An Efficient Moth Flame Optimization Algorithm using Chaotic Maps for Feature Selection in the Medical Applications

Ruba Abu Khurma, Ibrahim Aljarah and Ahmad Sharieh

King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan

Keywords: Moth Flame Optimization Algorithm (MFO), Dimensionality Problem, Classification, Optimization, Feature Selection (FS), Chaotic Maps.

Abstract: In this paper, multiple variants of the Binary Moth Flame Optimization Algorithm (BMFO) based on chaotic maps are introduced and compared as search strategies in a wrapper feature selection framework. The main purpose of using chaotic maps is to enhance the initialization process of solutions in order to help the optimizer alleviate the local minima and globally converge towards the optimal solution. The proposed approaches are applied for the first time on FS problems. Dimensionality is a major problem that adversely impacts the learning process due to data-overfit and long learning time. Feature selection (FS) is a preprocessing stage in a data mining process to reduce the dimensionality of the dataset by eliminating the redundant and irrelevant noisy features. FS is formulated as an optimization problem. Thus, metaheuristic algorithms have been proposed to find promising near optimal solutions for this complex problem. MFO is one of the recent metaheuristic algorithms which has been efficiently used to solve various optimization problems in a wide range of applications. The proposed approaches have been tested on 23 medical datasets. The comparative results revealed that the chaotic BMFO (CBMFO) significantly increased the performance of the MFO algorithm and achieved competitive results when compared with other state-of-the-arts metaheuristic algorithms.

1 INTRODUCTION

In recent years, due to advances in data collection methods, vast amounts of data have been stored in data repositories. This is negatively reflected in the size of datasets either by increasing the number of instances and/or increasing the number of features.

Curse of dimensionality is a challenging problem that causes many negative consequences for the datamining tasks (i.e classification, clustering) (Khurma et al., 2020). It implies the existence of some features that are unuseful for the learning process such as the redundant and irrelevant features. Redundant features do not add any new information to the learning process because they can be inferred from other features. On the other hand, irrelevant features are unrelated to the target class. These noisy features may mislead the learning algorithm when they are used in building the learning model. Furthermore, they adversely affect the learner’s performance and generate poor quality models due to data-overfit. Increasing dimensionality also consumes more learning time and increases the demand for specialized hardware resources.

Feature selection (FS) is a primary preprocessing stage in a datamining process that has two conflicting objectives: producing a smaller version of the dataset by minimizing it’s dimensions and simultaneously maximizing the learning performance (Faris et al., 2019; Al-Madi et al., 2018). This is accomplished by eliminating the noisy features (redundant or/and irrelevant) from original dataset without causing any loss of information. Formally speaking, for a dataset of \( N \) features, FS process selects \( n \) features from the original \( N \) features where \( n \leq N \) without causing any degradation in the learner’s performance. FS has the advantage that there is no generation of new feature combinations so the original meaning of the features is preserved. This is crucial for some fields which cares about the readability of a dataset such as bioinformatics and medicine.

The FS comprises four basic stages: subset generation, subset evaluation, checking a stopping criterion and validation stage (Dash and Liu, 1997). Subset generation is performed by a specific search technique (complete, heuristic) to generate candidate feature subsets. Subset evaluation determines the quality of a generated feature subset using a particular tech-
nique (filter or wrapper). FS is repeated until a specified condition is met (i.e. maximum number of iterations). The last stage is to validate the feature subset by comparing it with the domain knowledge gathered from experts.

With regard to the evaluation approaches. Filters are considered rank based methods because they rely on a predefined threshold to evaluate a feature. They don’t involve any learning algorithm but they use the intrinsic characteristics of the features. The absence of the learning process makes filters more time efficient. On the other hand, wrappers consider a learning algorithm to decide the quality of a feature subset. This contributes to better performance results but consumes more computational time.

