A new binary grasshopper optimization algorithm for feature selection problem

Haouassi Hichem *, Merah Elkamel, Mehdaoui Rafik, Maarouk Toufik Mesaaoud, Chouhal Ouahiba

Univ Khenchela, Fac. ST, Lab., BP 1252 El Houria, 40004 Khenchela, Algeria

A R T I C L E   I N F O

Article history:
Received 12 July 2019
Revised 9 November 2019
Accepted 11 November 2019
Available online xxxx

Keywords:
Swarm intelligence
Grasshopper optimization
Feature selection and binary search space

A B S T R A C T

The grasshopper optimization algorithm is one of the recently population-based optimization techniques inspired by the behaviours of grasshoppers in nature. It is an efficient optimization algorithm and since demonstrates excellent performance in solving continuous problems, but cannot resolve directly binary optimization problems. Many optimization problems have been modelled as binary problems since their decision variables varied in binary space such as feature selection in data classification. The main goal of feature selection is to find a small size subset of feature from a sizeable original set of features that optimize the classification accuracy. In this paper, a new binary variant of the grasshopper optimization algorithm is proposed and used for the feature subset selection problem. This proposed new binary grasshopper optimization algorithm is tested and compared to five well-known swarm-based algorithms used in feature selection problem. All these algorithms are implemented and experimented assessed on twenty data sets with various sizes. The results demonstrated that the proposed approach could outperform the other tested methods.

© 2019 The Authors. Production and hosting by Elsevier B.V. on behalf of King Saud University. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Contents

1. Introduction .............................................................................................................. 00
2. Related works. ......................................................................................................... 00
3. Grasshopper optimization algorithm ........................................................................ 00
4. Novel binary grasshopper optimization algorithm (NBGOA). ..................................... 00
  4.1. Binary encoded for the NBGOA ..................................................................... 00
  4.2. Initial population. .......................................................................................... 00
  4.3. Position updates ............................................................................................ 00
    4.3.1. Hamming distance between two binary bits ....................................... 00
  4.4. Fitness function ............................................................................................. 00
  4.5. Main procedure of the NBGOA for feature selection. ....................................... 00
5. Experimental results and performance comparison. .................................................. 00
  5.1. Data description ............................................................................................ 00
  5.2. Benchmark algorithms and parameters setting .................................................. 00
  5.3. Evaluation criteria ........................................................................................ 00
  5.4. Parameters study. ........................................................................................... 00

* Corresponding author.
E-mail addresses: houassi_h@yahoo.fr (H. Hichem), kmerah@yahoo.fr (M. Elkamel), mehdaoui.rafik@yahoo.fr (M. Rafik), toumaarouk@yahoo.fr (M.T. Mesaaoud), chouhal_wahiba@yahoo.fr (C. Ouahiba).

Peer review under responsibility of King Saud University.

https://doi.org/10.1016/j.jksuci.2019.11.007
1319-1578/© 2019 The Authors. Production and hosting by Elsevier B.V. on behalf of King Saud University. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Please cite this article as: H. Hichem, M. Elkamel, M. Rafik et al., A new binary grasshopper optimization algorithm for feature selection problem, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2019.11.007
1. Introduction

High dimensional data were containing a considerable number of attributes that complicates the machine learning tasks in data mining. Among the preprocessing functions in the data mining process is Feature Selection (FS) which aims to reduce the dimensionality of data by eliminating irrelevant, redundant or noisy features to enhance the effectiveness of the machine learning algorithms such as classification accuracy, CPU time and memory requirement. Formally, the feature selection algorithms aim to find a subset of relevant features from a given set of large numbers of attributes. In the feature selection problem, a dataset with n features includes 2^n subsets possible.

Since the main aim in FS is to minimize the number of selected features while maintaining or increasing the maximum classification accuracy, it can be considered as an optimization task. Many researchers model the feature selection as a combinatorial optimization problem (Guyon and Elisseeff, 2003). In practice, an exhaustive search to obtain the optimal solution is almost impossible because of the exponential increase in the number of features in recent datasets. One of the most recently used methods to solve this problem are Evolutionary Computation Algorithms (ECA); they are inspired from nature and simulate the behavior of biological, physical and ethological systems (Xue et al., 2016). Swarm Intelligence (SI), firstly proposed by Beni and Wang in 1993 (Beni and Wang, 1993), are population-based approaches where the swarm is composed by a population of artificial agents and used as a method for global optimization in order to mimic the behavior of animals to seek for their food like ants, cats, fish, birds, fireflies, etc. (Boussaid et al., 2013; Rajpurohit et al., 2017). In recent years, optimizing by swarm intelligence has became an interesting way in research.

Very recently, a novel swarm intelligence optimization algorithm, called Grasshopper Optimization Algorithm (GOA), has been proposed by Saremi et al. (Saremi et al., 2017) which is inspired by the behavior of grasshopper swarms in nature. This approach has demonstrated interesting results for continuous optimization problems as compared to recent algorithms in the literature. This fact motivates us to modify this approach to solve the feature selection problem modelled as binary optimization.

In this paper, we present our two main works. The first one is the New Binary version of the Grasshopper Optimization Algorithm (NBGOA) which can be used with any binary combinatorial optimization problem, and the other work is to use NBGOA as the swarm-based algorithm for feature selection problem.

The rest of the paper is organized as follows: Section 2 describes a review of related works. Section 3 presents the original GOA and the motivation for the development of its binary version. In Section 4, the proposed binary version of GOA is presented. Section 5 details and discusses the experimental results on the benchmark datasets using the proposed algorithm and other methods in the literature. Finally, Section 6 concludes the work and suggests several directions for future work.

2. Related works

With the use of high dimensional datasets, the feature selection optimization problem is becoming more complex. Then, traditional methods become insufficient to solve this complicated problem. Recently and to cope with this limitation, many swarm-based algorithms have been proposed in the literature, it used as effective solutions in the FS problem.

Binary Particle Swarm Optimization (PSO) and its variants have been widely used in the FS problem. The authors in (Moradi and Gholampour, 2016) proposed a hybrid PSO with a local search strategy to find the less correlated feature subset. Zhang et al. (Zhang et al., 2019) used filter-based bare-bone PSO. In this work, two filter-based strategies are proposed (average mutual information and feature redundancy) to improve the exploitation capability of the swarm. A Competitive Swarm Optimizer (CSO) for solving high-dimensional FS problem is proposed in (Shenkai et al., 2018). The authors in (Xue et al., 2014) proposed an initialization and updating mechanisms for the originally PSO. A hybrid binary bat enhanced particle swarm optimization algorithm called HBBPSO was proposed by Mohamed and Kevin (Mohamed and Kevin, 2018). The authors combine the bat algorithm and the PSO to solve the FS problem exploiting the best aspects of both these algorithms.

