JOIV : Int. J. Inform. Visualization, 8(1) - March 2024 299-308

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Development of a Java Library with Bacterial Foraging Optimization for Feature Selection of High-Dimensional Data

Tessy Badriyah [a], Iwan Syarif [a], Fitriani Rohmah Hardiyanti [b]

[a] *Department of Informatics and Computer Engineering, Politeknik Elektronika Negeri Surabaya, Surabaya 60112, Indonesia*
[b] *PT. Sinergi Informatika Semen Indonesia, Gresik, 61122, Indonesia*

*Corresponding author: *tessy@pens.ac.id*

***Abstract*—High-dimensional data allows researchers to conduct comprehensive analyses. However, such data often exhibits characteristics like small sample sizes, class imbalance, and high complexity, posing challenges for classification. One approach employed to tackle high-dimensional data is feature selection. This study uses the Bacterial Foraging Optimization (BFO) algorithm for feature selection. A dedicated BFO Java library is developed to extend the capabilities of WEKA for feature selection purposes. Experimental results confirm the successful integration of BFO. The outcomes of BFO's feature selection are then compared against those of other evolutionary algorithms, namely Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), and Ant Colony Optimization (ACO). Comparison of algorithms conducted using the same datasets. The experimental results indicate that BFO effectively reduces features while maintaining consistent accuracy. In 4 out of 9 datasets, BFO outperforms other algorithms, showcasing superior processing time performance in 6 datasets. BFO is a favorable choice for selecting features in high-dimensional datasets, providing consistent accuracy and effective processing. The optimal fraction of features in the Ovarian Cancer dataset signifies that the dataset retains a minimal number of selected attributes. Consequently, the learning process gains speed due to the reduced feature set. Remarkably, accuracy substantially increased, rising from 0.868 before feature selection to 0.886 after feature selection. The classification processing time has also been significantly shortened, completing the task in just 0.3 seconds, marking a remarkable improvement from the previous 56.8 seconds.**

***Keywords*—Feature selection; high dimensional data; bacterial foraging optimization evolutionary algorithm; evolutionary algorithm.**

Manuscript received 27 Sep. 2023; revised 7 Dec. 2023; accepted 23 Feb. 2024. Date of publication 31 Mar. 2024.

I. INTRODUCTION

The high dimensionality of data can be reduced through dimensionality reduction techniques. In machine learning and statistics, dimensionality reduction refers to reducing the number of variables according to specific criteria or finding a concise representation of high-dimensional data[1]. Dimensionality reduction can be categorized into two main types: feature selection and feature extraction [1]-[4]. Feature extraction aims to minimize the resources needed to represent extensive data. In contrast, feature selection combines search techniques to obtain a new subset of features, evaluated by an evaluation measure that assesses different feature subsets.

Feature selection is a crucial preprocessing step for managing high-dimensional data to identify and retain influential features that impact classification results. This process reduces data dimensionality by removing irrelevant features, thus improving data effectiveness and accuracy [5], [6]. Feature selection can be categorized into two main approaches: filters and wrappers. Filters operate independently of the model by selecting variables based on general characteristics such as correlation with predicted variables. This method discards the least informative variables while retaining others for use in the classification or regression model. It is highly efficient in terms of computation time and resistant to overfitting. However, filter methods may include redundant variables as they overlook inter-variable relationships. Hence, they are commonly employed as preprocessing steps. On the other hand, wrappers assess subsets of potential variables based on the estimated accuracy of the target learning algorithm [7].

Feature extraction holds significant importance in machine learning endeavors. In their study, Guyon et al. [1] provide a comprehensive analysis of various feature extraction methods, drawing from papers presented at the NIPS 2003 workshop on feature extraction. Their work is a foundational reference for comprehending feature extraction techniques

and their real-world implementations. Additionally, Wang et al. [4] focus on feature selection techniques in their publication within the Encyclopedia of Machine Learning and Data Mining. They offer valuable insights into the diverse methods and approaches employed in feature selection.

One of the methods of feature selection is Bacterial Foraging Optimization (BFO). BFO is an optimization algorithm that draws inspiration from the foraging behavior of bacteria. It has been employed in various domains, including image segmentation, power management, and parameter optimization. Kumar and Vishwakarma [8] propose a multi-level crop image segmentation method utilizing BFO and minimum cross entropy, demonstrating encouraging results in image segmentation. Zhang et al. [9] concentrate on enhancing the performance of BFO by incorporating a multi-colony cooperation strategy, thereby improving the algorithm's optimization capabilities. Dubuisson et al. [10] utilize BFO for predictive control in a standalone microgrid, demonstrating its effectiveness in managing power systems. Subhashini et al. [11] utilize BFO to fine-tune parameters of an artificial neural network (ANN) through adaptive Harris Hawks weight optimization, resulting in enhanced performance of the ANN model. Meanwhile, Zhang et al. [12] propose a multi-objective BFO algorithm tailored for cognitive emergency communication networks, prioritizing optimizing practical areas as a pivotal aspect.

Several researchers have improved BFO by modifying the steps of the BFO algorithm. These enhancements were diverse, from integrating Chaotic chemotaxis step length, Gaussian mutation, and chaotic local search into BFO as demonstrated in [13] and adopting a discrete approach for community detection in networks [14]. Furthermore, modifications such as incorporating adaptive chemotaxis processes and proposing new strategies for bacteria fitness assignment and selection were introduced in [15]. Similarly, improvements such as combining gravitational search and swarm diversity strategies were made [16]. Additionally, enhancements targeted specific applications, such as using BFO in robotic cells with sequence-dependent setup times and multi-objective multi-echelon supply chain optimization problems. These adaptations were validated through rigorous testing on various benchmark problems and real-world scenarios. Moreover, efforts were made to refine BFO's exploration ability through chemotactic strategies based on Gaussian distribution and swarm diversity in reproduction strategies, as highlighted in [17]. Furthermore, BFO was combined with genetic algorithms in [18] to improve multi-objective optimization in multiple sequence alignment tasks. Lastly, advancements such as incorporating adaptive step lengths in chemotaxis were introduced in [19], yielding superior results to the original BFO algorithm. Another improvement of BFO is that a refined version of BFO, termed ChaoticBFO, integrates two chaotic strategies to strike a better balance between exploitation and exploration[20]. Validated across 23 numerical benchmark functions, this improvement was compared against ten competitive metaheuristic algorithms. In another stride forward, enriching individual diversity within BFO to prevent entrapment in local optima was proposed [21]. Additionally, the segmentation and adjustment of bacteria step sizes based on fitness values were introduced to accelerate convergence and enhance search capabilities. Furthermore, dynamic variations in search scope and chemotaxis steps were introduced, significantly accelerating convergence and improving search precision, showcasing high efficiency, rapid convergence, and a strong capability for global search [22]. Finally, an iterative process where a dimension-by-dimension update evaluation strategy combined updated values to form new solutions was adopted [23]. Experimental results illustrated the effectiveness of this strategy in improving convergence speed and solution quality within the BFO algorithm.

