Automatic Clustering of DNA Sequences With Intelligent Techniques

YASMIN A. BADR 1, KHALED T. WASSIF 2, AND MAHMOUD OTHMAN 1

1 Department of Computer Science, Faculty of Computers and Information Technology, Future University in Egypt, Cairo 11835, Egypt
2 Department of Computer Science, Faculty of Computers and Artificial Intelligence, Cairo University, Giza 12613, Egypt
Corresponding author: Yasmin A. Badr (yasmin.badr@fue.edu.eg)

ABSTRACT
With the discovery of new DNAs, a fundamental problem arising is how to categorize those DNA sequences into correct species. Unfortunately, identifying all data groups correctly and assigning a set of DNAs into k clusters where k must be predefined are one of the major drawbacks in clustering analysis, especially when the data have many dimensions and the number of clusters is too large and hard to guess. Furthermore, finding a similarity measure that preserves the functionality and represents both the composition and distribution of the bases in a DNA sequence is one of the main challenges in computational biology. In this paper, a new soft computing metaheuristic framework is introduced for automatic clustering to generate the optimal cluster formation and to determine the best estimate for the number of clusters. Pulse coupled neural network (PCNN) is utilized for the calculation of DNA sequence similarity or dissimilarity. Bat algorithm is hybridized with the well-known genetic algorithm to solve the automatic data clustering problem. Extensive computational experiments are conducted on the expanded human oral microbiome database (eHOMD). A comparative study between the experimental results shows that the proposed hybrid algorithm achieved superior performance over the standard genetic algorithm and bat algorithm. Moreover, the hybrid performance was compared with competing algorithms from the literature review to ascertain its superiority. Mann-Whitney-Wilcoxon rank-sum test is conducted to statistically validate the obtained clusters.

INDEX TERMS DNA sequences, automatic clustering, pulse coupled neural network, genetic algorithm, bat algorithm.

I. INTRODUCTION
The clustering problem is an unsupervised problem, which aims at assigning similar groups together to discover unlabeled similar structures in data without any prior knowledge [1], [2]. Generally, many clustering algorithms have been developed in such a manner that objects in the same cluster should be similar to each other while objects in different clusters should be dissimilar [3], [4]. Clustering algorithms have been widely applied in solving many problems in various fields such as data analysis [5], [6], data mining [7], [8], machine learning [9], and image retrieval [10]. DNA is the code of life; it is composed of a sequence of four nucleotides. They are adenine (A), guanine (G), cytosine (C), and thymine (T). To learn an unknown DNA sequence and reveal the evolutionary information of the same gene in several species, researchers tended to seek the functions that describe it as well as comparing it with existing known DNAs based on a proximity measure either a similarity or dissimilarity following certain strategies. Recently, clustering of DNA sequences is considered an essential task in computational biology and bioinformatics. In spite of the fact that a massive amount of DNA/RNA is being sequenced, most of them have unknown structures and functions [11]. In solving this problem, traditional methods have been conducted such as k-means and its alternatives. Although the k-means has solved many clustering problems, it converges to a local minimum under certain conditions [12], and it is very sensitive to the presence of noise. Moreover, the number of clusters must be predetermined, and it is hard to provide an appropriate number of clusters that will obtain the optimal cluster formation, especially when the user must guess it where it is a tedious trial–error work. Besides, finding an optimal solution for this problem is NP-hard when the number of clusters is...
greater than 3 [13]. To overcome this inconvenience, many attempts to solve the clustering problems using metaheuristic techniques were developed. Researchers viewed the clustering problem as an optimization problem, which aims at optimizing the objective function. From a simple perspective, optimization algorithms are classified into deterministic or stochastic. If an algorithm works without any randomness, it is called deterministic. If there is a random nature in the algorithm, it is called stochastic such as genetic algorithms, particle swarm optimization (PSO) [14], firefly [15] and bat algorithm. Algorithms with stochastic nature are often referred to as metaheuristics in the recent literature [16], [17].

In this study, a new method based on pulse coupled neural network introduced by Xin Jin et al. [18] is applied to find similarity or dissimilarity of DNA sequences where DNA is transformed into a numeral sequence using four number mapping schemes representing the DNA effectively without losing any genetic information. It processes on DNAs with several sizes taking into consideration the local and global features; therefore, it is adopted. Also, a hybrid of two algorithms, genetic algorithm and bat algorithm, is introduced in this paper; it is referred to as GABAT. The proposed architecture is capable of finding a proper number of clusters and classifying the DNAs into these clusters at the same time.

The rest of this paper is organized as follows: Section II introduces the soft computing techniques utilized in this research. Section III, a review of related work is demonstrated. Section IV illustrates the proposed framework. Section V explicates the chosen data set, and it discusses the results of the conducted experiment. Finally, section VI highlights the conclusion.

II. SOFT COMPUTING TECHNIQUES
A. PULSE COUPLED NEURAL NETWORK (PCNN)
In 1993, pulse coupled neural network (PCNN) was introduced based on visual cortex theory. In this study, PCNN is utilized to get the distance between DNAs as proposed in [18]. PCNN is used in image processing where the image can either be colored or gray scaled. After n iterations applied with PCNN on the original image, a sequence of n binary (0,1) images results. The number of 1’s in each binary image is counted and a vector of numbers results. This vector is called the signature of an image. More information about PCNN is found in [19].

B. GENETIC ALGORITHM (GA)
Genetic Algorithm is a numerical randomized search and optimization algorithm that was introduced by John Holland in the early 1970s. GA emulates the principle of Darwin’s classical theory of natural evolution such as inheritance, mutation, selection, and crossover [20]. It has been widely applied in many problems (e.g., clustering, classification, etc.) to find a near-optimal solution [21]. In general, the genetic strategy starts by generating a set of strings called population. Each string is encoded randomly as a binary or floating-point string and called an individual or a chromosome. The number of iterations that a genetic algorithm run is known as generation and it may be specified by the user. In each generation, a new population emerges based on the previous one through the following steps:
1. fitness evaluation: each individual in the population is evaluated using a fitness function to measure the degree of goodness of a string where individuals with better fitness are chosen.
2. selection: apply the roulette wheel to select the parents for the reproduction phase.
3. mating: a crossover between selected individuals is applied to produce a new offspring for the next generation.
4. mutation: each individual in the offspring undergoes mutation with a predetermined percentage. It is continuously repeated till finding a satisfactory solution or a fixed number of generations have elapsed. Many clustering techniques based on genetic algorithms are studied on different data sets to find the optimal solution [22].

