9-12-2023

Bare-Bones Based Salp Swarm Algorithm for Text Document Clustering

Mohammed Azmi Al-Betar
Ajman University

Ammar Kamal Abasi
Mohamed Bin Zayed University of Artificial Intelligence

Ghazi Al-Naymat
Ajman University

Kamran Arshad
Ajman University

Sharif Naser Makhadmeh
Ajman University

Recommended Citation

M. A. Al-Betar, A. K. Abasi, G. Al-Naymat, K. Arshad and S. N. Makhadmeh, "Bare-Bones Based Salp Swarm Algorithm for Text Document Clustering," in *IEEE Access,* vol. 11, pp. 100010-100028, 2023, doi: 10.1109/ACCESS.2023.3314589

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 25 July 2023, accepted 5 September 2023, date of publication 12 September 2023, date of current version 19 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3314589

RESEARCH ARTICLE

Bare-Bones Based Salp Swarm Algorithm for Text Document Clustering

MOHAMMED AZMI AL-BETAR[1,2,3], (Member, IEEE), AMMAR KAMAL ABASI[4], GHAZI AL-NAYMAT[1,2], KAMRAN ARSHAD[1,2], (Senior Member, IEEE), AND SHARIF NASER MAKHADMEH[2,5]

[1]Department of Information Technology, College of Engineering and Information Technology, Ajman University, Ajman, United Arab Emirates
[2]Artificial Intelligence Research Center (AIRC), Ajman University, Ajman, United Arab Emirates
[3]Department of Information Technology, Al-Huson University College, Al-Balqa Applied University, Irbid 19117, Jordan
[4]Machine Learning Department, Mohamed Bin Zayed University of Artificial Intelligence (MBZUAI), Abu Dhabi, United Arab Emirates
[5]Department of Data Science and Artificial Intelligence, University of Petra, Amman 11196, Jordan

Corresponding author: Mohammed Azmi Al-Betar (m.albetar@ajman.ac.ae)

This work was supported by the Deanship of Graduate Studies and Research (DGSR), Ajman University, Ajman, United Arab Emirates, under Grant 2021-IRG-ENIT-6.

ABSTRACT Text Document Clustering (TDC) is a challenging optimization problem in unsupervised machine learning and text mining. The Salp Swarm Algorithm (SSA) has been found to be effective in solving complex optimization problems. However, the SSA's exploitation phase requires improvement to solve the TDC problem effectively. In this paper, we propose a new approach, known as the Bare-Bones Salp Swarm Algorithm (BBSSA), which leverages Gaussian search equations, inverse hyperbolic cosine control strategies, and greedy selection techniques to create new individuals and guide the population towards solving the TDC problem. We evaluated the performance of the BBSSA on six benchmark datasets from the text clustering domain and six scientific papers datasets extracted from the top eight UAE universities. The experimental results demonstrate that the BBSSA algorithm outperforms traditional SSA and nine other optimization algorithms. Furthermore, the BBSSA algorithm achieves better results than the five traditional clustering techniques.

INDEX TERMS Global optimization, salp swarm algorithm, bare bones, greedy selection strategy, text document clustering.

I. INTRODUCTION

Every day, a massive quantity of digital data is stored and collected via the internet from smart devices in a world where the web is widely available and accessible to millions of users through ever-increasing mobile software. Text analysis is essential in text mining, which involves innovative ways and methods for processing massive volumes of text documents with a profusion of e-data [1]. Text clustering (TC) is a proficient approach to categorizing unknown text documents into a subset of comparable and associated clusters in text mining, machine learning, and pattern recognition. For textual data, web pages, such as learning sites, virtual libraries, business sites, and smart applications like instructional and banking, are the significant cause for the enormous quantities of text documents posted online [2].

When dealing with large amounts of text data, an unsupervised machine learning technique known as text document clustering (TDC) can be employed to divide a set of text documents into consonant clusters [3], [4]. This strategy makes it easier for users to manage text documents by grouping related papers based on critical content. This approach is used to deal with a collection of unlabeled textual documents in which the class label of the document is unknown [5], [6]. This approach is widely used to support various systems, including information retrieval, search engines, text categorization and classification, and image segmentation.

The Vector Space Model (VSM) is utilized in the TDC to evaluate the similarity of documents, represented as vectors with cluster centroids for text documents. VSM is a

The associate editor coordinating the review of this manuscript and approving it for publication was Utku Kose.

commonly used concept for TDC in which all text documents are characterized as a vector of word weights using a simple weighting model [7].

TDC are represented in a multidimensional space, with each dimension's location value linked to its weight value. Thousands of text features can be used to specify the features created by unique text terms. The technique of TC strategies, on the other hand, does not include feature selection. Furthermore, the dimensionality reduction provides for the presence of obscure and huge text features and unclear features and high-dimensional information in TC, both of which are noisy, distributed unevenly, irrelevant, and redundant in features [8]. Typically, feature selection provides the TC's best and most relevant features. Hundreds of text features can be used to define the text features created by many text terms.

The techniques utilized to achieve effective clustering are k-means, an efficient, robust and resilient local search model to clarify the TDC problem. The evolution of metaheuristic optimization algorithms assists in solving several problems associated with different fields like TDC. Unfortunately, early convergence or premature are challenges for these algorithms [9], [10]. Recently, new and modified metaheuristic algorithms are introduced, such as Krill Herd Algorithm (KHA) [11], Moth-flame optimization (MFO) [12], Grey wolf optimizer (GWO) [13], Multi-Verse Optimize (MVO) [14], Genetic Algorithm (GA) [15], Particle Swarm Optimization (PSO) [16], Harmony Search (HS) [17], Bat algorithm (BAT) [18], Firefly algorithm (FFA) [19], Lemurs optimizer (LO) [20]. These methods were used to solve a large number of optimization problems [21], [22], for example, feature selection [8], [23], [24], EEG signals denoising [25], [26], [27], and scheduling problems in smart home [28], [29], [30], [31].

The work in [32] presented a multi-strategy co-evolutionary differential evolution algorithm (MSCoDE) tailored for mixed-variable optimization problems involving continuous and discrete decision variables. This algorithm utilizes a mixed-variable co-evolutionary approach and dynamically adapts mutation strategies and crossover operators based on the problem. Furthermore, a statistics-based local search is employed to enhance discrete variable optimization. Comparative analysis of benchmark functions and engineering design problems demonstrates the superior effectiveness and efficiency of MSCoDE compared to existing algorithms such as AEDA, CoDE, and MOSDE. Another work in [33] introduced a multi-strategy firefly algorithm with the selective ensemble (MSEFA) to address the challenge of balancing exploration and exploitation in complex engineering optimization. This algorithm maintains a pool of three novel search strategies and uses a priority roulette method to select appropriate strategies for different search stages. Empirical results demonstrate that MSEFA can obtain high-quality solutions by harmonizing search strategy balance. Also, a multi-strategy serial cuckoo search algorithm (MSSCS) has been proposed in [34] to prevent premature convergence by incorporating new learning strategies based on cuckoo behaviors. The algorithm leverages a serial framework and adaptive parameter regulation to maximize the performance of each strategy. Evaluations on test suites demonstrate that MSSCS outperforms standard cuckoo searches in terms of optimization performance.

TDC has been extensively studied using various nature-inspired optimization algorithms, such as the ant colony optimization (ACO) and particle swarm optimization (PSO) algorithms, which have been used for feature optimization and selection [35], [36], [37], [38]. PSO has also been hybridized with chaotic maps, opposition-based learning, and mutation for text clustering feature selection [39]. The social spider optimization (SSO) algorithm, which mimics the cooperative intelligence of spider social groups, has been used for text clustering and demonstrates significant improvements over k-means [40], [41]. Fuzzy C-Means (FCM) clustering has been hybridized with the whale optimization algorithm (WOA), demonstrating effectiveness for text clustering [42]. The multi-verse optimizer (MVO) algorithm has been applied to text clustering for feature selection and outperforms PSO, GA, and k-means [8], [43]. A link-based MVO algorithm with a neighborhood selection strategy further improves the performance [43]. The whale optimization algorithm (MWOA) has been hybridized with FCM for text clustering and insurance scam detection, outperforming CATBoost, XGBoost, Random Forest, and LightGBM [44]. A hybrid swarm intelligence approach combining a k-means algorithm has been proposed for text clustering and evaluated on CEC2019 benchmark functions and text datasets, outperforming state-of-the-art approaches [1]. Moreover, a hybrid β-hill climbing algorithm (BHC) and salp swarm algorithm (SSA) approach has been proposed for text clustering. BHC locally searches around initial solutions and the SSA's best solution at each iteration. The hybrid approach outperforms k-means, GA, CMAES, MVO, PSO, and KHA, achieving higher clustering performance based on eight datasets, [45]. These algorithms have been evaluated on various datasets, and their performances have been compared using appropriate metrics.

In [46], the Salp Swarm Algorithm (SSA) was proposed as one of many members of a larger group of swarm intelligence algorithms. The SSA approach represents a population-based evolutionary technique inspired by the salps' swarming behavior. Instead of using evolutionary operators to manage the individuals, each salp in SSA, as in other swarm algorithms, updates the position in a search space based on its attributes and the past attributes of the better individuals (i.e., solutions). Compared to other algorithms, the SSA method provides several advantages. It can maintain a good balance between exploration and exploitation during a specific search. Only one parameter needs to be established at the starting point of algorithm execution. Furthermore, it is adaptable, scalable, and flexible, and no mathematical derivation to a specific problem is needed. As a result, SSA

has been used to handle various optimization problems, for example, engineering optimization problems [47], feature selection [48], unrelated parallel machine scheduling [49], and power flow problem [50].

Since 2017, numerous techniques and modifications have been suggested to improve SSA performance [48]. Moreover, according to the no-free-lunch theorem, it is unlikely for a single global optimizer to solve all optimization problems equally effectively [43]. Like other optimization methods, SSA aims to achieve an optimal balance between exploration and exploitation during the search process. Exploration refers to the SSA's capacity to examine various regions of the search space, often facilitated by the c_1 parameter, which introduces randomness. Conversely, exploitation involves the SSA's ability to conduct a deep search within each region it converges to by leveraging the best solution obtained using the c_1 parameter. However, SSA has shortcomings, such as premature convergence and getting trapped in local optima. Many studies have attempted to overcome these limitations by strengthening the exploitation phase. This paper presents a modified version of SSA called the Bare-Bones Salp Swarm Algorithm (BBSSA) to solve the TDC problem and address the traditional SSA's weaknesses. The BBSSA approach combines the optimization features of Bare-Bones with the conventional SSA to enhance the convergence rate, reliability, and efficiency of the traditional SSA. This paper outlines three key contributions of the BBSSA algorithm, a novel modification of the SSA algorithm that offers superior performance in optimization tasks. These contributions include:

1) The proposal of the inverse hyperbolic cosine control strategy modifies the guided direction parameter (i.e., c_1) of the population search to improve the performance of traditional SSA.
2) The incorporation of the optimization property of Bare-Bones into the updated position equation of the BBSSA algorithm balances exploration and exploitation to prevent getting trapped in local optima.
3) Utilizing a greedy selection approach during the update phase maintains stability in the search strategy, reducing the likelihood of the algorithm getting stuck in local optima and improving its convergence speed.

A case study was conducted on the scientific publications of the top 8 universities in the United Arab Emirates (UAE) based on QS ranking to demonstrate the applicability and efficacy of the proposed BBSSA algorithm. Twelve text datasets were compared, comprising six benchmark datasets and the scientific publications datasets of the top 8 UAE universities. The proposed approach was shown to be superior in terms of performance. A performance analysis was also carried out to compare the proposed BBSSA algorithm with other optimization methods. The results indicated that the BBSSA algorithm outperforms other convergence speed and effectiveness optimization methods.

The paper's organization is as follows: The TDC problem is discussed in Section II. Section III describes the salp swarm algorithm. Section explains the proposed technique for TDC. Section V provides the results and conversations, and the paper conclusions are given in section VI.

II. TEXT DOCUMENT CLUSTERING PROBLEM

TDC is a fundamental task in text mining and information retrieval that involves grouping a collection of text documents into clusters based on their similarity. The goal is to group documents that are semantically related while separating those that are dissimilar. The clustering process involves identifying the most relevant features or terms that describe the content of the documents and grouping them based on their similarity in these features.

TDC has many applications in various fields, such as document organization, information retrieval, and data mining. For example, in document organization, clustering can group similar articles, news, or scientific papers, making browsing and retrieving relevant information easier. In information retrieval, clustering can be used to group search results based on their topics, allowing users to locate the most relevant documents quickly. In data mining, clustering can be used to find patterns and insights in large text datasets.

Several methods have been proposed to solve the text document clustering problem, including traditional clustering algorithms such as K-means and hierarchical clustering and metaheuristic algorithms such as PSO, GA, and SSA. The algorithm choice depends on the dataset's characteristics and the specific requirements of the clustering task.

A. FORMULATION OF THE PROBLEM

The task of TDC is to partition a corpus of text documents into a predetermined number of clusters, aiming to maximise the similarity within each cluster and minimise the similarity between different clusters. The corpus D is represented as a list of documents ($D = (D_1, D_2, \ldots, D_i, \ldots, D_d)$), where D_i is the i^{th} document, and d is the total number of documents in D. Each cluster K is represented by a cluster centroid (K_{cnt}), which is defined by a set of terms (weights vector) of length f ($\overrightarrow{k_{cnt}} = (K_{cnt1}, K_{cnt2}, \ldots, K_{cntn})$). Here, K_{cnt} refers to the K_{th} cluster centroid, and k_{cnt1} denotes the value at position 1 in the cluster centroid k [51].