In addition, BFO has been implemented in several case studies such as the following research endeavors: In [24], BFO was implemented in cellular manufacturing systems (CMS), and its performance was compared with other commonly used algorithms in the literature such as GA and K-Means. Furthermore, in [25], BFO was utilized to predict protein structures, improving search quality by minimizing the free energy level of overall structure with 17 proteins. Additionally, BFO implementation in menu planning problems was observed in [26], where a mathematical model satisfying the nutritional needs of individuals while enforcing the "Laws of Nutrition" was designed. A menu generator software prototype was developed to create custom menus with different characteristics for 15 users, yielding satisfactory results from an expert's perspective.

This study employs the Bacterial Foraging Optimization (BFO) algorithm as an evolutionary technique to improve the Feature Selection process. The primary objective of incorporating this evolutionary algorithm is to boost the effectiveness of removing irrelevant features while simultaneously enhancing the speed and accuracy of information acquisition through the wrapper technique search method. The BFO algorithm chosen for feature selection was then implemented by creating a Java library as one of the feature selection capabilities in WEKA tools[27]. WEKA is an open-source tool used for data analysis and machine learning. WEKA stands for "Waikato Environment for Knowledge Analysis," noting its origin at the University of Waikato in New Zealand, where the software was developed. WEKA offers various features and algorithms for data analysis and machine learning. It provides an intuitive interactive environment that allows users to run multiple data analysis experiments without writing code from scratch. One of WEKA's valuable capabilities is feature selection. In addition to the use of the Bacterial Foraging Optimization (BFO) algorithm for feature reduction across different open datasets, this study also evaluates how BFO performs in comparison to four other widely recognized feature selection methods: Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), and Ant Colony Optimization (ACO).

II. Materials and Method

In this section, we delve into the system design, the operation of the Bacterial Foraging Optimization (BFO) algorithm, the dataset used, the performance metrics employed, and the creation of the BFO Java library within the WEKA environment.

A. System Design

The system design of this research is illustrated in Fig. 1. There are crucial processes involved in feature selection, namely attribute selection using an attribute evaluator and subset selection using the Bacterial Foraging Optimization (BFO) search method.

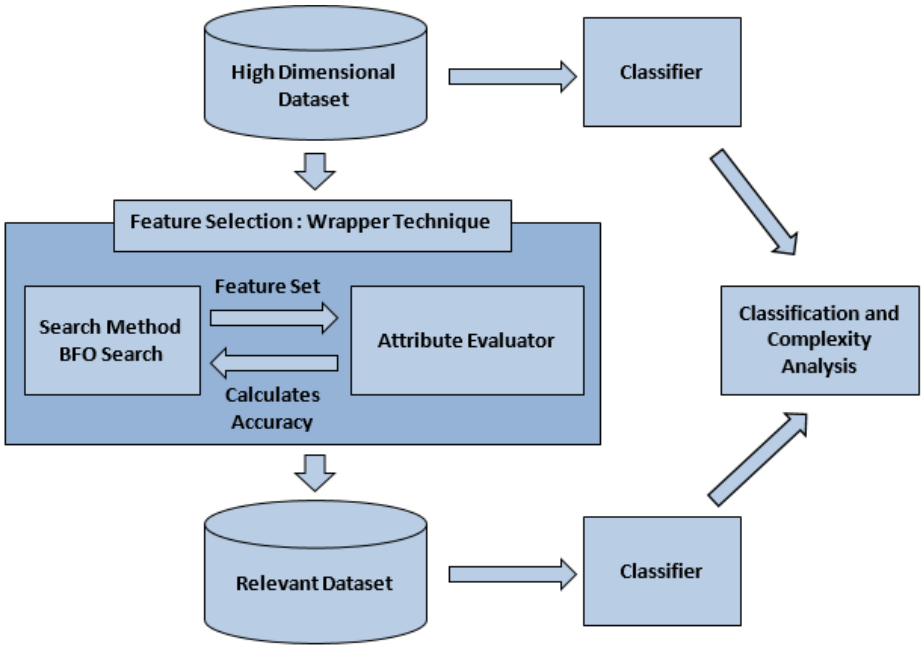

Fig. 1 System Design

The following is an explanation of the system design:

- High-dimensional datasets (with many features) will undergo classifier testing, which will later be compared with datasets processed using feature selection
- The attribute evaluator functions to provide a merit value for each subset, and subsets with high merit values are sought. In this research, the algorithm used is Correlation Based Feature Selection (CFS)
- After finding subsets with high merit values, the search method will explore and select the best subset with the top fitness value and the minimum number of features. In this research, the algorithm used is Bacterial Foraging Optimization (BFO)
- Points b and c are continuously performed until all relevant features are found and irrelevant features are removed.
- After successfully processing the dataset using feature selection, classification will be conducted using a classifier to measure the accuracy level. In this research, the algorithm used is Support Vector Machine (SVM).
- Classification and complexity analysis will be performed on the results of classification between the unprocessed and processed datasets to compare various aspects, such as precision, recall, accuracy, time classification process, and fraction of features

B. Bacterial Foraging Optimization (BFO) Algorithm

The Bacterial Foraging Optimization (BFO) algorithm was proposed by Kevin Passino [28]. Compared to optimization algorithms inspired by natural swarms, such as Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO), BFO is relatively new. The key concept is to mimic the foraging behavior of Escherichia coli bacteria to optimize multi-optimal functions. Bacteria search for nutrients to maximize energy acquisition per unit of time. Individual bacteria communicate with each other by sending signals. During the nutrient search process, bacteria move by taking small steps in their environment, a process known as chemotaxis, which is the main idea behind BFO [11]. The steps of the BFO algorithm include Chemotaxis, Swarming, Reproduction, and Elimination Dispersal.