C. BAT ALGORITHM
Bat algorithm is a relatively recent nature-inspired metaheuristic algorithm authentically developed by Xin-She Yang in 2010. Lately, the bat algorithm caught the scientists’ interest because of their good performance in solving unimodal problems such as clustering. It was inspired by the behavior of the microbats in avoiding objects and detecting prey via echolocation. Bat algorithm idealized some rules can be summarized as follows:
1. All bats use echolocation to avoid obstacles or detect prey and they also know the difference between both.
2. Bats fly randomly with a velocity \( v_i \) at the position \( X_i \) with a varying frequency between \( f_{\text{min}} \) and \( f_{\text{max}} \). Also, they can automatically adjust the rate of pulse emission \( r \in [0, 1] \) depending on how far they reached the target and a varying loudness \( A_0 \) to detect the prey location.
3. Loudness varies from \( A_0 \) to a minimum constant value \( A_{\text{min}} \).

More information about the nature inspired bat algorithm is found in [23].

III. RELATED WORK
During the last decades, research and development in cluster analysis have been committed to assigning the data elements in heterogeneous groups through evolutionary metaheuristic techniques and their applications. However, a few attempts have been recorded to automatically determine the optimal number of clusters. Most of the studies suggest clustering techniques, which require the number of clusters must be predetermined. In 2001, Tseng and Yang [24] presented the earliest attempt for automatic clustering based on a genetic approach, namely, CLUSTERING; its algorithm consists of two stages: in the first one, the authors compute the distance between the objects based on the average of the nearest neighbor to find the connected components where objects...
close to each other are grouped. This stage aims to reduce the complexity in the second stage. The second stage is the genetic algorithm where the small clusters obtained are used as a seed for generating larger ones. The user may specify the number of generations that was equal to 100 in this experiment. The user also may specify a parameter that affects the compactness or the enlargement of the clusters. The authors considered two artificial data sets that consist of a group of points with different sizes and one real-life data set containing 20,000 spectral feature vectors derived from 40 speeches.

CLUSTERING algorithm is compared to k-means, single-link, complete link. The run time of the proposed algorithm is \( O(n^2 + G N m^2) \). The first stage of the algorithm spends \( O(n^2) \) where \( n \) is the size of the data set and the second stage complexity is \( O(N m^2) \) where \( N \) denotes the population size, \( m \) represents the chromosome length, and finally, \( G \) specifies the number of the generations the user wants to run the algorithm. A Hierarchical genetic algorithm is applied by Lai [25] for automatic clustering. In HGA, the chromosome was made up of two parts - the control genes and the parametric genes. The control genes are coded as binary strings that play a role in activating the associated parametric genes provided the value of a control gene is “1” or deactivate it otherwise “0”. The total number of “1” represents the number of clusters and the parametric genes represent the real coordinates of clusters that were randomly picked up from the data set.

The author used the Davies-Bouldin index as a fitness function which is equal to the ratio of the sum of within-cluster scatter to between-cluster separation. The generation number in this experiment was 500 while the population size was 20. The proposed approach was applied to five artificial and two real-life data sets and it showed promising results.

Maulik and Bandyopadhyay presented a genetic algorithm for clustering in [26], namely, GA-clustering. The authors attempt to search for appropriate cluster centers where \( K \) must be determined prior. Floating-point representation in chromosome design was developed where the cluster centers were randomly picked from the data set. The algorithm aims to minimize the summation of the absolute Euclidean distances between each point and their respective centers. A comparison between the proposed algorithm and k-means was applied on four artificial and three real-life data sets. Both algorithms terminated after a fixed number of iterations, which was equal to 100. The genetic algorithm did not show any undesirable output while k-means got stuck at sub-optimal solutions. Although the algorithm shows a good performance, the user will face an obvious obstacle to choose a proper number of clusters for the unlabeled data sets [27].

In 2008, Liu et al. introduced a genetic algorithm called Automatic Genetic Clustering for unknown \( K \) (AGCUK) [28]. The algorithm was able to detect the number of clusters without any prior knowledge about the correct number of clusters. The chromosomes were made up of a random number of clusters that range between \( [2, \sqrt{N}] \), where \( N \) equals the number of objects to be clustered. Each cluster represents the coordinates of an object that was randomly picked from the data set. Davies-Bouldin (DB) index was applied to evaluate the fitness of a chromosome and measure the validity of a cluster. In the selection phase, they adopted noise selection to assure that the population is not occupied by the fittest individuals. Afterwards, they applied division-absorption mutation that consists of two operations: Division operation was applied where the sparser cluster was the most possible to be partitioned by k-means. Absorption operation where the closer is two clusters to each other, the more possible for a cluster to be absorbed by another. The Algorithm termination took place after a fixed number of generations which is equal to 50. The time complexity of this algorithm is \( O(G P K m N) \) Where \( G \) equals the number of generations, \( P \) equals population size, \( K \) denotes the number of clusters, and \( m \) represents the number of object attributes.

In 2012, Satish in [29] proposed a genetic algorithm called (OCGA) that estimates the number of clusters. Firstly, hierarchical clustering is applied to the data set to obtain a dendrogram where each level represents a different number of clusters and different clusters formation. Then, a genetic algorithm is applied to determine at which \( K \) level the dendrogram is cut to obtain \( K \) clusters. The initial population is initialized to a level in the dendrogram randomly. Dunn index was used to measure the fitness of a chromosome. The proposed algorithm was applied on two different artificial data sets and it determined the number of clusters correctly. Also, the algorithm took less time for computing the optimal clusters compared to evaluating the clusters at each level in the dendrogram. The author suggests using other optimization methods and different cluster validation indices. Chehouri et al. [30] introduce a genetic algorithm that uses cluster analysis to sort the population and select the parents to apply crossover referred to as KGA. The performance of KGA was tested on a class of unconstrained optimization problems through two proposed versions using either a fixed number of clusters or an optimal clusters number. The proposed technique is able to permit the evaluation of multimodal functions. However, a future investigation is needed in solving constrained optimization problems and multi-objective formulations.