Several conditions must be met to ensure a valid and meaningful clustering solution. First, similar documents must belong to the same cluster, and dissimilar documents must belong to different clusters. Second, the union of all cluster centroids k_{cnt} must be equal to 0. Third, the intersection of any two different cluster centroids k_{cnt} and $k_{cnt'}$ must be empty, i.e., $k_{cnt} \bigcap k_{cnt'} = \varnothing$ if $K \neq K'$. Finally, each cluster centroid k_{cnt} must be non-empty, i.e., $k_{cnt} \neq \varnothing$.

To apply a metaheuristic algorithm to the TDC problem, the algorithm must be tailored to handle the unique characteristics of the task. This includes defining a fitness function that evaluates the quality of the clustering solution and encoding the solution space in a way that is compatible with the algorithm's operators [52].

The fitness function typically measures the similarity between the clustering solution and the ground truth labels. This can be done using various metrics, such as SSE. The solution space can be encoded as a set of cluster centroids, where each centroid is represented by a set of terms (weights vector) of length f ($\overrightarrow{k_{cnt}} = (K_{cnt1}, K_{cnt2}, \ldots, K_{cntn})$). The assignment of each document to a cluster is determined by its distance to the cluster centroid.

The metaheuristic algorithm then iteratively searches the solution space by updating the candidate solutions using the defined operators, such as mutation, crossover, and selection. The distance between each document and its assigned cluster centroid is calculated, and the SSE is computed as the sum of the squared distances for all documents in the corpus.

The algorithm continues to search the solution space until a stopping criterion is met, such as a maximum number of iterations or reaching a satisfactory level of SSE. The clustering solution with the lowest SSE is selected as the final solution.

The SSE measures the sum of the squared distances between each document in a cluster and its cluster centroid. Mathematically, the SSE is defined as:

$$SSE = \sum_{i=1}^{K} \sum_{d \in C_i} ||d - K_{cnt_i}||^2 \tag{1}$$

where K is the total number of clusters, C_i is the i^{th} cluster with centroid K_{cnt_i}, and d represents a document in the corpus. The expression $||d - K_{cnt_i}||^2$ calculates the squared Euclidean distance between the document d and the cluster centroid K_{cnt_i}. The SSE measures the within-cluster variation and provides a quantitative measure of the quality of the clustering solution, where a lower SSE indicates a better clustering solution.

B. TEXT DOCUMENT PREPROCESSING

Before implementing the clustering technique, the preprocess of TD incorporates a few steps, such as stop word removal and tokenization, stemming, and term weighting. Then the preprocessed TD will be converted into a numerical representation using a word embedding method called term frequency-inverse document frequency (TF-IDT). The latter transforms the TD into a numerical matrix representing each word's importance (weight) in each document [53].

1) SOLUTION REPRESENTATION

In general, the solution representation is an essential part of designing the optimization algorithm to solve any optimization problem, such as the TDC problem and significantly impacts these algorithms' performance. Each solution must be described by connectivity, completeness, and feasibility during the execution. In TDC, each solution is presented as a vector of the entire documents in the dataset $\vec{x}$ and each cell value in this vector presents any cluster number. As shown in Figure 1, the ID row represents the number of documents in the dataset, and k row represents the cluster number containing the particular documents. Each dimension can be addressed in a unique document. As seen in Figure 1, the solution $\vec{k}$ has twenty documents and these documents may be distributed among five clusters, e.g., document ten (i.e., k_{10}) is assigned to cluster number 4, and cluster number four contain three documents (i.e., 9, 10, 17).

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
k	3	3	2	3	2	1	1	2	4	4	5	1	2	2	5	5	4	1	1	1

FIGURE 1. Solution representation.

C. CLUSTERING ALGORITHM AND SIMILARITY MEASURES

In TDC, choosing an appropriate clustering algorithm and similarity measure is crucial in generating clusters based on document features and their similarities. TDC is typically addressed using an unsupervised learning approach, and determining the optimal way to partition a collection of documents is essential. To achieve this, TDC employs a set of evaluation criteria, such as cost or fitness function [54]. Similarity measures are often defined in terms of distance or dissimilarity, such as the SSE metric, a standard measurement used in many text clustering methods. The SSE metric can be implemented as an objective/cost function, allowing cluster centroids and document similarity to be computed. The objective of the cost function is to minimize the distance between documents within a cluster. Ultimately, selecting an appropriate clustering algorithm and similarity measure depends on the specific characteristics of the dataset and the goals of the text document clustering task.

III. SALP SWARM ALGORITHM (SSA)

The salp swarm algorithm (SSA) is a nature-inspired algorithm proposed by Mirjalili et al. [46]. Salps are marine organisms belonging to the Salpidae family, with a translucent barrel-shaped body that closely resembles that of jellyfish. They move similarly to jellyfish, using their body to pump water for propulsion [55]. Although scientific research on salps is still in its early stages, their swarming behavior is one of the fascinating aspects of the species. Salps often form a swarm known as a salp chain in deep waters, such as oceans. While the exact reasons for this behavior are unknown, some researchers believe it is used to enhance movement through coordinated adjustments and foraging. Figure 2 illustrates the principle of a salp chain. The unique characteristics of salps have inspired the development of the SSA algorithm, which mimics the swarming behavior of salps to solve optimization problems. The SSA algorithm has shown promising results in various optimization tasks due to its ability to balance exploration and exploitation. However, further research is necessary to understand and utilize the SSA algorithm's potential fully.

The following steps summarize the basic steps of the original SSA:

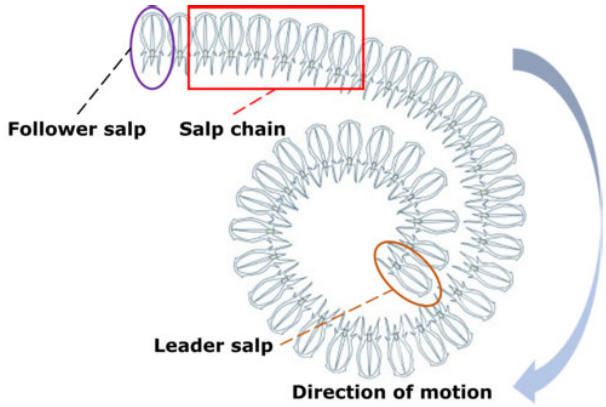

FIGURE 2. salps chain, follower, and leader concept.

Step 1: ***Parameters Initialization***.

In the context of TDC, the initialization step of SSA involves generating an initial population of candidate solutions, where each solution represents a possible clustering of the text documents. The initialization step typically involves randomly assigning each document to a cluster, which generates an initial set of candidate solutions. The initial population is then evaluated using a fitness function, such as the Sum of Squared Error (SSE), which measures the quality of the clustering solution. The candidate solutions are ranked based on their fitness, and the fittest solutions are selected to form the next generation of candidate solutions. To improve the diversity of the population, SSA also employs a local search operator, which perturbs the fittest solutions to explore the adjacent search space. The local search operator helps to prevent premature convergence and increases the chances of finding a better clustering solution. The initialization step of SSA plays an essential role in determining the quality of the final clustering solution. A well-designed initialization step can generate a diverse set of candidate solutions that cover the search space effectively. This increases the chances of finding a high-quality clustering solution and improves the algorithm's convergence rate.

Step 2: ***The salp chains model***.

In this step, the SSA involves classifying the solutions into two groups, the leader and the followers, based on their objective function values. The leader group is placed at the front of the salp chain to lead the other solutions. The population of solutions is represented as a 2-dimensional matrix, where each row represents a number of solutions, and each solution is presented as a d-dimensional vector, where d is the variable decision size for the optimization problem (Eq. 2). The leader group's position is updated using Equation 3,

Algorithm 1 SSA Pseudo-Code

Input: Population matrix $SSAM$, c_1, L_B,U_B
Output: Best Solution F
while *The end condition is not met* **do**
  Step 1: Compute each search agent's fitness (salp) based on the problem objective function.
  Step2: Determine the best salp (solution) and assign it to F vector.
  Step3: Use Eq.4 to update c_1.

  for *Each solution* x_i *in Population matrix SSAM* **do**
    if $x_i < M$ **then**
      Use Eq.3 to update the leading salp.
    end
    else
      Use Eq.5 to update the follower salp.
    end
  end
end

where x_d^1 represents a leader solution, c_2 and c_3 are random numbers, $F(d)$ denotes the decision variable d in the food source, and c_1 is a coefficient determined by Equation 4. The follower group's solutions are updated using Equation 5, where x_d^m represents the d^{th} dimension of the current solution at iteration m, and x_d^{m-1} represents the d^{th} dimension of the previous solution at iteration $m-1$. The SSA algorithm updates solutions during its execution to explore the search space of the optimization problem based on the search strategy of the algorithm.

$$\text{SSAM} = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_d^1 \\ x_1^2 & x_2^2 & \cdots & x_d^2 \\ \vdots & \vdots & \cdots & \vdots \\ x_1^m & x_2^m & \cdots & x_d^m \end{bmatrix}. \tag{2}$$

$$x_d^1 = \begin{cases} \lfloor F(d) + c_1 \times ((ub_d - lb_d) \times c_2 + lb_d) \rfloor \\ \qquad c_3 \geq 0, \\ \lfloor F(d) - c_1 \times ((ub_d - lb_d) \times c_2 + lb_d) \rfloor \\ \qquad c_3 < 0 \end{cases} \tag{3}$$

$$c_1 = 2 \times exp^{-(\frac{4l}{L_{\max}})^2} \tag{4}$$

$$x_d^m = \frac{1}{2}(x_d^m + x_d^{m-1}) \tag{5}$$

Step 3:***Stop condition***.

In this step, the exaction of the SSA (i.e., Step 2) is stopped based on the quality of the final results or the reaching of the predefined number of iterations.

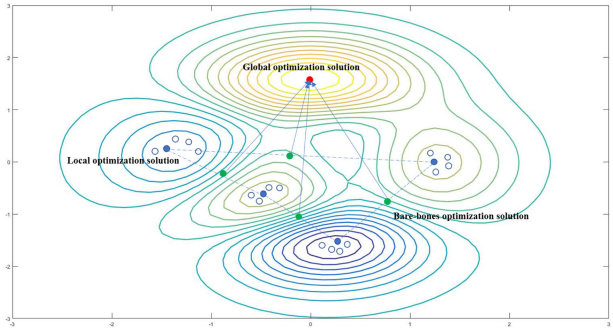

FIGURE 3. The impact of a bare-bones approach on determining the next position of a solution.

IV. THE PROPOSED METHOD

This section describes the proposed algorithm named Bare Bones Salp Swarm Algorithm (BBSSA) in detail. The motivation behind the idea is presented first, then the modified position equation method and the modifications of the guidance direction parameter are given. Finally, the BBSSA framework is illustrated.

A. MOTIVATIONS

Like many other optimization algorithms, the SSA algorithm needs several shortcomings, such as a slow convergence rate, time inefficiency, local optima trapping, and a lack of balance between exploration and exploitation phases. The $c1$ parameter is crucial in updating solutions towards the best solution during the search in SSA. However, experimental evidence has shown that the current equation for $c1$ can lead to local optima trapping [45]. This paper proposes a new modification to the c_1 equation to address this issue to improve the balance between the exploration and exploitation phases. Optimization problems typically involve unknown search spaces with multiple local areas, making it challenging for algorithms to jump out of local optima, as shown in Fig.3. To improve the search capability, a bare-bones strategy is added to the traditional SSA equations in this paper by calculating the average position between the current and best solutions, which helps the algorithm move towards the best solution direction. In the final enhancement, a greedy selection method is employed by the BBSSA to reduce randomness and enhance the search capability of the traditional SSA algorithm. In each iteration, only solutions with better objective functions are maintained by the BBSSA. The less promising solutions are discarded, reducing the time and resources spent exploring unpromising areas of the search space. By selecting only the most promising solutions and reducing randomness, the performance of SSA in solving complex optimization problems such as Text Document Clustering (TDC) is aimed to be improved by the BBSSA algorithm. Overall, these modifications aim to improve the performance of the SSA algorithm by addressing its shortcomings and enhancing its search capability.

B. MODIFYING THE SSA PARAMETER

As discussed in the previous section, managing the balance between exploration and exploitation is crucial for performing evolutionary algorithms like SSA. The c_1 parameter in SSA controls the individual search behavior of salps and significantly impacts the algorithm's exploration-exploitation trade-off. In this section, we propose modifying the c_1 parameter using an inverse hyperbolic cosine control strategy to improve SSA's performance in text document clustering. The inverse hyperbolic cosine control strategy dynamically modifies the c_1 parameter during the search process based on the following equation:

$$\bar{c_1} = a \times \ln\left(\left(\frac{4l}{L_{max}}\right)^2 + \sqrt{\left(\frac{4l}{L_{max}}\right)^4 - 1}\right) \quad (6)$$

where $\bar{c_1}$ is the value of the c_1 parameter at iteration l, a is a scaling factor, l is the current iteration number, and L_{max} is the maximum number of iterations.

This strategy initializes the c_1 parameter to a high value, promoting the exploitation of the search space. As the search progresses, c_1 gradually decreases, enabling increased exploration and escape from local optima. The scaling factor a controls the rate of decrease, while the maximum number of iterations L_{max} determines when c_1 reaches its minimum value. By dynamically modifying c_1 during the search, the inverse hyperbolic cosine control strategy helps balance exploration and exploitation in SSA. At the start of the search, the high c_1 value encourages exploration through individual salp behavior. As the search continues, the decreasing c_1 promotes exploitation, allowing escape from local optima. The inverse hyperbolic cosine control strategy has improved SSA's performance on various optimization problems, including TDC. By modifying c_1 during the search, this strategy helps increase SSA's convergence rate and solution quality.

