Contemporary Mathematics
https://ojs.wiserpub.com/index.php/CM/

Research Article

Improved Fire Hawk Optimizer with Crossover Scheme for Text Document Clustering

Mohammed M. Msallam[1,2], Zakir Hussain Ahmed[3*], Habibollah Bin Haron[1], Syahril Anuar Bin Idris[1], Harish Garg[4]

[1] School of Computing and Digital Technology, University Malaysia of Computer Science & Engineering (UNIMY), Petaling Jaya 46200, Malaysia
[2] Department of Computer and Information Sciences, Al-Aqsa University, Gaza 4051, Palestine
[3] Department of Mathematics and Statistics, College of Science, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh 11432, Saudi Arabia
[4] Department of Mathematics, Thapar Institute of Engineering and Technology (Deemed University), Patiala, Punjab 147004, India
E-mail: zaahmed@imamu.edu.sa

Received: 3 November 2025; **Revised:** 14 January 2026; **Accepted:** 30 January 2026

Abstract: Text Document Clustering (TDC) is an important task in document analysis, which groups unstructured text documents based on their similarities. The Fire Hawk Optimizer (FHO) has recently demonstrated strong performance in continuous optimization. However, the original FHO encounters difficulties in maintaining population diversity over time, so getting stuck in the local optimum is likely. This paper proposes an improved version of the FHO algorithm with several strategies to solve the TDC issue. It starts with a guided initialization strategy for enhancing the initial population quality. Furthermore, it integrates multiple crossover operators between the best global solution and any random individual in the population to enhance population diversity and search efficiency without a notable computational overhead. The improved FHO was tested on ten benchmark TDC datasets using four standard clustering metrics: accuracy, F-measure, entropy, and purity. In particular, the improved FHO with one-point crossover achieved average improvements of more than 12% across all metrics, with statistical testing confirming robustness and generalization. The study is one of the first successful applications of FHO to text clustering and demonstrates clear superiority over the state of the art.

***Keywords*:** fire hawk optimizer, text document clustering, crossover operator

MSC: 90C59, 68T05, 62H30

1. Introduction

The volume of online textual data has increased exponentially over the past few years and has become increasingly challenging for users to locate relevant information efficiently. Consequently, Text Document Clustering (TDC) has emerged as a necessity for structuring and grouping large document collections and making them more accessible and manageable [1]. It involves clustering of documents according to their similarities such that the documents belonging to a cluster are of high semantic similarity and the documents belonging to different clusters are of high dissimilarity [2, 3]. It is an important tool for analyzing documents to be used for purposes such as classifying different documents, finding

Copyright ©2026 Zakir Hussain Ahmed, et al.
DOI: https://doi.org/10.37256/cm.7320269070
This is an open-access article distributed under a CC BY license
(Creative Commons Attribution 4.0 International License)
https://creativecommons.org/licenses/by/4.0/

content duplication, extracting topics, recommending Web pages to users, detecting spam emails, summarizing text, and improving search results [4–7].

Typically, the clustering algorithms can fall into one of the following two classes: hierarchical and partitional. For the hierarchical clustering algorithm, a cluster tree or dendrogram can be generated through either the agglomeration of small clusters into larger ones or the division of big clusters into small ones [8]. On the other hand, partitional clustering simply partitions the documents with non-overlapping clusters and requires a specific objective function, iteratively refining the quality of the partition. Generally speaking, hierarchical methods are of higher clustering quality while computationally more expensive, whereas partitional approaches are faster, scalable, and can efficiently handle large-scale datasets [9].

The TDC is an Non-deterministic Polynomial time-hard (NP) optimization problem with traditional deterministic techniques often unable to reach globally optimum solutions [10]. Metaheuristic techniques have gained prominence in addressing such challenges because they possess the capacity to explore complex search spaces efficiently. These algorithms have also been successfully applied across diverse domains, including wireless sensor networks, feature selection, and scheduling, as well as engineering design optimization. More recently, the metaheuristics have also been used with TDC, with prominent methods ranging from Particle Swarm Optimization (PSO) [11], the Firefly Algorithm (FFA) [12], the Multi-Verse Optimizer (MVO) [13], the Jaya Optimization Algorithm (JOA) [14], Genetic Algorithm (GA) [15], and the Krill Herd Algorithm (KHA) [16]. These algorithms with local search methods can further improve their efficiency and effectiveness and yield improved TDC solutions.

Although there exist a number of metaheuristic algorithms for the TDC problem and their relatively promising performance, they still face fundamental problems of finding an appropriate exploration-exploitation balance, escaping local minima traps, and dealing with significant computational overhead [9–11, 17]. In light of these limitations and in line with the No-Free-Lunch (NFL) theorem [18], there remains a critical need to develop more effective metaheuristic solutions tailored to the TDC problem. For this reason, this research contributes to modifying Fire Hawk Optimizer (FHO) with enhanced exploration and exploitation strategies to improve the clustering outcome as well as computational efficiency in solving the TDC problem.

The FHO is among the recent nature-inspired computational methods that have been designed to mimic the natural fire hawk behaviors, especially their searching and fire spreading patterns to drive out the prey [19]. The FHO algorithm offers multiple benefits, such as rapid convergence, fewer objective function evaluations, and parameter-free, making it simple to apply with fewer settings and tunings. In addition, the flexibility of the FHO algorithm makes it easy to adapt for application in various fields. The method is quite efficient at rapidly producing the optimum (or near-optimal) solutions. Consequently, it has been employed in solving various optimization tasks like text feature selection problem [3], dynamic cluster-based routing [20], energy-efficient Internet of Things (IoT) network clustering [21], flowshop permutation scheduling problem [22], Quality of Service (QoS) provisioning in cloud computing environment [23], human activity recognition [24], early-stage diabetic retinopathy detection [25], and fake Arabic news detection [26].

This paper proposes an improved variant of the FHO specifically designed to address the TDC problem. Even though the traditional FHO has demonstrated effectiveness in continuous optimization problems, its application to the TDC problem has not been explored. Moreover, many metaheuristic algorithms suffer from limitations, such as a poor exploration-exploitation trade-off, premature convergence or stagnation to local optima, and high parameter sensitivity. To alleviate these drawbacks while enhancing computational efficiency, three improved variants of the FHO are proposed in this research paper. In all of them, a straightforward crossover method, i.e., one-point, two-point, or multi-point crossover, is utilized. These operators are used selectively between the global optimum solution and a random solution randomly chosen from the population. The suggested modifications have the intention of maximizing population diversity and making local optima avoidance easier. In addition, a centroid selection-guided initialization strategy is presented to enhance the quality of the initial solution population. The proposed algorithm is also parameter-free in nature, giving merit to its robustness and flexibility to be used across various problem domains without any need for large-scale parameter tuning. The key contributions of the current work are outlined as follows:

- FHO adaptation to address the TDC issue: The current work is among the first applications of the FHO to TDC by transforming its original continuous optimization framework to discrete search spaces efficiently through required algorithmic modifications.

- Three crossover strategies integration: Three versions of FHO using one-point, two-point, and multi-point crossover are proposed. They increase population diversity and mitigate premature convergence without incurring large computational complexity.

- Guided initialization strategy introduction: It contributes to generating a higher-quality initial population and reduces stochastic variability in the metaheuristic search process.

- Systematic empirical evaluation: The aforementioned FHO variations are extensively compared on ten TDC benchmark datasets under different measures of clustering quality. Their stability and performance are statistically validated with non-parametric tests, including the Friedman test and Holm's post hoc procedures.

Following this introduction, the second section reviews the relevant literature and previous studies in the field. The third section provides a detailed discussion of the TDC problem. The fourth section describes the basic FHO algorithm, whereas the fifth section describes the proposed clustering approach. The sixth section is dedicated to the experimental setup, which was designed to validate the efficacy of the proposed approach. The seventh section includes experimental findings and a discussion. Building upon the results discussed, the final section offers concluding remarks and identifies promising paths for future work.

2. Related work

This section provides an overview of the metaheuristic algorithms employed to tackle TDC problems. Generally, metaheuristics are partitioned into two categories: metaheuristics population-based and single-solution-based metaheuristics. The population-based algorithms generate a random solution set within an undertaken search space and update their solutions as they iterate until the best solution is found. In contrast, single-solution-based algorithms start from one solution and refine it iteratively to reach the best solution. While many metaheuristics are inherently stochastic, some methods can be purely deterministic, such as classical tabu search, which may start from a solution generated by deterministic techniques like the greedy algorithm. In both cases, problem-specific strategies can be incorporated to guide the search effectively.

In the literature, many metaheuristic algorithms have been developed to address the TDC. For instance, the MVO algorithm is based on the multiverse physics theory [27]. The authors in [28] proposed a hybrid MVO based on K-means for text clustering. A hybridization process is applied to both the initial population to generate initial clusters and the best solution derived from MVO in each generation to enhance exploitation. In the reference [13], the authors developed the link-based MVO algorithm to strengthen the local search efficiency of the original MVO algorithm in solving text clustering challenges. It incorporates a probability factor named the Neighborhood Selection Strategy (NSS), which operates in three phases: comparing solution similarity, calculating inter-solution links, and selecting the best-ranked neighboring solution. While the proposed algorithm has demonstrated superior performance over traditional clustering techniques and optimization algorithms, it also exhibits notable drawbacks. Specifically, the integration of NSS increases computational complexity, and limited discussion on parameter sensitivity, which may affect the algorithm's robustness across different datasets.

The Grey Wolf Optimizer (GWO) is a swarm intelligence algorithm that mimics grey wolves hunting [29]. The authors in [30] proposed an extension of GWO by adding a prey judgment step and combining it with a K-means algorithm for clustering text documents. Reference [31] suggested combining semantic word processing with a hybrid particle GWO to improve clustering results, and entropy-based feature selection is used to improve performance by reducing dimensions. It increases the computational complexity due to hybrid optimization and is parameter sensitive. Additionally, the evaluation was conducted on a small dataset that can affect generalizability of the results.

The PSO mimics natural swarm behavior like flocking birds, which addresses the TDC problem [32]. In [33], the authors suggested two hybrid algorithms for solving TDC: the first uses fuzzy clustering with PSO, whereas the second uses K-means and PSO. Initially, fuzzy-C-means or K-means clustering is used to generate clusters, which are then used to enhance global search options. According to experiments, the first proposed algorithm was better than the second.

However, the evaluation was conducted based on a limited number of datasets, and the impact of parameter sensitivity, particularly inertia weight, was not explored.

The Salp Swarm Algorithm (SSA) is a nature-inspired algorithm that mimics the behavior of salps [34]. A hybrid type of the SSA based on B-hill climbing is presented for solving the TDC problem. A hybridization aims to enhance the initial solutions as well as the best solution in every iteration [35], whereas the authors in [36] developed an improved version of the original SSA by using a unique probability component called neighborhood selection that balances exploration and exploitation in order to solve text clustering. Another work in [37] proposed an approach that incorporates a bare-bones strategy, inverse hyperbolic cosine control strategy, and a greedy selection to improve the performance of SSA. These improvements aim to investigate the search space effectively in solving the TDC problem. Computational results indicate that the improved SSA surpasses the conventional SSA as well as other optimization algorithms in terms of clustering quality. It is also effective across diverse datasets. However, greedy exploitation may lead to local optima as well as exhibiting sensitivity to initial population settings, which may reduce the reliability.

The JOA is one of the population-based algorithms that improves the solution using the best and worst solutions in the current population [38]. The hybrid version of JOA was used to solve text clustering by using swap mutation and one-point crossover, which are applied only to the best and worst candidate solutions. The proposed algorithm used the silhouette index as the objective function [14]. A hybrid algorithm with features from both the JOA and GWO was proposed for TDC. The objective function used is the weighted similarity measure, which includes intra-cluster and inter-cluster similarity [39]. The hybrid approach combines the strengths of both the JOA and GWO algorithms to enhance clustering performance. It shows notable effectiveness on text datasets. However, the improvement is achieved with the cost of increased complexity through hybridization and may require careful balancing of the two algorithms for best performance.

The FFA is a nature-inspired metaheuristic algorithm that works by mimicking the behavior of fireflies at night [40]. One of the primary challenges in document clustering is deciding the number of clusters to use on a specific text collection. The FFA is suggested as a dynamic text clustering method that clusters text documents automatically without any prior knowledge [41]. In this method, cosine similarity is used in the determination of high-scoring documents as centers to create clusters. Experimental results demonstrate that the suggested algorithm generates higher quality clusters than bisecting K-means and the dynamic approach. It is suitable for real-world applications because it can handle dynamically changing content. However, its performance degrades in the event of high-dimensional data. It may require multiple runs to achieve stable outcomes. In addition, its performance is parameter such as light absorption and attractiveness, as improper setting of parameters can degrade the trade-off between exploration and exploitation.

The KHA method is biologically inspired, drawing directly from the swarming and herding mechanisms of krill [42]. Two approaches are developed to resolve the TDC challenge based on the KHA. The first algorithm includes genetic mutation and crossover operators, whereas the second one does not. Another work [43] presented a hybrid version of the KHA by combining the KHA with a harmony search operator. This combination aims to improve the global searching capability of KHA as well as the quality of the resulting clusters. This hybrid algorithm outperforms several clustering approaches in terms of both cluster quality and convergence speed. However, the efficacy of the method is highly dependent on the appropriate configuration of its parameters in both the KHA and harmony search components, and no experiments were conducted to determine optimal parameter settings.