1) Chemotaxis: This stage replicates the motion of E. coli cells by incorporating swimming and tumbling motions using flagella. Here is the formula for chemotaxis:

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \quad (1)$$

where $\theta^i(j+1,k,l)$ is he latest position of bacteria i after chemotaxis. $\theta^i(j+1,k,l)$ comprises the position of bacteria i during chemotaxis to j, reproduction to k, and elimination dispersal to l. $C(i)$ is the steps taken by bacteria i when performing tumbling or swimming. While $\Delta(i)$ is the random value between -1 to 1 for each bacteria.

2) Swarming: When placed in the center of a semi-solid matrix containing a single nutrient chemo effector, E. coli cells will aggregate into a ring-like structure as they amplify the nutrient gradient. Cells sensing elevated succinate levels release aspartate attractant, facilitating their integration into the group and movement in a concentric pattern characterized by high bacterial density.

3) Reproduction: Weaker bacteria perish over time, while each healthier bacterium undergoes a process of asexual reproduction, dividing into two new bacteria positioned in the same location. This mechanism ensures a consistent size for the bacterial colony.

4) Elimination Dispersal: A small probability is employed to eliminate a few bacteria to simulate this occurrence randomly. Simultaneously, new replacements are randomly introduced into the search space. When comparing the Bacterial Foraging Optimization (BFO) algorithm to optimization algorithms inspired by natural swarms, such as Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO), BFO is relatively new. The details of the Bacterial Foraging Optimization (BFO) algorithm are shown in the following pseudocode:

Parameters:

Initialization of parameters: $p,\ S,\ Nc,\ Ns,\ Nre,\ Ned,\ Ped,\ C(i)(i = 1,2,\dots S), \theta.$

Algorithm:

a. Elimination dispersal loop: $l = l + 1$
b. Reproduction loop: $k = k + 1$
c. Chemotaxis loop: $j = j + 1$
   1.) For $i = 1,2,\dots S$ take the Chemotaxis step for bacterium i
   2.) Calculate fitness function, $J(i,j,k,l)$

$$=> J(i,j,k,l) + J_{cc}\left(\theta^i(j,k,l), P(j,k,l)\right) \quad (2)$$

   3.) $Jlast = J(i,j,k,l)$ save values that allow us to find the best costs
   4.) *Tumble*: *generate* random vector $\Delta(i) \in R^p$ with each element $\Delta m(i),\ m = 1,2,\dots p$ being a random value between $-1 \dots 1$
   5.) Move: $\theta^i(\mathrm{j}+1,k,l)$

$$=>\ \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \quad (3)$$

   This result is used in the tumble of bacterium i
   6.) Calculate $J(i,j+1,k,l)$ and also calculate

$$=> J(i,j,k,l) + J_{cc}\begin{pmatrix}\theta^i(j+1,k,l),\\ P(j+1,k,l)\end{pmatrix} \quad (4)$$

   7.) *Swim*
   $m = 0$ (counter the length of swim)
   while $m < Ns$
      1) $m = m + 1$
      2) if $J(j+1,k,l) < Jlast$
      then $Jlast = J(i,j+1,k,l)$ and $\theta^i(j+1,k,l)$

$$=>\ \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \quad (5)$$

      And use $(\theta^i(j+1,k,l))$ to calculate a new $J(i,j+1,k,l)$
      3) Else, $m = Ns$. This is the final part of *the whole statement*.

d. If $j < Nc$, do step number 3. In this case, the continuation of bacteria's Chemotaxis does not end.
e. Reproduction
   1.) For each $i = 1,2,\dots,S$

$$J^i_{health} = \sum_{j=1}^{Nc+1} J(i,j,k,l) \quad (6)$$

   The definition of health of bacterium i is a measure of how much nutrition it obtains during its lifetime and how successful it is in avoiding toxic substances.
   2.) The Sr bacteria with the highest Jhealth value will die and the remaining Sr with the best value will be exchanged.
f. If $k < Nre$, then do step number 2.
g. Elimination – dispersal.

For $i = 1,2,\dots,S$ with the probability Ped, Elimination and dispersal of each bacterium. If a bacterium is eliminated, perform simple dispersal to a random location within the optimization domain. If $l < Ned$, do step number 1.

C. Dataset Used

The dataset used in this study utilizes open datasets TurkishTextCategorizationProject [10] and microarray datasets [11] obtained from the link: https://csse.szu.edu.cn/staff/zhuzx/Datasets.Html. A total of 9 datasets were used with the number of attributes ranging from 24,481 (highest) to 4,026 (lowest). Table I shows the list of datasets used.

TABLE I
THE DATASETS

#	Dataset Name	Instances	Attributes	Source
1	Breast Cancer	97	24481	[29]
2	Lung Cancer	181	12532	[29]
3	ALL-AML-3	72	7129	[29]
4	ALL-AML-4	72	7129	[29]
5	ALL-AML	72	7129	[29]
6	Lymphoma	66	4026	[29]
7	MLL	72	12582	[29]
8	Ovarian Cancer	253	15154	[29]
9	Zemberek-Stemmed	3600	5693	[30]

D. Performance Measurement

This study employs performance evaluation using metrics such as Accuracy, Precision, and Recall derived from the confusion matrix.

TABLE II
CONFUSION MATRIX

		Actual Values	
		TRUE	**FALSE**
Prediction values	**TRUE**	TP (True Positive):Correct result	FP (False Positive): Unexpected result
	FALSE	FN (False Negative): Missing result	TN (True Negative): Correct absence of result

The equations for computing the values of Accuracy, Precision, and Recall from the confusion matrix are provided in Table 3 below, including the time classification process and fraction of features.

TABLE III
MEASUREMENT CLASSIFICATION PERFORMANCES

Measurement	Formula
Accuracy	TP+TN/ TP+TN+FP+FN
Precision	TP / TP+FP
Recall	TP/TP+FN
Time	Time classification process
Fraction of Features	Reduced Feature / Full Feature * 100%

E. Development of Bacterial Foraging Optimization Java library on Weka

The following figure (Fig 2) is the design of the BFSearch Method implemented in WEKA.