FS search methods play a key role in controlling the complexity of the FS process. Brute force methods create the feature space by generating all the possible feature subsets from the original features set. Formally speaking, for \( n \) features, there are \( 2^n \) feature subsets that can be generated. FS process that involves a complete search procedure needs an exponential running time to exhaustively traverse all the generated feature subsets. This is computationally expensive and impractical with medium and large datasets making the FS an NP-Hard problem.

Metaheuristic algorithms are stochastic search methods that effectively generate promising solutions (near optimal) in a less time effort. Metaheuristic algorithms include a population based algorithms that initialize multiple solutions during the optimization process and update them in each iteration until the global solution is best approximated. Population based algorithms are further classified based on the source of inspiration into Evolutionary Algorithms (EA) and Swarm Intelligence algorithms (SI). SIs are inspired from the social intelligence that can be observed from the interactions between the groups of creatures such as flock of wolves, swarm of fish and colony of bee. A well known example for SI paradigm is the Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1997).

There are many metaheuristic algorithms that have been adopted as search engines in a wrapper framework and proved their effectiveness to limit the complexity of FS problem and provide acceptable solutions within a bounded time frame. These include the well-regarded algorithms such as GA (Huang and Wang, 2006) and PSO (Jain et al., 2018) and the recent metaheuristic algorithms such as Whale Optimization Algorithm (WOA) (Sayed et al., 2018a), Multi-Verse Optimization algorithm (MVO) (Ewees et al., 2019) and Salp Swarm Algorithm (SSA) (Sayed et al., 2018b).

Moth Flame Optimization algorithm (MFO) is a recently developed SI algorithm that mimics the navigation method of moths at night. MFO proved it’s effectiveness in optimizing various complex optimization problems in different fields (Mirjalili, 2015). Like any population based paradigm, the MFO optimizer has two conflicting milestones in the optimization process called exploration and exploitation. In the exploration phase, the search space is searched to identify promising regions where the best solution may exist while in the exploitation phase, the found solutions are further improved. The main target in the optimization process is to maintain a balance between exploration and exploitation and smoothly alternates between them. Too much exploration loses the optimal solution while too much exploitation causes stagnation in a local minima.

The MFO algorithm has many advantages that motivated us to select it as a search algorithm in wrapper frameworks: First, the search engine of the MFO methodology relies on a spiral position update procedure that can change the positions of moths in a manner that achieves a promising trade off between the exploration and exploitation and adaptively converges toward the optimal solution. Second, it had been used to solve many problems with unknown and constrained search spaces (Mirjalili, 2015). Third, MFO algorithm is equipped with adaptive parameters that increase exploration phase in the early stages of the optimization process and increase the exploitation in the final stages. Fourth, MFO always maintains the best solutions obtained and reduces their numbers in each iteration so that it gets one global best solution in the final stage. Despite the promising characteristics of the MFO algorithm, the search agents still have a chance of being entrapped in local minima.

Chaotic maps are commonly used operators that have been used to replace random components and support convergence of multiple metaheuristic algorithms. In 2007, Chaotic maps were hybridized with PSO algorithm to develop a feasible approach for FS and classification of the hyper spectral image data (Yang et al., 2007). In 2011, a hybrid model for FS and classification of large-dimensional microarray data sets was developed by using correlation-based FS (CFS) and the Taguchi Chaotic Binary PSO (TCBPSO) (Chuang et al., 2011). The same author, in the same year, designed a chaotic BPSO (CBPSO) based on two kinds of chaotic maps called logistic maps and tent maps that were integrated with BPSO to determine the inertia weight of the BPSO to enhance the FS process. In 2017, chaotic maps were integrated with the MVO algorithm in the the context of FS to cope with slow convergence and local minima.
problems. The used chaotic maps were Tent, Logistic, Singer, Sinusoidal and Piecewise (Ewees et al., 2019). An improved SSA was developed in 2018 to handle the FS problem. Chaos theory was integrated into the algorithm to replace the random variables with chaotic variables. The developed approach was able to efficiently mitigate the local minima and low convergence problems (Sayed et al., 2018b). In 2018, a new wrapper FS approach was developed based on WOA and chaotic theory named CWOA in the medical application (Sayed et al., 2018a).