C. MODIFICATION OF THE UPDATED POSITION EQUATION

To overcome the limitations of the standard SSA algorithm, a modified search equation based on the particle position update method and inspired by the Gaussian bare-bones search equation of BBPSO [56] is proposed. This modified equation, shown in Equation 7, incorporates more information from the global optimum solution to accelerate the convergence of the SSA algorithm. Using this updated equation, the SSA algorithm can effectively balance exploration and exploitation and improve the quality of solutions. This modification makes the SSA algorithm more efficient and suitable for solving complex optimization problems.

$$x_m^{l+1} = N\left(\frac{F(l) + x_m^l}{2}, |F(l) - x_m^l|\right) \quad (7)$$

In Equation 7, $N()$ represents the normal distribution, m denotes the current selected solution, and $F(l)$ represents the j^{th} variable of the best solution at the l^{th} iteration. Incorporating the Gaussian search equation in the SSA

algorithm has been shown to improve the best individuals in the entire population.

This modification allows the algorithm to explore the search space more effectively by incorporating information from the global optimum solution. As a result, the SSA algorithm can find better solutions to complex optimization problems.

While the SSA algorithm's ability to converge quickly is beneficial for simple multimodal and unimodal problems, its high exploitation capabilities can cause premature convergence in more complex multimodal problems. In such situations, the algorithm may get stuck in suboptimal local areas of the search space. To address this issue, this section proposes the Gaussian search equation to improve the exploitation phase of the SSA algorithm. Two additional strategies are proposed to enhance the algorithm's searchability during exploitation. These strategies are seamlessly integrated into the solution search equation and will be explained in detail in the following subsections. By utilizing these strategies, the SSA algorithm can effectively balance exploration and exploitation to find high-quality solutions to complex optimization problems.

D. THE STRATEGY OF GREEDY SELECTION

A greedy selection strategy can help SSA maintain high-quality solutions across generations. The key idea is to compare a salp's current solution with its modified candidate solutions, then select the position yielding the best objective function value. By choosing the locally optimal candidate at each step, greedy selection preserves the most promising solutions in the population from one iteration to the next. In SSA for text document clustering, the greedy selection is applied when updating each salp's position. The salp's current clustering solution is compared to its candidate solutions generated through particle mutation. Instead of randomly selecting a candidate, the greedy strategy evaluates all candidates and chooses the one with the highest fitness value. This helps ensure that improved or high-quality solutions are carried over to subsequent iterations. Over multiple iterations, greedily selecting the best candidate position allows incremental improvements to accumulate, significantly enhancing the final solution. However, greedily optimizing at each step may lead the search into local optima that cannot be escaped. To prevent this, greedy selection can be balanced with exploration methods, such as probabilistically choosing the best candidate at each step or reducing selection pressure over iterations. In summary, greedy selection maintains high-quality solutions in SSA by comparing a salp's current and candidate positions, then choosing the locally optimal candidate. This strategy aims to guide SSA toward improved clustering solutions by incrementally optimizing salp positions through greedy local selection. When properly balanced with exploration, the greedy selection is a promising method for enhancing solution accuracy in SSA for text document clustering. However, future work should examine the effects of different approaches for balancing greedy selection and explorative search.

The selection operator formula is as follows:

$$X_i^{l+1} = \begin{cases} U_m^l, & \text{fit}(\text{U}_\text{m}^\text{l}) \leq \text{fit}(\text{X}_\text{m}^\text{l}) \\ X_m^l, & \text{fit}(\text{U}_\text{m}^\text{l}) > \text{fit}(\text{X}_\text{m}^\text{l}) \end{cases} \tag{8}$$

where the fitness-related function is denoted by $fit()$, and the selection procedure focuses on minimizing the problem. The inequality "$\leq$" in Equation 8 helps move the entire population away from the flat search area of the fitness landscape, reducing the likelihood of stagnation in the search process. The proposed selection procedure improves the overall performance of evolutionary algorithms in minimizing the fitness-related function.

E. THE FRAMEWORK OF THE PROPOSED ALGORITHM

Like other population-based metaheuristic algorithms, BBSSA starts by creating a population of candidate solutions that are randomly dispersed throughout the search space. The effectiveness of the search process is then evaluated based on the performance of this initial population. Once the population has been initialized, the optimization procedure begins. BBSSA utilizes a modified search equation designed to enhance the algorithm's performance. This equation is presented in subsection IV-C and is a critical component of the proposed algorithm. The equation 9 presents the updated equation for the solutions in the BBSSA algorithm.

$$U_i^l = \begin{cases} Eq.7, & \text{rand} \leq RC_i \\ Eq.3, & \text{rand} > RC_i \end{cases} \tag{9}$$

BBSSA utilizes a fixed parameter RC_i set to 0.5, while *rand* is a random number between 0 and 1. Moreover, the proposed modified parameter $\bar{c_1}$ (Equation 6) replaces the original parameter c_1 (Equation 4). To enhance exploitation efficiency while retaining the simplicity and usability of the original SSA, BBSSA incorporates the Gaussian search equation. Additionally, the inverse hyperbolic cosine control method and greedy selection based on fitness values guide the search towards optimal solutions. As a result, BBSSA is a robust optimization algorithm capable of tackling complex optimization problems. The BBSSA complete flowchart is illustrated in Fig.4. The BBSSA pseudo-code is described in depth in Algorithm 2. The algorithm takes a population matrix, lower and upper bounds, and the number of clusters as input. The algorithm iteratively updates the solutions in the population matrix until the end condition is met. The fitness of each search agent is calculated, and the best solution is determined. The algorithm then updates the coefficient value and the leading or follower salp based on a random number and the position of the current solution. Then, the algorithm continues to iterate until the end condition is met, returning the best clustering solution.

V. EXPERIMENTS AND DISCUSSION

This section explains the experimental design of the dataset from QS, ranked top 8 UAE universities' scientific papers and

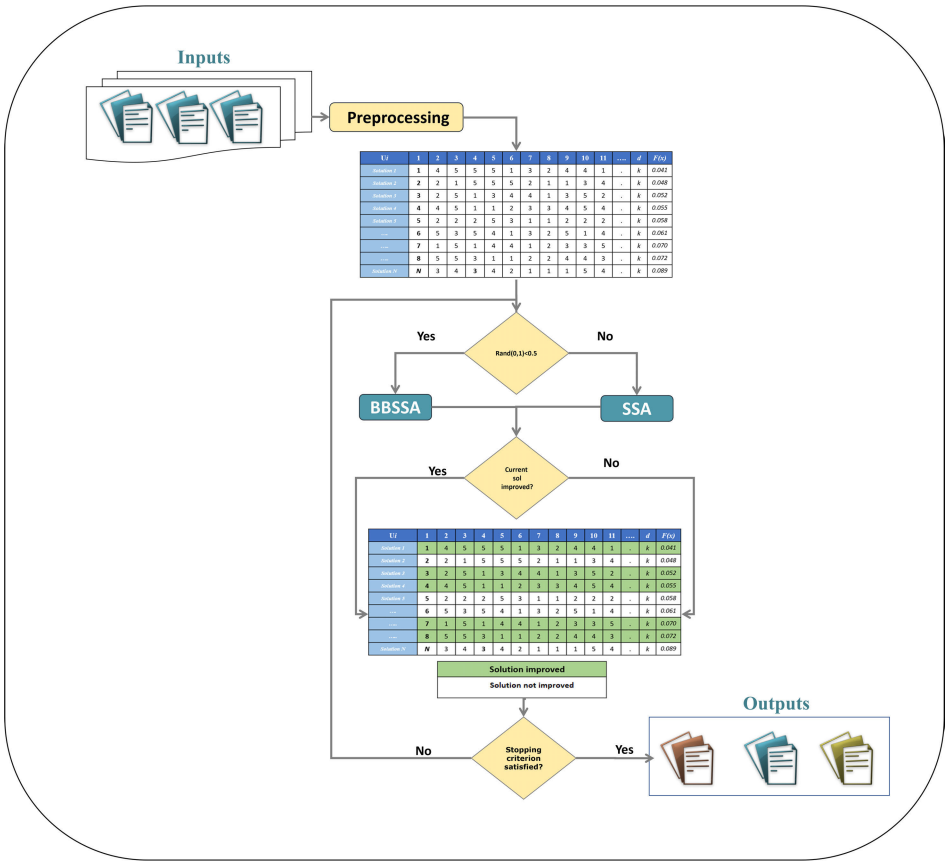

FIGURE 4. The proposed BBSSA Architecture.

TABLE 1. List of abbreviations for comparative methods, including clustering techniques and optimization algorithms.

Abbreviation	Method	Type	Reference
Agglomerative	Agglomerative algorithm	Clustering technique	[57]
DBSCAN	Density-based Spatial Clustering of Applications with Noise	Clustering technique	[58]
FFA	Firefly algorithm	Optimization algorithm	[19]
GA	Genetic Algorithm	Optimization algorithm	[15]
BAT	Bat algorithm	Optimization algorithm	[18]
GWO	Grey wolf optimizer	Optimization algorithm	[13]
Harmony Search (HS)	Harmony Search	Optimization algorithm	[17]
K-means	K-means algorithm	Clustering technique	[59]
K-means++	K-means++ algorithm	Clustering technique	[60]
Krill Herd Algorithm (KHA)	Krill Herd Algorithm	Optimization algorithm	[11]
MFO	Moth-flame optimization	Optimization algorithm	[12]
MVO	Multi-Verse Optimizer	Optimization algorithm	[14]
PSO	Particle Swarm Optimization	Optimization algorithm	[16]
Salp Swarm Algorithm (SSA)	Salp Swarm Algorithm	Optimization algorithm	[46]
Spectral	Spectral algorithm	Clustering technique	[61]

benchmark datasets, parameter configurations, the proposed approach's evaluation method, and the comparative algorithms. Furthermore, this section provides the convergence behaviour, followed by a discussion and evaluation of the results.

A. THE EXPERIMENTAL DESIGN

In this study, we aimed to evaluate the performance of the proposed BBSSA algorithm on the TDC problem. The experiments were conducted on a computer with a 2.8 GHz Intel Core i7 processor and 16 GB of RAM, running the Python programming language. Six benchmark datasets and six real datasets to evaluate the performance of clustering algorithms have been utilized. The benchmark datasets include commonly used datasets in the literature. Six external evaluation measures were employed to assess the clustering algorithms' performance: entropy, accuracy, purity, recall, precision, and F-measure. These measures were chosen to provide a comprehensive evaluation of the clustering results. For the diversification search method, 1000 iterations were deemed appropriate for each run to achieve convergence. For local-based clustering algorithms, 100 iterations per run were utilized. In total, seventeen clustering algorithms were employed in this study, including agglomerative, spectral,

Algorithm 2 BBSSA Pseudo-Code for TDC

Input: Population matrix $SSAM$, $\bar{c_1}$, L_B,U_B, Number of clusters k.
Output: Best Clustering Solution F
while *The end condition is not met* **do**
  Step1: Compute each search agent's fitness (salp) by Eq.1.
  Step2: Determine the best salp (solution) and assign it to F vector.
  Step3: Use Eq.6 to update $\bar{c_1}$.

  for *Each solution* x_i *in Population matrix SSAM* **do**
    if $rand > RC_i$ **then**
      if $x_i < M$ **then**
        Use Eq.3 to update the leading salp.
      end
      else
        Use Eq.5 to update the follower salp.
      end
    end
    else
      Use Eq.7 to update the leading or follower salp.
    end
  end
end

DBSCAN, k-means, and k-means++, as well as eleven optimization algorithms: GA, HS, PSO, WOA, FFA, GWO, MFO, MVO, BAT, KHA, and SSA. These algorithms were selected based on their popularity in the literature and their ability to handle different clustering problems. The experimental design involved conducting multiple runs of each algorithm on each dataset and recording the clustering results for each evaluation measure. The results were then statistically analyzed to compare the performance of the different algorithms. Table 1 illustrates abbreviated comparative methods with type and references.

This section evaluates and compares the proposed method with other state-of-the-art algorithms for text and scientific article clustering. Six text datasets and six scientific article datasets from top UAE universities are used to examine the performance of the proposed method. Each method is run thirty times with similar starting solutions for the meta-heuristic algorithms to ensure consistency in the clustering process. Six independent external measures, commonly used for evaluating clustering algorithms, namely agglomerative, spectral, DBSCAN, k-means, k-means++, and eleven optimization algorithms (GA, HS, PSO, WOA, FFA, GWO, MFO, MVO, BAT, KHA, and SSA) are employed to compare and validate the proposed solution. This approach ensures a fair comparison among all competing algorithms. For population-based algorithms, 1000 iterations for each run are appropriate for the convergence of the diversification search method. For local-based clustering algorithms, 100 iterations per run and 100 iterations were experimentally appropriate for the convergence of the intensification search algorithm.

TABLE 2. Dataset of scientific papers from the top 8 universities in the United Arab Emirates (UAE), including the university name and number of publications.

ID	University Name	Number of Publications
1	Khalifa University of Science and Technology	11572
2	University of Sharjah	7185
3	United Arab Emirates University	6554
4	American University of Sharjah	3326
5	Abu Dhabi University	1608
6	Zayed University	1783
7	Ajman University	1066
8	American University in Dubai	241

To ensure a fair comparison, all algorithms use identical datasets and parameters. Furthermore, they maintain consistent starting solutions and iterations.

B. CLUSTERING DATASETS

To prove the performance of the proposed method, benchmarks and real datasets are used to test the TDC problem. Based on the literature, Twelve datasets are selected as testbeds to be examined. The datasets belong to the literature and the field of research. These are widely used to test the performance of TDC algorithms and approaches. Text datasets can be obtained in numerical form and terms extraction from LABIC, which stands for Laboratory of Computational Intelligence. The following are some of the characteristics of text datasets:

Classic4: It contains 2000 documents and four classes, including CRAN and CACM, as well as MED and CISI (every single class contains 500 documents). Classic4 consists of 7,095 documents that can be assigned to any of the four previously described categories; Classic3 and Classic4 are two different variants of the same dataset.