The Ant Colony Optimization (ACO) was used in FS approaches by many researchers. In (Huijun et al., 2018) an ant colony optimization algorithm called FACO is proposed, they modified the original ACO algorithm to prevent it from falling into a local optimum prematurely. Youchuan et al. in (Youchuan et al., 2016) proposed a modified binary-coded ACO algorithm and combined it with genetic algorithm (MBACO). In (Forsati et al., 2014), the authors present a new variant of ACO called enRiched Ant Colony Optimization (RACO), which aims to prevent premature convergence. Besides, an improved ACO for the FS is proposed in (Zhao et al., 2014), whose objective is to find the near-optimal subsets in multi-character feature sets.

Binary Cuckoo Search Algorithm (BCSA) (Rodrigues et al., 2013), and its variants such as (Mohamed and Aboul, 2018) has successfully applied for FS problem in several works.

Binary Ant Lion Optimizer (BALO) is a recent meta-heuristic used as a features subset searching algorithm in (Emary et al., 2016a). A chaotic ALO approach was proposed for FS in (Zawbaa et al., 2016b). A modified BALO is an FS approach newly proposed in (Mingwei et al., 2019), it has used for the hyperspectral image. The GWO, as another recent swarm-based optimizer (Mirjalili et al., 2014), that has applied for FS in many works (Jingwei et al., 2018; Emary et al., 2016b; Chantar et al., 2019; Qasem et al., 2019).

Moth-Flame Optimizer is an algorithm that mimics the navigation method of moths and has been applied for the FS problem (Pauline et al., 2019; Zawbaa et al., 2016a).

Recent binary variants of Butterfly Optimization Algorithm (BOA) are proposed and used to select the optimal feature subset for classification (Sankalap and Priyanka, 2019). In this work, the BOA is used while its continuous steps are bounded in a threshold using a suitable threshold function after squashing them.

Grasshopper Optimization algorithm (GOA) is used for simultaneous feature selection and support vector machine optimization (Aljarah et al., 2018a). More recently, Mafarja et al. (Mafarja et al, 2019) proposed binary variants of GOA to develop a wrapper FS method. In this work, the authors propose two approaches to design a binary GOA. The first one is based on sigmoid (BGOA-S) and V-shaped transfer functions (BGOA-V). While the second approach, called BGOA-M, uses a stochastic mutation and has achieved the best results in almost all cases tested. In this second
approach, the mutation rate is changing depending on the step vector.

Moreover, many swarm-based FS approaches have been proposed in the literature, such as the Firefly Algorithm (FA) (Larabi et al., 2018), Multi-Verse Optimizer (MVO) (Faris et al. 2018a), Salp Swarm Algorithm (SSA) (Faris et al., 2019; Faris et al., 2018b; Aljarah et al., 2018b).

Some of these algorithms are combined to compose hybrid approaches to solve the FS problem. For instance, in (Zawbaa et al. 2018), a hybrid between GWO and ALO was proposed for wrapper FS. Besides, a combination between ALO and hill-climbing techniques for FS was introduced in (Mafarja and Mirjalili, 2018), Jona and Nagaveni hybridized ACO with cuckoo search (CS) to solve the FS problem in the digital mammogram (Jona and Nagaveni, 2014). Suguna and Thanuskodi use the Rough Set Theory (RST) with Bee Colony Optimization (BCO) for feature selection problem in the medical domain (Suguna and Thanuskodi, 2010). In (Chamisi and Benediktsson, 2015) a new approach based on the integration of a genetic algorithm and PSO is proposed.

3. Grasshopper optimization algorithm

There are many swarm-based algorithms proposed in the literature for solving complex and challenging optimization problems in different area search due to their simplicity and efficiency in global optimization. The number of these algorithms has a steep increase over the last decade (Rajpurohit et al., 2017). A variety of swarm-based optimization algorithms have been proposed in literature include, but not limited to PSO (Eberhart and Kennedy, 1995; Houassi et al., 2018), Ant Colony Optimization (ACO) (Dorigo et al., 1996), Flower Pollination Algorithm (FPA) (Yang, 2012), Grey Wolf Optimization (GWO) (Mirjalili et al., 2014), Ant Lion Optimization (ALO) (Mirjalili, 2015a), Moth-Flame Optimization (MFO) (Mirjalili, 2015b), Dragonfly Algorithm (DA), (Mirjalili, 2016), Bat Algorithm (BA) (Cai et al., 2016) and Whale Optimization Algorithm (WOA) (Mirjalili and Lewis, 2016; AlaM et al., 2018). Almost all of the above-cited approaches are originally proposed for continuous optimization problems and after that they where binarized to be used for the binary optimization problems such as feature selection in data classification which they show superior performance (Rajpurohit et al., 2017).

Grasshopper optimization algorithm firstly proposed by Saremi et al. (Saremi et al., 2017) is one of the new nature-inspired and population-based algorithms, which models the behaviour of grasshoppers swarms in nature. The two essential phases of optimization are exploration and exploitation of the search space; the grasshoppers provides these two phases during the food search through these social interactions. The main characteristics of the swarm in the larval stage are slow movement and small steps of the grasshoppers. In contrast, long-range and abrupt movement is the essential feature of the swarm in adulthood.

Based on the above description of grasshoppers, Saremi et al. propose three evolutionary operators in position updating of individuals in swarms (Saremi et al., 2017), the social interaction operator ($S_i$), the gravity force operator ($G_i$) and the wind advection operator ($A_i$) such as presented in Eq. (1).

$$X_i = S_i + G_i + A_i$$ (1)

where $X_i$ defines the position of the $i$th grasshopper.

Each of these behaviours is mathematically modelled as follows:

The interaction operator is calculated as follows in Eq. (2) (Saremi et al., 2017).

$$S_i = \sum_{j=1}^{N} \sum_{|j| \neq |i|} S(||X_j - X_i||) \frac{X_j - X_i}{d_{ij}}$$ (2)

where $N$ is the number of grasshoppers in the swarm, $d_{ij}$ represents the distance between the $i$th and the $j$th grasshopper, $S$ is a function that defines the strength of social forces, and that is calculated as in Eq. (3).

$$S(r) = \frac{e^r}{r}$$ (3)

where $f$ and $l$ are two constants that indicate respectively the intensity of attraction and the attractive length scale, and $r$ is a real value.