According to No-free-Lunch theorem (Wolpert et al., 1997), there is no metaheuristic algorithm that has the same performance with all optimization problems. Thus, the doors are still opened to propose new modifications to enhance metaheuristic algorithms. In this paper, chaotic maps have been proposed for the first time to enhance the MFO ability in the FS binary space. The main contribution is the development of four binary variants of the MFO algorithm through the use of four different chaotic maps. The main purpose is to improve the initialization strategy of the standard MFO algorithm by replacing the uniform random distribution with chaotic equations. The generated CBMFO variants are studied and compared to analyze the influence of the adopted operators on the BMFO performance while optimizing the feature space in the domain of disease diagnosis.

The paper is organized as follows: Section 2 gives an overview of the standard and binary MFO algorithm. Section 3 discusses the proposed approach. In Section 4, the experimental results are analyzed. Finally, in Section 5, conclusions and future works are outlined.

2 METHODOLOGY

2.1 Overview of MFO

Moth Flame Optimization (MFO) is one of the recent SI algorithms which was developed in (Mirjalili, 2015). The MFO methodology was inspired by the natural movements of moths at night. The moths are enabled to move long distances in straight line by maintaining the same angle with respect to moon light. This navigation method is called transfer orientation. However, transfer orientation has the shortcoming that nearby light sources such as candle fool the moths and force them to follow a spiral path until they eventually die.

Eq.1 describes mathematically the natural spiral motion of moths around a flame where $M_i$ represents the $i_{th}$ moth, $F_j$ represents the $j_{th}$ flame, and $S$ is the spiral function. Eq.2 formulates the spiral motion using a standard logarithmic function where $D_i$ is the distance between the $i_{th}$ moth and the $j_{th}$ flame as described in Eq.3, $b$ is a constant value for determining the shape of the logarithmic spiral, and $r$ is a random number in the range $[-1,1]$. The parameter $t = -1$ indicates the closest position of a moth to a flame where $t = 1$ indicates the farthest position between a moth and a flame. To achieve more exploitation in the search space the $t$ parameter is considered in the range $[r,1]$ where $r$ is linearly decreased over the course of iterations from -1 to -2. Eq.4 shows gradual decrements of the number of flames over the course of iterations where $l$ is the current number of iteration, $N$ is the maximum number of flames and $T$ is the maximum number of iterations. Algorithm 1 shows the entire pseudo code of the MFO algorithm. The steps of the MFO optimization starts by initializing the positions of moths. Each moth updates it’s position with respect to a flame based on a spiral equation. The $t$ and $r$ parameters are linearly decreased over iterations to emphasize exploitation. In each iteration, the flames list is updated and then sorted based on the fitness values of flames. Consequently, the moths update their positions with respect to their corresponding flames. To increase the chance of reaching to the global best solution, the number of flames is decreased with respect to the iteration number. Thus, a given moth updates it’s position using only one of the flames.

\[ Mi = S(M_i, F_j) \quad (1) \]
\[ S(M_i, F_j) = D_i e^{bt} \cdot \cos(2\pi) + F_j \quad (2) \]
\[ Di = |M_i - F_j| \quad (3) \]
\[ FlameNo = \text{round}(N - l \cdot (N - 1)/T) \quad (4) \]

2.2 Binary MFO (BMFO)

The original MFO algorithm was developed to solve global optimization problems where the components of a solution are real values. All what is required is to check that the upper and lower bounds are not exceeded during the initialization and update procedures. In the binary optimization problems, the case is different because the solutions have only binary elements (i.e either '0' or '1'). This restriction should be not violated while the moths change their positions in the binary search space. For achieving this purpose, some operators have to be integrated with MFO algorithm to allow it optimize in the binary search space.
Algorithm 1: Pseudo-code of the MFO algorithm.