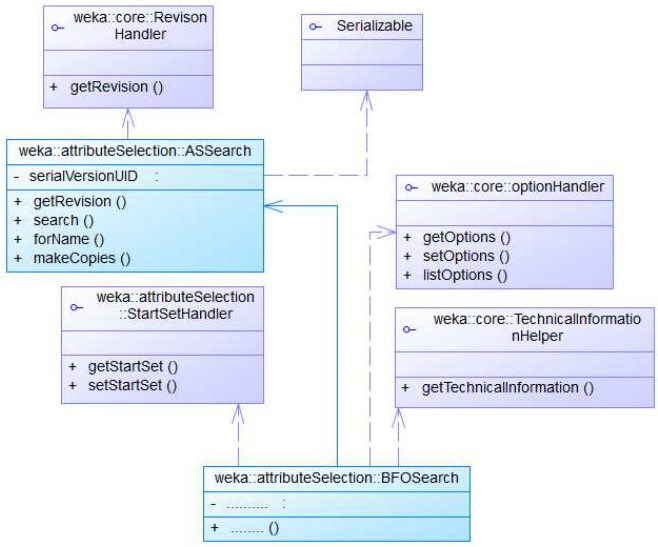

Fig. 2 BFOSearch Method Design

In Fig. 2, the main class BFOSearch implements several interfaces and methods, and there is the main class weka::attributeSelection::ASSearch which functions as the main process in Attribute Selection. In this class, the main method implemented is search(), which serves as a feature subset searcher using the BFO algorithm and returns the result as the best-performing feature subset. In addition to implementing the search() method, the weka::attributeSelection::ASSearch also extends several interfaces: StartSethandler, OptionHandler, Serializable, and TechnicalInformationHandler. The following is an explanation of each interface extended by the weka::attributeSelection::ASSearch:

1) weka::attributeSelection::StartSetHandler

This is an interface that plays a role for the search method to provide attributes. The implemented methods include setStartSet(), which is used to add a subset that the user wants to include as one of the solutions in the search process. The next method is getStartSet(), which returns the string representation of the feature subset entered by the user at the beginning of the search in the startSet parameter.

2) weka::core::optionHandler

This is an interface that provides the option setting for the implemented method. The methods included are as follows:

- getOptions(): This method returns the list of options that the user has input.
- listOptions(): It returns an object enumeration describing the options available in the search method, such as assigning values to variables.
- setOptions(): This method is used to provide a list of options to the search method being executed."

3) weka::core::TechnicalInformationHelper

This interface serves to display the publications of the author or authors of the Search Method. The implemented method is getTechnicalInformation(), which returns information about the constructed class.

4) Interface Serializable

This interface serves to perform Serialization. Serialization in Java functions as a process where the state of an object can be saved as a sequence of bytes, and vice versa.

The creation of the Library for Attribute Selection in Weka, carried out in this study, consists of two parts: Attribute Evaluator and Search Method:

1) Attribute Evaluator

The Evaluator functions to determine the merit of attribute selection. This Attribute Evaluator has superclasses, interfaces, and methods.

- Superclasses and Interfaces:
  weka.attributeSelection.ASEvaluation" is the parent class of all evaluators.
  Below are several interfaces most frequently utilized by evaluators:
  a. AttributeEvaluator: only evaluating a single attribute
  b. SubsetEvaluator: Evaluating a subset of attributes
  c. AttibuteTransformer: An evaluator that functions to transform input data.
- Methods
  Below are the methods used by the evaluator:
  a. buildEvaluator(Instances)
     This technique creates the attribute evaluator. Repeatedly employing this technique with identical data (and the same search algorithm) should yield consistent attribute selections. Additionally, this method verifies the capabilities of the data.
  b. postProcess(int[])
     It can serve as an optional step for post-processing the chosen attributes, intended to perform ranking.
  c. main(String[])
  d. Running the evaluator via the command line.

2) Search Method

The search algorithm determines heuristic exploration, encompassing methods like Exhaustive Search and Genetic Algorithm. Additionally, this study introduces a novel search approach termed BFOSearch. This Search Method has superclasses, interfaces, and methods:

- Superclasses dan Interfaces
  weka.attributeSelection.ASSearch is the parent class of all search algorithms.

  Interfaces that can be implemented and applied by search algorithms are as follows:

  a. RankedOutputSearch
     Displaying the output in the form of a list ranking of attributes.
  b. StartSetHandler:
     Search algorithms that require a start set can implement these interfaces
- Methods
  Only the following method needs to be implemented in Search Methods:

Search (ASEvaluation,Instances). This method is utilized to execute the search using the designated search algorithm. The desired search algorithm's code is inserted within this method. In this study, the search method with the BFO Java library is implemented, as explained in the previous section.

III. Results and Discussion

In the experiment, two types of tests were conducted: testing the BFO Java library on the WEKA tool, and subsequently evaluating the performance of the BFO algorithm as the one used in the feature selection process, compared to four other algorithms. Comparison of algorithms conducted using the same datasets as explained in section II.C.

A. Testing BFO Java library on WEKA's Feature Selection Capabilities

The following test is conducted to ensure that the BFO algorithm has been successfully added as one of the feature selection methods in WEKA.

The steps taken are as follows:

- In WEKA, use the Explorer menu option to perform the feature selection process, and from the available methods, select BFOSearch, which has been included previously.

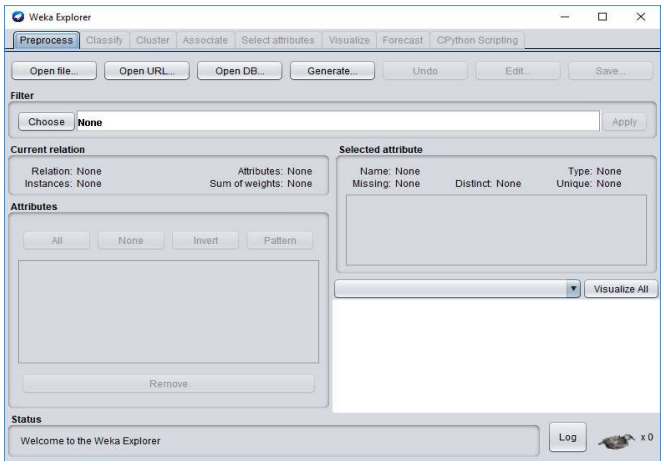

Fig. 3 WEKA Explorer

- Afterwards, the dataset to be utilized for feature selection should be chosen. This requires loading the dataset into Weka through the "Open File" button