Tr12, Tr41, and Wap (Datasets from Karypis Lab): To maximize diversity, the datasets are selected from a variety of sources. As a result, tr41 and tr12, as well as Wap, contain 878 and 313 documents, respectively, and 1560 documents belonging to 10 and 8, and 20 classes. More details are available in [62].

20Newsgroups:
This dataset contains more than ninety thousand documents from newsgroups that are split into 20 classes. To execute the experiment, the first 100 documents are selected from the dataset's first three classes: rec_autos classes, talk_politics_misc, and comp_windows_x.

CSTR:
This data comes from an interdisciplinary research center focusing on English language, linguistics, and informatics. In 1984, the CSTR was created. It focuses on research in various fields, including speech recognition and information access. There are 299 documents in this dataset, which are divided into four classes: artificial intelligence, theory, systems, and robotics.

TABLE 3. Overview of the datasets utilized in the experiment.

Type	ID	Datasets	Number of objects/Documents	Number of clusters (K)
Text Datasets				
UAE universities scientific papers datasets				
	DS1	2016	3335	3
	DS2	2017	4206	3
	DS3	2018	4904	3
	DS4	2019	6510	3
	DS5	2020	7734	3
	DS6	2021 (till 22/9/2021)	6045	3
Standard datasets				
	DS7	CSTR	299	4
	DS8	20Newsgroups	300	3
	DS9	tr12	313	8
	DS10	tr41	878	10
	D11	Wap	1560	20
	DS12	Classic4	2000	4

Six collections of scientific articles from the top eight QS-ranked universities in the United Arab Emirates were analyzed. The original classes were matched with a web of ''science core collection topic'' categories, and the results were presented in Table 2, which shows the number of published articles across three primary categories in all six datasets. For a comprehensive overview of each dataset, please refer to Table 3. The table provides information on various text datasets, including their ID, the year they were collected, the number of objects/documents in each dataset, and the number of clusters (K). The datasets are divided into two categories: UAE universities' scientific papers datasets and standard datasets. The UAE universities' scientific papers datasets include six datasets (DS1-DS6) from 2016 to 2021. The number of objects/documents in these datasets ranges from 3335 in DS1 to 7734 in DS5, and each dataset is divided into three clusters. These datasets likely contain scientific papers published by universities in the UAE. The standard datasets category includes six datasets (DS7-D12) with varying numbers of objects/documents and clusters. The CSTR dataset (DS7) contains 299 documents divided into four clusters, while the Classic4 dataset (DS12) contains 2000 documents divided into four clusters. The other datasets have varying numbers of documents and clusters, with tr41 (DS10) containing the most documents (878) and Wap (DS11) containing the most clusters (20). These datasets are likely used as benchmark datasets in text mining and clustering research. These datasets were found by conducting a Scopus query, specifying the university, the year, and the field of study. For instance, an example of the Scopus query used to collect articles from ''Ajman University'' in the field of engineering during the year 2021 is provided below:

(AF-ID (''Ajman University'' 60104134) OR AF-ID (''Khalifa University College of Medicine and Health Sciences'' 60231863) AND (LIMIT-TO (PUBYEAR, 2021) AND (LIMIT-TO (SUBJAREA, ''ENGI''))))

TABLE 4. Parametric values for all algorithms being compared.

Parameters	Algorithm	Value
A	BAT	0.5
r	BAT	0.5
$Qmin$	BAT	0
$Qmax$	BAT	2
$alpha$	FFA	0.5
$betamin$	FFA	0.2
$gamma$	FFA	1
$WEPMax$	MVO	1
$WEPMin$	MVO	0.2
p	MVO	6
$Vmax$	PSO	6
$wMax$	PSO	0.9
$wMin$	PSO	0.2
$c1$	PSO	2
$c2$	PSO	2
Vf	KHA	0.02
$Dmax$	KHA	0.002
$NMax$	KHA	0.05
Crossover probability	GA	0.8
Mutation probability	GA	0.02
$PARMin$	HS	0.45
$PARMax$	HS	0.9
$bwMin$	HS	0.1
$bwMax$	HS	1
HMCR	HS	0.9
Population size	All Optimization algorithms	30
Maximum number of iteration	All Optimization algorithms	500
runs	All Optimization algorithms	30

This query was used to collect articles from various universities and fields of study for the datasets. The data collection method ensures the relevance and diversity of the datasets used in the study.

It is essential to determine the clustering algorithms' parameter values. Table 4 contains all the algorithms' parameters. It is worth noting that the algorithm's author's recommendation determines all these parameter values.

C. CLUSTERING ALGORITHM EVALUATION MEASURES

When evaluating the efficacy of clustering algorithms, three distinct types of measurements are commonly utilized: external, internal, and relative. External measures involve quantitative analysis to determine how closely a clustering

algorithm corresponds to the underlying data structure. This is achieved by comparing the created clusters to established ground-truth clusters. Relevant external measures include purity, entropy, recall, Precision, F-measure, and accuracy, each of which serves to assess the performance of different algorithms [3]. In cases where external evaluation is not feasible, internal clustering quality criteria should be used to assess the validity of clustering. Relative measures are then obtained by comparing the scores of different clusters based on internal measures. Furthermore, if a single clustering algorithm consistently outperforms others across multiple measures, it can be confidently concluded that it represents the optimal choice. The specifics of these measures are given in Table 5.

D. ANALYSIS OF THE RESULTS

This section presents the experimental results obtained from evaluating the proposed method on TDC. The effectiveness and efficiency of the proposed method are evaluated based on two criteria: (a) clustering quality using benchmark datasets and (b) clustering quality using real scientific articles from the top eight QS-ranked UAE universities datasets. To assess the clustering quality, various benchmark datasets are used. These datasets are extensively used in the literature and provide a baseline for evaluating clustering algorithms. Additionally, real scientific article datasets from the top eight QS-ranked UAE universities are used to evaluate the performance of the proposed method on real-world data. The proposed method's effectiveness is measured by its ability to produce high-quality clusters, while its efficiency is evaluated based on the computational resources required to obtain these clusters. The results of the experiments provide insights into the performance of the proposed method and its potential for practical applications in TDC.

1) EFFICIENCY ANALYSIS OF THE IMPROVEMENT STRATEGY

To answer the first research question, the proposed algorithm is compared with current algorithms based on the quality of the produced clusters. Seven metrics, including purity, entropy, precision, recall, F-measure, accuracy, and SSE, are used to evaluate the effectiveness of the clustering results. The first six measures are taken from external quality measures, which provide a quantitative analysis of how well the clustering algorithm matches the underlying structure of the data. The SSE measure is an internal quality measure that quantifies the compactness of the clusters produced by the algorithm. This measure provides information on how well the algorithm can group similar data points while keeping different points apart. By comparing the proposed algorithm with current algorithms using these metrics, we can determine the effectiveness of the clustering findings and the potential of the proposed algorithm for practical applications.

Several metrics are used to evaluate the quality of clustering results. One such metric is purity, which measures the degree to which each cluster contains documents from only one class, i.e., the largest class in each cluster. The higher the purity value, the more influential the clustering solution. Tables 6 and 7 show the purity of clusters obtained from various datasets when the proposed algorithm is used. BBSSA outperformed all compared algorithms in four datasets (DS1, DS2, DS8, DS12) and was the second-best algorithm in DS6 and DS7 regarding cluster purity. Another widely used metric for clustering evaluation is the F-measure. Each cluster is considered a query result, and each class is considered a query's preferred document collection. The recall and precision of the clusters are then calculated for each class. Tables 6 and 7 illustrate the performance of the algorithms in the document collections in terms of the F-measure. BBSSA obtained the best F-measure compared to other algorithms due to the high quality of the clusters formed by the method. Accuracy is another metric used to evaluate clustering algorithms. Tables 6 and 7 show the accuracy of all comparison algorithms. In general, BBSSA outperformed the other approaches in terms of accuracy. Entropy is a metric that evaluates the quality of clusters clustering algorithms produce. The optimum clustering approach produces clusters that only contain documents from one class and have zero entropy. Tables 6 and 7 show the entropy of clusters obtained from various datasets. BBSSA outperformed other algorithms in terms of entropy. However, random initial solutions used in different runs to test clustering methods resulted in the worst entropy quality. The high entropy of the clusters created by clustering algorithms indicates that the documents in the generated clusters are from various document classes. It should be noted that the performance of clustering algorithms may vary significantly from one dataset to another. Therefore, the results presented in Tables 6 and 7 may differ slightly from other datasets. Nonetheless, the proposed BBSSA algorithm consistently outperformed other algorithms regarding clustering quality across various datasets. Based on the results presented in Table 6 on DS1, it can be observed that BBSSA has achieved the highest accuracy, precision, recall, F-measure, purity, and lowest entropy among all the algorithms tested. Specifically, the accuracy was 0.74722, precision was 0.587552, recall was 0.497061, F-measure was 0.543505, purity was 0.582263, and entropy was 0.476291. Furthermore, BBSSA outperformed other algorithms on DS2, DS3, DS4, DS5, and DS6 regarding accuracy, precision, recall, F-measure, purity, and entropy. Hence, BBSSA can be considered the top-performing algorithm among SSA, WOA, MVO, GA, and others for these datasets.

Based on Table 7, which presents the performance metrics results for six datasets (DS7-DS12), it can be observed that the clustering techniques and optimization algorithms used in the study had varying levels of success across the different datasets. For example, when considering accuracy as a performance metric, K-mean++, MVO, and WOA were among the top-performing optimization algorithms for DS7, while BBSSA, SSA, and HS achieved the highest accuracy scores for DS12. Meanwhile, DBSCAN was consistently among the worst-performing clustering techniques

TABLE 5. Six external evaluation measures:purity, recall, Precision, F-measure, and accuracy for TDC evaluation.

Measure	Equation	Description
Purity	$purity = \frac{1}{D}\sum_{i=1}^{k}\max(i,j)$	The purity metric is defined as the average of the maximum number of documents in each cluster belonging to a particular class. The total number of documents in the dataset is represented by the symbol D, while k represents the number of clusters. The notation $max(i,j)$ is used to denote the maximum number of documents from class i that are present in cluster j.
Entropy	$E(kj) = -\sum_{i=1} p(i,j)log(p(i,j))$	The entropy metric is calculated as the sum of the probability of each document in a cluster multiplied by the logarithm of that probability and then negated. The entropy of cluster j is denoted by $E(kj)$, where $p(i,j)$ represents the probability of a document from class i belonging to cluster j. The overall entropy for all clusters can be calculated as follows: $Entropy = -\sum_{i=1}^{k}\frac{D_i}{D}E(kj)$, where D is the total number of documents in the dataset, D_i is the number of documents in cluster j, and k represents the number of clusters.
Recall	$R(i,j) = \frac{D_{i,j}}{D_i}$	Recall is a metric used to evaluate the ability of a clustering algorithm to identify all relevant documents for a class and assign them to the correct cluster. The recall value is calculated by dividing the number of correctly assigned documents for a particular class in a cluster Di,j by the total number of documents in that class D_i.
Precision	$P(i,j) = \frac{D_{i,j}}{D_j}$	Precision is a metric used to evaluate the ability of a clustering algorithm to assign documents to their respective clusters accurately. It is calculated as the ratio of correctly assigned documents for a particular class i in cluster j to the total number of documents in that cluster j. A high precision value suggests that the algorithm effectively assigns relevant documents to their correct cluster, while a low precision value indicates that the algorithm is assigning irrelevant documents to clusters. Precision is just one of several metrics used to evaluate clustering algorithms, and it should be used alongside other metrics for a comprehensive assessment.
F-measure	$F(i,j) = \frac{2\times P(i,j)\times R(i,j)}{P(i,j)+R(i,j)}$	In addition to $R(i,j)$, which refers to the class recall i in the cluster j, $P(i,j)$ relates to the class precision i in the cluster j. The following formula can be used to calculate the F-measure of all clusters: $F = \sum_{i=1}^{k}\frac{D_i}{D}maxF(i,j)$
Accuracy	$Ac = \frac{1}{D}\sum_{j=1}^{k} D_{ij}$	The accuracy metric is calculated as the total number of correctly distributed documents for all classes and clusters divided by the total number of documents in the dataset D. The number of correctly distributed documents for a particular class i in cluster j is represented by D_{ij}, k represents the total number of clusters, and D represents the total number of documents in the dataset.

for most datasets. Precision, recall, and F-measure were also evaluated as performance metrics, and the results again showed that the success of the clustering techniques and optimization algorithms varied depending on the dataset. Purity was another metric used to evaluate the quality of the clustering results, and BBSSA, KHA and MVO were among the top-performing optimization algorithms for this metric.

2) CONVERGENCE ANALYSIS

The convergence behavior of clustering algorithms is a critical assessment criterion for finding the best solution. In this study, the convergence behavior of BBSSA was evaluated and compared to current state-of-the-art approaches. The convergence behavior of BAT, FFA, GA, GWO, HS, KHA, MFO, MVO, PSO, SSA, WOA, and BBSSA was tested on 12 datasets, and SSE values were plotted over 500 iterations.

TABLE 6. Performance metrics results for six datasets (DS1-DS6).