Input: Max_iteration, n (number of moths), d (number of dimensions)
Output: Approximated global solution

Initialize the position of moths

while $l \leq$ Max_iteration do
  Update flame no using Eq.4
  $OM = \text{FitnessFunction}(M)$;
  if $l \leq 1$ then
    $F = \text{sort}(M)$;
    $OF = \text{sort}(OM)$;
  else
    $F = \text{sort}(M_{l-1}, M_l)$;
    $OF = \text{sort}(OM_{l-1}, OM_l)$;
  end if
  for $i = 1$ to $n$ do
    for $j = 1$ to $d$ do
      Update $r$ and $t$;
      Calculate $D$ using Eq.3 with respect to the corresponding moth;
      Update $M(i, j)$ using Eqs.1 and Eqs.2 with respect to the corresponding moth;
    end for
  end for
  $l = l + 1$;
end while

The most common binary operator used for converting continuous optimizers into binary is the transfer function (TF) (Mirjalili and Lewis, 2013). The main reason for using TFs is that they are easy to implement without impacting the merit of the algorithm. In this paper, the used TF is the sigmoid function which was used originally in (Kennedy and Eberhart, 1997) to generate the binary PSO (BPSO). In the MFO algorithm, the first term of Eq.2 represents the step vector which is redefined in Eq.5. The function of the sigmoid is to determine a probability value in the range [0,1] for each element of the solution. Eq.6 shows the formula of the sigmoid function. Each moth updates it’s position based on Eq.7 which takes the output of Eq.6 as it’s input.

\[
\Delta M = Di \cdot e^{\beta t} \cdot \cos(2\pi) \quad (5)
\]

\[
TF(\Delta M_t) = \frac{1}{1 + e^{\Delta M_t}} \quad (6)
\]

\[
M^t_i(t+1) = \begin{cases} 
0, & \text{if } rand < TF(\Delta M^t_{i+1}) \\
1, & \text{if } rand \geq TF(\Delta M^t_{i+1}) 
\end{cases} \quad (7)
\]

2.3 Feature Selection based on BMFO

The FS problem must be represented correctly in order to facilitate the optimizer task in the feature space. There are two key issues to achieve this: properly representing the solution and evaluating it using a specific fitness function. The solution to the FS problem is represented as a binary vector where the length of the vector is equal to the dimensions of the data set. Thus, each element of the solution represents a feature that takes two values, either “1” if the feature is selected or “0” if the feature is not selected. The evaluation for the solution in the FS problem depends on combining two main objectives of the FS problem in one formula. These objectives are maximizing the performance of the classifier and simultaneously minimizing the number of dimensions in the dataset. Eq.8 formulates the FS problem where $\alpha \gamma (D)$ is the error rate of the classification produced by a classifier, $|R|$ is the number of selected features in the reduced dataset, and $|C|$ is the number of features in the original dataset, and $\alpha \in [0, 1]$, $\beta = (1 - \alpha)$ are two parameters for representing the importance of classification performance and length of feature subset based on recommendations (Mafarja and Mirjalili, 2018).

\[
\text{Fitness} = \alpha \gamma (D) + \beta |R| \quad (8)
\]

3 THE PROPOSED APPROACH

In this section, the proposed chaotic BMFO (CBMFO) approaches are presented.