However, the gravity operator is not considered by the authors (Saremi et al., 2017), and they assume that the wind direction is always towards a target. Then the Eq. (1) becomes as follows:

$$X_i^l = c \left( \sum_{j=1}^{N} \frac{1}{d_{ij}} \right) \left( X_j^l - X_i^l \right) + T_d$$ (4)

where $ub_d$ is the upper bound in the $d$th dimension, $lb_d$ is the lower bound in the $d$th dimension, $T_d$ is the value of the $d$th dimension in the target (best solution found so far), and the coefficient $c$ reduces the comfort zone proportional to the number of iterations and is calculated as follows in Eq. (5).

$$c = \frac{C_{max} - C_{min}}{l}$$ (5)

where $C_{max}$ is the maximum value, $C_{min}$ is the minimum value, $l$ indicates the current iteration, and $L$ is the maximum number of iterations. In (Saremi et al., 2017), they use $C_{max} = 1$ and $C_{min} = 0.00001$.

Eq. (4) shows that the next position of a grasshopper is defined based on its current position and the position of all other grasshoppers (first term in Eq. (4)) and the position of the target (second term).

The primary motivation in the development of the binary variant of the grasshopper optimization algorithm is the quality of exploration, local optimia avoidance, exploitation, and convergence of the GOA algorithm demonstrated in the paper (Saremi et al., 2017). These robust features originate from the high repulsion rate between grasshoppers and its efficient searching capabilities in nature. The second motivation is that the results showed that the GOA algorithm provides very competitive results compared to the five most well-known and recent algorithms in the literature.

4. Novel binary grasshopper optimization algorithm (NBGOA)

GOA algorithm is a population-based evolutionary search approach, which was initially proposed as an optimization technique to address continuous problems (Saremi et al., 2017). However, many optimization problems, such as feature selection are set in a binary space. So, to use the GOA in binary optimization problems, the solutions are restricted to the binary [0,1] values which we motivated to propose a binary version of the GOA. The binarization of continuous space transforms the continuous values of the continuous space into binary values 0 or 1 in the binary space.

In the literature, there are two different strategies to extend the continuous population-based evolutionary approaches to binary approaches. In the first strategy, the binary methods are proposed without modifying the structure of the continuous approaches. The position’s vectors are used as they in the continuous area and they must be converted from real values to binary values in each iteration for creating the solution vector using several binarization
methods (Krause et al., 2013). Including modified position equation (Pan et al., 2008), great value priority (Congying et al., 2011), angle modulation (Yavuz and Aydin, 2016) and transfer functions (Mirjalili and Lewis, 2013) which receive a real value as input and return a number in [0,1] and defines the probability of changing positions. The sigmoid function is the most used in the literature (Banati and Bajaj, 2011; Rodrigues et al., 2015; Kennedy and Eberhard, 1997; Feng et al., 2016; Pampara and Engelbrecht, 2011). The authors of (Mirjalili, 2016) used a V-shaped transfer function.

In the second strategy, the structure of the continuous approach is modified, in which, the position's vectors are modelled as binary vectors, and the main updating equation is reworded by using binary vectors and adapting the operators between these new vectors which can provide only binary values. Several operators are used in the literature: the difference between two vectors is modelled as a distance and defined as the Hamming distance between two binary strings. The addition of two vectors is modelled as a mutation with a probability in (Maalong et al., 2016), and the crossover between vectors is performed for placing the addition operator between vectors (Emary et al., 2016a; Emary et al., 2016b).

To extend the use of the GOA in the binary problems, we propose in this paper a binary version of GOA called (NBGOA) as an optimization algorithm for the feature selection problem using the second strategy. In the following subsections, we present the details of our proposed NBGOA approach for a feature selection problem.

4.1. Binary encoded for the NBGOA

In the optimization problems, particles of the swarm search optimal solution in the D-dimensional searching space and each particle in the swarm is looked like a point or a position in the D-dimensional space. The ith particle in the swarm represents a D-dimensional vector \( \mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iD}) \). In the optimization problems, the vector \( \mathbf{x}_i \) takes real values, but in the binary optimization problems, it takes binary values: 0 or 1.

In the Continuous Grasshopper Optimization (CGO) algorithm, the grasshopper’s vector takes real values, and the operators such as the social interaction and the wind advection are defined in continuous space. CGO cannot be used as is to optimize the discrete binary optimization problems. To tackle this problem, we propose a binary version of the GOA, where the grasshoppers take the values of binary vectors of length \( \mathbf{N} \) where \( \mathbf{N} \) is the number of features and the dimensionality of the search space. In our proposed NBGOA, each grasshopper’s position in the search space encodes the subset of features that it represents. For example, \( \mathbf{G}_i = (1, 1, 1, 0, 0, 0, 1, 0, 1) \) means that the feature \( f_1 \) is selected and \( f_2, f_6, f_7 \) and \( f_9 \) are not selected.

4.2. Initial population

The first step to design a population-based optimization algorithm is to initialize the population. In our approach, each grasshopper is in the swarm is randomly initialized by a binary vector \( \mathbf{X}_i \), \( (i = 1, 2, \ldots, \mathbf{N}) \), where \( \mathbf{N} \) is the size of the swarm. For each dimension \( j \) of grasshopper \( i \), a binary value 0 or 1 is assigned with a probability 0.5 as follows:

\[
X_i^j(0) = \begin{cases} 
1, & \text{if } R \text{ and}() > 0.5 \\
0, & \text{else} 
\end{cases} 
\]

where \( X_i^j(0) \) is the initial value of the \( j \)th dimension of the \( i \)th grasshopper, \( R \) returns a random value in \([0,1]\).

In Fig. 2, \( X_i \) represents the initial binary vector that corresponds to the \( i \)th grasshopper in the swarm. \( d_j \) is the dimension \( j \) that corresponds to the \( j \)th feature in the dataset.