Later, Dina and her colleagues proposed metaheuristic techniques that determine the number of clusters for a variable length chromosome [31]. Genetic algorithm, artificial immune system, and a hybrid between both, namely, IGA were applied on five real-world data sets. The fitness function of each chromosome aims to minimize the sum of distances between genes in the same cluster and to maximize the sum of distances between genes in the different clusters. The authors adopted the ERX as a crossover operator. They clarified the order of the hypermutation and the crossover in the IGA algorithm to prevent the premature convergence of the population. Mann-Whitney- Wilcoxon (MWW) rank-sum test was applied and showed significant results. In recent studies, a multi-objective genetic k-means clustering algorithm was presented by Hung Nguyen in [32]; the chromosome was divided into two parts. The first part consists of a sequence
of binary numbers either 0 or 1 indicating whether the cluster is active or inactive, respectively. The second part represents the centers of the clusters. The fitness of a chromosome is evaluated by three different benchmarks including

i) Sum of squares with cluster, ii) Davies-Bouldin index, and

iii) Silhouette index. Randomly selected parents that have the same number of clusters are selected for crossover. Gaussian noise is added to the centers of the clusters for mutation. Also, the number of clusters can be changed by activating or disabling a center. K-means operator is applied to each generation according to a user-defined probability. The proposed algorithm was applied on 16 disease data sets and 5 single-cell data sets.

Komarasamy and Wahi [33] introduce a new metaheuristic method called KMBA, a hybrid between the k-means and bat algorithm. The algorithm can detect the number of clusters without any previous knowledge. The clusters were evaluated using F-measure and they are tested against traditional k-means via iris data sets. In 2013, Sood and Bansal [34] proposed a hybrid approach between k-medoid and bat algorithm that improves the quality of the clusters and deduces the centers of the clusters correctly since k-medoid is highly affected by the initially selected objects. Later, Kumar and Kaur [35] proposed three variants of the bat algorithm to solve the clustering problem, namely, BA-CN, BA-C and BA-CNE. The performance of these variants was tested on twelve benchmark data sets. BA-CNE reported the best clustering solutions compared to the variants and other clustering algorithms.

A hybrid of atom search optimization (ASO) and sine-cosine algorithm (SCA) [36] are introduced to determine the number of clusters and their centers. Results are validated on sixteen benchmark data sets showing superior performance over competing algorithms. The proposed algorithm uses SCA as a local search method to improve the convergence of ASO, which leads to finding a globally optimum solution.

Absalom and others proposed a hybrid of two algorithms firefly and particle swarm optimization [37]. The hybrid FAPSO showed significant results based on two validity indices, namely, Davies-Bouldin (DB) and compact separated (CS) validity index. Results were validated on fourteen data set and compared to existing metaheuristic algorithms such as GCUK, DE, DCPSO and ACDE. These experiments prove that the CS index is the best one in terms of the clustering solutions compactness. In addition, a comparative study shows that the FAPSO hybrid outperformed FAIWO, FAABC, and FATLBO. On the contrary, FATKBI seems to have relative equivalent performance in terms of speed and clustering solutions [38].

In 2021, a new hybrid approach based on binary differential evolution and marine predators’ algorithm was introduced to solve the automatic clustering problem referenced as DEMP [39]. The proposed method was validated on eight multi-omics data sets from the cancer genome atlas (TCGA). Also, the proposed algorithm outperformed the competing algorithms and achieved better execution time.

A recent attempt for automatic clustering was presented by behanz and her colleague in [40]. The authors utilized a binary encoding scheme with a predetermined range \([k_{\text{min}}, k_{\text{max}}]\), where \(k_{\text{min}} = 1\) and \(k_{\text{max}} = \sqrt{m}\), and \(m\) is the number of data points. The proposed methodology showed superior performance over the competing algorithms, namely, BPSO, BGA, and BDA in all binary data sets from the UCI Machine learning and KEEL repositories.

A water wave optimization algorithm was proposed by kaur and his colleague in [41] for addressing the data clustering problem. The authors declared that WWO achieved significant results for most unconstrained and constrained optimization problems. However, WWO obtained less optimal solutions with complex optimization problems. The WWO was evaluated among thirteen clustering dataset using F-score and accuracy showing better clustering results compared to other algorithms. Finally, all the aforementioned methods showed significant performance, however, suffer from one or more drawbacks. Some algorithms could not determine the number of clusters optimally while others showed high execution time. Also, it became a fundamental problem to design a computer-aided technique to categorize the newly discovered DNAs into a proper cluster or category similar in functionality according to species. The designed technique running time should be minimized compared to the other traditional techniques that perform poorly due to the dramatic increase in time to find the required number of clusters [42]. Moreover, in the past decade, notable similarity measures have been developed due to the ever-growing demands nevertheless a few ones preserve the functionality of DNA sequences.

**IV. PROPOSED SYSTEM**

The proposed methodology aims to design a soft computing technique that determines the best estimate for the number of clusters and identifies the optimum cluster formation for a variable DNA length. To tackle this problem, PCNN is utilized to generate a signature vector for each DNA. The distances between DNA signature vectors are computed. A new design for the genetic algorithm is developed to optimize the PCNN parameters; it produces better signature vectors and cluster the DNAs. Also, bat algorithm is implemented to cluster the DNA signature vectors. Subsequently, a detailed description of the proposed hybrid GABAT clustering algorithm design is discussed.

**A. PULSE COUPLED NEURAL NETWORK (PCNN)**

Since PCNN cannot work with the alphabet DNA sequences directly, the DNA sequences are transformed into a vector of numbers. The bases A, G, C, T are encoded by 1, 2, 3, 4, respectively. Then, normalization of the resultant numerical DNA sequence will take place to be compatible with PCNN as it works with grey-scale images that range from 0 to 1. The rule applied to get the normalized sequence is denoted by (1):

\[
C = \frac{I_{\text{max}} - i}{I_{\text{max}} - I_{\text{min}}}
\]
where C is the normalized value, i is the encoded value, I_{\text{max}} and I_{\text{min}} are the maximum and minimum of all encoded values. Fig. 1 illustrates an example for DNA encoding and normalization.