Another research [44] developed a dynamic incremental technique with cuckoo search optimization and Latent Semantic Indexing (LSI). The LSI scheme is applied to decrease high dimensionality by selecting the most relevant features. The proposed method effectively identifies the number of clusters and detects outliers, and it showed improved performance over traditional algorithms. However, the LSI used does not naturally support dynamic or incremental updates without rerunning the process, so the method faces considerable computational overhead when handling dynamic data.

The whale optimization algorithm integrated with fuzzy C-means was proposed for TDC [45]. Other authors in [46] suggested an improved sine and cosine algorithm using the mutation and crossover operators for TDC. Another study introduced chaotic northern goshawk optimization by K-means [47]. The authors in [48] used the β-hill climbing procedure for solving TDC.

Despite there are several metaheuristic-based clustering methods with some success in enhancing clustering quality, they still exhibit at least one of the following limitations: (1) The exploration-exploitation trade-off always leads to degraded

convergence and solution quality; (2) Random population initialization can influence result consistency; and (3) Parameter tuning sensitivity incurs a huge amount of parameter tuning, reducing robustness and risking overfitting. Although efforts have been made to alleviate these shortcomings, most proposed variants, such as hybridization, adaptive parameter control, and initialization schemes, are achieved at the expense of higher computational complexity. Under these constraints and in consideration of the NFL theorem, which declares that there is no algorithm superior across all problem domains, there remains an ongoing need for research effort in developing more adaptive and effective metaheuristic methodologies for the TDC problem.

The proposed algorithm addresses the above-stated three shortcomings while maintaining a low computational cost, thereby positioning itself as a practical and efficient solution for TDC. First, the algorithm is not parameter-sensitive in its nature. Unlike most prevalent metaheuristic algorithms that require fine-tuning of control parameters (e.g., learning rates, inertia weights, attractiveness, absorption coefficient of light, and probability of mutation), the proposed method works best under fixed settings. This attribute not only reduces the computational burden of parameter tuning but also makes the algorithm more robust and generalizable to other datasets and problem instances without risk of overfitting. Second, the algorithm possesses a crossover mechanism thoughtfully designed to further enhance exploration capability. Rather than performing the crossover operations randomly on the whole population, the crossover is carried out selectively between the best global solution and an arbitrarily chosen candidate solution from the population. This focused approach has two main objectives: (1) it enhances population diversity by introducing new solution structures obtained from high-quality individuals and from randomly selected individuals, and (2) it prevents the heavy computational burden by limiting the crossover activity to only a subset of pairs. This realization ensures that the algorithm benefits from the exploratory potential of crossover without compromising efficiency. Thirdly, to overcome the shortcomings of random initialization, which have a tendency to lead to poor performance diversity and worse initial points, the algorithm adopts a guided initialization strategy. This strategy is employed to generate improved initial solutions. Through the improvement of the initial population quality, the algorithm reduces its dependency on randomness and increases the chances of faster convergence to optimal or near-optimal solutions. Therefore, the proposed approach effectively balances exploration and exploitation, overcomes key limitations of other clustering approaches based on metaheuristics, and does so without adding high computational overhead.

3. Fundamentals of text document clustering

This section presents the common aspects of the text clustering challenge, including problem formulation, followed by an explanation of the solution representation, text preprocessing, and objective function.

3.1 *Problem definition*

TDC is an unsupervised learning technique for partitioning documents based on certain criteria. The underlying principle of TDC is that since the documents are unstructured, they can be represented as a vector $D = (d_1, d_2, d_3, \ldots, d_n)$ where n represents the number of documents. The goal is to join these documents into k clusters. Every cluster c_k has a centroid that represents a vector of feature weights $c_k = \left(c_{k1}, c_{k2}, c_{k3}, \ldots, c_{kf}\right)$, where c_k refers to the k^{th} centroid, c_{kf} denotes the position value in the centroid k, and f represents the number of features. The following conditions are used to decide a valid partition:

• The documents that are similar must be grouped into the same cluster, whereas the documents that are dissimilar should be grouped separately.

• Each cluster must have at least one document, i.e., $c_k \neq \varnothing$.

• Each document must belong to exactly one cluster, i.e., for any two clusters c_k and $c_{k'}$, $c_k \cap c_{k'} \neq \varnothing$ when $k \neq k'$.

3.2 *Solution representation*

The chosen solution representation directly influences the performance and success of an optimization process. In TDC, any candidate solution is an integer vector. In such a case, the solution vector is decided on the basis of the complete count of documents in the dataset. Suppose the dataset consists of ten documents and needs to be divided into three clusters. Assume that the proposed solution vector is: [2, 3, 1, 2, 2, 1, 2, 3, 1, 3]. Cluster 1 contains documents 3, 6, and 9. Documents 1, 4, 7, and 5 belong to cluster 2. Documents 2, 8, and 10 belong to cluster 3. Figure 1 illustrates the proposed solution vector.

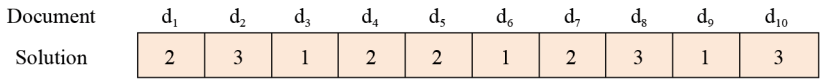


Figure 1. Text clustering solution representation

3.3 *Text pre-processing*

Pre-processing of the text steps must be performed before the actual clustering process because the documents are unstructured. Some of these are tokenization, stop word removal, stemming the words of the document, and term weighting. They are necessary to transform the dataset into lower dimensionality. Hence, clustering is a faster and improved process. The following sections briefly outline each one of these steps.

• Tokenization: It refers to the operation of dividing the text into individual tokens (terms). It facilitates text content analysis and manipulation.

• Removal of stop words: Frequently occurring words such as pronouns and prepositions (i.e., "the", "is", "it") usually don't contribute much information for clustering. Removing stop words reduces dimensionality while computationally simplifying the process.

• Stemming: It is utilized for the removal of prefixes and suffixes of words in order to change them into the root word.

• Weighting: Each feature (term) is assigned a numeric weight depending on its importance. The Term Frequency-Inverse Document Frequency (TF-IDF) is a popular algorithm that keeps track of both the term frequency within a document and within the whole dataset. The weight of a feature can be computed as (Equation (1)):

$$w_{i,\,j} = TF(i,\,j) \times \log \frac{n}{DF(j)} \tag{1}$$

where $TF(i,\,j)$ refers to the frequency of the feature j in a document i, and n represents the total number of text documents. $DF(j)$ is the number of text documents containing the feature j. Finally, each document is transformed into a Vector Space Model (VSM), where it is represented as a vector of weighted features.

3.4 *Objective function*

The proposed clustering algorithm evaluates candidate solutions using the Average Distance of Documents to the Cluster centroid (ADDC). The degree of closeness between the cluster centroids and the documents is measured by an objective function. The proposed algorithm updates the cluster centroids iteratively to minimize the ADDC. By minimizing ADDC, the algorithm ensures that documents within a cluster are more similar, which leads to more coherent clusters and improves clustering performance. The ADDC is defined as (Equation (2)):

$$\min f(x) = \frac{\sum_{i=1}^{k}\left(\frac{1}{d_i}\sum_{j=1}^{d_i} ED(c_i, d_j)\right)}{k} \tag{2}$$

where $f(x)$ denotes the fitness function, k is the number of clusters, d_i is the number of documents in cluster c_i, $ED(c_i, d_j)$ is the Euclidean distance between the document d_j and its corresponding cluster centroid c_i. The cluster centroids for each cluster are calculated by dividing their average by all the documents assigned to each cluster, as formulated in Equation (3) that follows:

$$c_{kj} = \frac{1}{d_k}\sum_{i=1}^{d_k} d_{ij} \tag{3}$$

where d_k denotes the total count of documents assigned to kth cluster j, and d_{ij} denotes the jth feature weight of document i.

4. Fire hawk optimizer

The FHO imitates the natural behaviors of hawks, which light small fires to drive prey into a trap so that the fire will spread. Firstly, the FHO process begins with initialization using Equation (4), where a number of candidate solutions (X) is defined based on the fire hawks' and their prey's location vectors.

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix} \Rightarrow x_i^j(0) = L_j + \text{rand} \cdot (U_j - L_j), \quad \begin{cases} i = 1, 2, \ldots, N \\ \\ j = 1, 2, \ldots, D \end{cases} \tag{4}$$

where the i^{th} potential solution is denoted by X_i in the search space, the total number of potential solutions is denoted by N, x_i^j refers to the j^{th} decision variable of the i^{th} potential solution, $x_i^j(0)$ denotes an initialized location for the potential solution. U_j and L_j are the upper and lower limits of the j^{th} decision parameter, respectively. rand represents a random number between 0 and 1 using uniform distribution, and d is the dimension of the problem. Afterward, using an objective function that evaluates each X, the fire hawks are identified as representing the best solutions (Equation (5)), and the prey are the other solutions (Equation (6)).

$$FH = \begin{bmatrix} FH_1 \\ FH_2 \\ \vdots \\ FH_l \\ \vdots \\ FH_F \end{bmatrix}, \quad l = 1, 2, \ldots, H \tag{5}$$

$$PR = \begin{bmatrix} PR_1 \\ PR_2 \\ \vdots \\ PR_K \\ \vdots \\ PR_P \end{bmatrix}, \quad k = 1, 2, \ldots, P \tag{6}$$

where PR_K represents the kth prey in the search space out of a total of P preys, and FH_l represents the lth fire hawk out of a total of H fire hawks in the search space. The distance between the Fire Hawks (FH) and each Prey (PR) is systematically evaluated to decide which prey is positioned closest, as indicated by the Equation (7).

$$D_k^l = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}, \quad \begin{cases} l = 1, 2, \ldots, H \\ \\ k = 1, 2, \ldots, P \end{cases} \tag{7}$$

where m and n indicate the overall population count of PR and FH, respectively, and (x_1, y_1) and (x_2, y_2) denote the location of FH and PR in the search area. Afterward, the position of each FH is updated with Equation (8).

$$FH_l^{\text{new}} = FH_l + (r_1 \times GB - r_2 \times FH_{\text{Near}}), \quad l = 1, 2, \ldots, H \tag{8}$$

where GB represents the best global solution, and FH_{Near} refers to one of the other fire hawks, r_1 and r_2 represent stochastic variables uniformly distributed within the interval [0, 1]. In the next step, the prey must determine where they can meet together in safety, and mathematically, it is expressed as follows (Equations (9)-(10)):

$$SP_l = \frac{1}{r} \sum_{q=1}^{r} PR_q, \quad \begin{cases} q = 1, 2, \ldots, R \\ \\ l = 1, 2, \ldots, H \end{cases} \tag{9}$$

$$SP = \frac{1}{m} \sum_{k=1}^{m} PR_k, \quad k = 1, 2, \ldots, P \tag{10}$$

where PR_q refers to the qth prey encircled by the $lth\ FH$, R is the number of prey within the $lth\ FH$ area, PR_k denotes the $kth\ PR$ in the search space. After that, the FH updates the prey's location within its area using Equation (11).

$$PR_q^{\text{new}} = PR_q + (r_3 \times FH_l - r_4 \times SP_l), \quad \begin{cases} l = 1, 2, \ldots, H \\ \\ q = 1, 2, \ldots, R \end{cases} \tag{11}$$

where PR_q^{new} refers to the positional update of the $qth\ PR$ surrounded by the $lth\ FH$; r_3 and r_4 denote randomly generated values constrained to the interval [0, 1]; and SP_l represents a safe region under the $lth\ FH$ area. After that, PR outside the FH area updates their positions using Equation (12).

$$PR_q^{\text{new}} = PR_q + (r_5 \times FH_{\text{Alter}} - r_6 \times SP), \quad \begin{cases} l = 1, 2, \ldots, H \\ \\ q = 1, 2, \ldots, R \end{cases} \tag{12}$$

where FH_{Alter} represents another FH agent present in the exploration domain; r_5 and r_6 denote randomly generated values constrained to the interval [0, 1]; and SP is a safe region out of the $lth\ FH$ territory.

5. Proposed clustering approach

This section offers a comprehensive description of our proposed clustering method, which is structured into two main phases. In the first phase, the standard FHO is adapted to effectively address the TDC problem. To enhance the initial population's quality, a guided initialization strategy is applied. Subsequently, the FHO is further improved by integrating multiple crossover strategies, with a theoretical justification for their inclusion provided in this section. Finally, the computational efficiency of the proposed algorithms is analyzed.

5.1 *Adapted FHO for text document clustering*

The FHO is designed to handle problems with continuous search spaces, whereas the TDC is formulated and analyzed as a discrete optimization problem. Therefore, FHO needs to be modified to address TDC by changing the algorithm itself; hence, the following equations need to be changed.

The initialization phase of FHO includes generating a set of initial solution vectors for the optimization process. For solving the TDC problem, Equation (4), which generates the solutions, is modified as follows (Equation (13)):

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix} \Rightarrow x_i^j(0) = \lfloor L_j + \text{rand} \cdot (U_j - L_j + 1) \rfloor, \quad \begin{cases} i = 1, 2, \ldots, N \\ \\ j = 1, 2, \ldots, D \end{cases} \tag{13}$$

where U_j represents a number of clusters.