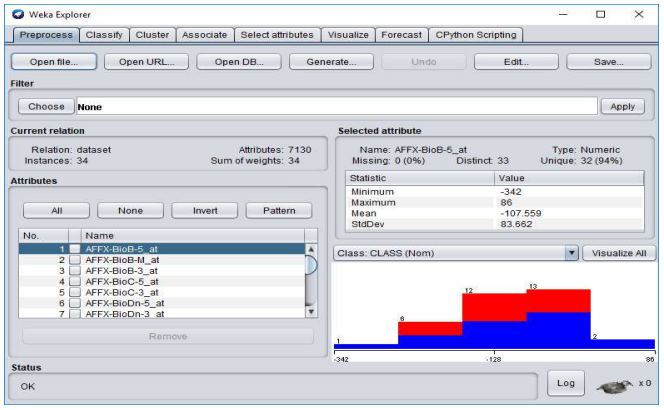

Fig. 4 Input the Dataset

- Select the Select Attributes tab, and choose BFOSearch as the Search Method

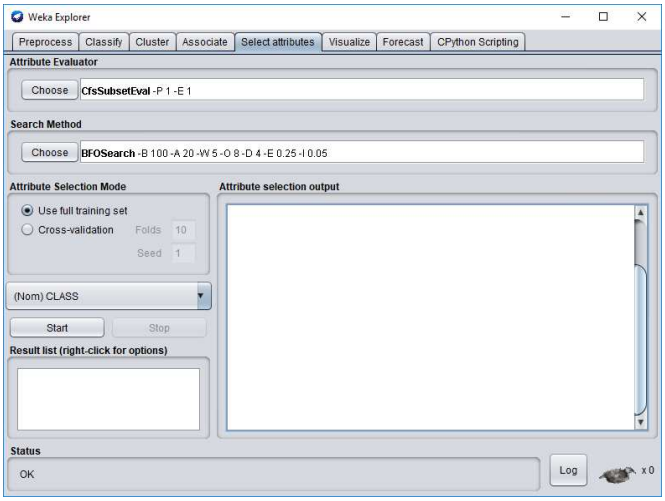

Fig. 5 Choose BFOSearch Method

- Next, specify the parameters for BFOSearch, which include: bacterial movement, chemotactic steps, colony count, dispersal, probability of dispersal, reproduction steps, swim steps, and the desired subset to be added

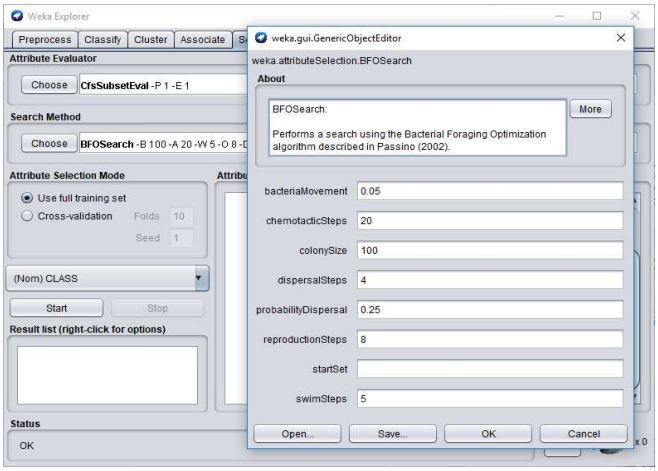

Fig. 6 Parameter BFOSearch Method

- Then, initiate the feature selection process by clicking the start button, and wait until the process is completed.

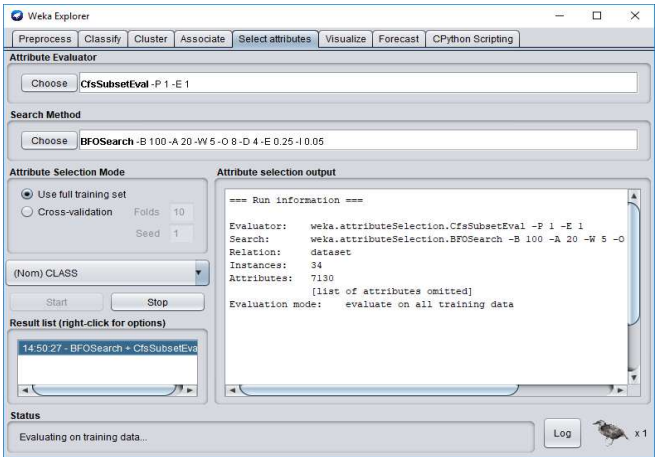

Fig. 7 Feature Selection Process

- Next, save the results of the feature selection by clicking the "Save reduced data".

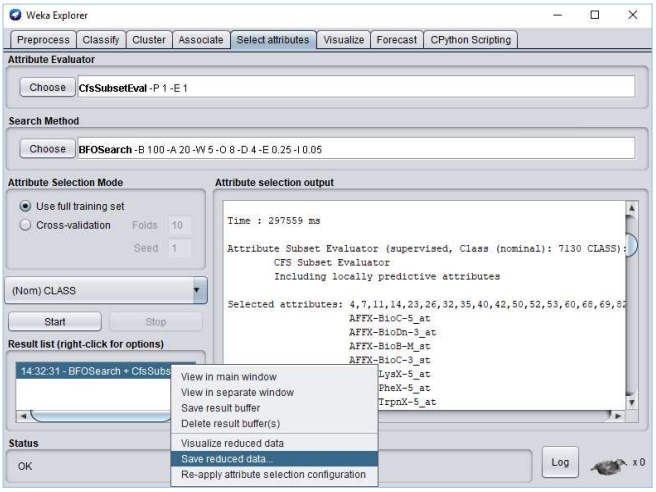

Fig. 8 Save Reduced Data

B. Evaluating the Performance of BFO in Feature Selection

In this assessment, the efficacy of BFO in the feature selection process is compared with four alternative algorithms, namely: Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), and Ant Colony Optimization (ACO). These five algorithms have different parameters in the feature selection process, which will be used as benchmarks for the calculations.

The specified parameters for each algorithm are as follows:

1) Bacterial Foraging Optimization Parameter

- bacteria movement: 0.05
- chemotactic steps: 20
- colony size: 100
- dispersal steps: 4
- dispersal probability: 0.25
- reproduction steps: 8
- swim steps: 5

2) Genetic Algorithm Parameter

- population size: 20
- max generations: 20
- crossover probability: 0.6
- mutation probability: 0.033
- Random seed: 1
- Report frequency: 20

3) Particle Swarm Optimization Parameter

- Individual weight: 0.34
- Inertial weight: 0.33
- iterations: 30
- mutation probability: 0.01
- mutation type: bit-flip
- population size: 20
- Random seed: 1
- Report frequency: 20
- Social weight: 0.33

4) Artificial Bee Colony Parameter

- food number: 30
- max foraging cycle: 30
- max trial: 6
- modification rate: 0.2
- Random seed: 1

5) Ant Colony Optimization Parameter

- chaotic coefficient: 4
- Accelerate type: normal
- Chaotic parameter type: normal
- Chaotic population type: normal
- Chaotic type: logistic map
- Evaporation: 0.9
- Heuristic: 0.7
- Iterations: 20
- Log File: dist.
- mutation probability: 0.01
- mutation type: bit-flip
- Pheromone: 2
- population size: 20
- Random seed: 1

After performing feature selection using the Bacterial Foraging Optimization (BFO) algorithm, the results of the feature selection are shown in Table IV. The performance measurement uses fraction of features. The feature fraction represents the proportion of features remaining after the selection process. A lower feature fraction indicates a smaller number of successfully selected features. Therefore, based on its definition, the smaller the value of the fraction of features, the better the results of the feature selection process.