3.1 Chaotic Maps

Chaotic maps are mathematical systems that describe a dynamic deterministic process which has a high sensitivity to initial conditions (dos Santos Coelho and Mariani, 2008). Even though the process is deterministic but the outcomes are unpredictable. Chaotic maps have proved their effectiveness in improving the performance of metaheuristic algorithms when they are integrated with them for solving a specific optimization problem. They have been applied to replace the random components of the metaheuristic algorithm to provide a higher convergence capability and alleviate the local minima problem by getting closer to the position of the optimal solution. In this paper, the impact of chaotic maps are studied on MFO optimizer in the FS search space. Four different chaotic maps have been selected called circle, logistic, piece-wise and tent as formulated in equations Eq.9, Eq.10,
Eq.11, Eq.12, respectively. Fig 1 visually presents these chaotic maps. The developed chaotic MFO variants are called CBMFO1, CBMFO2, CBMFO3 and CBMFO4 respectively. In these variants, the chaotic maps are used to initiate the positions of moths instead of using the uniform random distribution. The idea is to improve the initialization process of the MFO and reduce the uncertainty of the optimizer. This is done by replacing the initial random positions of moths generated by uniform random distribution with positions generated by chaotic maps.

\[ \text{Circle} : x_1 = \text{mod}(X_1 + b - (a2\pi) \sin(2\pi x_1), 1) \]
where: \(a = 0.5, b = 0.2\).

\[ \text{Logistic} : x_1 + 1 = a x_1 (1 - x_1) \]
where: \(a = 4\).

\[ \text{Piecewise} : x_{i+1} = \begin{cases} \frac{x_i}{p}, & \text{if } 0 \leq x_i < p \\ \frac{p-x_i}{p-p}, & \text{if } p \leq x_i < 0.5 \\ \frac{x_i-p}{p-p}, & \text{if } 0.5 \leq x_i < 1-p \\ \frac{1-x_i}{p}, & \text{if } 1-p \leq x_i < p \end{cases} \]
where: \(p = 0.4\).

\[ \text{Tent} : x_{i+1} = \begin{cases} \frac{x_i}{p}, & \text{if } x_i < 7 \\ \frac{10}{(1-x_i)}, & \text{if } x_i \geq 7 \end{cases} \]

4 EXPERIMENTAL RESULTS

In this paper, 23 medical datasets were downloaded from UCI (Asuncion and Newman, 2007), Keel (Alcalá-Fdez et al., 2011) and Kaggle (Goldbloom et al., 2017) data repositories to evaluate the proposed wrapper approaches. Table 1 lists these datasets along with their number of features, instances and classes. All the datasets are characterized by balanced distribution of their classes. To validate the proposed approaches, three well known metaheuristic algorithms were used for comparison purposes: BCGWO, BCS and BBA. Their parameter settings are as follows: the value of \(\alpha\) parameter in GWO is in [2.0]. For BA, the Qmin Frequency minimum value is 0, Qmax Frequency maximum is 2, A Loudness value is 0.5 and r Pulse rate is 0.5. For CS, the \(pa\) value is 0.25 and the value of \(\beta\) parameter is 3/2.

All the experiments were executed on a personal machine with AMD Athlon Dual-Core QL-60 CPU at 1.90 GHz and memory of 2 GB running Windows7 Ultimate 64 bit operating system. The optimization algorithms are all implemented in Python in the EvoloPy-FS framework (Khurma et al., 2020). The maximum number of iterations and the population size were set to 100 and 10 respectively. In this work, the K-NN classifier (where \(K = 5\) (Mafarja and Mirjalili, 2018)) is used to evaluate individuals in the wrapper FS approach. Each dataset is randomly divided in two parts: 80% for training and 20% for testing. To obtain statistically significant results, this division was repeated 30 independent times. Therefore, the final statistical results were obtained over 30 independent runs. The \(\alpha\) and \(\beta\) parameters in the fitness equation is set to 0.99 and 0.01, respectively (Emary et al., 2016). The used evaluation measures are fitness values, classification accuracy, number of selected features and CPU time.