PCNN model is summarized by the following equations:

\[ F_{ij}(n) = V^F \sum_{kl} M_{ijkl} Y_{kl}(n-1) + e^{-\alpha^F} F_{ij}(n-1) + S_{ij} \]

\[ L_{ij}(n) = V^L \sum_{kl} W_{ijkl} Y_{kl}(n-1) + L_{ij}(n-1) e^{-\alpha^L} \]

\[ U_{ij}(n) = F_{ij}(n) \{ 1 + \beta L_{ij}(n) \} \]

\[ \theta_{ij}(n) = e^{-\alpha} \theta_{ij}(n-1) + V^F_{ij} Y_{ij}(n-1) \]

\[ Y_{ij}(n) = \begin{cases} 1, & U_{ij}(n) > \theta_{ij}(n) \\ 0, & \text{otherwise} \end{cases} \]

where the subscripts i, j refers to the neuron location in a PCNN and n is the iteration number. The receptive field of a PCNN neuron consists of two main components one for linking where \( L_{ij} \) is the linking value associated with neuron (i, j) and the other one for feeding where \( F_{ij} \) is the feeding value associated with neuron (i, j), both known as L and F channels. Both channels communicate with the neighboring neurons through the synaptic weights W and M respectively where both W and M are equal and declared as one-dimensional array where the element of the matrix is equal to the reciprocal of the distance between the central neuron and its adjacent neurons. For the neuron (i, j) and the adjacent neuron (k, l) the element of the linking matrix can be denoted by equation (7). The length of W is 8 since each codon consists of three nucleotides (bases) and interacts with the adjacent two codons.

\[ W_{ijkl} = \frac{1}{(i-k)} \]

\( Y_{kl}'s \) are the outputs of neurons from a previous iteration \([n-1]\) and external stimulus S that represents the input quaternary code. \( V^L \) and \( V^F \) are normalizing constant values. \( \alpha^F \) and \( \alpha^L \) are the decay exponentials. The internal activity of a neuron \( U_{ij} \) is denoted by equation (4), which equals the combination of \( L, F \) and linking strength \( \beta \). The internal state of a neuron is compared to a threshold value \( \theta \) where \( V^\theta_{ij} \) is a constant value and \( \infty^\theta \) is the decay coefficient. The output of each neuron is either 1 or 0 that indicates pulse or non-pulse.

### B. ENTROPY OF DNA SEQUENCES

As mentioned above, the output of the PCNN is two-dimensional image that consists of a sequence of 0’s or 1’s. The resultant image describes the features in this image. Each DNA will have its signature according to PCNN characteristics. The distance between two DNAs is calculated according to the resultant signature vector based on two methods. The first method depends on calculating the entropy of the binary sequence on each iteration; for example, if the number of iterations equals N, one-dimensional array of size N is declared where elements of the array are denoted by equation (8).

\[ E[Y(n)] = -p_1 \log_2(p_1) - p_0 \log_2(p_0) \quad (8) \]

where \( E[Y(n)] \) is the entropy of at nth iteration, \( p_1 \) is the probability of (1) in a binary image and \( p_0 \) is the probability of (0) in a binary image [43]. Then, the Euclidean distance between the resultant vectors is calculated.

\[ Eucl(X, Y) = \sqrt{\sum_{i=1}^{k} (X_i - Y_i)^2} \quad (9) \]

\[ ES(n) = \sum_{1}^{N} E[Y(n)] \quad (10) \]

Equation (10) shows how the Euclidean distance works, such that \( X, Y \) are DNA signature vectors, \( k \) is the signature length. The second method calculates the Euclidean distance between the summation of the entropies at each iteration for every DNA sequence, where \( ES(n) \) is the entropy of DNA sequences denoted by equation (9). \( N \) is the total number of iterations, and \( n \) varies from 1 to \( N \). The PCNN performs in an iterative manner till a user-defined value. This value is defined as the signature length, which was selected as an average value within the range suggested and used by researchers in [19],[44] and equals 70.

### C. CLUSTERING WITH GENETIC ALGORITHM

We propose a new chromosome design that can identify the optimal number of clusters for variable-length chromosomes without any prior knowledge. Also, it is noticed that the PCNN generated signature is highly affected by \( \alpha^L, \alpha^F, \beta, V^L, V^F, V^\theta, \alpha^\theta \). Consequently, the proposed genetic algorithm is implemented in such a way that can adjust the PCNN parameters to the optimal values. Fig. 2 shows the genetic optimization mode. The chromosome is divided into three parts shown, in abstract form, as follows:

1. The first seven genes represent the PCNN parameters where each gene corresponds to one of the PCNN parameters that will be optimized; they are \( \alpha^L, \alpha^F, \beta, V^L, V^F, V^\theta, \alpha^\theta \), respectively.
2. A sequence of random 0’s and 1’s where (1) indicates that the cluster is valid or active and (0) indicates that the cluster is invalid or disabled with a predefined length \( K \) which is equal to total number of DNAs the algorithm runs on. \( K \) indicates the maximum number of clusters.
3. The centroids corresponding to each cluster, respectively, where a centroid act as the center of a cluster.

An Example that illustrates chromosome structure for proposed design is shown in Fig. 3.
This chromosome structure allows us to evaluate solutions with a different number of clusters in the same population. Besides, the mutation operator plays an important role in changing the number of clusters by either activating or disabling clusters preventing a certain number of clusters from dominating the population. Given a set of signature DNAs \( D \), each having \( n \) resultant entropies for \( n \) PCNN iterations \( D = \{ d_1, d_2, \ldots, d_n \} \); The aim is to partition those DNAs to \( k \) clusters where \( C = \{ c_1, c_2, \ldots, c_k \} \) for \( D \). A population of potential solutions is initialized where each chromosome is initialized randomly according to the previously mentioned design. The fitness function for each chromosome is computed for only valid clusters as follows:

\[
F_t = \sum_{i=1}^{k} \sum_{j \in C_i} \sqrt{\| D_j - c_i \|^2} \tag{11}
\]

where \( i, j, k \in \mathbb{N} \), \( \| D_j - c_i \|^2 \) is the Euclidean squared distance between a DNA and center \( c_i \), \( F_t \) is the fitness value and \( k \) is the number of clusters. The fitness function aims to minimize the average Euclidean distance within clusters sum of squares. Hence, DNAs with similar traits will be clustered together. The best fitness is the minimum one over population of chromosomes. Two types of mutation are applied, one for the binary segment and the other for the floating-point segment. The normal binary flip mutation is applied to the binary segment. Non-uniform mutation [45] is adopted to mutate floating-point segments formulated as follows:

\[
\Delta (y, t) = y(1 - r^{(1 - \frac{1}{T})}) \tag{12}
\]

where \( r \) is a random number between \([0,1]\), \( t \) denotes the current iteration number. \( T \) is the maximum number of iterations (generation number) and \( b \) measures the dependency factor on iteration number. In this experiment, we used a dependency factor that ranges from \([0.5-5]\). GA uses Roulette-wheel for selection. A normal single crossover is applied. Elitist strategy is adopted for a replacement that guarantees the fittest individual will pass its traits to the next generation. Fig. 4 shows a pseudo-code for the genetic algorithm implementation. Function run Evaluations are used to evolve the population replacing the older one till max generation.
size. After reaching the stopping criteria, the number of ones generated by the genetic algorithm indicates the optimal max number of clusters.