The position update equation for each fire hawk, originally defined in Equation (8), is modified to ensure that the new position always remains within the valid range of clusters. Specifically, if the updated position FH_l^{new} exceeds the number of clusters (k), it is wrapped back into the range using the modulo operation. Otherwise, the position is floored to the nearest integer. This modification guarantees that all updated fire hawk positions are valid cluster indices between 1 and k (Equation (14)).

$$FH_l^{\text{new}} = \lfloor (FH_l + (r_1 \times GB - r_2 \times FH_{\text{Near}})) \,\%k \rfloor \tag{14}$$

Equations (11) and (12) use the same mechanism to update prey positions using Equations (15)-(16), ensuring they remain within the valid search space through wrapping and flooring, thus preserving optimization validity.

$$PR_q^{\text{new}} = \left\lfloor (PR_q + (r_3 \times FH_l - r_4 \times SP_l))\%k \right\rfloor \tag{15}$$

$$PR_q^{\text{new}} = \left\lfloor (PR_q + (r_5 \times FH_{\text{Alter}} - r_6 \times SP))\%k \right\rfloor \tag{16}$$

Applying these modifications is necessary to enhance the algorithm's capability to deal with TDC problems by strengthening its search dynamics and ensuring more accurate clustering results.

5.2 *A guided initialization strategy*

Most clustering algorithms that are based on metaheuristics create an initial population solution randomly. This randomness is good for variety, but it frequently introduces inefficiencies into the search. The algorithm may require many iterations to explore the solution space before converging to an optimal or near-optimal solution. This delayed convergence can negatively affect the clustering performance as well as computational efficiency, especially in high-dimensional or complex datasets.

In order to mitigate these limitations, the proposed algorithm employs a guided initialization method that is inspired by the centroid initialization of the *K*-means algorithm. Instead of generating candidate solutions totally at random, the process begins with selecting an initial set of cluster centroids at random from the dataset. Then, each document is allocated to the closest centroid according to some predefined similarity or distance metric (e.g., cosine similarity or Euclidean distance). This strategy provides an initial cluster structure that is more accurate, so it represents the natural distribution of the data.

By creating a more structured and semantically more stable initial population, this guided method enhances the quality of the starting solutions. It reduces the stochastic variability that is typically associated with metaheuristic optimization in creating the initial population. As a result, the benefits of using this strategy in the proposed algorithm are represented in faster convergence, more consistent performance across different runs, and enhanced clustering accuracy. So, it has an extremely influential role in steering the optimization process toward promising areas of the search space from the beginning.

5.3 *Improved FHO algorithm with integrated crossover strategies*

Population diversity of any optimization algorithm contributes a critical character in boosting the algorithm's performance because it prevents premature convergence or stagnation to suboptimal solutions and increases the algorithm's exploration capability by visiting several regions in the search space. To achieve this goal, three crossover operators are integrated into the proposed clustering approach to increase the diversity of candidate solutions and increase the algorithm's efficiency in discovering the best possible solutions. In the proposed algorithm, the crossover strategy is applied only between the best global solution and a single candidate solution in each population at every iteration. Since one parent represents the best solution, the probability of generating a high-quality child solution is consequently increased. Figure 2 shows the three crossover operators, and the following paragraph explains these operators:

• One-point crossover: It involves randomly selecting a crossover location on the parent solution and exchanging it between the parents at this point.

• Two-point crossover: it involves selecting two random crossover points on the parent solution, and everything between the two points is exchanged between the parents.

• Multi-point crossover: It involves dividing the parent solutions into multiple segments based on several randomly selected crossover points. Then, the segments are alternately exchanged between the parents to generate new solutions.

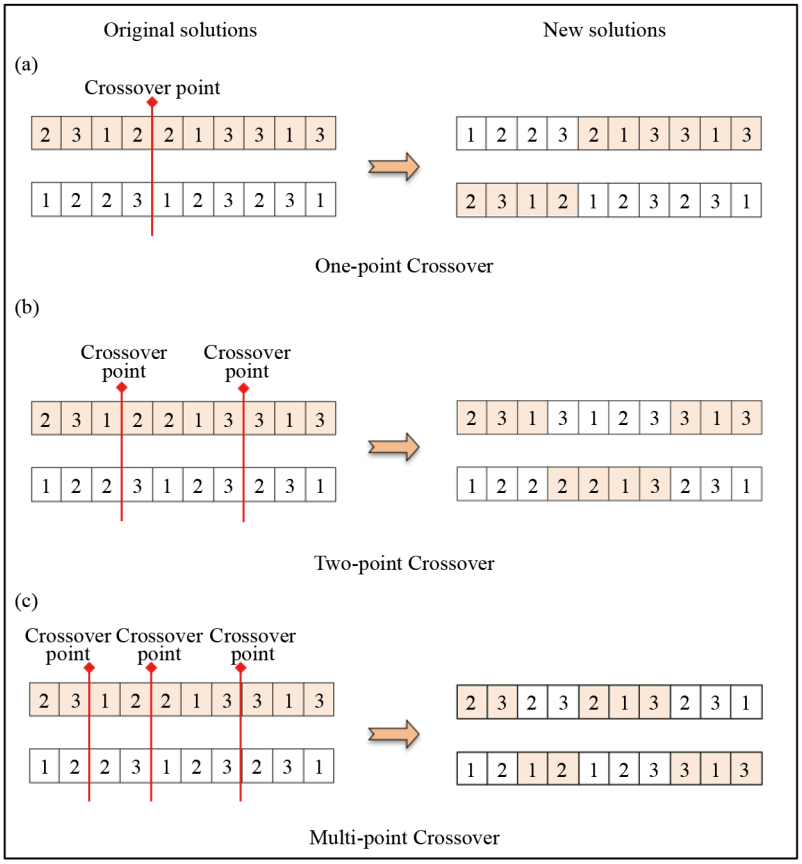


Figure 2. Crossover operators

Input: Parameter of FHO algorithm: dataset (D), class label (L), number of clusters (k), population size (N), maximum iterations (T_{max}).
Output: The best solution in all iterations.

1. Initialize the population:
   - **Select the initial k cluster centroids at random from the dataset.**
   - **Set each document to the nearest cluster centroid using the chosen distance metric.**
2. Evaluate each solution in a population using fitness.
3. Arrange all solutions in order of their fitness and determine the global optimum.
4. Repeat
5. Generate an arbitrary number (n) to decide the number of fire hawks.
6. Identify Fire Hawks (FH) and Prey (PR).
7. Determine the distance between each FH and its corresponding PR.
8. Identify the fire hawk's territory by dispersing the prey.
9. For each fire hawk in fire hawks
10. **Update the location of the fire hawk by Equation (14).**
11. For each prey in the prey
12. Compute the safe place under *fire hawk*$_i$, area by Equation (9).
13. **Update the location of prey by Equation (15).**
14. Compute the safe place out of *fire hawk*$_i$, area by Equation (10).
15. **Update the location of prey by Equation (16).**
16. End for
17. End for
18. **Apply the crossover operator.**
19. Evaluate each fire hawk and prey.
20. Identify the globally optimal solution and designate it as the primary fire hawk.
21. Until the stopping requirement is achieved
22. Return the best global solution.

Figure 3. A pseudo code of the proposed FHO steps

The detailed pseudo-code illustrating the main steps and procedures of the proposed approach is provided in Figure 3.

5.4 *Theoretical justification for the selected crossover operators in the FHO algorithm*

Incorporating one-point, two-point, and multi-point crossover operators into the FHO is intended to improve exploration capability while keeping stable convergence. Although FHO demonstrates effective searching, the inability to maintain population diversity over time can create problems in avoiding premature convergence or stagnation. The challenge is especially significant in high-dimensional and complex problems such as TDC. In these cases, the search space tends to be sparse, non-convex, or highly sensitive to the starting conditions.

To address this, the integration of these operators provides a structured strategy for creating new candidate solutions. These operators combine characteristics from the best global solution with those of an arbitrarily chosen individual. The hybrid method allows the maintenance of beneficial solution characteristics through the effect of the best global solution, while the inclusion of a randomly selected individual promotes diversity and exploration. Merging the best global solution and an arbitrary selected individual directs the search toward promising regions while preserving diversity. The algorithm can thus explore new regions of the solution space better.

The properties of these crossovers will positively contribute to improving FHO. One-point crossover is useful for making slight modifications to a solution and supports local exploration around high-quality regions. Two-point crossovers introduce a moderate increase in diversity by swapping larger sections of a solution. This process maintains many important characteristics of the original candidates. At the same time, it encourages exploration of alternative clustering structures. Multi-point crossover causes the most disruption and supports more global exploration. This is important for FHO to avoid local optima and to explore a wider range of new solutions.

Selected crossover operators provide a more structured form of variation in the search process. It typically preserves significant segments of the best global solution in the new solution. This allows useful components of the best-known structure to remain intact. In contrast, alternative diversity-enhancing techniques such as Lévy flights or chaotic maps introduce higher levels of randomness and large changes in the solution space [49]. Despite being useful in evading local optima, they also tend to affect and remove useful structures within well-formed parts. Hence, crossover is useful to provide a balance of exploration and exploitation. It introduces diversity into the population without completely disrupting the valuable components of high-quality solutions. Crossover is efficient in preserving the integrity of good solutions with an ongoing search for better ones.

5.5 *Complexity analysis of proposed algorithms*

The importance of algorithm complexity is its implication on performance, which is a function of the number of steps needed to reach the best solution for any given problem. It is usually measured in terms of Big O notation, which considers the worst-case scenario with a considerable problem size.

The complexity of the basic FHO algorithm includes the number of iterations (T), the number of fire hawks and prey (N), the number of features (F), and the dimension of the problem being addressed (D). Controlling these parameters is essential to maintaining the complexity of an optimization algorithm. In addition, the complexity of each algorithm depends on the number of operations conducted for solution updating and the respective cost of the fitness function. Based on Equation (17), we can estimate the FHO algorithm's basic computational complexity:

$$O(FHO) = O(\text{initialization phase}) + O(\text{population update}) + O(\text{fitness evaluation}) \tag{17}$$

The population initialization complexity (also fitness evaluation computation) is $O(N \times K \times D \times F)$, where K indicates the total count of clusters. The time complexity of the population update is the number of operations required to update the solution, which is $O(T \times N \times D \times 3)$. It must be noted that the constant value (3) is generally smaller than the iterations count, the problem size, and the population size. Therefore, the constant can be dropped from the time complexity calculation. Thus, according to the operations of FHO, the total complexity of FHO is estimated as given in Equation (18).

$$O(FHO) \cong O(N \times K \times D \times F) + O(T \times N \times D) + O(T \times N \times F \times D) \tag{18}$$

The overall computational time required by the Improved FHO (IFHO) that incorporates the one-point crossover mechanism can be represented in terms of its time complexity as follows (Equations (19)-(20)):

$$O(IFHOcr1) \cong O(\text{initialization phase}) + O(\text{population update}) + O(\text{fitness evaluation}) + O(\text{crossover}) \tag{19}$$

$$O(IFHOcr1) \cong O(N \times K \times D \times F) + O(T \times N \times D) + O(T \times N \times F \times D) + O(T \times D) \tag{20}$$

In the improved version, the introduction of crossover operations adds a marginal overhead. The operations are applied to a small number of individuals: the best global and one randomly selected candidate at each iteration. Therefore, the additional complexity is approximately $O(T \times D)$ in the worst case, which equals only a linear improvement over the prevailing $O(T \times N \times F \times D)$ term of fitness evaluation.

6. Experimental setup

This section provides a detailed description of the standard benchmark text datasets used in the experiments, along with the parameter configurations applied to all metaheuristic algorithms being compared. In addition, the evaluation measures adopted to assess clustering performance are also introduced.

6.1 *Text clustering datasets*

For the evaluation of the proposed clustering approach's effectiveness and its generality, ten well-known benchmark text datasets from publicly available repositories were selected [50]. The datasets were selected according to their occurrence in TDC literature as well as based on their diversity in structure, complexity, and topic. Specifically, the datasets represent a variety of text sources based on real-world including news articles, academic abstracts, social media, and web pages. This diversity allows us to evaluate the proposed approach across a range of real-world scenarios with varying linguistic characteristics and topic distributions.

The datasets exhibit considerable variation in the number of documents they contain, ranging from 299 to 2,000, features from 1,725 to 8,460, and clusters from just 3 up to 20. This wide range is perfect for testing how well a clustering algorithm can scale, stay stable, and adapt. Table 1 presents their essential properties. The large diversity of dataset characteristics enables us to test algorithmic behavior with varying levels of sparsity, dimensionality, and granularity of clusters.

The datasets employed in this study were obtained from two publicly accessible online sources. The first nine datasets were retrieved from a text collections repository [51], and the last dataset was obtained from a Kaggle dataset [52]. The preprocessing applied to these datasets included tokenization, stopword removal, and stemming using Porter's algorithm. In this study, the TF-IDF weighting scheme was subsequently applied to enhance feature representation.