Inspecting the results in Table 2, it seems clearly that the usage of chaotic operators have improved the performance of the BMFO algorithm in terms of the classification accuracy. By comparing CBMFO1, CBMFO2, CBMFO3 and CBMFO4 it appears that the CBMFO2 and CBMFO4 achieved promising results. Based on the ranking results, the Tent-based CBMFO4 achieved the highest classification performance in five out of 23 datasets then Logistic-based CBMFO2 which which was superior across four datasets. On the other hand, Circle-based CBMFO1 and Piecewise-based CBMFO3 outperformed other algorithms only across two datasets. By combining all the chaotic variants of the BMFO algorithm and comparing their results with the standard BMFO algorithm, it can be seen that the BMFO chaotic variants achieved an improvement over the standard BMFO equals 70%. Moreover, the standard BMFO was not better than any of the chaotic approaches on any
dataset. By comparing all BMFO-based approaches with other metaheuristic wrapper approaches, the BMFO approach was superior in 78% of the data sets. The improvement in the classification results of the CBMFO can be explained that the chaotic operators have effectively improved the initialization procedure by replacing the uniform random distribution with chaotic functions that have a greater sensitivity to initial conditions. This led to a greater exploration for the search space and better alleviating for the local minima problem. Furthermore, the optimizer was able to achieve a better trade-off between the two conflicting milestones: exploration and exploitation. This was realized by an improvement in the convergence behaviour of the MFO in the binary feature space.

The overall performance of the proposed approaches can be better realized when analyzing the fitness values results that combine both classification accuracy and the selection ratio. From the results in Table 3, it can be seen that CBMFO approaches got better results than other methods in twelve out of 23 datasets that are close to half of the datasets.
Table 4: Average Number of Selected Features from 30 Runs for All Approaches.

<table>
<thead>
<tr>
<th>No</th>
<th>Dataset Name</th>
<th>BMFO</th>
<th>CBMFO1</th>
<th>CBMFO2</th>
<th>CBMFO3</th>
<th>CBMFO4</th>
<th>BGWO</th>
<th>BCS</th>
<th>BBA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Breast Cancer (Original)</td>
<td>5.500</td>
<td>5.667</td>
<td>5.637</td>
<td>5.667</td>
<td>5.687</td>
<td>5.671</td>
<td>5.671</td>
<td>5.733</td>
</tr>
<tr>
<td>2</td>
<td>Breast Cancer (Wisconsin)</td>
<td>5.587</td>
<td>5.637</td>
<td>5.671</td>
<td>5.671</td>
<td>5.687</td>
<td>5.671</td>
<td>5.733</td>
<td>5.733</td>
</tr>
<tr>
<td>3</td>
<td>Breast Cancer (Prognosis)</td>
<td>5.687</td>
<td>5.671</td>
<td>5.671</td>
<td>5.671</td>
<td>5.687</td>
<td>5.671</td>
<td>5.733</td>
<td>5.733</td>
</tr>
<tr>
<td>4</td>
<td>Breast Cancer (Combina)</td>
<td>5.687</td>
<td>5.671</td>
<td>5.671</td>
<td>5.671</td>
<td>5.687</td>
<td>5.671</td>
<td>5.733</td>
<td>5.733</td>
</tr>
<tr>
<td>5</td>
<td>Breast Cancer (Combina)</td>
<td>5.687</td>
<td>5.671</td>
<td>5.671</td>
<td>5.671</td>
<td>5.687</td>
<td>5.671</td>
<td>5.733</td>
<td>5.733</td>
</tr>
<tr>
<td>6</td>
<td>Diabetic Retinopathy (DR)</td>
<td>7.567</td>
<td>7.637</td>
<td>7.671</td>
<td>7.671</td>
<td>7.733</td>
<td>7.733</td>
<td>7.733</td>
<td>7.733</td>
</tr>
<tr>
<td>7</td>
<td>Diabetic Retinopathy (DR)</td>
<td>7.637</td>
<td>7.671</td>
<td>7.733</td>
<td>7.733</td>
<td>7.733</td>
<td>7.733</td>
<td>7.733</td>
<td>7.733</td>
</tr>
<tr>
<td>8</td>
<td>Pima-indians-diabetes</td>
<td>6.000</td>
<td>6.000</td>
<td>6.000</td>
<td>6.000</td>
<td>6.000</td>
<td>6.000</td>
<td>6.000</td>
<td>6.000</td>
</tr>
<tr>
<td>9</td>
<td>Pima-indians-diabetes</td>
<td>6.000</td>
<td>6.000</td>
<td>6.000</td>
<td>6.000</td>
<td>6.000</td>
<td>6.000</td>
<td>6.000</td>
<td>6.000</td>
</tr>
<tr>
<td>12</td>
<td>Diabetes Retinopathy</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
</tr>
<tr>
<td>13</td>
<td>Diabetes Retinopathy</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
</tr>
<tr>
<td>14</td>
<td>Diabetes Retinopathy</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
</tr>
<tr>
<td>15</td>
<td>Diabetes Retinopathy</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
</tr>
<tr>
<td>16</td>
<td>Diabetes Retinopathy</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
</tr>
<tr>
<td>17</td>
<td>Diabetes Retinopathy</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
</tr>
<tr>
<td>18</td>
<td>Diabetes Retinopathy</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
</tr>
<tr>
<td>19</td>
<td>Diabetes Retinopathy</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
</tr>
<tr>
<td>20</td>
<td>Diabetes Retinopathy</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
</tr>
<tr>
<td>21</td>
<td>Diabetes Retinopathy</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
</tr>
<tr>
<td>22</td>
<td>Diabetes Retinopathy</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
</tr>
<tr>
<td>23</td>
<td>Diabetes Retinopathy</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
</tr>
<tr>
<td>24</td>
<td>Diabetes Retinopathy</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
</tr>
<tr>
<td>25</td>
<td>Diabetes Retinopathy</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
</tr>
<tr>
<td>26</td>
<td>Diabetes Retinopathy</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
<td>137.133</td>
</tr>
</tbody>
</table>