D. CLUSTERING WITH BAT ALGORITHM

Let the given DNAs \( D = \{d_1, d_2, \ldots, d_n\} \) be classified into non-overlapping clusters \( C = \{c_1, c_2, \ldots, c_k\} \); thus, the dimension of \( d_i \) (\( i = 1, 2, \ldots, m \)) is \( b \). A centroid acting as center of a cluster is \( P = \{p_1, p_2, \ldots, p_k\} \). For a \( b\)-dimensional DNA vector, the following rules are applied:

\[
\begin{align*}
C_i \cap C_j &= \emptyset, \quad i, j = 1, 2, \ldots, k \quad (13) \\
C_1 \cup C_2 \cup \ldots \cup C_k &= D \quad (14) \\
C_i \subseteq D \text{ and } C_i \neq 0, \quad i = 1, 2, \ldots \quad (15)
\end{align*}
\]

In the proposed bat algorithm, the number of bats/solutions are initialized through the objective function where \( f(x), X = (x_1, x_2, \ldots, x_n) \). As described above, each solution is \( k \times b \)-dimensional vector, \( D_{n\times b} \), where \( X_i = (x_{i1}, x_{i2}, \ldots, x_{ib}), (x_{i21}, x_{i22}, \ldots, x_{i2b}), \ldots (x_{ik1}, x_{ik2}, \ldots, x_{ikb}) \). The goal of the clustering method is carried over the fitness function defined in equation (11). Where the smaller the value of \( f_t \), the better the compactness. Velocities and frequencies are initialized randomly. Moreover, pulse rate and loudness are defined. The user may specify the generation size. While the number of iterations is less than the specified generation size, the algorithm proceeds to cluster the DNAs according to the following steps:

1. The Euclidean distance between \( K \) initial clusters locations and signature DNAs is calculated where each DNA signature is assigned to a cluster index that indicates the nearest centroid distance.
2. The fitness of a solution is computed as the average distance between the DNA signatures and their nearest centroids.
3. The bats can iteratively update their frequencies, velocities and locations based on equations (16-18).
4. The best solution is considered to be the minimum one over the generation.

\[
\begin{align*}
fi &= f_{\min} + (f_{\max} - f_{\min})\beta \\
v_i &= v_{i-1} + (x_i - x_*)fi \\
x_i &= x_{i-1} + v_i \\
\end{align*}
\]

where \( \beta \in [0, 1] \) is a random vector drawn from a uniform distribution, and \( x_* \) is the current best global solution.

4. The best solution is considered to be the minimum one over the generation.

E. HYBRID GENETIC BAT ALGORITHM (GABAT)

The hybridization technique discussed in this study focuses on exploiting the merits of both algorithms into a single efficient framework reducing the deficiencies that individual algorithms might have. Moreover, hybridizing the bat algorithm with the well-known genetic algorithm should improve the balance between exploitation and exploration, which is a challenging task for most metaheuristic algorithms. GABAT is proposed with its main aim to improve the premature convergence of the bat algorithm towards the local minima due to the random initialization of the centroids. The GABAT starts with computing the signature vectors using PCNN, and it calculates the distances between DNAs. The population of chromosomes (solutions) are initialized randomly.

Thereafter, the fitness value of each chromosome is computed using equation (11), after which the chromosomes are updated by genetic algorithm operators. The same process is

1: Function evaluateFitness()  
2: for (i=0 to NumberOfDNAs) do  
3: begin  
4: for (i=0 to ClustersCount) do  
5: begin  
6: Calculate the Euclidean distance between each DNA and clusters centers  
7: end for  
8: end for  
9: for (k=0 to NumberOfDNAs) do  
10: begin  
11: Assign each DNA to the nearest centroid  
12: Set the nearest centroid index to each DNA  
13: end for  
14: fitness ←average nearest centroid distances  
15: return fitness  
16: end Function  
17: Function runEvaluations(P)  
18: Population_size = size(P)  
19: for (j = 1 to NumberOfDNAs) do  
20: begin  
21: Create new solutions by adjusting frequencies, velocities and solutions  
22: \( P[i] = P_{\text{min}} + (P_{\text{max}} - P_{\text{min}}) \times \beta \) //update frequency  
23: for (k = 0 to signature_dimension) do  
24: begin  
25: \( V[i, j] += P[i][k] - \text{BestSolution}[k] \times \text{Freq}[j] \) //update velocity  
26: \( \text{newSolution}[k] = P[n][d] + V[n, d] \) //update centroids  
27: end for  
28: if (Rand() > r)  
29: select a solution among BestSolution  
30: generate a local solution around BestSolution  
31: end if  
32: \( \text{newSolutionFitness} ← \text{evaluateFitness}(); \)  
33: if (newSolutionFitness < PnewFitness) \( \text{Rand}() < A \)  
34: Accept the new solution  
35: end if  
36: if (newSolutionFitness < BestSolutionFitness)  
37: BestSolution ← newSolution  
38: end if  
39: end for  
40: end Function

41: Function BatAlg()  
42: Objective function \( f(x), x = (x_1, \ldots, x_d) \);  
43: Initialize a population of \( N \) bats, \( X_i = (x_i, 1, 2, \ldots, N) \) and velocities \( V \);  
44: Define Frequency \( F_i \) at \( X_i \);  
45: Initialize pulse rates \( R_i \) and loudness \( A \);  
46: \( P_{\text{Fitness}} ← \text{evaluateFitness}(); \)  
47: BestSolution ← get the solution with minimum fitness  
48: for (gen = 1 to generation_size) do  
49: begin  
50: \( P_{\text{gen}} = \text{runEvaluations}(P_{\text{gen-1}}) \)  
51: end for  
52: end Function

FIGURE 5. Pseudo code for bat algorithm implementation.
sections are repeated iteratively till reaching a satisfactory fitness value or a maximum number of generations is reached. Subsequently, the PCNN is utilized to compute the signature vectors using the optimized parameters produced by GA. The bat algorithm uses the best solution generated by GA search results as its initial search population. Iteratively, the locations and velocities of the new solutions generated by the bat algorithm are updated through equations (16-18). Finally, the best candidate solution has the smallest fitness value. Fig. 6 illustrates the design of the proposed hybrid algorithm.