Table 1. Characteristics of the experimental text datasets

ID	Dataset	Number of documents	Number of clusters	Number of features
DS1	Syskill Webert	334	4	4,339
DS2	Dmoz-Computers	2,000	4	2,855
DS3	Tr12	313	8	5,805
DS4	Wap	1,560	20	8,460
DS5	Tr41	878	10	6,743
DS6	Re8	1,459	6	5,255
DS7	Reuters-21578	1,014	12	5,484
DS8	Computer Science Technical Reports (CSTR)	299	4	1,725
DS9	Tr21	336	6	7,902
DS10	20 Newsgroups	300	3	5,797

6.2 *Parameter configuration*

For the purpose of a fair and consistent comparison, our proposed procedure was compared to seven widely used algorithms that have been utilized in the domain of the TDC domain, including K-means, PSO, MVO, GWO, JOA, SSA, and FFA. All the algorithms were implemented using the Python programming language and executed on a standard computing environment consisting of an HP laptop with Windows 10 Pro operating system installed, having an Intel Core i5 processor with a clock speed of 1.00 GHz and 16 GB Random Access Memory (RAM). Each algorithm's parameters for configuration are listed in Table 2. Parameter values were selected based on previous research recommendations and benchmark settings utilized in the metaheuristic optimization literature to reflect generally accepted and established values. In instances where more than one configuration was reported in the literature, the most frequent or best-performing ones were utilized.

Table 2. The parameter configuration for the optimization methods

Method	Parameter	Value
Optimization algorithms	Size of the population	25
	Maximum iterations	1,000
	The total number of executions	20
PSO	Inertia weight (maximum)	0.9
	Inertia weight (minimum)	0.2
	C_1 (Personal learning coefficient)	2
	C_2 (Global learning coefficient)	2
MVO	Wormhole Existence Probability (WEP) (maximum)	1
	WEP (minimum)	0.2
GWO	a (controlling parameter)	$a \in [2, 0]$
FFA	α_0	0.5
	γ	1
	β_0	1

The configuration parameters were appropriately selected to achieve a balanced trade-off between computational cost and solution quality. A population size 25 was set to provide enough diversity for the candidate solutions without excessive computation required for larger populations. The iteration number was set to 1,000, providing sufficient time for the algorithms to completely search the search space and converge toward high-quality solutions without extending the runtime unnecessarily once convergence behavior was observed. To address the randomized nature of metaheuristic algorithms, each experiment was executed 20 times independently, ensuring statistical reliability and reducing the effect of random fluctuations. These parameter settings align well with established standards and recommendations found in the metaheuristic optimization literature.

6.3 *Evaluation metrics*

This paper identifies the primary extrinsic metrics commonly used in the literature to assess the effectiveness of the text clustering techniques, namely F-measure, accuracy, entropy, and purity. The following provides a concise description of each metric below.

6.3.1 *Accuracy*

Accuracy is obtained by taking the proportion of correctly classified documents relative to the total documents. The formula given in Equation (21) can be applied to compute the accuracy.

$$\text{Accuracy} = \frac{1}{n}\sum_{j=1}^{k} n_{i,\,j} \tag{21}$$

n is the overall total of text documents, k denotes the number of clusters, and $n_{i,\,j}$ shows the number of accurate documents in cluster j that belong to class i.

6.3.2 *F-measure*

The F-measure is obtained by calculating the harmonic mean between precision and recall values. It gives a broader perspective regarding the performance of an algorithm. The F-measure can be achieved by the Equation (22).

$$F(C_j) = \frac{2 \times P(i,\,j) \times R(i,\,j)}{P(i,\,j) + R(i,\,j)} \tag{22}$$

where $F(C_j)$ is the F-measure value for cluster j, $R(i,\,j)$ and $P(i,\,j)$ represent the recall and precision metrics, respectively, for class i within cluster j (Equation (23)–(24)).

$$P(i,\,j) = \frac{n_{i,\,j}}{n_j} \tag{23}$$

$$R(i,\,j) = \frac{n_{i,\,j}}{n_i} \tag{24}$$

where $n_{i,\,j}$ represents the count of correct documents located within cluster j associated with class i. n_j is the total count of documents in cluster j, whereas n_i denotes the total number of documents classified under class i. The F-measures for all classes are weighted by the maximum F-measure for each class as described in Equation (25).

$$F(C) = \sum_{i=1}^{m} \frac{n_i}{n} \max_{j=1} \{F(i, j)\} \tag{25}$$

where n and m are total number of documents and total number of classes, respectively, and the term n_i indicates the total number of documents that belong to a specific class i.

6.3.3 *Purity measure*

The metric of purity is defined as the average of the highest number of documents within each cluster that are associated with a specific class. The formula given in Equation (26) be employed to calculate the purity.

$$P = \frac{1}{n} \sum_{j=1}^{k} \max_{i} \left\{ n_{i, j} \right\} \tag{26}$$

where n and k are the number of documents and the total count of clusters, respectively. The notation $\max_i \{n_{i, j}\}$ is utilized to indicate the highest number of documents associated with class i that are identified within cluster j.

6.3.4 *Entropy measure*

The entropy metric is essential to understanding the nature of a cluster. It essentially quantifies the level of disorder present. When a cluster has low entropy, it means the documents inside it are very similar, resulting in a coherent and structured grouping. One can calculate the entropy for a cluster j using Equation (27):

$$E(C_j) = -\sum_{i} p_{i, j} \log p_{i, j} \tag{27}$$

where $p_{i, j}$ denotes the probabilistic value associated with the documents within cluster j that are classified under class i. The entropy values of the clusters are determined using Equation (28).

$$E = \sum_{j=1}^{k} \frac{n_j}{n} E(C_j) \tag{28}$$

where k, n, and n_j are the total number of clusters, the total document count in the dataset and the count of documents within the cluster j respectively.

7. Results and discussions

The experiments are divided into three main parts to evaluate the efficacy of the proposed algorithms. The first part involves a comparison of the adapted FHO with several metaheuristic optimization techniques for evaluating baseline competitiveness. The second section involves a comprehensive comparison between the different proposed variations of the FHO algorithm to examine the impact of improvement on clustering performance. Finally, the top-performing FHO variant is compared with recently developed clustering algorithms reported in the literature.

7.1 *Evaluation of the adapted FHO against well-established text algorithms*

This section presents a comprehensive comparison between the adapted FHO algorithm and seven established text clustering algorithms: *K*-means, PSO, MVO, JOA, GWO, SSA, and FFA. All methods are evaluated in terms of *F*-measure, accuracy, entropy, and purity. For a fair comparison, the parameter settings and population initialization strategy of these comparative algorithms are identical to the proposed approach. Each algorithm is executed on the same datasets, and the results are obtained as the mean of 20 independent executions per dataset. Tables 3 and 4 present the results according to the evaluation conditions for all text clustering algorithms across ten text datasets, with the best-performing results underlined in boldface.

Table 3. Performance metrics results for five datasets (DS1–DS5)

Datasets	Measure	Methods							
		K-means	PSO	SSA	JOA	GWO	MVO	FFA	FHO
DS1	Accuracy	0.4136	0.5117	0.4763	0.4752	0.4608	0.4769	0.5138	**0.5563**
	F-measure	0.4374	0.5117	0.4864	0.5048	0.4756	0.4932	0.5199	**0.5459**
	Purity	0.4219	0.5141	0.4810	0.4922	0.4672	0.4835	0.5234	**0.5588**
	Entropy	0.5624	0.5024	0.5200	0.5037	0.5229	0.5125	0.4828	0.4714
	Rank	8	3	6	4	7	5	2	1
DS2	Accuracy	0.3626	0.3000	0.3013	0.3226	0.327	0.3066	0.3597	**0.3987**
	F-measure	**0.4539**	0.3934	0.4058	0.4102	0.4192	0.4118	0.4272	0.4307
	Purity	0.3646	0.303	0.3017	0.3241	0.3286	0.3075	0.3628	**0.4019**
	Entropy	**0.5370**	0.5834	0.5867	0.5815	0.5736	0.5800	0.5596	0.5546
	Rank	**1**	7	7	5	4	5	3	**1**
DS3	Accuracy	0.3780	0.4099	0.3850	0.3112	0.3866	0.3984	0.4179	**0.4268**
	F-measure	0.4052	0.4022	0.3785	0.3514	0.3860	0.4047	0.4154	**0.4261**
	Purity	0.3930	0.4246	0.4099	0.3879	0.4214	0.4329	0.4444	**0.4585**
	Entropy	**0.6384**	0.6831	0.7011	0.7044	0.6874	0.6591	0.6540	0.6522
	Rank	5	4	7	8	6	3	2	**1**
DS4	Accuracy	0.4970	0.5963	0.6129	0.6314	0.6189	0.6434	**0.6776**	0.6696
	F-measure	0.5039	0.6337	0.6389	0.6558	0.6386	0.6700	0.6919	**0.6938**
	Purity	0.5354	0.6998	0.7012	0.7026	0.6886	0.7151	0.7406	**0.7552**
	Entropy	0.5232	0.3067	0.3100	0.3066	0.3132	0.2981	**0.2604**	0.2766
	Rank	8	6	5	4	6	3	**1**	1
DS5	Accuracy	0.4433	0.4256	0.3628	0.3468	0.4115	0.4023	0.4620	**0.4691**
	F-measure	0.4466	0.4087	0.3572	0.3436	0.3961	0.3973	**0.4528**	0.4504
	Purity	0.4580	0.4596	0.3966	0.3975	0.4516	0.4294	0.4907	**0.4946**
	Entropy	0.6153	0.6765	0.7196	0.7384	0.6866	0.6871	0.6441	0.6482
	Rank	3	4	7	8	5	6	**1**	**1**
Average rank		5	4.8	6.4	5.8	5.6	4.4	1.8	**1**
Final rank		5	4	8	7	6	3	2	**1**

Table 4. Performance metrics results for five datasets (DS6–DS10)

Datasets	Measure	Methods							
		K-means	PSO	SSA	JOA	GWO	MVO	FFA	FHO
DS6	Accuracy	0.3074	0.3321	0.3247	0.3234	0.3486	0.2903	0.3332	**0.3840**
	F-measure	0.3554	0.3720	0.3598	0.3567	0.3865	0.3323	0.3543	**0.3904**
	Purity	0.3262	0.3351	0.3292	0.3308	0.3540	0.2934	0.3473	**0.3976**
	Entropy	0.6655	0.6732	0.6816	0.6849	0.6648	0.6969	0.6686	**0.6522**
	Rank	6	3	5	6	2	8	4	**1**
DS7	Accuracy	0.3623	0.3715	0.3598	0.3492	0.3545	0.3641	0.4022	**0.4054**
	F-measure	0.3521	0.3467	0.3528	0.3537	0.3381	0.3453	0.3891	**0.3952**
	Purity	0.3886	0.3821	0.3831	0.3940	0.3831	0.3790	**0.4283**	0.4243
	Entropy	0.7590	0.7835	0.7668	0.7621	0.7749	0.7800	0.7298	**0.7289**
	Rank	3	6	5	4	7	7	2	**1**
DS8	Accuracy	0.4472	0.4637	0.4540	0.4498	0.4550	0.4488	0.487	**0.5001**
	F-measure	0.4803	0.4826	0.4743	0.4819	0.4764	0.4840	0.4835	**0.4873**
	Purity	0.4580	0.4826	0.4577	0.4535	0.4589	0.4518	0.4938	**0.5058**
	Entropy	0.5088	0.5099	0.5171	0.5160	0.5155	0.5140	0.4985	**0.4981**
	Rank	5	3	7	5	7	4	2	**1**
DS9	Accuracy	0.6360	0.6455	0.6333	0.5214	0.6274	0.6274	0.7015	**0.7076**
	F-measure	0.5933	0.6193	0.5969	0.5304	0.6215	0.5931	**0.6481**	0.6356
	Purity	0.6949	0.7179	0.7021	0.6935	**0.7318**	0.7018	0.7268	0.7135
	Entropy	0.4337	0.4061	0.4247	0.4229	**0.3856**	0.4255	0.4025	0.4208
	Rank	6	4	5	8	2	7	**1**	2
DS10	Accuracy	0.3452	0.388	0.3868	0.3768	0.3778	0.3858	0.4135	**0.4757**
	F-measure	0.4783	0.4981	0.4877	0.4970	0.4869	0.4837	0.4927	**0.5031**
	Purity	0.3462	0.3882	0.388	0.3775	0.3783	0.3863	0.4153	**0.4857**
	Entropy	0.4712	0.4619	0.4627	0.4645	0.4655	0.4658	0.4529	**0.4422**
	Rank	8	3	4	5	6	6	2	**1**
Average rank		5.6	3.8	5.2	5.6	4.8	6.4	2.2	**1.2**
Final rank		6	3	5	6	4	8	2	**1**

The rankings in these tables were computed using a Friedman-based multi-criteria ranking procedure. For each dataset and metric, methods were ranked. Higher values indicate better performance for all metrics except entropy, for which lower values are better. The ranks across the four metrics were averaged to give one rank per method for each dataset. Finally, the average of these ranks across all datasets determined the final ranking, with lower values indicating better overall performance.

The results clearly demonstrate that FHO consistently outperforms the competing algorithms across nearly all datasets. This superiority is reflected in its higher accuracy and purity, lower entropy, and superior *F*-measure scores, indicating the overall quality and reliability of its clustering performance. These findings validate the efficiency and robustness of the FHO algorithm in addressing diverse text clustering challenges.

In terms of accuracy, FHO achieved the highest values in all datasets except DS4. The FFA algorithm recorded the second-best accuracy, followed by PSO and GWO. Specifically, FHO exhibited accuracy gains ranging from 4% to 14% on DS1 and from 3.5% to 9.4% on DS6, confirming its reliable performance. Conversely, FFA outperformed FHO only in DS4, where FHO ranked second. Overall, these results highlight FHO's consistent dominance in clustering accuracy.

