, A.H. *Cuckoo search algorithm, a metaheuristic approach to solve structural optimization problems*, Eng. Comput, 29 (2013), 17–35.

[16] Hu, F., Liu, J., Li, L. and Liang, J. *Community detection in complex networks using Node2vec with spectral clustering*, Physica A: Statistical Mechanics and its Applications, 545 (2020), 123633.

[17] Jadhav, A.N. and Gomathi, N. *Kernel-based exponential grey wolf optimizer for rapid centroid estimation in data clustering*, Jurnal Teknologi, 78 (11) (2016), 65–74.

[18] Jadhav, A.N. and Gomathi, N. *WGC: Hybridization of exponential grey wolf optimizer with whale optimization for data clustering*, Alex. Eng. J, 57 (2018), 1569–1584.

[19] Jain, A.K. *Data clustering: 50 years beyond K-means*, Pattern Recognit. Lett, 31 (2010), 651–666.

[20] Jain, A.K. and Dubes, R.C. *Algorithms for Clustering Data*, Englewood Cliffs, NJ, USA: Prentice-Hall, (1988).

[21] Kumar, N. and Kumar, H. *A fuzzy clustering technique for enhancing the convergence performance by using improved Fuzzy c-means and Particle Swarm Optimization algorithms*, Data & Knowledge Engineering, 140 (2022), 102050.

[22] Kumar, Y. and Sahoo, G. *Hybridization of magnetic charge system search and particle swarm optimization for efficient data clustering using neighborhood search strategy*, Soft Computing, 19 (12) (2015), 3621–3645.

[23] Kushwaha, N., Pant, M., Kant, S. and Jain, V.K. *Magnetic optimization algorithm for data clustering*, Pattern Recognition Letters, 115 (2018), 59–65.

[24] Liang, J., Suganthan, P. and Deb, K. *Novel composition test functions for numerical global optimization, in Swarm Intelligence Symposium*, 2005. SIS 2005, Proceedings 2005 IEEE, (2005), 68–75.

[25] Luque-Chang, A., Cuevas, E., Fausto, F., Zaldívar, D. and Pérez, M. *Social spider optimization algorithm: modifications, applications*, and perspectives, Mathematical Problems in Engineering, 1 (2018), 6843923.

[26] Lv, Z., Liu, T., Shi, C., Benediktsson, J.A. and Du, H. *Novel land cover change detection method based on k-Means clustering and adaptive majority voting using bitemporal remote sensing images*, IEEE Access, 7 (2019), 34425–34437.

[27] MacQueen, J. *Some methods for classification and analysis of multivariate observations*, in Proc. 5th Berkeley Symp. Math. Statist. Probab, 1 (1967), 281–297.

[28] Mansalis, S., Ntoutsi, E., Pelekis, N. and Theodoridis, Y. *An evaluation of data stream clustering algorithms*, Statistical Analysis and Data Mining: The ASA Data Science Journal, 11 (2018), 167–187.

[29] Meng, Y., Liang, J., Cao, F. and He, Y. *A new distance with derivative information for functional k-means clustering algorithm*, Inf. Sci. 463 (2018), 166–185.

[30] Mirjalili, S. *Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm*, Knowledge-based systems. 89 (2015), 228–249.

[31] Mirjalili, S. and Lewis, A. *The whale optimization algorithm*, Adv. Eng. Softw. 95 (2016), 51–67.

[32] Mirjalili, S., Mirjalili, S.M. and Lewis, A. *Grey wolf optimizer*, Advances in engineering software. 69 (2014), 46–61.

[33] Nanda, S.J. and Panda, G. *A survey on nature inspired metaheuristic algorithms for partitional clustering*, Swarm Evol. Comput. 16 (2014), 1–18.

[34] Nasiri, J. and Khiyabani, F.M. *A whale optimization algorithm (WOA) approach for clustering*, Cogent Math. Stat. 5 (2018), 1483565.

[35] Rodriguez, M.Z., Comin, C.H.v Casanova, D., Bruno, O.M., Amancio, D.R., Costa, L.D.F. and Rodrigues, F.A. *Clustering algorithms: A comparative approach*, PLoS ONE 14(1), e0210236 (2019). https://doi.org/10.1371/journal.pone.0210236

[36] Saida, I.B., Nadjet, K. and Omar, B. *A new algorithm for data clustering based on cuckoo search optimization, In Genetic and Evolutionary Computing*, Springer: Berlin/Heidelberg, Germany. (2014), 55–64.

[37] Sayed, G.I. and Hassanien, A.E. *A hybrid SA-MFO algorithm for function optimization and engineering design problems*, Complex & Intelligent Systems. 4 (2018), 195–212.

[38] Shojaee, Z., Shahzadeh Fazeli, S.A., Abbasi, E. and Adibnia, F. *Feature Selection based on Particle Swarm Optimization and Mutual Information*, AI and Data Mining. 9(1) (2021), 39–44.

[39] Singh, T. *A novel data clustering approach based on whale optimization algorithm*, Expert Syst. 38 (2020), e12657.

[40] Singh, T. and Mishra, K.K. *Data clustering using environmental adaptation method In International Conference on Hybrid Intelligent Systems*, Springer. (2019), 156–164.

[41] Singh, T., Mishra, K.K. and Ranvijay. *A variant of EAM to uncover community structure in complex networks*, Int. J. Bio-Inspired Comput. 16 (2020), 102–110.

[42] Singh, T., Saxena, N., Khurana, M., Singh, D., Abdalla, M. and Alshazly, H. *Data clustering using moth-flame optimization algorithm*, Sensors. 21(12) (2021), 4086.

[43] Suganthan, P.N., Hansen, N., Liang, J. J., Deb, K., Chen, Y.P., Auger, A. and Tiwari, S. *Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization*, KanGAL report. 2005005 (2005).

[44] Talevi, A. and Bellera, CL. *Clustering of small molecules: new perspectives and their impact on natural product lead discovery*, Frontiers in Natural Products. 3 (2024), 1367537.

[45] Wolpert, D.H. and Macready, W.G. *No free lunch theorems for optimization*, IEEE Trans Evol Comput. 1 (1997), 67–82.

[46] Zhang, Q.H., Li, B.L., Liu, Y.J., Gao, L., Liu, L.J. and Shi, X.L. *Data clustering using multivariant optimization algorithm*, Int. J. Mach. Learn. Cybern. 7 (5) (2016), 773–782.

[47] Zhu, J., Jiang, Z., Evangelidis, G.D., Zhang, C., Pang, S. and Li, Z. *Efficient registration of multi-view point sets by K-means clustering*, Inf. Sci. 488 (2019), 205–218.