Dataset	Measure	Clustering techniques					Optimization algorithms										
		Agglomerative	Spectral	DBSCAN	K-mean	K-mean++	KHA	MFO	GWO	MVO	GA	PSO	HS	BAT	FFA	WOA	SSA	BBSSA
DS1	Accuracy	0.4318555	0.436035	0.4005455	0.435515	0.357304	0.366301	0.345901	0.435552	0.339855	0.364885	0.446415	0.459309	0.432641	0.465265	0.463509	0.474722	**0.532492**
	Precision	0.3597169	0.442255	0.3392604	0.39526	0.4091383	0.417101	0.447701	0.534021	0.441674	0.421304	0.423504	0.571539	0.571105	0.583596	0.577039	0.587552	**0.629962**
	Recall	0.4925472	0.466586	0.4255906	0.407591	0.3091629	0.316201	0.342801	0.435961	0.341778	0.535534	0.506034	0.482948	**0.674476**	0.486605	0.486748	0.497061	0.593253
	F-measure	0.3970561	0.326595	0.3045996	0.3546	0.3459954	0.348001	0.393601	0.481903	0.388623	0.413943	0.337743	0.524392	0.46078	0.536649	0.531492	0.543505	**0.601509**
	Purity	0.4485001	0.481639	0.4076436	0.409644	0.3524738	0.362501	0.406001	0.495318	0.405017	0.387387	0.435487	0.56845	0.499328	0.577607	0.57065	0.582263	**0.654351**
	Entropy	0.4892585	0.507597	0.458602	0.524602	0.8201057	0.497101	0.452001	0.619904	0.717035	0.434445	0.478645	0.520691	0.610411	0.482891	0.501991	0.476291	**0.42001**
	Rank	11	10	12	11	13	11	10	7	9	7	7	5	5	3	3	2	1
DS2	Accuracy	0.3632522	0.405485	0.3038065	0.378419	0.3180393	0.319001	0.368601	0.349806	0.367571	0.321571	0.312185	0.404419	0.491502	0.409185	0.380252	0.440485	**0.557816**
	Precision	0.3423996	0.399033	0.309429	0.365166	0.3121482	0.313101	0.428901	0.413429	0.420922	0.382922	0.360133	0.439166	0.41656	0.445133	0.4154	0.474033	**0.590868**
	Recall	0.3280106	0.357644	0.3016993	0.366177	0.3099926	0.312001	0.375601	0.349699	0.367606	0.313606	0.317044	0.384177	0.387348	0.387044	0.359011	0.414644	**0.442957**
	F-measure	0.3135804	0.354813	0.3193157	0.361947	0.3406494	0.348601	0.403601	0.380316	0.393556	0.299556	0.321414	0.410947	0.390667	0.412414	0.39258	0.437813	**0.497587**
	Purity	0.3109961	0.441729	0.3026774	0.413363	0.3741288	0.382101	0.409101	0.409677	0.408091	0.342091	0.335529	0.434363	0.452491	0.438529	0.410996	0.470729	**0.523472**
	Entropy	0.6124685	0.595268	0.7472693	0.661068	0.8028197	0.726801	0.677701	0.772269	0.754695	0.676695	0.648068	0.712068	0.750853	0.692068	0.717468	0.684268	**0.504933**
	Rank	12	5	15	8	13	12	6	8	7	8	7	5	4	3	3	2	1
DS3	Accuracy	0.3373444	0.448051	0.3045257	0.379511	0.2971487	0.302101	0.376701	0.407526	0.36768	0.33568	0.377577	0.448511	0.408577	0.462577	0.411344	0.490051	**0.542424**
	Precision	0.450757	0.492264	0.3217809	0.421524	0.3522029	0.361201	0.419801	0.429781	0.412801	0.374801	0.43849	0.507524	0.421895	0.51749	0.470757	0.541264	**0.589106**
	Recall	0.3055487	0.434055	0.3233684	0.377815	0.2944349	0.300401	0.360901	0.426368	0.354918	0.309918	0.345282	0.439815	0.398606	0.449282	0.400549	0.480055	**0.693395**
	F-measure	0.3781003	0.409007	0.4047721	0.417567	0.3221783	0.330201	0.389601	0.427772	0.382607	0.291607	0.447033	0.470567	0.438393	0.480033	0.4281	0.510007	**0.637608**
	Purity	0.413171	0.555278	0.472799	0.480838	0.3907924	0.395801	0.456301	0.487799	0.451294	0.380294	0.498604	0.544838	0.542588	0.554604	0.504171	**0.583278**	0.511413
	Entropy	0.4932334	**0.397833**	0.5009972	0.509433	0.7137675	0.534801	0.501301	0.571997	0.623305	0.518305	0.451733	0.522433	0.577219	0.507733	0.536233	0.497833	0.443782
	Rank	12	4	12	9	13	12	9	8	8	8	6	4	5	3	3	2	1
DS4	Accuracy	0.3786896	0.461013	0.3704436	0.42603	0.4125524	0.421601	0.441001	0.487013	0.432005	0.405444	0.359005	0.46303	0.493379	0.472444	0.45069	0.503444	**0.612424**
	Precision	0.3630419	0.344508	0.4241959	0.376882	0.3944651	0.404501	0.424001	0.450508	0.414011	0.419096	0.371011	0.456882	0.480788	0.466196	0.443042	0.498096	**0.544511**
	Recall	0.34801	0.348657	0.429064	0.35585	0.3812988	0.384301	0.408801	0.449657	0.400764	0.457864	0.388764	0.44185	0.558485	0.459064	0.42201	0.483864	**0.594719**
	F-measure	0.3691419	0.418721	0.4151959	0.429882	0.3876084	0.397601	0.414101	0.449721	0.40712	0.413696	0.37012	0.456882	0.456207	0.474196	0.437142	0.500696	**0.566968**
	Purity	0.4865921	0.475999	0.6025461	0.547132	0.4107521	0.413801	0.568301	0.578999	0.560251	0.606346	0.486251	0.608132	0.450345	0.620546	0.591592	**0.649346**	0.561397
	Entropy	0.5386321	0.482138	0.4776299	0.47453	0.5874242	0.594401	0.548901	0.539138	0.546931	**0.40413**	0.453931	0.53553	0.517934	0.51863	0.623632	0.49513	0.430268
	Rank	16	13	8	11	13	12	9	7	8	5	7	5	4	3	3	2	1
DS5	Accuracy	0.4905456	0.493298	0.484633	0.49371	0.5011585	0.511201	0.532601	0.562298	0.531633	0.519132	0.503243	0.529132	**0.647406**	0.523546	0.51771	0.559243	0.581291
	Precision	0.4454473	0.447943	0.5003685	0.491312	0.462637	0.467601	0.534401	0.524943	0.531369	0.477333	0.463045	0.521333	**0.565088**	0.521447	0.513312	0.555045	0.518689
	Recall	0.3423349	0.447062	0.4175523	0.3508	0.4010829	0.410101	0.480601	0.481062	0.470552	0.410621	0.444732	0.449621	0.458571	0.448335	0.4438	0.477732	**0.560201**
	F-measure	0.4140516	0.448696	0.4877675	0.446216	0.4314881	0.441501	0.501801	0.501696	0.499767	0.413138	0.461149	0.483138	0.486616	0.483052	0.474216	0.511149	**0.585785**
	Purity	0.5141566	0.544488	0.40069	0.579721	0.4759058	0.484901	0.492701	0.612488	0.49169	0.593943	0.558854	0.606943	0.542761	0.601157	0.603721	**0.635854**	0.46528
	Entropy	0.5468173	0.487536	0.5706077	0.596117	0.7043592	0.545401	0.529601	0.576536	0.621608	0.559517	0.547017	0.662517	0.578425	0.649817	0.659117	0.626017	**0.392884**
	Rank	16	9	14	13	13	12	5	4	6	8	7	5	2	3	3	2	1
DS6	Accuracy	0.6326076	0.668294	0.6205796	0.679908	0.5858086	0.593801	0.669101	0.636339	0.662058	0.576058	0.596339	0.704294	0.625792	0.731608	0.67758	0.749908	**0.791093**
	Precision	0.6620295	0.644915	0.5708015	0.688029	0.5698953	0.577901	0.678601	0.660384	0.672583	0.624583	0.606384	0.691915	0.613809	0.724029	0.662801	**0.747029**	0.719972
	Recall	0.6527976	0.622384	0.5795696	0.702798	0.5259185	0.528901	0.642001	0.616364	0.631986	0.554986	0.567364	0.684384	0.648523	0.718798	0.65457	0.731798	**0.783203**
	F-measure	0.6763864	0.628072	0.6172584	0.691986	0.5471582	0.555201	0.654901	0.637712	0.651873	0.552873	0.578712	0.688072	0.621824	0.720386	0.655258	**0.741986**	0.740015
	Purity	0.6352375	0.581223	0.6089095	0.622737	0.5938009	0.599801	0.636001	0.624279	0.631995	0.554995	0.598279	0.674223	0.576705	0.701237	0.648909	**0.723737**	0.720264
	Entropy	0.4001709	0.403271	0.4853709	**0.353971**	0.5600567	0.566101	0.582101	0.530601	0.578076	0.532076	0.494601	0.511271	0.462749	0.462171	0.531371	0.439971	0.40803
	Rank	6	7	11	4	13	12	6	7	6	8	7	4	5	3	3	2	1
Average		12.17	8.00	12.00	9.33	13.00	11.83	7.50	6.83	7.33	7.33	6.83	4.67	4.17	3.00	3.00	2.00	1.00
Final Rank		16	12	14	12	13	12	11	7	8	8	7	6	5	3	3	2	1

Note: The lowest ranked algorithm is the best one

TABLE 7. Performance metrics results for six datasets (DS7-DS12).

Dataset	Measure	Clustering techniques					Optimization algorithms										
		Agglomerative	Spectral	DBSCAN	K-mean	K-mean++	KHA	MFO	GWO	MVO	GA	PSO	HS	BAT	FFA	WOA	SSA	BBSSA
DS7	Accuracy	0.436035	0.431855	0.4005455	0.357304	0.435515	0.364885	0.432641	0.463509	0.459309	0.339855	0.435552	0.446415	0.345901	0.366301	0.465265	0.474722	**0.642966**
	Precision	0.442255	0.359717	0.3392604	0.4091383	0.39526	0.421304	0.571105	0.577039	0.571539	0.441674	0.534021	0.423504	0.447701	0.417101	0.583596	**0.587552**	0.568746
	Recall	0.466586	0.492547	0.4255906	0.3091629	0.407591	0.535534	**0.674476**	0.486748	0.482948	0.341778	0.435961	0.506034	0.342801	0.316201	0.486605	0.497061	0.617415
	F-measure	0.326595	0.397056	0.3045996	0.3459954	0.3546	0.413943	0.46078	0.531492	0.524392	0.388623	0.481903	0.337743	0.393601	0.348001	0.536649	0.543505	**0.765406**
	Purity	0.481639	0.4485	0.4076436	0.3524738	0.4096444	0.387387	0.499328	0.57065	0.568345	0.405017	0.495318	0.435487	0.406001	0.362501	0.577607	**0.582263**	0.578362
	Entropy	0.507597	0.489258	0.458602	0.8201057	0.524602	**0.434445**	0.610411	0.501991	0.520691	0.717035	0.619904	0.478645	0.452001	0.497101	0.482891	0.476291	0.525404
	Rank	10	11	14	17	13	8	6	4	5	16	7	9	12	15	3	2	1
DS8	Accuracy	0.405485	0.363252	0.3038065	0.3180393	0.378419	0.321571	0.491502	0.380252	0.404419	0.367571	0.349806	0.312185	0.368601	0.319001	0.409185	0.440485	**0.629516**
	Precision	0.399033	0.3424	0.309429	0.3121482	0.365166	0.382922	0.41656	0.4154	0.439166	0.420922	0.413429	0.360133	0.428901	0.313101	0.445133	0.474033	**0.622968**
	Recall	0.357644	0.328011	0.3016993	0.3099926	0.366177	0.313606	0.387348	0.359011	0.384177	0.367606	0.349699	0.317044	0.375601	0.312001	0.387044	0.414644	**0.715357**
	F-measure	0.354813	0.31358	0.3193157	0.3406494	0.361947	0.299556	0.390667	0.39258	0.410947	0.393556	0.380316	0.321414	0.403601	0.348601	0.412414	0.437813	**0.737188**
	Purity	0.441729	0.310996	0.3026774	0.3741288	0.413363	0.342091	0.452491	0.410996	0.434363	0.408091	0.409677	0.335529	0.409101	0.382101	0.438529	0.470729	**0.652272**
	Entropy	0.595268	0.612468	0.7472693	0.8028197	0.661068	0.676695	0.750853	0.717468	0.712068	0.754695	0.772269	0.648068	0.677701	0.726801	0.692068	0.684268	**0.442733**
	Rank	5	12	17	16	9	14	4	8	6	10	11	13	7	15	3	2	1
DS9	Accuracy	0.448051	0.337344	0.3045257	0.2971487	0.379511	0.33568	0.408577	0.411344	0.448511	0.36768	0.407526	0.377577	0.376701	0.302101	0.462577	0.490051	**0.63595**
	Precision	0.492264	0.450757	0.3217809	0.3522029	0.421524	0.374801	0.421895	0.470757	0.507524	0.412801	0.429781	0.43849	0.419801	0.361201	0.51749	0.541264	**0.598737**
	Recall	0.434055	0.305549	0.3233684	0.2944349	0.377815	0.309918	0.398606	0.400549	0.439815	0.354918	0.426368	0.345282	0.360901	0.300401	0.449282	0.480055	**0.661946**
	F-measure	0.409007	0.3781	0.4047721	0.3221783	0.417567	0.291607	0.438393	0.4281	0.470567	0.382607	0.427772	0.447033	0.389601	0.330201	0.480033	0.510007	**0.673994**
	Purity	0.555278	0.413171	0.472799	0.3907924	0.480838	0.380294	0.542588	0.504171	0.544838	0.451294	0.487799	0.498604	0.456301	0.395801	0.554604	**0.583278**	0.463723
	Entropy	**0.397833**	0.493233	0.5009972	0.7137675	0.509433	0.518305	0.577219	0.536233	0.522433	0.623305	0.571997	0.451733	0.501301	0.534801	0.507733	0.497833	0.605168
	Rank	4	12	14	17	10	15	8	6	5	13	9	7	11	16	3	2	1
DS10	Accuracy	0.461013	0.37869	0.3704436	0.4125524	0.42603	0.405444	0.493379	0.45069	0.46303	0.432005	0.487013	0.359005	0.441001	0.421601	0.472444	0.503444	**0.574988**
	Precision	0.344508	0.363042	0.4241959	0.3944651	0.376882	0.419096	0.480788	0.443042	0.456882	0.414011	0.450508	0.371011	0.424001	0.404501	0.466196	0.498096	**0.739493**
	Recall	0.348657	0.34801	0.429064	0.3812988	0.35585	0.457864	0.558485	0.42201	0.44185	0.400764	0.449657	0.388764	0.408801	0.384301	0.459064	0.483864	**0.671344**
	F-measure	0.418721	0.369142	0.4151959	0.3876084	0.429882	0.413696	0.456207	0.437142	0.456882	0.40712	0.449721	0.37012	0.414101	0.397601	0.474196	0.500696	**0.60528**
	Purity	0.475999	0.486592	0.6025461	0.4107521	0.547132	0.606346	0.450345	0.591592	0.608132	0.560251	0.578999	0.486251	0.568301	0.413801	0.620546	**0.649346**	0.605002
	Entropy	0.482138	0.538632	0.4776299	0.5874242	0.47453	**0.40413**	0.517934	0.623632	0.53553	0.546931	0.539138	0.453931	0.548901	0.594401	0.51863	0.49513	0.559863
	Rank	13	16	8	17	12	5	4	9	6	11	7	14	10	15	3	2	1
DS11	Accuracy	0.493298	0.490546	0.484633	0.5011585	0.49371	0.519132	**0.647406**	0.51771	0.529132	0.531633	0.562298	0.503243	0.532601	0.511201	0.523546	0.559243	0.530703
	Precision	0.447943	0.445447	0.5003685	0.462637	0.491312	0.477333	0.565088	0.513312	0.521333	0.531369	0.524943	0.463045	0.534401	0.467601	0.521447	0.555045	**0.608058**
	Recall	0.447062	0.342335	0.4175523	0.4010829	0.3508	0.410621	0.458371	0.4438	0.449621	0.470552	0.481062	0.444732	0.480601	0.410101	0.448335	0.477732	**0.557939**
	F-measure	0.448696	0.414052	0.4877675	0.4314881	0.446216	0.413138	0.486616	0.474216	0.483138	0.499767	0.501696	0.461149	0.501801	0.441501	0.483052	0.511149	**0.552305**
	Purity	0.544488	0.514157	0.40069	0.4759058	0.579721	0.593943	0.542761	0.603721	0.606943	0.49169	0.612488	0.558854	0.492701	0.484901	0.601157	**0.635854**	0.493513
	Entropy	**0.487536**	0.546817	0.5706077	0.7043592	0.596117	0.559517	0.578425	0.659117	0.662517	0.621608	0.576536	0.547017	0.529601	0.545401	0.649817	0.626017	0.583465
	Rank	9	16	15	17	14	12	2	10	7	8	4	11	5	13	6	3	1
DS12	Accuracy	0.371707	0.632608	0.6205796	0.5858086	0.679908	0.576058	0.625792	0.67758	0.704294	0.662058	0.636339	0.596339	0.669101	0.731608	0.668294	**0.749908**	
	Precision	0.389086	0.662029	0.5708015	0.5698953	0.688029	0.624583	0.613809	0.662801	0.691915	0.672583	0.660384	0.606384	0.678601	0.577901	0.724029	0.644915	**0.747029**
	Recall	0.454617	0.652798	0.5795696	0.5259185	0.702798	0.554986	0.648523	0.65457	0.684384	0.631986	0.616364	0.567364	0.642001	0.528901	0.718798	0.622384	**0.731798**
	F-measure	0.434929	0.676386	0.6172584	0.5471582	0.691986	0.552873	0.621824	0.655258	0.688072	0.651873	0.637712	0.578712	0.654901	0.555201	0.720386	0.628072	**0.741986**
	Purity	0.451778	0.635237	0.6089095	0.5938009	0.622737	0.554995	0.576705	0.648909	0.674223	0.631995	0.624279	0.598279	0.636001	0.599801	0.701237	0.581223	**0.723737**
	Entropy	0.68173	0.400171	0.4853709	0.5600567	**0.353971**	0.532076	0.462749	0.531371	0.511271	0.578076	0.530601	0.494601	0.582101	0.566101	0.462171	0.403271	0.439971
	Rank	17	5	12	16	3	14	11	6	4	9	10	13	8	15	2	7	1
Average		9.67	12.00	13.33	16.67	10.17	11.33	5.83	7.17	5.50	11.17	8.00	11.17	8.83	14.83	3.33	3.00	1.00
Final Rank		9	14	15	17	10	13	5	6	4	11	7	11	8	16	3	2	1