V. RESULTS AND DISCUSSION
A. DATA SET DESCRIPTION
The series of experiments were tested on the expanded Human Oral Microbiome Database (eHOMD) [46]. It provides information about bacterial species found in the human aerodigestive tract (ADT) including the nasal passages, sinuses, throat, esophagus, mouth, and lower respiratory tract. eHOMD includes a total of 775 microbial species and more than 1,000 microbial DNAs. The size of each DNA ranges between 1 10MB. A subset data set of 100 DNAs is used to conduct the tests.
TABLE 1. Parameter setting for pulse coupled neural network and classical bat algorithm.

| Parameter | PCNN Before | After | Bat Algorithm Parameter | Val |
|-----------|-------------|-------|-------------------------|-----|
| $a^i$     | 0.2         | 4.7305| Loudness $A$            | 0.5 |
| $a^g$     | 0.2         | 6.1736| Pulse rate $r$         | 0.5 |
| $\beta$   | 0.001       | 0.0072| Frequency $f_{min}$    | 0   |
| $\nu^i$   | 1           | 0.3297| Frequency $f_{max}$   | 2   |
| $\nu^g$   | 15          | 15.2614| Population          | 20  |
| $\alpha^g$| 0.54        | 12.2329| -                     | -   |
| $\nu^f$   | 0.1         | 0.4352| -                     | -   |

TABLE 2. Parameter setting for classical genetic algorithm.

| Settings | Mutation probability | Crossover probability | Population |
|----------|----------------------|-----------------------|------------|
| GA       | 0.2                  | 0.6                   | 20         |

B. SYSTEM CONFIGURATION AND PARAMETER SETTING

Tables 1 and 2 demonstrate the parameters settings in the conducted experiments.

The execution of the evolutionary algorithms was repeated ten times for the mentioned settings to assure the accuracy of the results and observe if any of the algorithms showed any unexpected behavior. The average best fitness was recorded. All tests were performed on an Intel Core i7-9750H (9th generation) CPU, under a 64-bit Windows 10 with 16 GB Ram using C# on Microsoft Visual Studio 2017 and Java on Eclipse 2018; the database was managed by MYSQL. The statistical test for the validation of the obtained GABAT results was carried out via IBM SPSS Version 27.

C. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, the experimental results are divided into two main subsections: the first section demonstrates the DNA entropy values and shows the similarity distances between DNA samples. The second section illustrates the experimental results of the evolutionary algorithms.

1) DNA ENTROPIES USING PCNN

As previously mentioned, the execution of the proposed techniques is applied to the eHOMD data set. Fig. 7 shows the resultant entropies for a sample of 50 DNAs before and after the enhancement of PCNN parameters with the help of genetic algorithm. The distance between two DNAs is either calculated using Euclidean distance of signature vectors or entropy. Both distances are computed before and after the optimization of the PCNN parameters. Similar strains evidently have closer entropy values and a high degree of similarity. Tables 3, 4, 5 and 6 show the similarity matrix between DNA samples. The Euclidean distance between the same strains such as 1054 and 1099 has the smallest similarity distance compared to the others.

2) METAHEURISTIC ALGORITHMS

The aforementioned evolutionary algorithms’ settings were tested ten times on different PCs with similar specifications, and the outputs were averaged to acquire the best results from each technique. Figures 8 and 9 show the convergence performance of the algorithms with respect to the eHOMD.
The execution of the algorithms was halted after 10000 evaluations. The obtained results indicate that GABAT yielded better fitness rather than both the standard bat and genetic algorithms.

Figures 10, 11, and 12 show the resultant fitness values with respect to the number of clusters generated using Entropy as a measure distance between DNAs.

Figures 13, 14 and 15 below show the resultant fitness values with respect to the number of clusters generated using Euclidean distance to calculate similarity between DNAs.

Figures 16 and 17 below show the distribution of eHOMD elements in the obtained clusters in respect of the minimum fitness achieved by GABAT.

Mann-Whitney-Wilcoxon (MWW) rank-sum test is conducted to statistically validate the obtained clusters. In our study, MWW is preferably used rather than a T-test due to the non-deterministic nature of the metaheuristic techniques since non-parametric tests are believed to be more resilient [47]. Table 7 shows the results of the test conducted. A p-value less than 0.05 is statistically significant. As shown in Table 7, it is evident that the bat algorithm outperformed the GA and the GABAT outperformed the bat. This

**TABLE 3.** Similarity matrix using Euclidean distance after optimization.

| ID   | GENUS          | 1006 | 1016 | 1052 | 1054 | 1099 | 1100 | 1139 | 1205 |
|------|----------------|------|------|------|------|------|------|------|------|
| 1006 | Capncytophaga  | 0.666222 | 0.279294 | 0.3801 | 0.372343 | 0.507034 | 0.375392 | 0.476771 |
| 1016 | Eikenella      | 0.666222 | 0.382503 | 0.510229 | 0.525782 | 0.421661 | 1.006397 | 0.440401 |
| 1052 | Listeria       | 0.279294 | 0.825033 | 0.403993 | 0.384974 | 0.544596 | 0.275948 | 0.510565 |
| 1054 | Yersinia       | 0.3801 | 0.510229 | 0.403993 | 0.038407 | 0.149942 | 0.601461 | 0.115284 |
| 1099 | Yersinia       | 0.372343 | 0.525782 | 0.384974 | 0.038407 | 0.149942 | 0.601461 | 0.115284 |
| 1100 | Neisseria      | 0.507034 | 0.421661 | 0.544596 | 0.149942 | 0.168408 | 0.748074 | 0.035574 |
| 1139 | Bacillus       | 0.375392 | 1.006397 | 0.275948 | 0.601461 | 0.59279 | 0.748074 | 0.714778 |
| 1205 | Escherichia    | 0.476771 | 0.440401 | 0.510565 | 0.115284 | 0.132857 | 0.035574 | 0.714778 |