TABLE IV
FEATURE SELECTION USING BFO

#	Dataset Name	Full Attributes	Reduced Attributes	Fraction of Features
1	Breast Cancer	24482	1323	5.4%
3	ALL-AML 3c	7130	2243	31.4%
4	ALL-AML 4c	7130	1969	27.6%
5	ALL-AML	7130	2261	31.7%
6	Lung	12601	4203	33.3%
7	Lymphoma	4027	248	6.1%
8	MLL	12583	189	1.5%
9	Ovarian Cancer	15155	34	0.2%
10	Zemberek-Stemmed	5693	725	12.7%
Average ± standard deviation				15.8% ± 0.0854

Table IV shows the results of feature selection using Bacterial Foraging Optimization, resulting in an average fraction of features of 15.8%. In Table IV, the best result of the feature selection process is obtained in the ovarian cancer dataset, as indicated by the smallest value of its fraction of features of 0.2%. Furthermore, the evaluation of accuracy changes before and after performing feature selection using Bacterial Foraging Optimization (BFO) is presented in Table V.

TABLE V
FEATURE SELECTION ACCURACY USING BFO

#	Dataset Name	Accuracy Before	Accuracy After
1	Breast Cancer	0.527	0.477
2	ALL-AML-3	0.529	0.529
3	ALL-AML-4	0.529	0.529
4	ALL-AML	0.654	0.654
5	Lung	0.685	0.685
6	Lymphoma	1	1
7	MLL	0.389	0.389
8	Ovarian Cancer	0.868	0.886
9	Zemberek-Stemmed	0.852	0.718

Table V shows the accuracy before and after feature selection using BFO (Bacterial Foraging Optimization). Based on the table, it can be analyzed that the accuracy generally remains consistent before and after feature selection, as experienced by 5 (five) datasets, namely ALL-AML-3, ALL-AML-4, Lung, Lymphoma, and MLL. There are two datasets, Breast Cancer and Zemberek-Stemmed, that experience a decrease in accuracy, while the Ovarian Cancer dataset shows an increase in accuracy. In Table 5, it can be observed that the best value of the fraction of features in the Ovarian Cancer dataset means that this dataset has the fewest selected features. As a result, the learning process will be faster with significantly fewer features, and surprisingly, the accuracy value shows a substantial improvement, increasing from 0.868 before feature selection to 0.886 after feature selection. This indicates that the use of feature selection results in attributes that are relevant to the outcome variable, leading to an increase in accuracy.

The comparison of classification process time before and after feature selection using Bacterial Foraging Optimization (BFO) is presented in Table VI.

TABLE VI
CLASSIFICATION PROCESSING TIME AFTER FEATURE SELECTION

#	Dataset Name	Time (s) Before	Time (s) After
1	Breast Cancer	26.6	1.6
2	ALL-AML-3	4.8	1.8
3	ALL-AML-4	6	1.8
4	ALL-AML	4	1.7
5	Lung	75	21
6	Lymphoma	2	0.6
7	MLL	9.6	0.3
8	Ovarian Cancer	56.8	0.3
9	Zemberek-Stemmed	104.9	17.9

Table VI shows the comparison of classification processing time before and after feature selection. It is evident that there is a significant change in the processing time in all datasets. With significantly fewer features in the Ovarian Cancer dataset, the obtained classification processing time is much shorter, taking only 0.3 seconds compared to the previous 56 seconds. The comparison of the reduced features in five algorithms is shown in Table VII.

TABLE VII
COMPARISON OF REDUCED FEATURES IN FIVE ALGORITHMS

#	Dataset Name	Full Attrib.	Reduced Attributes BFO	GA	PSO	ABC	ACO
1	Breast Cancer	24482	1323	2782	4254	8600	5007
2	ALL-AML-3	7130	2243	2146	2109	2545	2258
3	ALL-AML-4	7130	1969	971	2275	2389	2372
4	ALL-AML	7130	2261	2236	2065	2612	2361
5	Lung	12601	4203	4618	2777	4490	4708
6	Lymphoma	4027	248	1166	1279	1385	1238
7	MLL	12583	189	193	189	653	189
8	Ovarian Cancer	15155	34	3749	504	4257	1219
9	Zemberek-Stemmed	5693	725	917	997	571	1334

BFO = Bacterial Foraging Optimization
GA = Genetic Algorithm
PSO = Particle Swarm Optimization
ABC = Artificial Bee Colony
ACO = Ant Colony Optimization

Table VII shows the comparison of the number of reduced features after performing feature selection using 5 (five) algorithms, namely GA, PSO, ABC, ACO, and BFO. Based on the results, it can be analyzed that BFO performs effectively in reducing features. Out of the 9 (nine) tested datasets, BFO outperformed in 4 (four) datasets.

TABLE VIII
COMPARISON OF ACCURACY IN FIVE ALGORITHMS

#	Dataset Name	Accuracy Before	Accuracy After BFO	GA	PSO	ABC	ACO
1	Breast Cancer	0.527	0.477	0.467	0.508	0.53	0.474
2	ALL-AML-3	0.529	0.529	0.529	0.529	0.529	0.529
3	ALL-AML-4	0.529	0.529	0.529	0.529	0.529	0.529
4	ALL-AML	0.654	0.654	0.654	0.654	0.654	0.654
5	Lung	0.685	0.685	0.685	0.685	0.685	0.685
6	Lymphoma	1	1	1	1	1	1
7	MLL	0.389	0.389	0.389	0.389	0.389	0.389
8	Ovarian Cancer	0.868	0.886	0.923	0.907	0.9	0.896
9	Zemberek-Stemmed	0.852	0.718	0.73	0.76	0.706	0.782

BFO = Bacterial Foraging Optimization
GA = Genetic Algorithm
PSO = Particle Swarm Optimization
ABC = Artificial Bee Colony
ACO = Ant Colony Optimization

Table VIII shows a comparison of accuracy after the feature selection process using the Reduced Dataset with 5 different algorithms. Based on the results, it can be observed that the accuracy before and after feature selection remained unchanged in six datasets, namely ALL-AML-3, ALL-AML-

4, ALL-AML, Lung, Lymphoma, and MLL. However, in the breast cancer and Zemberek-Stemmed datasets, there was a slight decrease in accuracy, although not significantly. On the other hand, the Ovarian Cancer dataset exhibited an increase in accuracy for all algorithms, with the highest accuracy achieved by the GA algorithm at 0.923. However, this accuracy value is not considerably different from BFO, which had an accuracy of 0.886 after the feature selection process. With the best fraction of feature value at 0.2% in the Ovarian Cancer dataset, BFO managed to significantly reduce the classification processing time from 56.8 seconds to just 0.3 seconds, as shown in Table IX.