Regarding purity, FHO also shows notable superiority across most datasets. For instance, in DS1, it achieved improvements of 13.7%, 9.2%, 7.8%, 7.5%, 6.7%, 4.5%, and 3.5% compared to *K*-means, GWO, SSA, MVO, JOA, PSO, and FFA, respectively. This demonstrates that FHO not only improves cluster assignment accuracy but also enhances cluster homogeneity.

For the *F*-measure, FHO attained the highest scores among all algorithms, reflecting its balanced performance between precision and recall. It ranked second in DS2, DS5, and DS9, suggesting that its performance may slightly decline in datasets with structural complexities, such as class imbalance or uneven document distributions. Nonetheless, its overall strong performance highlights the need for robust metaheuristic algorithms capable of maintaining stability under diverse data characteristics.

Finally, considering entropy, which measures the quality and consistency of clustering results, FHO produced the lowest values among all compared algorithms, followed by FFA, *K*-means, PSO, GWO, MVO, JOA, and SSA. This further validates the stability and precision of FHO in generating well-defined and homogeneous clusters.

FHO naturally exhibits good overall performance due to its learning process, where the worst solutions (prey) iteratively improve by following the guidance of higher-quality solutions (fire hawks). This process introduces a form of selective pressure that drives the population toward higher-quality solutions over successive iterations. Moreover, unlike many algorithms that rely on parameter tuning, FHO features a straightforward update process that does not include fine-tuned control parameters. This straightforward nature makes the algorithm stable and reliable, particularly when working with the high-dimensional and sparse data typically encountered in text clustering.

Looking at Tables 3 and 4, the algorithms are evaluated and ranked based on metrics that include accuracy, entropy, purity, and *F*-measure. As per the results, the proposed FHO algorithm had the topmost overall ranking, proving its good clustering capability across various datasets. This is followed by FFA, PSO, and GWO, which also ranked well with competitive performance. As a contrast, *K*-means and MVO achieved moderate rankings with lower performance but still acceptable results. Finally, JOA and SSA obtained the lowest ranks, suggesting that their clustering was less efficient on average. It is clear that this ranking remains steady across all ten datasets used in the study.

The values of the observed metrics are comparatively lower due to the complexity of the datasets with higher complexity and high dimensionality. This increased complexity makes it much more difficult to detect an optimal solution to the clustering problem. In particular, the basic FHO algorithm does not do well in searching the entire search space because it lacks strong mechanisms to maintain diversity among the solutions. This leads to its being trapped in local optima, failing to identify the best global clustering arrangement for these complex data sets.

7.1.1 *Convergence analysis*

The clustering convergence rate is also a significant factor for the performance evaluation in finding the best possible solution. Figures 4 and 5 depict the convergence characteristics of the proposed FHO algorithm in comparison with other existing algorithms across ten text datasets. The *x*-axis is used to represent the number of iterations. The *y*-axis is used to represent the average fitness values presented by ADDC. All the curves in the plots represent the performance trajectory of a specific algorithm while optimizing the fitness function. The algorithm with the lowest fitness value is considered the best one. All the competing methods adopt a unique strategy to search through the search space of the TDC problem under the control of its particular exploration and exploitation components. From Figures 4 and 5, it is clear that the proposed FHO algorithm converges uniformly to improved solutions with more iterations. Considering individual curves closely, it is evident that FHO has the minimum fitness values in most of the datasets. This finding validates the algorithm's strong convergence and optimization effectiveness.

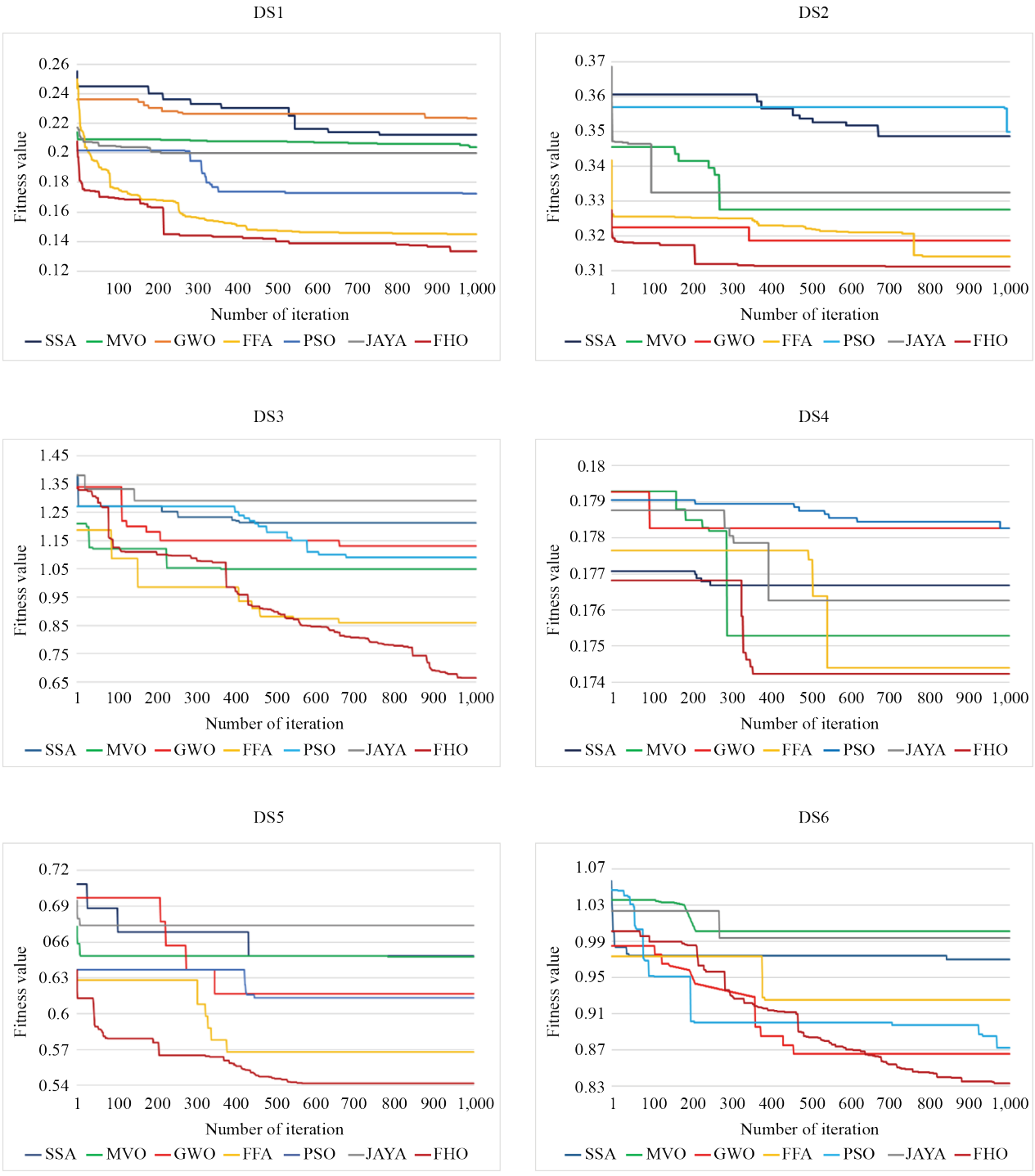


Figure 4. Comparison of the convergence behavior between the FHO algorithm and other optimization algorithms across six datasets

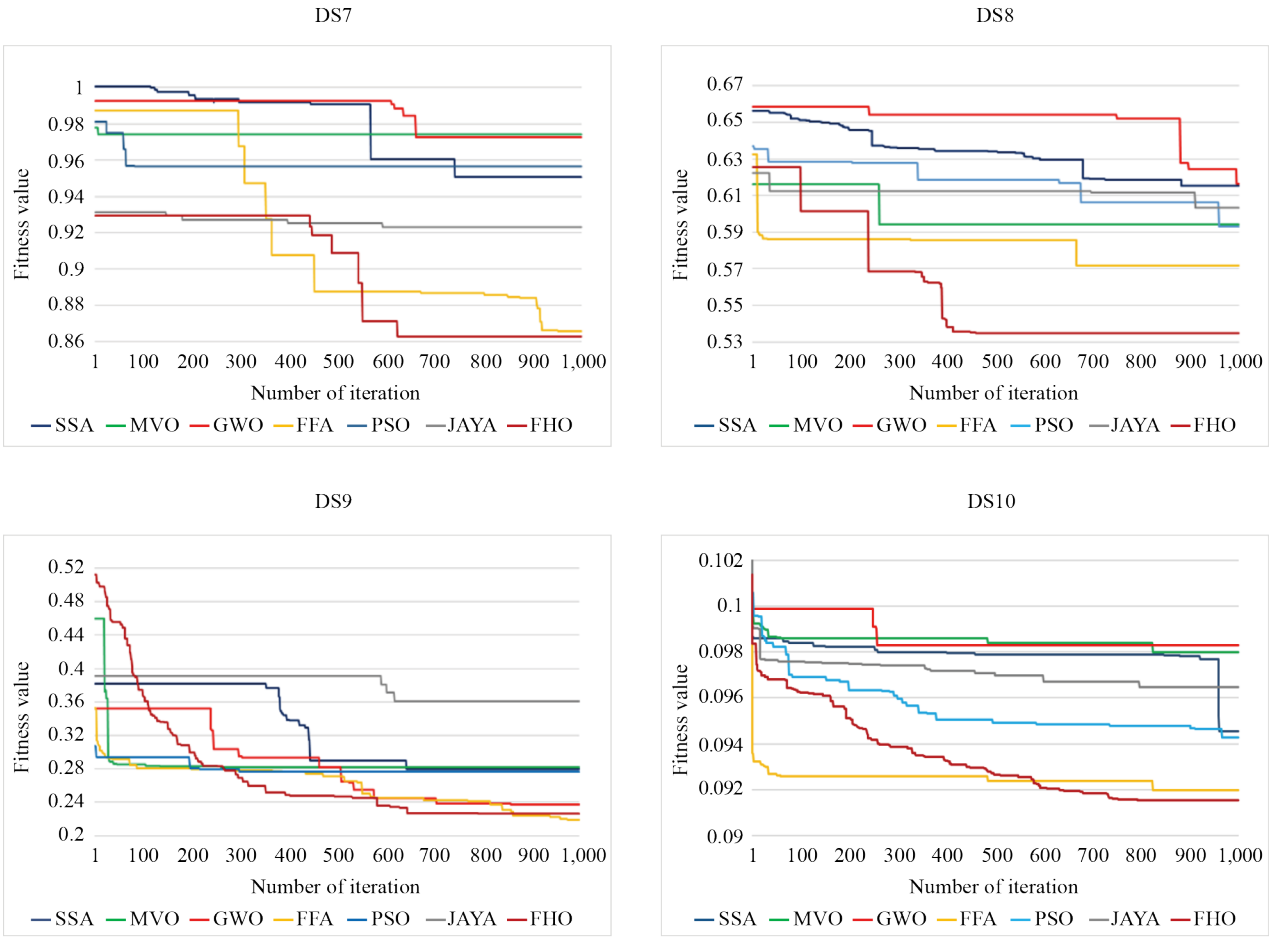


Figure 5. Comparison of the convergence behavior between the FHO algorithm and other optimization algorithms across four datasets (DS7–DS10)

7.1.2 *Statistical significance analysis*

To verify the statistical superiority of FHO against other competitors' algorithms, the Friedman non-parametric test was applied. There is no substantial difference among the algorithms under the null hypothesis, but there is a substantial difference under the alternative hypothesis. The ultimate choice relies on the computed value of ρ because it determines whether the null hypothesis should be rejected or not. If the value of ρ drops below 0.05, the null hypothesis is rejected in favor of the alternative hypothesis. The statistical analysis was performed using the ranks obtained from all performance metrics. The statistical results of the Friedman test and the value of ρ are presented in Table 5.

Table 5 reports the mean rank and standard deviation obtained from the Friedman-based analysis. The mean rank reflects the overall performance of each algorithm, while the standard deviation indicates the stability of this performance across datasets, where lower values imply more consistent behavior. As shown in the table, FHO achieves the best performance with the lowest mean rank and standard deviation, followed by FFA, confirming their superiority and reliability compared to the other methods.

Test results indicated that the algorithms differ considerably; thus, the null hypothesis was rejected because the ρ value (0.00003) was less than the presumed significance level of 0.05. For a verification of the statistically significant differences between FHO and others, the Holm Step-Down Procedure was applied using a significance threshold of $\alpha = 0.05$.

Table 5. Friedman-based statistical comparison of the algorithms

Algorithm	Mean rank	Standard deviation	Ranking
K-means	5.3	2.41	5
PSO	4.3	1.49	3
GWO	5.2	1.93	4
MVO	5.4	1.71	6
SSA	5.8	1.14	8
JOA	5.7	1.7	7
FFA	2	0.94	2
FHO	1.2	0.32	1
ρ value	0.000003		

Note: The best method is the lowest-ranked algorithm

This post-test was conducted after the Friedman test, which first pointed out considerable differences in the algorithms' performance. For this analysis, the method of control employed was FHO due to its top-ranked performance. The Standard Error (SE) of the differences of mean ranks was calculated and found to be approximately 1.095. Based on this value, the *z*-scores are calculated, which are the standard deviations between FHO and all the other algorithms. The *p*-values were calculated from the *z*-scores for all pair-wise comparisons. These *p*-values indicate whether the observed performance differences are statistically significant or not. A detailed summary of the results is presented in Table 6.