**TABLE 4.** Similarity matrix using Euclidean distance before optimization.

| ID   | GENUS          | 1006 | 1016 | 1052 | 1054 | 1099 | 1100 | 1139 | 1205 |
|------|----------------|------|------|------|------|------|------|------|------|
| 1006 | Capncytophaga  | 0 | 0.695833 | 0.216164 | 0.299638 | 0.318967 | 0.48512 | 0.239791 | 0.445558 |
| 1016 | Eikenella      | 0.695833 | 0 | 0.911996 | 0.396195 | 0.376866 | 0.210713 | 0.935652 | 0.250275 |
| 1052 | Listeria       | 0.216164 | 0.911996 | 0 | 0.515802 | 0.53513 | 0.701283 | 0.023641 | 0.661721 |
| 1054 | Yersinia       | 0.299638 | 0.396195 | 0.515802 | 0 | 0.019329 | 0.185482 | 0.539427 | 0.14592 |
| 1099 | Yersinia       | 0.318967 | 0.376866 | 0.53513 | 0.019329 | 0 | 0.166153 | 0.558755 | 0.126591 |
| 1100 | Neisseria      | 0.48512 | 0.210713 | 0.701283 | 0.185482 | 0.166153 | 0 | 0.724908 | 0.039562 |
| 1139 | Bacillus       | 0.239791 | 0.935652 | 0.023641 | 0.539427 | 0.558755 | 0.724908 | 0 | 0.685346 |
| 1205 | Escherichia    | 0.445558 | 0.250275 | 0.661721 | 0.14592 | 0.126591 | 0.039562 | 0.685346 | 0 |

**TABLE 5.** Similarity matrix using entropy after optimization.

| ID   | GENUS          | Entropy | 1006 | 1016 | 1052 | 1054 | 1099 | 1100 | 1139 | 1205 |
|------|----------------|---------|------|------|------|------|------|------|------|------|
| 1006 | Capncytophaga  | 19.04075 | 0 | 0.668511 | 0.988626 | 0.239067 | 0.2341 | 0.167764 | 1.088295 | 0.177821 |
| 1016 | Eikenella      | 19.70927 | 0.668511 | 0 | 1.657137 | 0.907579 | 0.902611 | 0.836275 | 1.756807 | 0.846332 |
| 1052 | Listeria       | 18.05213 | 0.988626 | 1.657137 | 0 | 0.749559 | 0.754527 | 0.820863 | 0.099669 | 0.810805 |
| 1054 | Yersinia       | 18.80169 | 0.239067 | 0.907579 | 0.749559 | 0 | 0.004968 | 0.071304 | 0.849228 | 0.061246 |
| 1099 | Yersinia       | 18.80665 | 0.2341 | 0.902611 | 0.754527 | 0.004968 | 0 | 0.066336 | 0.854196 | 0.056279 |
| 1100 | Neisseria      | 18.87299 | 0.167764 | 0.836275 | 0.820863 | 0.071304 | 0.066336 | 0 | 0.920532 | 0.010057 |
| 1139 | Bacillus       | 17.95246 | 1.088295 | 1.756807 | 0.099669 | 0.849228 | 0.854196 | 0.920532 | 0 | 0.910474 |
| 1205 | Escherichia    | 18.86293 | 0.177821 | 0.846332 | 0.810805 | 0.061246 | 0.056279 | 0.010057 | 0.910474 | 0 |
shows that GABAT is performing relatively better than its rivals.

Table 8 show the performance of the GABAT against prementioned algorithms in the literature review to ascertain the superiority of the new hybrid. These algorithms are genetic algorithm, bat algorithm, firefly, particle swarm optimization and the hybrid FAPSO [37]. It is evident that...
GABAT achieved optimal mean and StDev. As previously mentioned, the proposed metaheuristic algorithms are applied on 100-dimension data set and the tests are halted after 10000 maximum evaluations. A stopping criterion is automatically enforced when a maximum number of fitness-function evaluations is achieved, or the optimal fitness is reached. All tests are repeated ten independent times to assure our statistical analysis. The performance measure UE represents the consumed number of fitness-function evaluations to achieve fitness F. For more explanation, it represents the minimum number of evaluations that took the algorithm to achieve such fitness. T represents the elapsed time in seconds to reach fitness F. This procedure is demonstrated as shown in Tables 9 and 10.
TABLE 8. Performance of GABAT against competing algorithms using Euclidean distance and Entropy.

| Algorithm   | Euclidean distance | Entropy |
|-------------|--------------------|---------|
|             | Mean   | StDev. | Mean   | StDev. |
| GABAT       | 0.40954 | 0.0197 | 0.012312 | 0.003918 |
| Classical GA| 0.48388 | 0.0235 | 0.102329 | 0.030316 |
| Classical Bat| 0.46766 | 0.0421 | 0.061044 | 0.007331 |
| FAPSO       | 0.45568 | 0.0364 | 0.047590 | 0.004076 |
| Classical FA| 0.46885 | 0.0459 | 0.071286 | 0.005724 |
| Classical PSO| 0.49510 | 0.0412 | 0.09418 | 0.025632 |

TABLE 9. Performance results using Entropy.

| # Run | F   | UE  | T    |
|-------|-----|-----|------|
| 1     | 0.012302 | 8376 | 5.180497 |
| 2     | 0.011927 | 5651 | 3.473747 |
| 3     | 0.011454 | 5798 | 5.50342 |
| 4     | 0.010304 | 10000 | 6.308225 |
| 5     | 0.022618 | 1603 | 1.036014 |
| 6     | 0.008383 | 10000 | 9.316141 |
| 7     | 0.008485 | 10000 | 8.094985 |
| 8     | 0.013126 | 10000 | 7.932604 |
| 9     | 0.014676 | 6315 | 6.024015 |
| 10    | 0.009842 | 10000 | 9.356377 |

TABLE 10. Performance results using Euclidean distance.

| # Run | F   | UE  | T     |
|-------|-----|-----|-------|
| 1     | 0.396668 | 1230 | 17.01665 |
| 2     | 0.421618 | 2778 | 39.06622 |
| 3     | 0.409096 | 1464 | 22.61327 |
| 4     | 0.403187 | 3339 | 52.20347 |
| 5     | 0.392321 | 1385 | 29.87602 |
| 6     | 0.461341 | 1406 | 30.87958 |
| 7     | 0.418404 | 2056 | 45.25209 |
| 8     | 0.391355 | 1465 | 22.7945 |
| 9     | 0.401147 | 2140 | 32.62871 |
| 10    | 0.400262 | 2921 | 45.20117 |