4.3. Position updates

During the search process, the grasshoppers move towards the target (Saremi et al., 2017), that is the best position obtained so far by the swarm. The swarm is initialized with a population of random solutions (binary vectors), and it searches for the best solution by updating the position of each particle according to the Eq. (5) in continuous GOA. Updating in binary search space is very different than in a continuous space. In continuous search spaces, the grasshoppers can update their positions by adding the value of the first term in Eq. (4) to the value of their corresponding bit in the target vector \( T_a \). However, in a binary search space, the position of grasshopper cannot be updated by adding values since the position vectors of grasshoppers can only contain 0 or 1 then, if we consider the first term in Eq. (4) as the difference between the two binary vectors \( \mathbf{X}_d \) and \( T_a \), we can transform the update equation of the grasshopper’s position in the original GOA as follows in Eq. (7).

\[
X_{d_i} - T_a = c \left( \sum_{j=1}^{N} \frac{c_{d_j} - c_{b_j}}{2} \frac{1}{5} \left| X_{d_j} - X_{a_j} \right| \frac{X_{d_j} - X_{j}}{d_j} \right) 
\]

The term on the right side of Eq. (7) indicates a distance between the bits \( X_{d_i} \) and \( T_a \), since, in the binary version of GOA these two bits are binary, we can use the Hamming distance between these two binary bits.

4.3.1. Hamming distance between two binary bits

The grasshoppers are presented as binary strings, and the distance between two grasshoppers is defined as the Hamming distance between the two binary strings which corresponds the two grasshoppers. The Hamming distance is calculated as the number of different bits in the two strings. For example, grasshopper i is represented as \( \mathbf{G}_i = (X_{i1}, X_{i2}, \ldots, X_{iD}) \) where \( X_{ij} \in \{0,1\} \), and it has D features, and \( X_{ij} \) refers to the \( j \)th dimension of the position of the \( i \)th grasshopper.

We can get the Hamming distance equation as follows in Eq. (8).
\[ |X_i - X_j| = dH(X_i, X_j) = \sum_{d=1}^{D} X_{id}^2 X_{jd}^2 \]  
(8)

where \( d_{ij}(d) \) is the Hamming distance between the grasshoppers \( i \) and \( j \), \( d \) is the index which varies from 1 to the total number of features \( D \).

For the particular case that \( X_{id} \) and \( X_{jd} \) are single bits, the Hamming distance is calculated as in Eq. (9).

\[
d_{ij}(X_{id}, X_{jd}) = \begin{cases} 0 & \text{if } X_{id} = X_{jd} \\ 1 & \text{if } X_{id} \neq X_{jd} \end{cases} \quad (9)
\]

Because \( d_{ij}(d) \) is Hamming distance between two bits, \( d_{ij}(d) \) must be 0 or 1, which is the reason for the use of transformation function \( F(Dist) \) to transform a real number \( Dist \) to a binary digit 0 or 1 in binary version. Then the Eq. (7) can be written again in binary version as followed in Eq. (10), Eq. (11), Eq. (12) and Eq. (13).

\[ |X_i - T_d| = F(Dist) \quad (10)\]

Where:

\[ |X_i - T_d| = d_{ij}(X_{id}, T_d) \]

is the Hamming distance between the bits \( X_{id} \) and \( T_d \), Dist is calculated as in Eq. (11).

\[
Dist = c \left( \sum_{j=1}^{N} \frac{u_{bd} - l_{bd}}{2} S\left( \left| X_{jd}^d - X_{id}^d \right| \right) \right) \quad (11)
\]

Such as, \( Dist \) is the same first term in the right as in Eq. (4). In the binary version, the upper bound \( u_{bd} \) in any dimension is 1, and the lower bound \( l_{bd} \) is 0, then the term \( \frac{u_{bd} - l_{bd}}{2} \) in Eq. (11) is equal \((-1) / 2 = 0.5 \).

\[
u_{bd} - l_{bd} = 0.5 \quad (12)
\]

The function \( S() \) in the Eq. (11) is used in (Saremi et al., 2017) as a function to transform value from the interval \([2,4]\) to a new value in the range \([0,0.02]\), but the term \( |X_{jd}^d - X_{id}^d| \) in Eq. (11) indicates the distance between the binary vectors \( X_j \) and \( X_i \), its value belongs to the interval \([0, Dim]\) in the binary version, \( Dim \) is the number of dimensions, then we must transform the value of \( |X_{jd}^d - X_{id}^d| \) from the interval \([0, Dim]\) to a value in the interval \([2,4]\). For that, we use the function \( G() \) as presented in Eq. (13).

\[
G(x) = \frac{2}{Dim} x + 2 \quad (13)
\]

Where \( Dim \) is the number of dimensions. By using Eq. (12) and Eq. (13), Eq. (11) becomes as in the Eq. (14).

\[
Dist = c \left( \sum_{j=1}^{N} \frac{u_{bd} - l_{bd}}{2} S\left( G\left( \left| X_{jd}^d - X_{id}^d \right| \right) \right) \right) \quad (14)
\]

Because \( Dist \) indicates the distance between two binary vectors, we must use a function to transform the value of \( Dist \) to 1 or 0. We propose the function \( F(Dist) \) as presented in the Eq. (15). All values of \( Dist \) are limited in the range \([\text{Max}(Dist), \text{Min}(Dist)]\). When the value approaches \( \text{Max}(Dist) \) its corresponding feature is a candidate to be selected in classification task. Otherwise the corresponding feature isn’t a candidate to be chosen. \( F(Dist) \) is computed as in Eq. (15).

\[
F(Dist^d) = \begin{cases} 1, & \text{if } Dist^d > \frac{\text{Max}(Dist^d) - \text{Min}(Dist^d)}{2} \\ 0, & \text{if } Dist^d < \frac{\text{Max}(Dist^d) - \text{Min}(Dist^d)}{2} \\ 0, & \text{if } \text{Rand}() < 0.5 \\ 1, & \text{Else} \end{cases} \quad (15)
\]

where \( \text{Max}(Dist^d) = 2 \times \frac{0.02}{Dim} \) and \( \text{Min}(Dist^d) = 0 \) are respectively, the maximum and the minimum value of \( Dist. \text{Rand()} \) is a random number drawn from a uniform distribution in \([0,1]\), and \( Dim \) is the number of dimensions of the problem. Then, the main updating equation can be formulated, as shown in Eq. (16).

\[
X_{id} = \begin{cases} T_d, & \text{if } F(Dist^d) = 0 \\ 1 - T_d, & \text{else} \end{cases} \quad (16)
\]

where \( T_d \) is the value of the \( d \)th dimension in the target (best solution found so far) and \( F(Dist) \) is calculated using Eq. (15).