Table 6. Results of the Holm step-down statistical test

Algorithm	Mean rank	*Z*-value	*p*-value	Threshold	Significant
SSA	5.8	4.20	0.00003	0.0071	Yes
JOA	5.7	4.11	0.00004	0.0083	Yes
GWO	5.4	3.83	0.0001	0.0100	Yes
K-means	5.2	3.65	0.0003	0.0125	Yes
MVO	5.0	3.47	0.0005	0.0167	Yes
PSO	4.3	2.83	0.0046	0.025	Yes
FFA	2.0	0.73	0.46	0.05	No

The Holm's step-down post-hoc test demonstrated that FHO performed significantly better than six popular algorithms, SSA, JOA, MVO, GWO, *K*-means, and PSO, but not FFA. This result highlights the robustness and consistent superiority of FHO across the benchmark datasets. The statistical significance of these findings is supported by the *p*-values obtained from the test, which represent the probability that the observed performance differences occurred by chance. In all six significant comparisons, the *p*-values were well below the Holm-adjusted thresholds, confirming that the improvement achieved by FHO is not due to random variation but reflects a statistically significant enhancement in clustering effectiveness.

7.2 *Comparative analysis between proposed FHO variants*

In this section, the proposed enhancements to the FHO algorithm are analyzed and compared. Three improved variants of FHO are presented as follows: (I) FHO with a one-point crossover operator (IFHOcr1); (II) FHO with a two-point crossover operator (IFHOcr2); and (III) FHO with a multi-point crossover operator (IFHOcrn). The performance of all

the suggested algorithms on ten text collections is provided in Tables 7 and 8, in terms of accuracy, F-measure, entropy, and purity. 20 independent runs are calculated for each dataset to assess the performance of the proposed algorithm. The experimental comparison indicates the superiority of the IFHOcr1 over the majority of datasets and clustering metrics. Especially, it achieved the best accuracy on seven out of the ten datasets, namely DS1, DS3, DS5, DS7, DS8, DS9, and DS10. This is proof of the extremely good generalizability and strong robustness of the algorithm to various text clustering scenarios. For instance, in DS1, it achieved a significant accuracy improvement of between 4.2% and 19.6% relative to other methods. Similarly, on DS3, IFHOcr1 achieved improvements of between 3% and 16.8%. This enhancement is for the effectiveness of the one-point crossover in guiding the search towards high-quality clusters without any loss of diversity.

Table 7. Comparative performance results based on 20 independent executions for five datasets (DS1–DS5)

Datasets	Measure	Methods			
		FHO	IFHOcr1	IFHOcr2	IFHOcrn
DS1	Accuracy	0.5563	**0.7525**	0.7102	0.6608
	F-measure	0.5459	**0.7340**	0.7104	0.6572
	Purity	0.5588	**0.7527**	0.7115	0.6617
	Entropy	0.4714	**0.3216**	0.3543	0.3995
	Rank	4	**1**	2	3
DS2	Accuracy	0.3987	0.4605	**0.5150**	0.4350
	F-measure	0.4307	0.5008	**0.5066**	0.4603
	Purity	0.4019	0.4651	**0.4857**	0.4381
	Entropy	0.5546	0.4927	**0.4767**	0.5332
	Rank	4	2	**1**	3
DS3	Accuracy	0.4268	**0.5946**	0.5642	0.5118
	F-measure	0.4261	**0.5854**	0.5479	0.5036
	Purity	0.4585	**0.6058**	0.5700	0.5275
	Entropy	0.6522	**0.4963**	0.5513	0.5863
	Rank	4	**1**	2	3
DS4	Accuracy	0.6696	0.6808	**0.6941**	0.6514
	F-measure	0.6919	0.6998	**0.7113**	0.6754
	Purity	0.7552	0.7524	**0.7527**	0.7135
	Entropy	0.2766	**0.2664**	0.2684	0.2857
	Rank	3	2	**1**	4
DS5	Accuracy	0.4691	**0.6579**	0.5781	0.4861
	F-measure	0.4504	**0.6484**	0.5742	0.4712
	Purity	0.4946	**0.6802**	0.6081	0.5173
	Entropy	0.6482	**0.4521**	0.5285	0.6293
	Rank	4	**1**	2	3
Average rank		3.8	**1.4**	1.6	3.2
Final rank		4	**1**	2	3

Table 8. Comparative performance results based on 20 independent executions for five datasets (DS6–DS10)

Datasets	Measure	Methods			
		FHO	IFHOcr1	IFHOcr2	IFHOcrn
DS6	Accuracy	0.3840	0.5417	**0.5679**	0.5086
	F-measure	0.3904	0.5279	**0.5666**	0.4777
	Purity	0.3976	0.5543	**0.5842**	0.5306
	Entropy	0.6522	0.5382	**0.5020**	0.5446
	Rank	4	2	**1**	3
DS7	Accuracy	0.4054	**0.4830**	0.4401	0.4140
	F-measure	0.3952	**0.4722**	0.4332	0.4001
	Purity	0.4243	**0.5082**	0.4622	0.4390
	Entropy	0.7289	**0.6372**	0.6837	0.7000
	Rank	4	**1**	2	3
DS8	Accuracy	0.5001	**0.6460**	0.6290	0.5273
	F-measure	0.4873	**0.6429**	0.6194	0.5287
	Purity	0.5058	**0.6530**	0.6348	0.5360
	Entropy	0.4981	**0.3857**	0.4141	0.4784
	Rank	4	**1**	2	3
DS9	Accuracy	0.7076	**0.8396**	0.8235	0.7497
	F-measure	0.6356	**0.8320**	0.8213	0.7517
	Purity	0.7135	**0.8521**	0.8360	0.7732
	Entropy	0.4208	**0.2216**	0.2375	0.2984
	Rank	4	**1**	2	3
DS10	Accuracy	0.4757	**0.6515**	0.5898	0.6067
	F-measure	0.5031	**0.6797**	0.6353	0.6192
	Purity	0.4857	**0.6528**	0.5908	0.6073
	Entropy	0.4422	**0.3185**	0.3621	0.3875
	Rank	4	**1**	3	2
Average rank		4	**1.2**	2	2.8
Final rank		4	**1**	2	3

On the other hand, IFHOcr2 handled DS2, DS4, and DS6 best. This indicates that two-point crossover might work more effectively in specific scenarios where structural balance and moderate diversity contribute to improved clustering performance. This result implies that while IFHOcr1 ensures good general performance, it is not uniformly the best. Document sparsity or dimensionality in datasets can influence the relative performance of different crossover strategies.

According to purity and *F*-measure results, IFHOcr1 ranked first in seven out of ten datasets, followed by IFHOcr2 and IFHOcrn. Moreover, the entropy levels of IFHOcr1 were the lowest for eight datasets. Notably, FHO provided the highest entropy in all cases compared with other proposed algorithms, which reflects the worst cluster separability and stability.

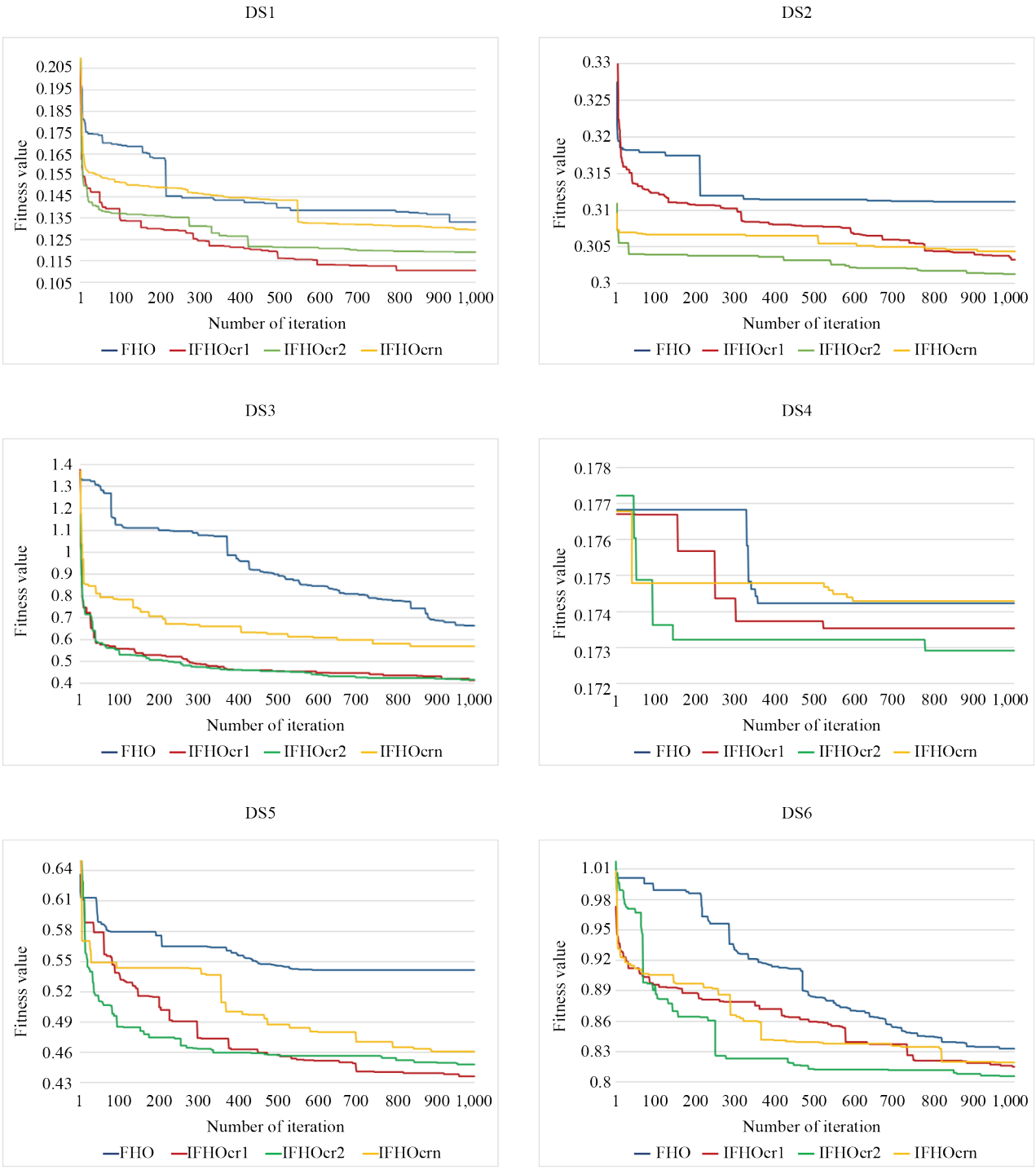


Figure 6. Comparison of the convergence behavior among the proposed FHO approaches across six datasets (DS1–DS6)

From an algorithmic perspective, IFHOcr1's performance advantage comes from the careful integration of the one-point crossover operator, which provides a more favorable balance between preserving building blocks and introducing useful diversity. The operator strategically merges segments from the best global solution with those from a randomly chosen candidate. This strategy can improve the quality of solutions without sacrificing stability, and progress can be maintained as a result. In contrast to this, IFHOcr2's two-point crossover operator produces more variation than that of

IFHOcr1's single-point technique. However, more variation in some situations can increase diversity and be beneficial for the overall search. On the other hand, more randomness may also disrupt useful solution structures, which may lead to slow convergence and unstable results.

In the case of IFHOcrn, the multi-point crossover introduces extremely high randomness into the search process, and this can disrupt the optimization process and cause early stagnation. This crossover behavior variation is the reason why IFHOcr1 has better stability and clustering consistency because it retained the best average fitness values and faster convergence on different datasets.

In summary, when the performance of the proposed IFHOcr1 algorithm is compared with the original FHO, it shows a clear and measurable advantage. On average, IFHOcr1 delivers about a 13.1% higher accuracy, achieves roughly a 13.7% gain in the F-measure, improves the purity of clusters by around 12.8%, and lowers the entropy by approximately 12.1%, all of which highlight its stronger and more reliable clustering capability.

Figures 6 and 7 illustrate the average convergence behaviors of the proposed FHO variants employing different crossover operators across all datasets. The convergence curves of the algorithms reveal noticeable differences in both optimization progress and population diversity. Among the proposed algorithms, IFHOcr1 demonstrates the best overall performance. It converges at the fastest rate and achieves the lowest final fitness value, indicating superior solution quality. The algorithm exhibits consistent and stable improvement throughout the optimization process, suggesting a well-balanced trade-off between exploration and exploitation, and effectively reaching a near-global optimum.