Note: The lowest ranked algorithm is the best one

BBSSA outperformed the swarm algorithms in terms of clustering quality, although it was slower in approximating local optima than some state-of-the-art approaches. The convergence behavior of BBSSA was investigated, and it was found to obtain the best performance results quickly. Therefore, BBSSA was able to achieve high-quality clusters compared to other methods.

3) STATISTICAL SIGNIFICANCE

Statistical significance is a term used to describe the probability that an observed difference between two groups or variables is not due to chance. In other words, it indicates whether the difference will likely be a natural effect or a coincidence. To determine statistical significance, statistical tests are used. These tests provide a p-value, which measures

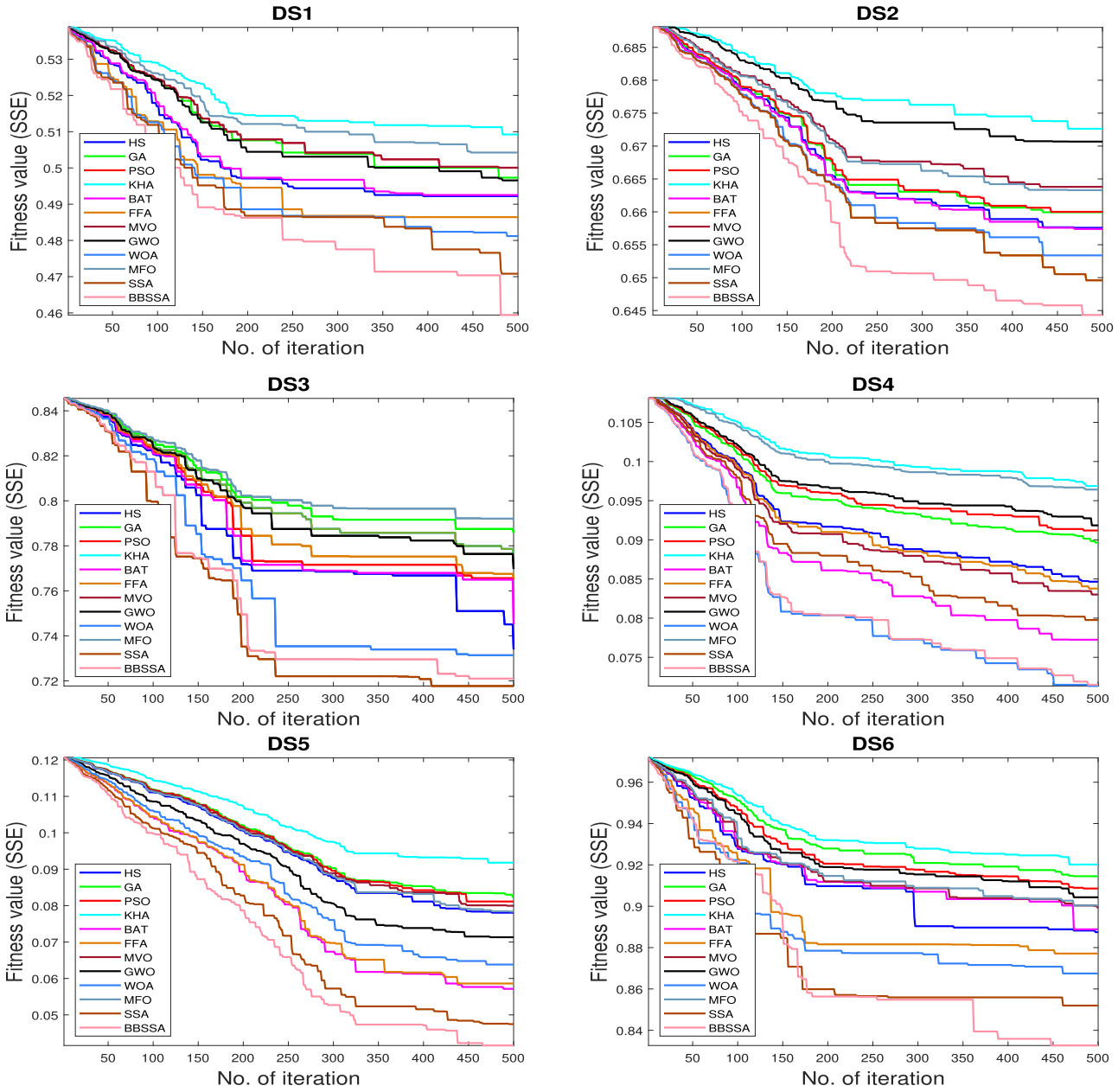

FIGURE 5. Comparison of convergence behavior for six text clustering datasets (DS1-DS6) between the proposed algorithm and other optimization techniques.

the probability of obtaining the observed results if the null hypothesis (the hypothesis that there is no difference between the groups) is accurate. A small p-value (typically less than 0.05) indicates that the observed difference is unlikely to have occurred by chance. Therefore, the null hypothesis can be rejected, and the alternative hypothesis (the hypothesis that there is a significant difference between the groups) can be accepted. To test the statistical significance of the proposed method, non-parametric Mann-Whitney-Wilcoxon tests [63] were conducted at a significance level of 5% (0.05). These tests were used to determine whether the improvements achieved by the BBSSA algorithm were statistically significant or occurred by chance. The non-parametric Mann-Whitney-Wilcoxon test is a statistical method that generates a p-value to test the null hypothesis that there is no significant difference between the median values of the two groups. In this study, the first group consisted of purity, entropy, recall, precision, F-measure, and accuracy values obtained by the proposed algorithm, while the second group consisted of the same values generated by various algorithms. The obtained p-values are displayed in Tables 8 and 9. The smallest p-value indicates that the BBSSA algorithm significantly outperformed the other algorithms. The Mann-Whitney-Wilcoxon test investigated the null hypothesis (β) and the

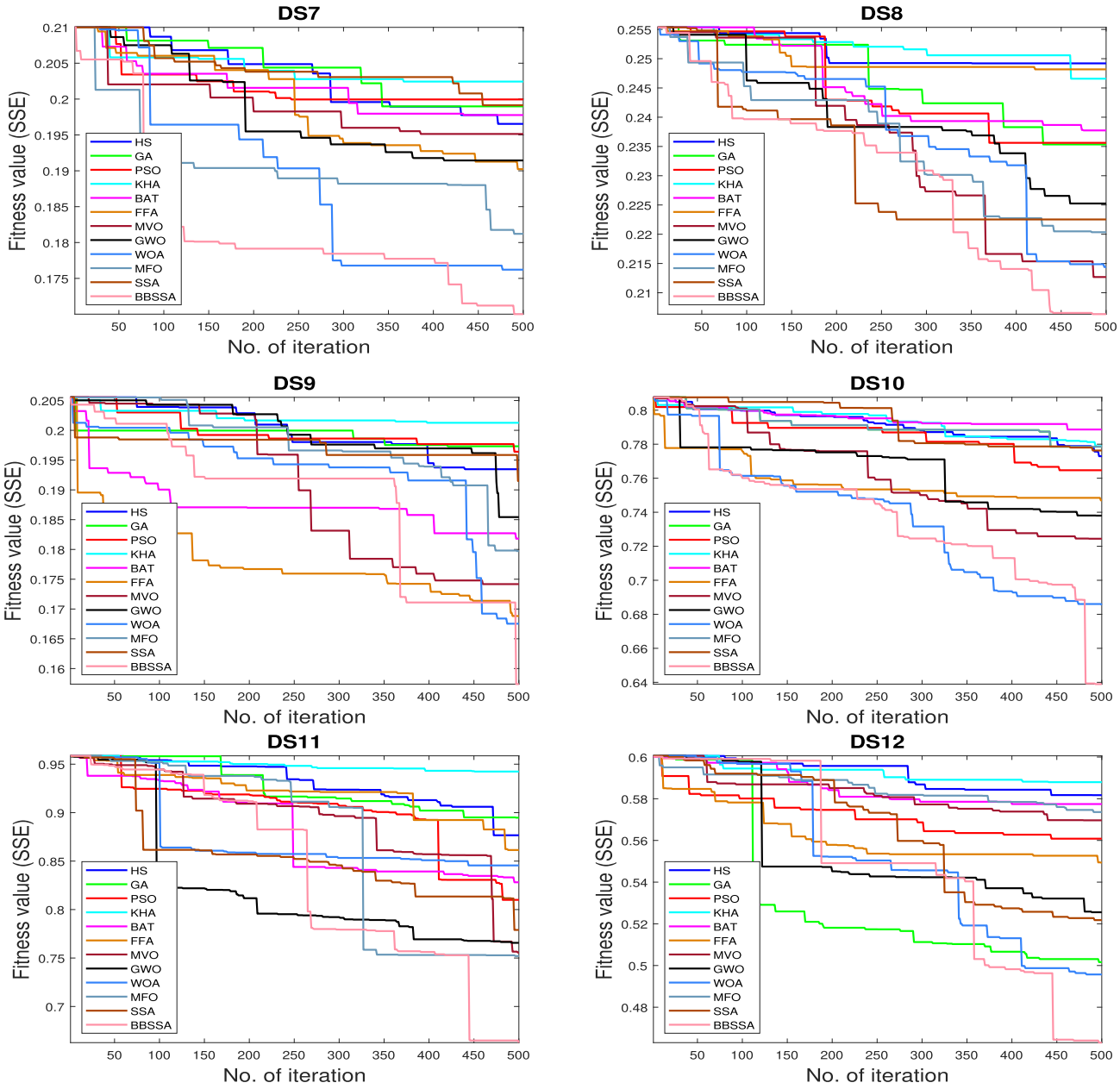

FIGURE 6. Comparison of convergence behavior for six text clustering datasets (DS7-DS12) between the proposed algorithm and other optimization techniques.

alternative hypothesis (α). The alternative hypothesis suggests a significant difference in the median values of the two groups.