It can be seen in Fig. 3 that the binary bits are calculated from the target vector according to the value of function \( F(Dist) \) calculated as in Eq. (15).

4.4 Fitness function
The fitness function considers the classification accuracy and the number of selected features. It maximizes the classification accuracy and minimizes the set size of the selected features. Therefore, the following fitness function (Mafarja et al., 2019) is used to evaluate individual solutions, as shown in Eq. (17).

\[
\text{Fitness} = x \times \text{ErrorRate} + (1 - x) \times \frac{\#SF}{\#All_F} \quad (17)
\]

where \( \text{ErrorRate} \) means the classification error rate using the selected features. \( \text{ErrorRate} \) is calculated as the percentage of incorrect classified (by 5-ANN classifier) to the number of classifications made, expressed as a value between 0 and 1. \( \text{ErrorRate} \) is the complement of the classification accuracy, \#SF is the number of selected features and \#All_F is the total number of attributes in the original dataset. \( x \) is used to control the importance of classification quality and subset length. In our experiments, \( x \) is set to 0.9.

You can calculate classification error as the percentage of incorrect predictions to the number of predictions made, expressed as a value between 0 and 1.

4.5 Main procedure of the NBGOA for feature selection
In this subsection, Algorithm 1 shows the general process of the proposed NBGOA for the FS. The NBGOA, firstly generates randomly the initial population of \( N \) grasshoppers (solutions). Each solution may contain a subset of features, and it is evaluated according to the fitness function, which depends on the solution’s accuracy obtained by the KNN classifier and the number of selected features in the solution as in Eq. (17). After the initialization step, the evolutionary process is repeated until a predefined stopping criterion is satisfied. In each iteration, grasshoppers update their positions according to all other grasshoppers based on Eq. (4), Eq. (7) and Eq. (8). After each iteration, the position of the best target obtained so far was updated. Finally, the position and fitness value

\[
\begin{array}{cccccccc}
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
\end{array}
\]

**Fig. 3.** Current position updates.
of the best target could be returned as the best solution for the FS problem. Flowchart of the NBGOA is also presented in Fig. 4.

**Table 1** List of used datasets.

<table>
<thead>
<tr>
<th>No</th>
<th>Dataset</th>
<th># Features</th>
<th># Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Arrhythmia</td>
<td>297</td>
<td>452</td>
</tr>
<tr>
<td>2</td>
<td>Breastcancer</td>
<td>9</td>
<td>699</td>
</tr>
<tr>
<td>3</td>
<td>BreastEW</td>
<td>30</td>
<td>596</td>
</tr>
<tr>
<td>4</td>
<td>CleanI</td>
<td>166</td>
<td>476</td>
</tr>
<tr>
<td>5</td>
<td>Colon</td>
<td>2000</td>
<td>62</td>
</tr>
<tr>
<td>6</td>
<td>German</td>
<td>24</td>
<td>1000</td>
</tr>
<tr>
<td>7</td>
<td>HeartEW</td>
<td>13</td>
<td>270</td>
</tr>
<tr>
<td>8</td>
<td>Ionosphere</td>
<td>34</td>
<td>351</td>
</tr>
<tr>
<td>9</td>
<td>Leukemia</td>
<td>7129</td>
<td>72</td>
</tr>
<tr>
<td>10</td>
<td>Libras</td>
<td>90</td>
<td>360</td>
</tr>
<tr>
<td>11</td>
<td>Madelon</td>
<td>500</td>
<td>4400</td>
</tr>
<tr>
<td>12</td>
<td>Parkinsons</td>
<td>22</td>
<td>195</td>
</tr>
<tr>
<td>13</td>
<td>PenglungEW</td>
<td>325</td>
<td>73</td>
</tr>
<tr>
<td>14</td>
<td>Semeion</td>
<td>265</td>
<td>1593</td>
</tr>
<tr>
<td>15</td>
<td>SonarEW</td>
<td>60</td>
<td>208</td>
</tr>
<tr>
<td>16</td>
<td>SpectEW</td>
<td>22</td>
<td>267</td>
</tr>
<tr>
<td>17</td>
<td>Spectf</td>
<td>44</td>
<td>276</td>
</tr>
<tr>
<td>18</td>
<td>Tic-tac-toe</td>
<td>9</td>
<td>958</td>
</tr>
<tr>
<td>19</td>
<td>WaveletEW</td>
<td>40</td>
<td>5000</td>
</tr>
<tr>
<td>20</td>
<td>Zoo</td>
<td>16</td>
<td>101</td>
</tr>
</tbody>
</table>

**Algorithm 1** (Pseudo codes of the binary grasshopper optimization algorithm).

**input:** Number of grasshoppers $(N)$, Maximum number of iterations for optimization $(\text{Max}_{\text{it}})$, $C_{\text{max}}$ and $C_{\text{min}}$.

**output:** Optimal grasshopper’s binary position.

Initialize randomly a population of $N$ grasshopper’s positions $X_i(t = 1, 2, \cdots, n)$ at from $\{0, 1\}$.

Calculate the fitness value of each grashopper.

Find the best solution $T$ based on fitness function.

Set the best solution as the Target

$it = 0$;

while $(it < \text{Max}_{\text{it}})$

Update $c$ using Eq. (5)

for each grasshopper in the population do

Normalize the distance between grasshoppers in $[2,4]$

Calculate $\text{Dist}$ value and $F(\text{Dist})$ using Eq. (14) and Eq. (15)

Update the position vector using Eq. (16)

end for

Update the Target by the best solution

$it = it + 1$

end while

Return $T$
5. Experimental results and performance comparison

5.1. Data description

In this section, the performance of the proposed NBGOA algorithm is tested and compared to some algorithms cited in the related work section, in the feature selection field. To show that NBGOA has better performance on problems with various sizes compared to other algorithms, we choose 20 data sets with multiple sizes obtained from the University of California at Irvine (UCI) machine learning repository (Lichman, 2013). Table 1 reports the description of the datasets used in the tests.

For the training/testing methodology, we apply 20-folds cross-validation, each dataset is randomly divided into two sets with different proportions: 80% of the instances in the datasets were used for training task, and 20% are used for the testing task (Friedman et al., 2001).

5.2. Benchmark algorithms and parameters setting

The performance of our developed approach NBGOA is measured and compared to five recent and well-regarded methods in the FS field: the recent binary Grasshopper Optimisation Algorithm (BGOA-M) (Mafarja et al., 2018b), the binary Grey Wolf Optimization Approach (bGWOA) (Emary et al., 2016b) and the Novel Binary Particle Swarm Optimization (NBPSO) (Hoai et al., 2016).