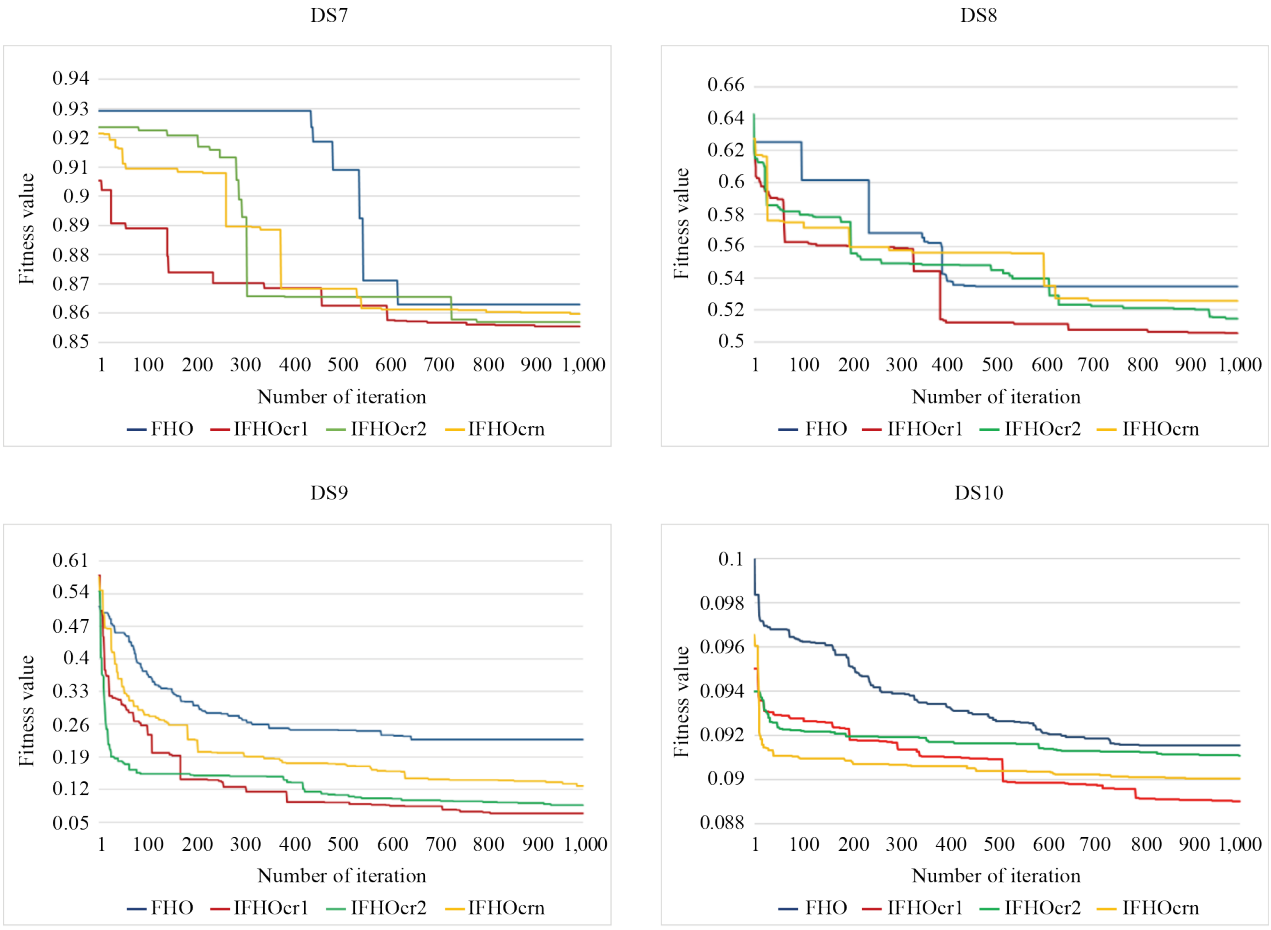


Figure 7. Comparison of the convergence behavior among the proposed FHO approaches across four datasets (DS7–DS10)

IFHOcr2 shows a moderate yet stable convergence behavior. It preserves sufficient population diversity throughout the search process, enabling continuous improvement, although its convergence rate is slightly slower than IFHOcr1. In contrast, IFHOcrn displays a different trend: it converges rapidly in the early iterations, but its progress slows down in later stages, implying a reduction in search diversity that restricts further enhancement of solution quality. The original FHO algorithm demonstrates the weakest convergence performance, showing minimal improvement over iterations and yielding relatively higher final fitness values.

Table 9. Friedman-based statistical comparison of the algorithms

Algorithm	Mean rank	Ranking
FHO	3.9	4
IFHOcr1	1.3	1
IFHOcr2	1.8	2
IFHOcrn	3	3
ρ value	0.000017	

Note: The best one is the lowest-ranked algorithm

Statistical analysis supports these observations. The Friedman test, as shown in Table 9, revealed a significant difference between the algorithms ($p = 0.000017 < 0.05$). The post-hoc comparison using Holm's Step-Down Procedure, summarized in Table 10, showed that IFHOcr1 significantly outperformed FHO ($p = 0.0000$) and IFHOcrn ($p = 0.0032$), while the difference between IFHOcr1 and IFHOcr2 was not statistically significant ($p = 0.3865$). These results provide strong statistical evidence that IFHOcr1 consistently achieves superior performance, confirming its stability and high optimization capacity.

Table 10. Results of the Holm step-down statistical test

Algorithm	Mean rank	Z-value	p-value	Threshold	Significant
FHO	3.9	4.5033	0.000	0.0167	Yes
IFHOcrn	3.0	2.9445	0.0032	0.0250	Yes
IFHOcr2	1.8	0.8660	03865	0.0500	No

7.3 *Comparison with state-of-the-art published algorithms*

In order to demonstrate the efficacy and strength of the proposed improved FHO with crossover scheme (IFHOcr1), a comparative study has been conducted with four state-of-the-art methods from literature, i.e., Link-Based Multi-Verse Optimizer (LBMVO), Hybrid Enhanced Fruit-Fly Firefly (HEFFF), Bare Bones-Based Salp Swarm Algorithm (BBSSA), and Sin Cosine Algorithm-Genetic Algorithm (SCA-GA), as presented in Table 11. The comparison is conducted on five datasets (DS3, DS4, DS5, DS8, and DS10). These datasets are established in the research community and form a good basis for objective comparison of performance. It must be noted that results on just five of these datasets were presented in the original experiments of the methods being compared. The remaining datasets (DS1, DS2, DS6, DS7, and DS9) were not considered in these works found in the literature; therefore, no comparative results exist on them. The summarized results across the five datasets are presented in Table 12.

Table 11. Summary of compared algorithms with descriptions, publication years, and references

Key	Algorithm	Year	Reference
IFHOcr1	Improved Fire Hawk Optimizer with one-point crossover	This study	-
BBSSA	Bare Bones-Based Salp Swarm Algorithm	2023	[37]
SCA-GA	Sin Cosine Algorithm-Genetic Algorithm	2023	[46]
HEFFF	Hybrid Enhanced Fruit-Fly Firefly	2021	[53]
LBMVO	Link-Based Multi-Verse Optimizer	2020	[13]

Table 12. Performance comparison of algorithms across multiple datasets and evaluation metrics

Dataset	Measure	Algorithms				
		LBMVO	HEFFF	BBSSA	SCA-GA	IFHOcr1
DS3	Accuracy	0.4901	0.6398	0.6359	**0.6497**	0.5946
	F-measure	0.5100	0.6290	**0.6739**	0.6392	0.5854
	Purity	0.5833	0.5744	0.4637	0.5471	**0.6058**
	Entropy	0.4918	0.6965	0.6051	**0.3549**	0.4963
	Rank	5	3	3	1	2
DS4	Accuracy	0.5592	0.6398	0.5307	0.6598	**0.6808**
	F-measure	0.5111	0.629	0.5523	0.6485	**0.6998**
	Purity	0.6359	0.5744	0.4935	0.6985	**0.7524**
	Entropy	0.6260	0.6965	0.5834	0.4029	**0.2664**
	Rank	4	3	4	2	1
DS5	Accuracy	0.5034	0.6487	0.5749	0.6538	**0.6579**
	F-measure	0.5007	0.638	0.6052	0.6319	**0.6484**
	Purity	0.6493	0.5791	0.6050	0.6715	**0.6802**
	Entropy	0.4951	0.313	0.5598	**0.3005**	0.4521
	Rank	4	3	4	2	1
DS8	Accuracy	0.4747	0.5964	0.6429	0.5988	**0.6460**
	F-measure	0.5435	0.5755	**0.7654**	0.5991	0.6429
	Purity	0.5823	0.6140	0.5783	0.6395	**0.6530**
	Entropy	0.4763	0.3361	0.5254	**0.3122**	0.3857
	Rank	4	4	2	2	1
DS10	Accuracy	0.4405	0.5833	0.6295	0.5996	**0.6515**
	F-measure	0.4378	0.5750	**0.7372**	0.5620	0.6797
	Purity	0.4707	0.5100	0.6522	0.5098	**0.6528**
	Entropy	0.6843	0.3267	0.4427	**0.3118**	0.3185
	Rank	5	4	2	3	1
Average rank		4.4	3.4	3	2	1.2
Final rank		5	4	3	2	1

From Table 12, IFHOcr1 performs better on the majority of the benchmark datasets than the other algorithms. Notably, it is the most accurate in four out of the five datasets. For instance, in DS4, IFHOcr1 achieves accuracy gains of 2.1% to 15% over the other algorithms. The situation is similar in DS10, where the gains are 2.2% to 21.1%.

It achieves the highest F-measure in all the algorithms experimented on DS4 and DS5. Even though it ranks second best for DS8 and DS10, the performance remains highly competitive. This stability is due to crossover being able to preserve the best gene segments as well as sufficiently search in the solution space, a characteristic evident in IFHOcr1's overall good performances for all the tested datasets.

IFHOcr1 forms more uniform clusters compared to the other algorithms and achieves the highest purity scores on all datasets being analyzed. For example, it achieves between 5.4% and 25.9% purity gain on DS4 compared to the competitor algorithms. Likewise, on DS3, IFHOcr1 achieves between 2.3% and 14.2% purity gain compared to the other algorithms, indicating its stable and uniform performance on different datasets. Among the comparison algorithms, SCA-GA ranked second with its purity scores, followed by LBMVO. Conversely, HEFFF and BBSSA consistently recorded the lowest purity values on all the datasets.

For entropy, SCA-GA recorded the lowest for all datasets except DS4. IFHOcr1acheived the second-lowest entropy values, followed by BBSSA and HEFFF. In contrast, LBMVO recorded the highest entropy levels across all datasets, which indicates lower clustering quality compared to the other methods.

In terms of overall performance, IFHOcr1 is superior compared to other state-of-the-art solutions to the text clustering problem. It scores the highest for accuracy, F-measure, and purity, which indicate its effectiveness in creating well-clustered groups. The accuracy indicates that most of the documents are correctly mapped to their respective clusters, and the good F-measure indicates an appropriate trade-off between recall and precision. The high purity score confirms that clusters are mostly comprised of documents from a single actual category, resulting in clear and consistent results.

The improved performance is largely due to the novel structure of the proposed IFHOcr1 algorithm, where one-point crossover operator is used between the best global solution and any randomly chosen individual from the population. This strategic operator has three crucial advantages. It initially maintains genetic diversity at low computational expense. Second, it retains the high-quality components of solutions in reproduction. Third, it prevents premature convergence.

8. Conclusion and future directions

In this paper, an improved version of the FHO is proposed for addressing the TDC problem. Three crossover operations, one-point, two-point, and multi-point, are included in the suggested enhancement. They are applied selectively between a randomly selected member of the candidate population and the best global solution. The effective search space exploration is made possible by the selective approach, which also guarantees the exploitation of promising solutions. This enhances solution diversity and avoids premature convergence. Besides, a centroid-based guided initialization technique is introduced to achieve a high-quality and diverse initial population.

The performance of the proposed variants was thoroughly evaluated on ten benchmark TDC datasets against four popular clustering metrics: accuracy, F-measure, entropy, and purity. Experimental findings show that the improved FHO variants achieve superior performance compared with state-of-the-art metaheuristic algorithms in most datasets. Specifically, the one-point crossover variant achieved the best performance. It attained the average improvement of 13.1% in accuracy, 13.7% in F-measure, 12.8% in purity, and 12.1% in entropy over the FHO. Besides, the proposed algorithms exhibited better convergence rates and solution stability. All these characteristics make them candidate solutions for stable and effective document clustering.

There are several promising avenues for future research. First, using semantic-based representation schemes, such as Word2Vec, GloVe, or contextual embeddings (e.g., Bidirectional Encoder Representations from Transformers (BERT)), may allow the algorithm to capture deeper semantic relationships within text data. Second, developing automatic methods for determining the optimal number of clusters could reduce the reliance on prior knowledge and improve adaptability. Third, applying the proposed algorithm to real-world text clustering problems can demonstrate its practical applicability and generalization. In addition, its scalability and computational efficiency on large-scale data should be evaluated for real-time

and big data scenarios. Lastly, the integration of deep learning approaches, particularly transformer-based architectures, with metaheuristic optimization can be further explored to enhance clustering performance in high-dimensional text data.

Funding

This work was supported and funded by the Deanship of Scientific Research at Imam Mohammad Ibn Saud Islamic University (IMSIU) (grant number IMSIU-DDRSP-RP25).

Conflict of interest

The authors declare no competing financial interest.

References

[1] Selvaraj S, Choi E. Swarm intelligence algorithms in text document clustering with various benchmarks. *Sensors*. 2021; 21(9): 3196. Available from: https://doi.org/10.3390/s21093196.

[2] Abualigah L, Gandomi AH, Elaziz MA, Al Hamad H, Omari M, Alshinwan M, et al. Advances in meta-heuristic optimization algorithms in big data text clustering. *Electronics*. 2021; 10(2): 101. Available from: https://doi.org/10.3390/electronics10020101.

[3] Msallam MM, Bin Idris S. Unsupervised text feature selection by binary fire hawk optimizer for text clustering. *Cluster Computing*. 2024; 27: 7721–7740. Available from: https://doi.org/10.1007/s10586-024-04364-z.