VI. CONCLUSION AND FUTURE WORK

In this study, a new soft computing technique is proposed to solve the automatic clustering problem for a variable DNA length. Despite that many clustering algorithms have been proposed, most of those algorithms require prior knowledge for the number of clusters. In this paper, pulse-coupled neural networks are utilized to measure the similarity between the DNAs of the eHOMD data set. A new design for the genetic algorithm is implemented for clustering and PCNN parameters optimization. Also, a hybrid of two algorithms, genetic algorithm and bat algorithm referred to as GABAT is implemented. The simulation results have shown that the hybrid GABAT outperformed the two state-of-the-art clustering algorithms and other metaheuristic algorithms, namely, firefly, particle swarm optimization and FAPSO. Wilcoxon test is conducted to statically validate the obtained clusters, and it showed a significant p-value of less than 5%. Finally, further research can extend to this area by exploring other clustering techniques such as fuzzy logic, neural networks, and other metaheuristic optimization techniques capable of detecting the number of clusters for a given data set. Besides, combining different metaheuristic approaches can obtain a hybrid system with efficient and enhanced capabilities.
A. Singh, S. Sharma, and J. Singh, “Nature-inspired algorithms for wireless sensor networks: A comprehensive survey,” *Comput. Sci. Rev.*, vol. 39, Feb. 2021, Art. no. 100342.

A. E. Ezugwu, “Nature-inspired metaheuristic techniques for automatic clustering: A survey and performance study,” *Social Netw. Appl. Sci.*, vol. 2, pp. 1–10, Feb. 2020.

X. Jin, D. Zhou, S. Yao, R. A. W. Q. Nie, and K. He, “Analysis of similarity/dissimilarity of dna sequences based on pulse coupled neural network,” in *Proc. Int. Workshop Multi-Disciplinary Trends Artif. Intell.*, Cham, Switzerland: Springer, 2016, pp. 279–287.

T. Lindblad, J. M. Kinser, T. Lindblad, and J. Kinser, *Image Processing Using Pulse Coupled Neural Networks*, Berlin, Germany: Springer, 2005.

S. Katoch, S. S. Chauhan, and V. Kumar, “A review on genetic algorithm: Past, present, and future,” *Multimedia Tools Appl.*, vol. 80, pp. 8091–8126, Oct. 2020.

D. A. Coley, *An Introduction to Genetic Algorithms for Scientists and Engineers*. Singapore: World Scientific, 1999.

Y. Liu, M. Ye, J. Peng, and H. Wu, “Finding the optimal number of clusters using genetic algorithms,” in *Proc. IEEE Conf. Cybern. Intell. Syst.*, Sep. 2008, pp. 1325–1330.

M. A. Elaziz, N. Nabil, A. A. Ewees, and S. Lu, “Automatic data clustering algorithms and bioinformatics,” in *Proc. Int. Conf. Soft Comput. Probl. Solving (SocProS)*, New Delhi, India: Springer, 2012, pp. 295–303.

A. Chidouri, R. Younes, J. Khoder, J. Perron, and A. Illica, “A selection process for genetic algorithm using clustering analysis,” *Algorithms*, vol. 10, no. 4, p. 123, Nov. 2017.

D. A. Moussa, N. S. Eissa, H. Abounaser, and A. Badr, “Design of novel metaheuristic techniques for clustering,” *IEEE Access*, vol. 6, pp. 77350–77358, 2018.

H. Nguyen, S. J. Louis, and T. Nguyen, “MGKA: A genetic algorithm-based clustering technique for genomic data,” in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2019, pp. 103–110.

G. Komarasyamy and A. Wahi, “An optimized K-means clustering technique using bat algorithm,” *Eur. J. Sci. Res.*, vol. 84, no. 2, pp. 263–273, 2012.

M. Sood and S. Bansal, “K-medoids clustering technique using bat algorithm,” *Int. J. Appl. Inf. Syst.*, vol. 5, no. 8, pp. 20–22, Jun. 2013.

Y. Kumar and A. Kaar, “Variants of bat algorithm for solving partitional clustering problems,” *Eng. Comput.*, vol. 37, pp. 1–27, Mar. 2021.

M. A. Elaziz, N. Nabil, A. A. Ewees, and S. Lu, “Automatic data clustering based on hybrid atom search optimization and sine-cosine algorithm,” in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2019, pp. 2315–2322.

M. B. Agbaje, A. E. Ezugwu, and R. Els, “Automatic data clustering using hybrid firefly particle swarm optimization algorithm,” *IEEE Access*, vol. 7, pp. 184963–184984, 2019.

A. E.-S. Ezugwu, M. B. Agbaje, N. Aljojo, R. Els, H. Chiroma, and M. A. Elaziz, “A comparative performance study of hybrid firefly algorithms for automatic data clustering,” *IEEE Access*, vol. 8, pp. 121089–121118, 2020.

M. Ghoneimy, H. Hassan, and E. Nabil, “A new hybrid clustering method of binary differential evolution and marine predators algorithm for multi-omics datasets,” *Int. J. Intell. Eng. Syst.*, vol. 14, no. 2, pp. 421–431, Apr. 2021.

B. Merikhi and M. R. Soleymani, “Automatic data clustering framework using nature-inspired binary optimization algorithms,” *IEEE Access*, vol. 9, pp. 93703–93722, 2021.

A. Kaur and Y. Kumar, “A new metaheuristic algorithm based on water wave optimization for data clustering,” *Evol. Intell.*, vol. 14, pp. 1–25, Jun. 2021.

S. J. Nanda and G. Panda, “A survey on nature inspired metaheuristic algorithms for clustering problems,” * Swarm Evol. Comput.*, vol. 16, pp. 1–18, Jun. 2014.

C. E. Shannon, “A mathematical theory of communication,” *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.

M. F. Tolba, M. Abdellahab, M. Aboul-Ela, and A. Samir, “Image signature improving by PCNN for Arabic sign language recognition,” *Can. J. Artif. Intell., Mach. Learn. Pattern Recognit.*, vol. 1, no. 1, pp. 1–6, 2010.

X. Zhao, “Convergent analysis on evolutionary algorithm with non-uniform mutation,” in *Proc. IEEE Congr. Evol. Comput.*, IEEE World Congr. Comput. Intell., Jun. 2008, pp. 940–944.

Y. A. Badr, *HOMD*. Access, Jan. 4, 2021. [Online]. Available: http://www.homd.org/ftp/HOMD_prokka_genomes/fna/

J. Derrac, S. García, D. Molina, and F. Herrera, “A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms,” *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011.