For evaluation, all tests have completed under the same conditions and parameters values. Moreover, for all algorithms, the population size is set to 50 particles, and the number of iterations is set to 100. These values are selected after performing an experimental study, as will be explained in subsection 5.4.

During the test experiments, the 5-Nearest-Neighbour (5-NN) classification algorithm is used, and all statistical results are recorded over 40 independent runs to minimize random-effect. The experiments were performed on a PC with an Intel Core i5-4200U, 1.60 GHz, 4 GB RAM, and Microsoft Windows 10. The parameters of all used algorithms are outlined in Table 2.

5.3. Evaluation criteria

The performance of the proposed approach is evaluated and compared in terms of the following criteria:

a) Average fitness function

The average fitness function is the average value of the fitness function obtained from M runs of the optimization algorithm. Table 3 presents the average fitness function results.

b) Average classification accuracy

The average classification accuracy \( A_{ACC} \) is the average of solutions obtained from M runs of optimization algorithm and can be formulated as in Eq. (18).

\[
A_{ACC} = \frac{1}{M} \sum_{i=1}^{M} ACC_i
\]  

(18)

where \( M \) is the number of runs of the optimization algorithm to select a feature subset. \( ACC \) is the accuracy of the best solution obtained from the \( i^{th} \) run. Table 4 displays the results of Eq. (18) over the testing sets for the different benchmark algorithms used.

c) Average Features selection size

The Average Features Selection Size (AFSS) is the average number of selected features set obtained by the best solutions to the total number of features from \( M \) runs. This criterion can be calcu-
The results of this criterion are reported in Table 5.

\[
\text{AFSS} = \frac{1}{M} \sum_{i=1}^{M} \frac{\text{Size}(i)}{D}
\]  \hspace{1cm} (19)

where \(M\) is the number of runs of the optimization algorithm, \(\text{Size}(i)\) returns the number of features selected in the best solution from the \(i\)th run, and \(D\) is the size of the original dataset.

d) Statistical standard Deviation (StD)

Statistical standard deviation is reported for all approaches to indicate variation and robustness (Emary et al., 2016b) of the obtained best solutions for \(M\) different runs. Whereas StD is small presented in Table 4.

\[
\text{StD} = \sqrt{\frac{1}{M-1} \sum_{i=1}^{M} (\text{Fitness}_i - \text{Mean})^2}
\]  \hspace{1cm} (20)

where \(M\) is the number of runs of the optimization algorithm, \(\text{Fitness}_i\) is the optimal solution resulted from the \(i\)th run, and \(\text{Mean}\) is the average fitness function obtained at the different \(M\) runs of the optimization algorithm, \(\text{Mean}\) is calculated as in Eq. (21).

\[
\text{Mean} = \frac{1}{M} \sum_{i=1}^{M} \text{Fitness}_i
\]  \hspace{1cm} (21)
where \( \text{Fitness}_i \) is the best fitness function value obtained from the \( i \)th run by the optimization algorithm.

e) Average computational time

The Average Computational Time (ACT) is the average of computational time in seconds gained when algorithms run 40 times in the same computing platform, and it is calculated as follows:

\[
ACT = \frac{1}{M} \sum_{i=1}^{M} CT_i
\]

where \( M \) is the number of runs and \( CT_i \) is the value of computational time gained at \( i \)th run.

5.4. Parameters study

To study the impact of the main standard parameters (population size and a number of iterations) in the algorithm on the performance, a set of experiments has been performed. We are using different values of parameters on six datasets: two small datasets (Breastcancer and HeartEW), two medium datasets (SonarEW and Clean 1) and two large datasets (Leukemia and WaveforEW).

The number of iterations parameter is allowed to take five different values: 30, 60, 90, 100 and 120, and population size parameter is allowed to take six different values: 10, 20, 30, 40, 50 and 60.

Figs. 5 and 6 show the performance of NBGOA in terms of fitness function values when varying, respectively, the number of iterations and population size parameters for datasets selected. We observe from Fig. 5 that in 100 iterations the algorithm obtains the best results in most of the cases.

In Fig. 6, a population size of 50 shows best results compared to a smaller population sizes in almost all datasets tested.

5.5. Experimental results and discussions

In this subsection, we present numerical results obtained by NBGOA, as well as those obtained by the other benchmark algorithms; the objective is to compare these results and show the improvements of our algorithm. The comparison was made using the criteria presented in subsection 5.3.

Tables 3–6 present the simulation results obtained for each algorithm and each dataset cited in Table 1.

For each data set, the tested algorithms are run 40 times according to parameters in Table 2.

Note that in the following tables, the best values are formatted in bold underlined.

We firstly examine the average fitness function values of all compared algorithms. The results are shown in Table 3. The average values are calculated using fitness function defined in Eq. (17). From this table, we can notice that the NBGOA algorithm is more efficient than the other algorithms for most of the datasets tested, in terms of the quality of average fitness function values found. Indeed, NBGOA has achieved better results on 12 datasets from 20 datasets tested, and followed by BGOA-M, which gives best results in 6 datasets. But the approaches bGWOA and BDO gets better results for the datasets “German” and “SonarEW” respectively. Besides, our algorithm finds better outcomes in the six largest datasets named “Leukemia”, “ Colon”, “Madelon”, “PenglungEW”, “Arrhythmia” and “Semeion” with more than 265 features.

The performance of the proposed NBGOA is also evaluated on the tested datasets, as shown in Table 4. The table outlines the average classification accuracy and the standard deviations presented in Eq. (18) and Eq. (20) respectively. From 40 runs on each dataset, we can see that NBGOA improves the results for most datasets; it is observed that the NBGOA can outperform all other algorithms in terms of average classification accuracy and standard deviations metrics in 11 and 10 datasets respectively.