E. COMPUTATION COST OF BBSSA

Time complexity measures the time an algorithm requires to solve a problem as a function of the input size. It is often expressed using big O notation, which provides an upper bound on the growth rate of the algorithm's running time relative to the input size. The time complexity of an algorithm is determined by analyzing the number of operations it performs as it processes the input data. The time complexity can be affected by various factors, including the number of inputs, the size of the input data, the number of iterations, and the efficiency of the algorithm's operations. Analyzing the time complexity of an algorithm provides insight into its efficiency and scalability. Algorithms with lower time complexity typically run faster and more efficiently than those with higher ones. Therefore, time complexity is essential when designing and evaluating algorithms, especially for large-scale applications like TDC, where performance is critical.

In this subsection, the time complexity of the BBSSA algorithm is analyzed. The bare-bones algorithm has a time complexity of $O(d \times m)$ or $O(\log d \times m)$, depending on the random value. During each iteration, the salps are

TABLE 8. Wilcoxon statistical test pvalues comparing the proposed BBSSA algorithm with GA, GWO, KHA, MFO, MVO, and PSO.

Dataset	KHA		MFO		GWO		MVO		GA		PSO	
	P Value	Results	P Value	Results	P Value	Results	P Value	Results	P Value	Results	P Value	Results
DS1	N/A	β	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	N/A	β	N/A	β	$<1\times10^{-5}$	α
DS2	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	N/A	β	N/A	β	$<1\times10^{-5}$	α
DS3	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α
DS4	0.5	β	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α
DS5	N/A	β	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α
DS6	$<1\times10^{-5}$	α	0.5	β	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α
DS7	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	N/A	β	N/A	β	0.31	β
DS8	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	N/A	β	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α
DS9	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	0.5	β	0.5	β	$<1\times10^{-5}$	α
DS10	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α
DS11	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	0.0128	β	0.0128	β	$<1\times10^{-5}$	α
DS12	0.0128	β	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α

TABLE 9. Wilcoxon statistical test pvalues comparing the proposed BBSSA algorithm with SSA, WOA, FFA, BAT, and HS.

Dataset	HS		BAT		FFA		WOA		SSA	
	P Value	Results	P Value	Results	P Value	Results	P Value	Results	P Value	Results
DS1	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α
DS2	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α
DS3	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	N/A	β	$<1\times10^{-5}$	α
DS4	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α
DS5	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α
DS6	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	β	0.5	β
DS7	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α
DS8	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α
DS9	0.12843	β	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	0.12843	β
DS10	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α
DS11	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α
DS12	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	$<1\times10^{-5}$	α	N/A	β	$<1\times10^{-5}$	α

sorted using the Sort function, which has a time complexity of $O(m \log m)$. The overall time complexity of BBSSA is influenced by two parameters: the number of salps (m) and the maximum iterations (L). Therefore, the overall time complexity of BBSSA can be expressed as follows:

$$O(BBSSA) = O(t(2(d \times m) + O(sort) + O(ObjectiveFunction) \times m)) \quad (10)$$

The equation in Eq. 10 indicates that the BBSSA algorithm may take longer to reach the optimal solution than the SSA algorithm. However, it is essential to note that the time complexity of BBSSA is calculated separately from the time complexity of the objective function in other optimization problems. Additionally, the performance of clustering algorithms, including BBSSA, can be affected by dataset characteristics. Therefore, it is crucial to consider dataset characteristics when evaluating the performance of clustering algorithms.

F. INTERPRETATION OF CLUSTERING RESULTS ON TOP 8 UAE UNIVERSITIES SCIENTIFIC PAPERS DATASETS

The visualization of BBSSA clusters in Fig. 7 relies on fingerprints. The algorithm proposed for these clusters operates on the premise that their most informative labels possess a high term frequency score. Generally, these labels have values ranging from 10 to 100, and the high peaks at a few, sometimes distinct labels offer significant insight into the nature of these clusters. Based on our analysis, we can ascertain that in 2016, the focus was on 'polymorphisms and reinforcement', while in 2017, it was about 'intersection'. In 2018, the focus shifted to 'maintenance'; in 2019, it was about 'infrastructures', and in 2020, it was about 'system', whereas in 2021, it was about 'system and wavelets'. It is worth noting that researchers can use various topic extraction methods to extract unique labels other than those depicted in the figure. Different approaches have different ways of determining cluster

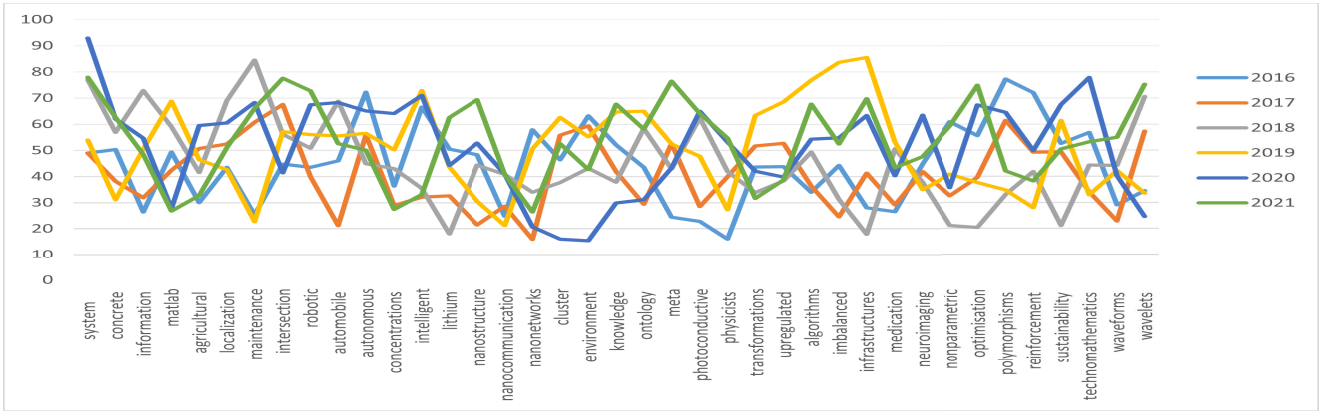

FIGURE 7. Fingerprints for topics BBSSA on top 8 UAE universities scientific papers datasets.

labels, which underscores the significance of this paper to researchers.

VI. CONCLUSION AND FUTURE DIRECTIONS

This paper introduced BBSSA, a novel population-based metaheuristic algorithm that used two different approaches to produce possible prospect solutions during the search in the TDC search space. The standard SSA system is one approach, while the bare-bones approach scheme with the Gaussian search formula is another. The direction guidance parameters for the original scheme are changed to preserve the population's diversity. At the same time, BBSSA utilized a greedy selection strategy, which theoretically provided that BBSSA predicts the global optimum solution utilizing possibility. The paper's findings can be summarized as follows:

- The BBSSA demonstrated efficiency by autonomously partitioning six text benchmark datasets and six scientific article datasets as a case study from the top eight UAE institutions.
- Several measures were used to assess the quality of the produced results: purity, entropy, precision, recall, F-measure, and accuracy. Furthermore, statistical analysis and convergence behavior were demonstrated.
- The results were compared to other state-of-the-art algorithms, such as optimization and clustering algorithms. The findings showed that the proposed method could obtain the optimal global solution for all datasets, whereas other methods were stuck at local optima.
- The findings showed that the proposed approach is suitable for automatically splitting datasets by making clusters less similar.
- We revealed that the BBSSA significantly improved the quality of six external metrics while maintaining a high convergence rate.

The current research will be expanded to include the elements listed below in the future. Firstly, the population guidance method is based on the bare-bones model; it may be compared to other guiding approaches. Second, topic extraction methods to label the cluster can be adapted to extract the topics that provide an overview of each cluster. Thirdly, an adaptive adjustment method can be used to improve the search capability of the specification $\bar{c_1}$. Finally, the introduced BBSSA algorithm will undoubtedly be used in various applications, including feature selection, multi-objective optimization, specification optimization, and constrained optimization problems.

REFERENCES

[1] T. Bezdan, C. Stoean, A. A. Naamany, N. Bacanin, T. A. Rashid, M. Zivkovic, and K. Venkatachalam, "Hybrid fruit-fly optimization algorithm with K-means for text document clustering," *Mathematics*, vol. 9, no. 16, p. 1929, Aug. 2021.

[2] W. Song, Y. Qiao, S. C. Park, and X. Qian, "A hybrid evolutionary computation approach with its application for optimizing text document clustering," *Expert Syst. Appl.*, vol. 42, no. 5, pp. 2517–2524, Apr. 2015.

[3] A. K. Abasi, A. T. Khader, M. A. Al-Betar, S. Naim, Z. A. A. Alyasseri, and S. N. Makhadmeh, "A novel hybrid multi-verse optimizer with K-means for text documents clustering," *Neural Comput. Appl.*, vol. 32, no. 23, pp. 17703–17729, Dec. 2020.

[4] V. H. A. Soares, R. J. G. B. Campello, S. Nourashrafeddin, E. Milios, and M. C. Naldi, "Combining semantic and term frequency similarities for text clustering," *Knowl. Inf. Syst.*, vol. 61, no. 3, pp. 1485–1516, Dec. 2019.

[5] A. K. Abasi, A. T. Khader, M. A. Al-Betar, S. Naim, S. N. Makhadmeh, and Z. A. A. Alyasseri, "A novel ensemble statistical topic extraction method for scientific publications based on optimization clustering," *Multimedia Tools Appl.*, vol. 80, no. 1, pp. 37–82, Jan. 2021.

[6] A. Albahr, D. Che, and M. Albahar, "A novel cluster-based approach for keyphrase extraction from MOOC video lectures," *Knowl. Inf. Syst.*, vol. 63, no. 7, pp. 1663–1686, Jul. 2021.

[7] O. M. Alyasiri, Y.-N. Cheah, and A. K. Abasi, "Hybrid filter-wrapper text feature selection technique for text classification," in *Proc. Int. Conf. Commun. Inf. Technol. (ICICT)*, Jun. 2021, pp. 80–86.

[8] A. K. Abasi, A. T. Khader, M. A. Al-Betar, S. Naim, S. N. Makhadmeh, and Z. A. A. Alyasseri, "A text feature selection technique based on binary multi-verse optimizer for text clustering," in *Proc. IEEE Jordan Int. Joint Conf. Electr. Eng. Inf. Technol. (JEEIT)*, Apr. 2019, pp. 1–6.

[9] A. K. Abasi, M. Aloqaily, and M. Guizani, "Grey wolf optimizer for reducing communication cost of federated learning," in *Proc. IEEE Global Commun. Conf.*, Dec. 2022, pp. 1049–1154.

[10] A. K. Abasi, M. Aloqaily, M. Guizani, and F. Karray, "Sine cosine algorithm for reducing communication costs of federated learning," in *Proc. IEEE Int. Medit. Conf. Commun. Netw. (MeditCom)*, Sep. 2022, pp. 55–60.

[11] A. H. Gandomi and A. H. Alavi, "Krill herd: A new bio-inspired optimization algorithm," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 17, no. 12, pp. 4831–4845, Dec. 2012.

[12] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowl.-Based Syst.*, vol. 89, pp. 228–249, Nov. 2015.

[13] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.

[14] S. Mirjalili, S. M. Mirjalili, and A. Hatamlou, "Multi-verse optimizer: A nature-inspired algorithm for global optimization," *Neural Comput. Appl.*, vol. 27, no. 2, pp. 495–513, Feb. 2016.

[15] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Mach. Learn.*, vol. 3, nos. 2–3, pp. 95–99, 1988.

[16] S. Rana, S. Jasola, and R. Kumar, "A review on particle swarm optimization algorithms and their applications to data clustering," *Artif. Intell. Rev.*, vol. 35, no. 3, pp. 211–222, Mar. 2011.

[17] O. Moh'd Alia, M. A. Al-Betar, R. Mandava, and A. T. Khader, "Data clustering using harmony search algorithm," in *Proc. Int. Conf. Swarm, Evol., Memetic Comput.* Cham, Switzerland: Springer, 2011, pp. 79–88.

[18] X. Yang and A. H. Gandomi, "Bat algorithm: A novel approach for global engineering optimization," *Eng. Comput.*, vol. 29, no. 5, pp. 464–483, Jul. 2012.

[19] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *Proc. Int. Symp. Stochastic Algorithms*. Cham, Switzerland: Springer, 2009, pp. 169–178.

[20] A. K. Abasi, S. N. Makhadmeh, M. A. Al-Betar, O. A. Alomari, M. A. Awadallah, Z. A. A. Alyasseri, I. A. Doush, A. Elnagar, E. H. Alkhammash, and M. Hadjouni, "Lemurs optimizer: A new metaheuristic algorithm for global optimization," *Appl. Sci.*, vol. 12, no. 19, p. 10057, Oct. 2022.

[21] A. K. Abasi, M. Aloqaily, and M. Guizani, "Optimization of CNN using modified honey badger algorithm for sleep apnea detection," *Expert Syst. Appl.*, vol. 229, Nov. 2023, Art. no. 120484.

[22] A. K. Abasi, M. Aloqaily, B. Ouni, and M. Hamdi, "Optimization of CNN-based federated learning for cyber-physical detection," in *Proc. IEEE 20th Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2023, pp. 1–6.