[4] Verma P, Verma A, Pal S. An approach for extractive text summarization using fuzzy evolutionary and clustering algorithms. *Applied Soft Computing*. 2022; 120: 108670. Available from: https://doi.org/10.1016/j.asoc.2022.108670.

[5] Alami N, Meknassi M, En-nahnahi N, El Adlouni Y, Ammor O. Unsupervised neural networks for automatic Arabic text summarization using document clustering and topic modeling. *Expert Systems with Applications*. 2021; 172: 114652. Available from: https://doi.org/10.1016/j.eswa.2021.114652.

[6] Shringi S, Sharma H. Detection of spam reviews using hybrid grey wolf optimizer clustering method. *Multimedia Tools and Applications*. 2022; 81(27): 38623–38641. Available from: https://doi.org/10.1007/s11042-022-12848-6.

[7] Abasi AK, Khader AT, Al-Betar MA, Naim S, Makhadmeh SN, Alyasseri ZAA. A novel ensemble statistical topic extraction method for scientific publications based on optimization clustering. *Multimedia Tools and Applications*. 2021; 80: 37–82. Available from: https://doi.org/10.1007/s11042-020-09504-2.

[8] Karna A. Automatic identification of the number of clusters in hierarchical clustering. *Neural Computing and Applications*. 2021; 34(1): 119–134. Available from: https://doi.org/10.1007/s00521-021-05873-3.

[9] Kumar Y, Kaur A. Variants of bat algorithm for solving partitional clustering problems. *Engineering with Computers*. 2021; 38: 1973–1999. Available from: https://doi.org/10.1007/s00366-021-01345-3.

[10] Hassan IH, Mohammed A, Ali YS, Jeremiah I, Abdulraheem SA. Metaheuristic algorithms in text clustering. In: *Advances in Metaheuristic Optimization Algorithms*. Academic Press; 2023. p.131–152. Available from: https://doi.org/10.1016/B978-0-323-91781-0.00007-7.

[11] Janani R, Vijayarani S. Text document clustering using spectral clustering algorithm with particle swarm optimization. *Expert Systems with Applications*. 2019; 134: 192–200. Available from: https://doi.org/10.1016/j.eswa.2019.05.030.

[12] Xie H, Zhang L, Peng C, Yu Y, Liu C, Liu H, et al. Improving *K*-means clustering with enhanced firefly algorithms. *Applied Soft Computing Journal*. 2019; 84: 105763. Available from: https://doi.org/10.1016/j.asoc.2019.105763.

[13] Abasi AK, Khader AT, Al-Betar MA, Naim S, Makhadmeh SN, Alyasseri ZAA. Link-based multi-verse optimizer for text documents clustering. *Applied Soft Computing Journal*. 2020; 87: 106002. Available from: https://doi.org/10.1016/j.asoc.2019.106002.

[14] Thirumoorthy K, Muneeswaran K. A hybrid approach for text document clustering using Jaya optimization algorithm. *Expert Systems with Applications*. 2021; 178: 115040. Available from: https://doi.org/10.1016/j.eswa.2021.115040.

[15] Mustafi D, Mustafi A, Sahoo G. A novel approach to text clustering using genetic algorithm based on the nearest neighbour heuristic. *International Journal of Computer Applications in Technology*. 2022; 44(3): 291–303. Available from: https://doi.org/10.1080/1206212X.2020.1735035.

[16] Abualigah LM, Khader AT, Hanandeh ES. A combination of objective functions and hybrid krill herd algorithm for text document clustering analysis. *Engineering Applications of Artificial Intelligence*. 2018; 73: 111–125. Available from: https://doi.org/10.1016/j.engappai.2018.05.003.

[17] Abualigah L, Gandomi AH, Elaziz MA, Hussien AG, Khasawneh AM, Alshinwan M, et al. Nature-inspired optimization algorithms for text document clustering—a comprehensive analysis. *Algorithms*. 2020; 13(12): 345. Available from: https://doi.org/10.3390/a13120345.

[18] Joyce T, Herrmann JM. A review of no free lunch theorems, and their implications for metaheuristic optimisation. In: *Nature-Inspired Algorithms and Applied Optimization*. Springer; 2017. p.27–51. Available from: https://doi.org/10.1007/978-3-319-67669-2_2.

[19] Azizi M, Talatahari S, Gandomi AH. Fire hawk optimizer: A novel metaheuristic algorithm. *Artificial Intelligence Review*. 2023; 56: 287–363. Available from: https://doi.org/10.1007/s10462-022-10173-w.

[20] Hosseinzadeh M, Yoo J, Ali S, Lansky J, Mildeova S, Yousefpoor MS, et al. A cluster-based trusted routing method using fire hawk optimizer (FHO) in wireless sensor networks (WSNs). *Scientific Reports*. 2023; 13: 13046. Available from: https://doi.org/10.1038/s41598-023-40273-8.

[21] Gupta IK, Mishra AK, Diwan TD, Srivastava S. Unequal clustering scheme for hotspot mitigation in IoT-enabled wireless sensor networks based on fire hawk optimization. *Computers and Electrical Engineering*. 2023; 107: 108615. Available from: https://doi.org/10.1016/j.compeleceng.2023.108615.

[22] Baihaqi MA, Utama DM. No-wait flowshop permutation scheduling problem: Fire hawk optimizer vs beluga whale optimization algorithm. *Jurnal Ilmiah Teknik Industri*. 2023; 22(1): 124–136. Available from: https://doi.org/10.23917/jiti.v22i1.21128.

[23] Gupta M, Singh D, Gupta B. Modified fire hawks gazelle optimization (MFHGO) algorithm based optimized approach to improve the QoS provisioning in cloud computing environment. *International Journal of Communication Networks and Applications*. 2023; 10(3): 383–400. Available from: https://doi.org/10.22247/ijcna/2023/221896.

[24] Alonazi M, Alnfiai MM. Fire hawk optimizer with deep learning enabled human activity recognition. *Computer Systems Science and Engineering*. 2023; 45(3): 3135-3150. Available from: https://doi.org/10.32604/csse.2023.034124.

[25] Huang C, Sarabi M, Ragab AE. MobileNet-V2/IFHO model for accurate detection of early-stage diabetic retinopathy. *Heliyon*. 2024; 10(17): e37293. Available from: https://doi.org/10.1016/j.heliyon.2024.e37293.

[26] Elaziz MA, Dahou A, Orabi DA, Alshathri S, Soliman EM, Ewees AA. A hybrid multitask learning framework with a fire hawk optimizer for Arabic fake news detection. *Mathematics*. 2023; 11(2): 258. Available from: https://doi.org/10.3390/math11020258.

[27] Mirjalili S, Mirjalili SM, Hatamlou A. Multi-verse optimizer: A nature-inspired algorithm for global optimization. *Neural Computing and Applications*. 2016; 27: 495–513. Available from: https://doi.org/10.1007/s00521-015-1870-7.

[28] Abasi AK, Khader AT, Al-Betar MA, Naim S, Alyasseri ZAA, Makhadmeh SN. A novel hybrid multi-verse optimizer with *K*-means for text documents clustering. *Neural Computing and Applications*. 2020; 32(23): 17703–17729. Available from: https://doi.org/10.1007/s00521-020-04945-0.

[29] Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. *Advances in Engineering Software*. 2014; 69: 46–61. Available from: https://doi.org/10.1016/j.advengsoft.2013.12.007.

[30] Wang J, Pan C, Shi J. *K*-means text clustering method based on decision grey wolf optimization. *ACM Transactions on Asian and Low-Resource Language Information Processing*. 2024. Available from: https://doi.org/10.1145/3689210.

[31] Vidyadhari C, Sandhya N, Premchand P. Particle grey wolf optimizer (PGWO) algorithm and semantic word processing for automatic text clustering. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*. 2019; 27(2): 201–223. Available from: https://doi.org/10.1142/S0218488519500090.

[32] Kennedy J, Eberhart R. Particle swarm optimization. In: *Proceedings of ICNN'95-International Conference on Neural Networks*. Perth, WA, Australia: IEEE; 1995. p.1942–1948. Available from: https://doi.org/10.1109/ICNN.1995.488968.

[33] Karol S, Mangat V. Evaluation of text document clustering approach based on particle swarm optimization. *Central European Journal of Computer Science*. 2013; 3(2): 69–90. Available from: https://doi.org/10.2478/s13537-013-0104-2.

[34] Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM. Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software*. 2017; 114: 163–191. Available from: https://doi.org/10.1016/j.advengsoft.2017.07.002.

[35] Abasi AK, Khader AT, Al-Betar MA, Alyasseri ZAA, Makhadmeh SN, Al-Laham M, et al. A hybrid salp swarm algorithm with β-hill climbing algorithm for text documents clustering. In: *Evolutionary Data Clustering: Algorithms and Applications*. Singapore: Springer; 2021. p.129–161. Available from: https://doi.org/10.1007/978-981-33-4191-3_6.

[36] Abuain WA. Improved salp swarm algorithm for text document clustering. *Journal of Theoretical and Applied Information Technology*. 2024; 102(15): 5396–5407.

[37] Al-Betar MA, Abasi AK, Al-Naymat G, Arshad K, Makhadmeh SN. Bare-bones based salp swarm algorithm for text document clustering. *IEEE Access*. 2023; 11: 100010–100028. Available from: https://doi.org/10.1109/ACCESS.2023.3314589.

[38] Rao RV. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *International Journal of Industrial Engineering Computations*. 2016; 7: 19–34. Available from: https://doi.org/10.5267/j.ijiec.2015.8.004.

[39] Venkanna G, Bharati DKF. Optimal text document clustering enabled by weighed similarity oriented Jaya with grey wolf optimization algorithm. *The Computer Journal*. 2021; 64(6): 960–972. Available from: https://doi.org/10.1093/comjnl/bxab013.

[40] Yang XS. Firefly algorithms for multimodal optimization. In: *Stochastic Algorithms: Foundations and Applications*. Berlin, Heidelber: Springer; 2009. p.169–178. Available from: https://doi.org/10.1007/978-3-642-04944-6_14.

[41] Mohammed AJ, Yusof Y, Husni H. Determining number of clusters using firefly algorithm. In: *Advances in Visual Informatics*. Cham: Springer; 2015. p.14–24. Available from: https://doi.org/10.1007/978-3-319-25939-0_2.

[42] Gandomi AH, Alavi AH. Krill herd: A new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation*. 2012; 17(12): 4831–4845. Available from: https://doi.org/10.1016/j.cnsns.2012.05.010.

[43] Abualigah LM, Khader AT, Hanandeh ES, Gandomi AH. A novel hybridization strategy for krill herd algorithm applied to clustering techniques. *Applied Soft Computing*. 2017; 60: 423–435. Available from: https://doi.org/10.1016/j.asoc.2017.06.059.

[44] Boushaki SI, Kamel N, Bendjeghaba O. High-dimensional text datasets clustering algorithm based on cuckoo search and latent semantic indexing. *Journal of Information and Knowledge Management*. 2018; 17(3): 1850033. Available from: https://doi.org/10.1142/S0219649218500338.

[45] Gopal J, Singh AK. Text clustering algorithm using fuzzy whale optimization algorithm. *International Journal of Intelligent Engineering and Systems*. 2019; 12(2): 278–286. Available from: https://doi.org/10.22266/ijies2019.0430.27.

[46] Radomirović B, Jovanović V, Nikolić B, Stojanović S, Venkatachalam K, Zivkovic M, et al. Text document clustering approach by improved sine cosine algorithm. *Information Technology and Control*. 2023; 52(2): 541–561. Available from: https://doi.org/10.5755/j01.itc.52.2.33536.

[47] Dodda R, Babu AS. Text document clustering using chaotic northern goshawk optimization with *K*-means algorithm. *International Journal of Intelligent Engineering and Systems*. 2024; 17(3): 720–731. Available from: https://doi.org/10.22266/ijies2024.0630.56

[48] Abualigah LM, Sawaie AM, Khader AT, Rashaideh H, Al-Betar MA, Shehab M. β-hill climbing technique for the text document clustering. In: *Proceedings of the New Trends in Information Technology (NTIT-2017)*. Amman, Jordan: The University of Jordan; 2017. p.60-66.

[49] He Q, Liu H, Ding G, Tu L. A modified Lévy flight distribution for solving high-dimensional numerical optimization problems. *Mathematics and Computers in Simulation*. 2023; 204: 376–400. Available from: https://doi.org/10.1016/j.matcom.2022.08.017.

[50] Rossi RG, Marcacini RM, Rezende SO. *Benchmarking Text Collections for Classification and Clustering Tasks*. University of Sao Paulo; 2013.

[51] University of Sao Paulo. *LABIC-text collections*. Available from: http://sites.labic.icmc.usp.br/text_collections/ [Accessed 23rd March 2025].

[52] Abbasi A. *20Newsgroups_300_articles*. Kaggle. 2018. Available from: https://www.kaggle.com/ammarabbasi/20newsgroups-300-articles [Accessed 23rd March 2025].

[53] Bezdan T, Stoean C, Al Naamany A, Bacanin N, Rashid TA, Zivkovic M, et al. Hybrid fruit-fly optimization algorithm with *K*-means for text document clustering. *Mathematics*. 2021; 9(16): 1929. Available from: https://doi.org/10.3390/math9161929.