BGOA-M comes in the second place where it could outperform other approaches in 5 and 8 datasets respectively, while BDO comes next with 2 and 3 datasets respectively. NBGOA shows the best accuracy results with higher stability because of the small values for the standard deviations metric in half of the datasets and
Table 6
Average computational time.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>NBGOA</th>
<th>BGOA-M</th>
<th>BDO</th>
<th>BGOA</th>
<th>bGWOA</th>
<th>NBPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrhythmia</td>
<td>20.112</td>
<td><strong>20.005</strong></td>
<td>23.526</td>
<td>24.032</td>
<td>25.302</td>
<td>26.32</td>
</tr>
<tr>
<td>Breastcancer</td>
<td><strong>7.124</strong></td>
<td>7.635</td>
<td>8.03</td>
<td>7.895</td>
<td>8.365</td>
<td>9.632</td>
</tr>
<tr>
<td>Clean1</td>
<td>17.987</td>
<td><strong>17.62</strong></td>
<td>19.236</td>
<td>18.026</td>
<td>18.32</td>
<td>21.365</td>
</tr>
<tr>
<td>Colon</td>
<td><strong>53.325</strong></td>
<td>59.302</td>
<td>62.351</td>
<td>61.892</td>
<td>65.325</td>
<td>71.365</td>
</tr>
<tr>
<td>German</td>
<td>4.365</td>
<td><strong>3.689</strong></td>
<td>4.362</td>
<td>4.985</td>
<td>5.032</td>
<td>6.012</td>
</tr>
<tr>
<td>HeartEW</td>
<td>4.203</td>
<td>4.625</td>
<td><strong>4.103</strong></td>
<td>4.205</td>
<td>4.985</td>
<td>5.321</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>5.362</td>
<td>5.461</td>
<td>5.789</td>
<td><strong>5.102</strong></td>
<td>5.398</td>
<td>5.987</td>
</tr>
<tr>
<td>Leukemia</td>
<td><strong>256.632</strong></td>
<td>287.365</td>
<td>298.32</td>
<td>291.0123</td>
<td>312.001</td>
<td>310.235</td>
</tr>
<tr>
<td>Libras</td>
<td>15.023</td>
<td>18.365</td>
<td>19.036</td>
<td>19.001</td>
<td>20.325</td>
<td>22.036</td>
</tr>
<tr>
<td>Madelon</td>
<td>62.351</td>
<td><strong>60.532</strong></td>
<td>64.032</td>
<td>63.235</td>
<td>66.892</td>
<td>67.989</td>
</tr>
<tr>
<td>Parkinsons</td>
<td>4.365</td>
<td><strong>4.025</strong></td>
<td>5.036</td>
<td>4.958</td>
<td>4.985</td>
<td>5.986</td>
</tr>
<tr>
<td>PenglungEW</td>
<td><strong>3.692</strong></td>
<td>4.025</td>
<td>4.362</td>
<td>4.026</td>
<td>5.02</td>
<td>4.032</td>
</tr>
<tr>
<td>Semeion</td>
<td><strong>41.302</strong></td>
<td>45.302</td>
<td>43.325</td>
<td>41.325</td>
<td>42.325</td>
<td>48.325</td>
</tr>
<tr>
<td>Spectft</td>
<td><strong>7.235</strong></td>
<td>7.568</td>
<td>7.469</td>
<td>7.321</td>
<td>8.023</td>
<td>7.89</td>
</tr>
<tr>
<td>WaveforEW</td>
<td><strong>33.265</strong></td>
<td>35.369</td>
<td>38.036</td>
<td>34.025</td>
<td>36.265</td>
<td>38.921</td>
</tr>
<tr>
<td>Zoo</td>
<td>4.036</td>
<td><strong>3.962</strong></td>
<td>4.952</td>
<td>4.325</td>
<td>5.641</td>
<td>5.82</td>
</tr>
<tr>
<td>Total</td>
<td>571.687</td>
<td>616.106</td>
<td>645.126</td>
<td>627.8113</td>
<td>668.441</td>
<td>691.799</td>
</tr>
</tbody>
</table>

Fig. 7. Comparison between proposed NBGOA and other algorithms on all datasets in terms of average classification accuracy.

Fig. 8. Comparison between proposed NBGOA and other algorithms on all datasets in terms of average selection ratio.
especially the datasets of large sizes such as “Leukemia” with 7129 features.

Regarding the average ratio of selected features to the total number of features, Table 5 outlines the average rate of features selected overall datasets for the different algorithms. Our NBGOA algorithm has a much-enhanced performance over the others where it outperformed all other approaches on 11 datasets from 20 datasets and also followed by the BGOA-M algorithm, which outperforms the other algorithms in 6 datasets. It is clear also then our algorithm finds better results in the six largest datasets except the dataset PenglungEW. In general, we notice a significant improvement of our algorithm on large size datasets such as “Leukemia”, “Colon” and “Madelon” with more than 500 features. On the other hand, we notice a decrease of the performance on some small size datasets which are the “HeartEW”, “Tic-tac-toe” and “Zoo”. It can be because of the binary initialization of particles in the swarm.

The results in Tables 4 and 5 prove that the selected features by the NBGOA algorithm are the most relevant features in the dataset. So, these best-obtained results are through the high repulsion rate between grasshoppers and its efficient searching capabilities.

Figs. 7 and 8 show the comparison between NBGOA, and other approaches from literature in terms of average classification accuracy and average selected feature ratio respectively, where our proposed approach outperforms other approaches on many datasets.

In addition, average computational time per second spent on running the proposed NBGOA and all other benchmarked algorithms are shown in Table 6. Inspecting the results in Table 6, it can be seen that, NBGOA is the fastest approach in the same computational environment with other algorithms. NBGOA outperforms all others on around 50% of the datasets with the best ranking, and the binary BGOA-M is placed at the next rank with 30%. On the other hand, the proposed NBGOA provides 571.687 s as a total run time for all datasets, while the full run time provided by BDO, BGOA, bGWOA and NBPSO are 645.126, 627.8113, 668.441 and 691.799 respectively. This considerable improvement by our