TABLE IX
COMPARISON OF CLASSIFICATION PROCESSING TIME IN FIVE ALGORITHMS

#	Dataset Name	Time (s) Before	Time (s) After Feature Selection				
			BFO	GA	PSO	ABC	ACO
1	Breast Cancer	26.6	1.6	3.7	4.1	8.5	4.8
2	ALL-AML 3c	4.8	1.8	2	1.8	2.6	2.1
3	ALL-AML 4c	6	1.8	1.3	2.1	2.2	2.4
4	ALL-AML	4	1.7	1.8	1.7	2.2	2.2
5	Lung	75	21	26.1	16	24.3	24.3
6	Lymphoma	2	0.6	1.1	1.1	1.3	0.6
7	MLL	9.6	0.3	0.9	0.3	0.6	0.9
8	Ovarian	56.8	0.3	11.4	1.9	13	3.5
9	Zemberek-Stemmed	104.9	17.9	56.4	23.6	16	27.7

BFO = Bacterial Foraging Optimization
GA = Genetic Algorithm
PSO = Particle Swarm Optimization
ABC = Artificial Bee Colony
ACO = Ant Colony Optimization

Table IX shows the comparison of the time in seconds required to perform the classification process on the dataset before and after feature selection using 5 algorithms. In most of the feature-selected datasets using BFO, the processing time is superior compared to the others, specifically in 6 datasets, as BFO has the greatest reduction in features and the smallest fraction of features, as indicated by Tables VII. This suggests that feature selection significantly reduces classification time, particularly noticeable in datasets with a high volume of instances, like the Zemberrek Stemmed dataset, which contains 3600 instances. The time required for classifying these datasets is almost 104 seconds or equivalent to 1 minute and 44 seconds. However, when feature selection is performed using BFO, the classification only takes 17.9 seconds. From Table IX, it can be seen that BFO achieved the fastest processing time in most datasets, 6 out of 9 datasets.

IV. CONCLUSION

This study aims to develop the BFO (Bacterial Foraging Optimization) Java library to extend WEKA's capabilities in Feature Selection and compare its performance with four other feature selection algorithms that use Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), and Ant Colony Optimization (ACO). From testing results, BFO algorithm has been successfully added as one of the feature selection methods in WEKA. The results obtained after conducting experiments using 9 (nine) high-dimensional data sets show that BFO successfully reduces the least number of features and has the fastest classification time compared to other algorithms.

From the experiments results, shows the accuracy before and after feature selection using BFO (Bacterial Foraging Optimization) remains consistent before and after feature selection therefore we can conclude that the Bacterial Foraging Optimization (BFO) algorithm is effective in feature selection. With significantly fewer features in the Ovarian Cancer dataset, the obtained classification processing time is much shorter, taking only 0.3 seconds compared to the previous 56.8 seconds.

When compared to other evolutionary algorithms such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), and Ant Colony Optimization (ACO), BFO performs effectively in reducing features. Out of the 9 tested datasets, BFO outperformed in 4 datasets. In most of the feature-selected datasets using BFO, the processing time is superior compared to the others, specifically in 6 datasets, as BFO has the greatest reduction in features and the smallest fraction of features. Additionally, the comparison of classification processing time among the five algorithms shows that BFO performs competitively in terms of time efficiency. Overall, the experiments demonstrate that BFO emerges as a promising choice for feature selection in datasets with high dimensions, providing stable accuracy and efficient processing time.

For further development, improvements can be made to the existing processes in Bacterial Foraging Optimization, specifically by enhancing one of the stages to achieve better feature selection, which involves modifying the chemotaxis process into an adaptive one.

REFERENCES

[1] I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, *Feature extraction. Foundations and applications. Papers from NIPS 2003 workshop on feature extraction, Whistler, BC, Canada, December 11–13, 2003. With CD-ROM*. 2006.

[2] I. Guyon, J. Li, T. Mader, P. A. Pletscher, G. Schneider, and M. Uhr, "Competitive baseline methods set new standards for the NIPS 2003 feature selection benchmark," *Pattern Recognition Letters,* vol. 28, no. 12, pp. 1438-1444, 2007/09/01/ 2007, doi:10.1016/j.patrec.2007.02.014.

[3] A. Jain and D. Zongker, "Feature selection: evaluation, application, and small sample performance," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 19, no. 2, pp. 153-158, 1997, doi: 10.1109/34.574797.

[4] M. Kudo and J. Sklansky, "Comparison of algorithms that select features for pattern classifiers," *Pattern Recognition,* vol. 33, no. 1, pp. 25-41, 2000/01/01/ 2000, doi:10.1016/S0031-3203(99)00041-2.

[5] L. Ke, Z. Feng, and Z. Ren, "An efficient ant colony optimization approach to attribute reduction in rough set theory," *Pattern Recognition Letters,* vol. 29, no. 9, pp. 1351-1357, 2008/07/01/ 2008, doi:10.1016/j.patrec.2008.02.006.

[6] D. Mittal and M. Bala, "Hybrid feature selection approach using bacterial foraging algorithm guided by Naive Bayes classification," in *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 3-5 July 2017 2017, pp. 1-7, doi: 10.1109/ICCCNT.2017.8204178.

[7] B. Kumari and T. Swarnkar, "Filter versus wrapper feature subset selection in large dimensionality micro array: A review," *International Journal of Computer Science and Information Technologies,* vol. 2, pp. 1048-1053, 01/01 2011.

[8] A. Kumar and A. K. Vishwakarma, "Multilevel Crop Image Segmentation using Bacterial Foraging Optimization Based on Minimum Cross Entropy," in *2021 International Conference on Control, Automation, Power and Signal Processing (CAPS)*, 10-12 Dec. 2021 2021, pp. 1-6, doi: 10.1109/CAPS52117.2021.9730680.