According to Table 4, it seems clear that the BBA algorithm outperformed other wrapper approaches across 57% of datasets. By comparing the CBMFO approaches to each other in terms of selection ratio, the Circle-based CBMFO2 outperformed the CBMFO1 approach in terms of selection ratio. On the other hand, the BCS algorithm outperformed the CBMFO2 approach in terms of selection ratio.

By looking at Table 5, it seems clear from the ranking results that the CBMFO algorithm outperformed other algorithm across 57% of datasets. It is seen that the BMFO approaches were not able to outperform these approaches on any of the datasets. By comparing the CBMFO approaches to each other in terms of running time, the Logistic-based CBMFO2 and Circle-based CBMFO3 achieved better results compared with other chaotic variants. The CBMFO2 consumed the shortest optimization time to find the near optimal feature subset across ten datasets and the CBMFO1 consumed the shortest optimization time across eight datasets. However, the BBA algorithm had much superior performance in general. On the other hand, CBMFO4 based on Tent map achieved the shortest CPU time for optimization across four datasets. For the CBMFO3, it is clear that this approach didn’t outperform any other approaches in terms of running time.

**5 CONCLUSIONS AND FUTURE WORK**

In this paper, multiple binary versions based on MFO algorithm have been proposed to address the FS problem. The chaotic maps have been adopted to enhance the MFO performance in the feature space. Specifically, the chaotic maps were used to enhance the initialization of moths and promote the convergence behaviour of the MFO algorithm. Therefore, the MFO can alleviate stagnation in local minima and reach to
a closer place near the global optima. To evaluate the proposed approaches, 23 medical datasets were used from well regarded data repositories including UCI, Keel and Kaggle. The comparative results showed that the chaotic operators have enhanced the performance of the standard BMFO when used to optimize the feature search space. For the future, the research line of metaheuristic based wrapper methods can be continued by proposing new modification strategies and adopting other metaheuristic algorithms to examine feature space.

REFERENCES