[23] A. K. Abasi, A. T. Khader, M. A. Al-Betar, S. Naim, S. N. Makhadmeh, and Z. A. A. Alyasseri, "An improved text feature selection for clustering using binary grey wolf optimizer," in *Proc. 11th Nat. Techn. Seminar Unmanned Syst. Technol.* Cham, Switzerland: Springer, 2021, pp. 503–516.

[24] O. A. Alomari, S. N. Makhadmeh, M. A. Al-Betar, Z. A. A. Alyasseri, I. A. Doush, A. K. Abasi, M. A. Awadallah, and R. A. Zitar, "Gene selection for microarray data classification based on gray wolf optimizer enhanced with TRIZ-inspired operators," *Knowl.-Based Syst.*, vol. 223, Jul. 2021, Art. no. 107034.

[25] Z. A. A. Alyasseri, A. T. Khader, M. A. Al-Betar, A. K. Abasi, and S. N. Makhadmeh, "EEG signals denoising using optimal wavelet transform hybridized with efficient metaheuristic methods," *IEEE Access*, vol. 8, pp. 10584–10605, 2020.

[26] Z. A. A. Alyasseri, A. T. Khadeer, M. A. Al-Betar, A. Abasi, S. Makhadmeh, and N. S. Ali, "The effects of EEG feature extraction using multi-wavelet decomposition for mental tasks classification," in *Proc. Int. Conf. Inf. Commun. Technol.*, Apr. 2019, pp. 139–146.

[27] Z. A. A. Alyasseri, A. K. Abasi, M. A. Al-Betar, S. N. Makhadmeh, J. P. Papa, S. Abdullah, and A. T. Khader, "EEG-based person identification using multi-verse optimizer as unsupervised clustering techniques," in *Evolutionary Data Clustering: Algorithms and Applications*, 2021, p. 89.

[28] S. N. Makhadmeh, A. T. Khader, M. A. Al-Betar, S. Naim, A. K. Abasi, and Z. A. A. Alyasseri, "Optimization methods for power scheduling problems in smart home: Survey," *Renew. Sustain. Energy Rev.*, vol. 115, Nov. 2019, Art. no. 109362.

[29] S. N. Makhadmeh, A. T. Khader, M. A. Al-Betar, S. Naim, Z. A. A. Alyasseri, and A. K. Abasi, "Particle swarm optimization algorithm for power scheduling problem using smart battery," in *Proc. IEEE Jordan Int. Joint Conf. Electr. Eng. Inf. Technol. (JEEIT)*, Apr. 2019, pp. 672–677.

[30] S. N. Makhadmeh, M. A. Al-Betar, Z. A. A. Alyasseri, A. K. Abasi, A. T. Khader, R. Damaševičius, M. A. Mohammed, and K. H. Abdulkareem, "Smart home battery for the multi-objective power scheduling problem in a smart home using grey wolf optimizer," *Electronics*, vol. 10, no. 4, p. 447, Feb. 2021.

[31] S. N. Makhadmeh, A. T. Khader, M. A. Al-Betar, S. Naim, A. K. Abasi, and Z. A. A. Alyasseri, "A novel hybrid grey wolf optimizer with min-conflict algorithm for power scheduling problem in a smart home," *Swarm Evol. Comput.*, vol. 60, Feb. 2021, Art. no. 100793.

[32] H. Peng, Y. Han, C. Deng, J. Wang, and Z. Wu, "Multi-strategy co-evolutionary differential evolution for mixed-variable optimization," *Knowl.-Based Syst.*, vol. 229, Oct. 2021, Art. no. 107366.

[33] H. Peng, W. Xiao, Y. Han, A. Jiang, Z. Xu, M. Li, and Z. Wu, "Multi-strategy firefly algorithm with selective ensemble for complex engineering optimization problems," *Appl. Soft Comput.*, vol. 120, May 2022, Art. no. 108634.

[34] H. Peng, Z. Zeng, C. Deng, and Z. Wu, "Multi-strategy serial cuckoo search algorithm for global optimization," *Knowl.-Based Syst.*, vol. 214, Feb. 2021, Art. no. 106729.

[35] A. Cobo and R. Rocha, "Document management with ant colony optimization metaheuristic: A fuzzy text clustering approach using pheromone trails," in *Soft Computing in Industrial Applications*. Cham, Switzerland: Springer, 2011, pp. 261–270.

[36] E. Hasanzadeh, "Text clustering on latent semantic indexing with particle swarm optimization (PSO) algorithm," *Int. J. Phys. Sci.*, vol. 7, no. 1, pp. 16–120, Jan. 2012.

[37] P. Nema and V. Sharma, "Multi-label text categorization based on feature optimization using ant colony optimization and relevance clustering technique," in *Proc. Int. Conf. Comput., Commun., Syst. (ICCCS)*, Nov. 2015, pp. 1–5.

[38] Y. Lu, M. Liang, Z. Ye, and L. Cao, "Improved particle swarm optimization algorithm and its application in text feature selection," *Appl. Soft Comput.*, vol. 35, pp. 629–636, Oct. 2015.

[39] K. K. Bharti and P. K. Singh, "Opposition chaotic fitness mutation based adaptive inertia weight BPSO for feature selection in text clustering," *Appl. Soft Comput.*, vol. 43, pp. 20–34, Jun. 2016.

[40] T. R. Chandran, A. V. Reddy, and B. Janet, "A social spider optimization approach for clustering text documents," in *Proc. 2nd Int. Conf. Adv. Electr., Electron., Inf., Commun. Bio-Inform. (AEEICB)*, Feb. 2016, pp. 22–26.

[41] T. R. Chandran, A. V. Reddy, and B. Janet, "Text clustering quality improvement using a hybrid social spider optimization," *Int. J. Appl. Eng. Res.*, vol. 12, no. 6, pp. 995–1008, 2017.

[42] J. Gopal, B. University, and S. Brunda, "Text clustering algorithm using fuzzy whale optimization algorithm," *Int. J. Intell. Eng. Syst.*, vol. 12, no. 2, pp. 278–286, Apr. 2019.

[43] A. K. Abasi, A. T. Khader, M. A. Al-Betar, S. Naim, S. N. Makhadmeh, and Z. A. A. Alyasseri, "Link-based multi-verse optimizer for text documents clustering," *Appl. Soft Comput.*, vol. 87, Feb. 2020, Art. no. 106002.

[44] S. K. Majhi, "Fuzzy clustering algorithm based on modified whale optimization algorithm for automobile insurance fraud detection," *Evol. Intell.*, vol. 14, no. 1, pp. 35–46, Mar. 2021.

[45] A. K. Abasi, A. T. Khader, M. A. Al-Betar, Z. A. A. Alyasseri, S. N. Makhadmeh, M. Al-Laham, and S. Naim, "A hybrid salp swarm algorithm with β-hill climbing algorithm for text documents clustering," in *Evolutionary Data Clustering: Algorithms and Applications*. Springer, 2021, p. 129.

[46] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems," *Adv. Eng. Softw.*, vol. 114, pp. 163–191, Dec. 2017.

[47] R. Salgotra, U. Singh, S. Singh, G. Singh, and N. Mittal, "Self-adaptive salp swarm algorithm for engineering optimization problems," *Appl. Math. Model.*, vol. 89, pp. 188–207, Jan. 2021.

[48] A. E. Hegazy, M. A. Makhlouf, and G. S. El-Tawel, "Improved salp swarm algorithm for feature selection," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 32, no. 3, pp. 335–344, Mar. 2020.

[49] A. A. Ewees, M. A. A. Al-qaness, and M. A. Elaziz, "Enhanced salp swarm algorithm based on firefly algorithm for unrelated parallel machine scheduling with setup times," *Appl. Math. Model.*, vol. 94, pp. 285–305, Jun. 2021.

[50] S. A. El-Sattar, S. Kamel, M. Ebeed, and F. Jurado, "An improved version of salp swarm algorithm for solving optimal power flow problem," *Soft Comput.*, vol. 25, no. 5, pp. 4027–4052, Mar. 2021.

[51] L. Duan, S. Ma, C. Aggarwal, and S. Sathe, "Improving spectral clustering with deep embedding, cluster estimation and metric learning," *Knowl. Inf. Syst.*, vol. 63, no. 3, pp. 675–694, Mar. 2021.

[52] M. A. Al-Betar, A. K. Abasi, G. Al-Naymat, K. Arshad, and S. N. Makhadmeh, "Optimization of scientific publications clustering with ensemble approach for topic extraction," *Scientometrics*, vol. 128, no. 5, pp. 2819–2877, May 2023.

[53] O. M. Alyasiri, Y.-N. Cheah, A. K. Abasi, and O. M. Al-Janabi, "Wrapper and hybrid feature selection methods using metaheuristic algorithms for English text classification: A systematic review," *IEEE Access*, vol. 10, pp. 39833–39852, 2022.

[54] C. Bouras and V. Tsogkas, "A clustering technique for news articles using WordNet," *Knowl.-Based Syst.*, vol. 36, pp. 115–128, Dec. 2012.
[55] L. P. Madin, "Aspects of jet propulsion in salps," *Can. J. Zool.*, vol. 68, no. 4, pp. 765–777, Apr. 1990.
[56] J. Kennedy, "Bare bones particle swarms," in *Proc. IEEE Swarm Intell. Symp.*, vol. 3, 2003, p. 26.
[57] I. Davidson and S. S. Ravi, "Agglomerative hierarchical clustering with constraints: Theoretical and empirical results," in *Proc. Eur. Conf. Princ. Data Mining Knowl. Discovery*. Cham, Switzerland: Springer, 2005, pp. 59–70.
[58] D. Ienco and G. Bordogna, "Fuzzy extensions of the DBSCAN clustering algorithm," *Soft Comput.*, vol. 22, no. 5, pp. 1719–1730, Mar. 2018.
[59] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. Berkeley Symp. Math. Statist. Probab.*, Oakland, CA, USA, vol. 1, no. 14, pp. 281–297, 1965.
[60] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proc. 18th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2007, pp. 1027–1035.
[61] S. Zeng, X. Tong, and N. Sang, "Study on multi-center fuzzy C-means algorithm based on transitive closure and spectral clustering," *Appl. Soft Comput.*, vol. 16, pp. 89–101, Mar. 2014.
[62] Y. Zhao and G. Karypis, "Criterion functions for document clustering: Experiments and analysis," Dept. Comput. Sci., Univ. Minnesota, Minneapolis, MN, USA, Tech. Rep. 01-040, 2001.
[63] F. Wilcoxon, "Individual comparisons by ranking methods," in *Breakthroughs in Statistics*. Cham, Switzerland: Springer, 1992, pp. 196–202.

MOHAMMED AZMI AL-BETAR (Member, IEEE) received the Ph.D. degree in artificial intelligence from Universiti Sains Malaysia (USM), in 2010. He was a Postdoctoral Research Fellow with USM, for three years, and invited as a Visiting Researcher two times. He is currently the Head of the Artificial Intelligence Research Center (AIRC), Ajman University, where he has been a full-time Faculty Member with the Master of Artificial Intelligence (MSAI) Program, since 2020. He is also the Head of the Evolutionary Computation Research Group (ECRG). He has more than 15 years of teaching experience in higher education institutions. He has taught several courses in computer science and artificial intelligence fields. He has published more than 175 scientific publications in high-quality and well-reputed journals and conferences. He ranked in Stanford's study of the world's top 2% of scientists. His main research interests include scheduling and optimization.

AMMAR KAMAL ABASI received the B.Sc. degree in computer information system from the Jordan University of Science and Technology, the M.Sc. degree in international business from The University of Jordan, and the Ph.D. degree in artificial intelligence and software engineering from Universiti Sains Malaysia. His research interests include evolutionary algorithms, nature inspired computation, artificial intelligence, and their applications to optimization problems.

GHAZI AL-NAYMAT received the Ph.D. degree from the School of Information Technologies, The University of Sydney, Australia, in May 2009. In 2015, he joined the Department of Computer Science, King Hussein School of Computing Sciences, Princess Sumaya University for Technology (PSUT). In addition, he was the Chair of the Computer Science Department, PSUT, from 2017 to 2019. He is currently an Associate Professor with the College of Engineering and Information Technology, Ajman University, United Arab Emirates. His research interests include data mining and machine learning, big data, and data science.

KAMRAN ARSHAD (Senior Member, IEEE) is currently the Dean of the Graduate Studies and Research and a Professor of electrical engineering with Ajman University, United Arab Emirates. Prior to joining Ajman University, in January 2016, he has been associated with the University of Greenwich, U.K., as a Senior Lecturer, and the Program Director of the M.Sc. Wireless Mobile Communications Systems Engineering. He is a Senior Fellow with the Higher Education Academy (SF-HEA), U.K. He has more than 130 technical peer-reviewed articles in top quality journals and international conferences. He led a number of locally and internationally funded research projects encompassing the areas of cognitive radio, LTE/LTE-advanced, 5G, optimization, and cognitive machine-to-machine (M2M) communications. He has contributed to several European and international large-scale research projects. He received the three Best Paper Awards and the one Best Research and Development Track Award. He Chaired Technical Sessions in several leading international conferences. He is an Associate Editor of the *EURASIP Journal on Wireless Communications and Networking*.

SHARIF NASER MAKHADMEH received the B.Sc. degree in computer science from Yarmouk University, Jordan, in 2013, the M.Sc. degree in information technology from Universiti Utara Malaysia (UUM), Malaysia, in 2015, and the Ph.D. degree in artificial intelligence from Universiti Sains Malaysia (USM), Malaysia, in 2020. He is currently a full-time Research Associate (FTRA) with the Artificial Intelligence Research Center (AIRC), Ajman University, United Arab Emirates. His research interests include optimization algorithms, artificial intelligence, engineering and scheduling problem, and smart home.