### Table 7

P-values of the Wilcoxon test of NBGOA average classification accuracy results vs. other algorithms (P ≤ 0.05 are underlined).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>BGOA-M</th>
<th>BDO</th>
<th>BGOA</th>
<th>bGWOA</th>
<th>NBPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrhythmia</td>
<td>2.86E-4</td>
<td>1.89E-5</td>
<td>2.81E-6</td>
<td>6.87E-2</td>
<td>3.32E-6</td>
</tr>
<tr>
<td>Breastcancer</td>
<td>7.86E-2</td>
<td>5.39E-3</td>
<td>5.61E-4</td>
<td>2.81E-4</td>
<td>4.78E-6</td>
</tr>
<tr>
<td>BreastEW</td>
<td>6.98E-6</td>
<td>7.98E-8</td>
<td>4.24E-7</td>
<td>1.20E-9</td>
<td>2.35E-5</td>
</tr>
<tr>
<td>Cleani</td>
<td>2.68E-8</td>
<td>2.68E-9</td>
<td>5.88E-8</td>
<td>8.64E-9</td>
<td>1.82E-10</td>
</tr>
<tr>
<td>Colon</td>
<td>6.18E-10</td>
<td>7.30E-11</td>
<td>1.24E-9</td>
<td>5.31E-7</td>
<td>5.30E-10</td>
</tr>
<tr>
<td>German</td>
<td>1.64E-5</td>
<td>1.08E-11</td>
<td>2.81E-6</td>
<td>7.20E-9</td>
<td>1.85E-9</td>
</tr>
<tr>
<td>HeartEW</td>
<td>2.34E-6</td>
<td>1.64E-9</td>
<td>8.20E-7</td>
<td>4.32E-8</td>
<td>1.21E-6</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>5.78E-6</td>
<td>6.38E-4</td>
<td>6.12E-4</td>
<td>5.62E-5</td>
<td>2.21E-6</td>
</tr>
<tr>
<td>Leukemia</td>
<td>6.12E-4</td>
<td>9.20E-4</td>
<td>4.98E-5</td>
<td>8.31E-6</td>
<td>5.89E-5</td>
</tr>
<tr>
<td>Libras</td>
<td>7.81E-2</td>
<td>5.25E-4</td>
<td>8.01E-2</td>
<td>8.21E-4</td>
<td>4.01E-4</td>
</tr>
<tr>
<td>Madelon</td>
<td>1.02E-8</td>
<td>9.18E-7</td>
<td>1.54E-10</td>
<td>7.10E-7</td>
<td>2.71E-10</td>
</tr>
<tr>
<td>Parkinsons</td>
<td>9.12E-5</td>
<td>2.05E-7</td>
<td>2.31E-8</td>
<td>7.81E-5</td>
<td>6.32E-6</td>
</tr>
<tr>
<td>PenglungEW</td>
<td>1.02E-7</td>
<td>2.85E-5</td>
<td>1.71E-5</td>
<td>8.62E-5</td>
<td>7.61E-4</td>
</tr>
<tr>
<td>Semeion</td>
<td>3.65E-6</td>
<td>2.31E-4</td>
<td>5.64E-5</td>
<td>1.35E-7</td>
<td>6.65E-5</td>
</tr>
<tr>
<td>SonarEW</td>
<td>1.32E-8</td>
<td>2.00E-6</td>
<td>7.35E-9</td>
<td>4.31E-7</td>
<td>3.11E-8</td>
</tr>
<tr>
<td>SpectEW</td>
<td>5.36E-5</td>
<td>1.42E-7</td>
<td>7.36E-6</td>
<td>7.3E-8</td>
<td>9.97E-6</td>
</tr>
<tr>
<td>Spectft</td>
<td>4.32E-7</td>
<td>2.18E-5</td>
<td>4.85E-6</td>
<td>7.10E-6</td>
<td>1.02E-4</td>
</tr>
<tr>
<td>Tic-tac-toe</td>
<td>7.91E-2</td>
<td>6.32E-2</td>
<td>1.20E-5</td>
<td>3.25E-5</td>
<td>9.35E-4</td>
</tr>
<tr>
<td>WaveforEW</td>
<td>8.62E-3</td>
<td>6.58E-4</td>
<td>5.32E-5</td>
<td>6.31E-5</td>
<td>9.94E-4</td>
</tr>
<tr>
<td>Zoo</td>
<td>1.32E-4</td>
<td>1.27E-4</td>
<td>2.63E-4</td>
<td>2.35E-5</td>
<td>9.34E-2</td>
</tr>
</tbody>
</table>

Fig. 9. Comparison between proposed NBGOA and other algorithms on all datasets in terms of average computational time.
NBGOA is due to the manipulation of binary values compared to other algorithms which manipulate real values and its simple operator used for updating position that depends only on target position. On the other hand, our algorithm does not use transfer functions to convert continuous values to binary values.

Fig. 9 visualizes the comparison between the proposed algorithm and other benchmark algorithms from literature in terms of computational time. In addition, due to the stochastic nature of these tested algorithms, the statistical test is must be done for proving that the results do not randomly happen. So, to confirm the significance and robustness of the results, we apply the Wilcoxon’s statistical test with 5% significance level on the obtained average accuracy results. Table 7 displays Wilcoxon’s test P-values between proposed NBGOA and other tested approaches. It should be noted that the P-values less than 0.05 (underlined in Table 7) indicate that NBGOA results have statistically significant differences compared to other methods. The P-values that are greater than 0.05 are not significant compared to other approaches. According to this table, NBGOA results are statistically significant in most of the data sets. The significant difference between our approach NBGOA and the recently proposed approach BGOA-M by Mafarja et al. is that our approach works in a binary space since the initialization step of the swarm and does not use transfer functions to convert continuous values to binary. Therefore, the frequency of changing the binary bits from 0 to 1 or from 1 to 0 will encourage more exploration of search space. Moreover, the none using of transfer functions by our algorithm improve also the computational time criterion as presented in Table 2.

The good results found and especially in the case of large bases in terms of average accuracy, average selected features number and computational time, show the merits of the proposed algorithms when solving big dimension FS problems.

6. Conclusion

In this work, we have proposed a new binary grasshopper optimization algorithm NBGOA to solve the feature selection problem. Our approach work in binary search space since the initialization step, it initializes the positions of grasshoppers with binary values, and uses simple operators to update the positions. The proposed NBGOA and five other approaches from literature are used to tackle 20 benchmark datasets from UCI datasets. The results of all tested algorithms were compared in terms of several evaluation criteria such as fitness function, average classification accuracy, average feature selected number statistical standard deviation and Wilcoxon’s statistical test (p-values). So, these comparisons show that the proposed algorithm has achieved the best results compared to other tested algorithms. The results proved that NBGOA could achieve the best values in most of the datasets for the five evaluated criteria and especially for the large size datasets such as “Leukemia”, “colon” and “madelon”. It was observed that the binary initialization of the swarm improves the exploration feature in our algorithm and reduces its computational time.

For our future work, the application of the proposed algorithm to other problems that have large size datasets can be investigated such as biomedical, zoo and games data. Furthermore, the hybridization of our algorithm with other recently proposed metaheuristics can also be investigated and can give good results.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References


Please cite this article as: H. Hichem et al. /Journal of King Saud University – Computer and Information Sciences xxx (xxxx) xxx.