[9] C. Zhang, J. Yu, and B. Niu, "Bacterial Foraging Optimization Based on Multi-colony Cooperation Strategy," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1-4 Dec. 2020 2020, pp. 1543-1548, doi: 10.1109/SSCI47803.2020.9308213.

[10] F. Dubuisson, A. Chandra, M. Rezkallah, and H. Ibrahim, "A Bacterial Foraging Optimization Technique and Predictive Control Approach for Power Management in a Standalone Microgrid," in *2020 IEEE Electric Power and Energy Conference (EPEC)*, 9-10 Nov. 2020 2020, pp. 1-7, doi: 10.1109/EPEC48502.2020.9320038.

[11] P. P. S. Subhashini, M. S. S. Ram, and D. S. Rao, "Bacterial Foraging Optimized Parameters for ANN using Adaptive Harris Hawks Weight Optimization," in *2021 6th International Conference on Inventive Computation Technologies (ICICT)*, 20-22 Jan. 2021 2021, pp. 849-854, doi: 10.1109/ICICT50816.2021.9358701.

[12] S. Zhang, X. Ji, L. Guo, and Z. Bao, "Multi-objective bacterial foraging optimization algorithm based on effective area in cognitive emergency communication networks," *China Communications,* vol. 18, no. 12, pp. 252-269, 2021, doi: 10.23919/JCC.2021.12.016.

[13] H. Chen, Q. Zhang, J. Luo, Y. Xu, and X. Zhang, "An enhanced Bacterial Foraging Optimization and its application for training kernel extreme learning machine," *Applied Soft Computing,* vol. 86, p. 105884, 2020/01/01/ 2020, doi:10.1016/j.asoc.2019.105884.

[14] B. Yang, X. Huang, W. Cheng, T. Huang, and X. Li, "Discrete bacterial foraging optimization for community detection in networks," *Future Generation Computer Systems,* vol. 128, pp. 192-204, 2022/03/01/ 2022, doi:10.1016/j.future.2021.10.015.

[15] M. Kaur and S. Kadam, "A novel multi-objective bacteria foraging optimization algorithm (MOBFOA) for multi-objective scheduling," *Applied Soft Computing,* vol. 66, pp. 183-195, 2018/05/01/ 2018, doi:10.1016/j.asoc.2018.02.011.

[16] W. Zhao and L. Wang, "An effective bacterial foraging optimizer for global optimization," *Information Sciences,* vol. 329, pp. 719-735, 2016/02/01/ 2016, doi:10.1016/j.ins.2015.10.001.

[17] L. Wang, W. Zhao, Y. Tian, and G. Pan, "A bare bones bacterial foraging optimization algorithm," *Cognitive Systems Research,* vol. 52, pp. 301-311, 2018/12/01/ 2018, doi:10.1016/j.cogsys.2018.07.022.

[18] P. Manikandan and D. Ramyachitra, "Bacterial Foraging Optimization –Genetic Algorithm for Multiple Sequence Alignment with Multi-Objectives," *Scientific Reports,* vol. 7, no. 1, p. 8833, 2017/08/18 2017, doi:10.1038/s41598-017-09499-1.

[19] X. Yan, Y. Zhu, H. Zhang, H. Chen, and B. Niu, "An Adaptive Bacterial Foraging Optimization Algorithm with Lifecycle and Social Learning," *Discrete Dynamics in Nature and Society,* vol. 2012, p. 409478, 2012/11/14 2012, doi: 10.1155/2012/409478.

[20] Q. Zhang, H. Chen, J. Luo, Y. Xu, C. Wu, and C. Li, "Chaos Enhanced Bacterial Foraging Optimization for Global Optimization," *IEEE Access,* vol. 6, pp. 64905-64919, 2018, doi:10.1109/access.2018.2876996.

[21] J. Jiang, X. Xiong, Y. Ou, and H. Wang, "An Improved Bacterial Foraging Optimization with Differential and Poisson Distribution Strategy and its Application to Nurse Scheduling Problem," in *Advances in Swarm Intelligence*, vol. 12145: Springer Nature Switzerland AG 2020., 2020, pp. 312-24.

[22] Y. Chen and W. Lin, "An improved bacterial foraging optimization," in *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 19-23 Dec. 2009 2009, pp. 2057-2062, doi:10.1109/robio.2009.5420524.

[23] M. He, J. Chen, and H. Deng, "Bacterial Foraging Optimization Algorithm with Dimension by Dimension Improvement," in *2019 4th International Conference on Computational Intelligence and Applications (ICCIA)*, 21-23 June 2019 2019, pp. 1-5, doi:10.1109/ICCIA.2019.00008.

[24] H. Nouri and T. S. Hong, "Development of bacteria foraging optimization algorithm for cell formation in cellular manufacturing system considering cell load variations," *Journal of Manufacturing Systems,* vol. 32, no. 1, pp. 20-31, 2013/01/01/ 2013, doi:10.1016/j.jmsy.2012.07.014.

[25] D. Ramyachitra and V. Veeralakshmi, "Bacterial Foraging Optimization for protein structure prediction using FCC & HP energy model," *Gene Reports,* vol. 7, pp. 43-49, 2017/06/01/ 2017, doi:10.1016/j.genrep.2017.01.005.

[26] B. Hernández-Ocaña, O. Chávez-bosquez, J. HernáNdez-Torruco, J. Canul-Reich, and P. Pozos-Parra, "Bacterial Foraging Optimization Algorithm for Menu Planning," *IEEE Access,* vol. 6, pp. 8619-8629, 2018, doi: 10.1109/access.2018.2794198.

[27] G. Holmes, A. Donkin, and I. H. Witten, "WEKA: a machine learning workbench," in "Computer Science Working Papers," Working Paper 1994. [Online]. Available: https://hdl.handle.net/10289/1138

[28] M. P. Kevin, "Bacterial Foraging Optimization," *International Journal of Swarm Intelligence Research (IJSIR),* vol. 1, no. 1, pp. 1-16, 2010.

[29] Z. Zhu, Y.-S. Ong, and M. Dash, "Markov blanket-embedded genetic algorithm for gene selection," *Pattern Recognition,* vol. 40, no. 11, pp. 3236-3248, 2007/11/01/ 2007, doi:10.1016/j.patcog.2007.02.007.

[30] "TurkishTextCategorizationProject - Browse /4. Zemberek-Stemmed at SourceForge.net."