Future Computing and Informatics Journal

Volume 3
Issue 1 (2018) *Volume 3, Issue 1, 2018*

Article 3

2018

Simultaneous ranking and selection of keystroke dynamics features through a novel multi-objective binary bat algorithm

Taha M. Mohamed
Faculty of Computers and Information, Helwan University, Egypt, tahamahdy3000@yahoo.com

Hossam M. Moftah
Faculty of Computers and Information, Beni Suef University, Egypt, hossamm@gmail.com

Available online at www.sciencedirect.com

ScienceDirect

Future Computing and Informatics Journal 3 (2018) 29–40
http://www.journals.elsevier.com/future-computing-and-informatics-journal/

Simultaneous ranking and selection of keystroke dynamics features through a novel multi-objective binary bat algorithm

Taha M. Mohamed [a,*], Hossam M. Moftah [b]

[a] *Faculty of Computers and Information, Helwan University, Egypt*
[b] *Faculty of Computers and Information, Beni Suef University, Egypt*

Received 26 October 2017; accepted 14 November 2017
Available online 23 December 2017

Abstract

In this paper, we propose a novel multi-objective binary bat algorithm for simultaneous ranking and selection of keystroke dynamics features. The proposed algorithm uses the *V shaped* binarization function. Simulation results show that, the proposed algorithm can efficiently identify the most important features of the data set. Of the three feature classes, the key down hold time features (H-features) are proofed to be the most dominant features. Using H-features only in classification decreases the mean square error (MSE) by 2% compared to choosing all features in classification. The UD features are the second ranked features. The worst features are the DD features which represent the largest MSE when being used individually in the classification process. The results are performed using two classifiers for comparisons; the linear and the quadratic classifiers. The quadratic classifier outperforms the linear classifier with respect to the mean square error (MSE) and the average number of features selected.

Keywords: Feature selection; Multi-objective bat algorithm; Keystroke dynamics features

1. Introduction

Keystroke dynamics refers to the process of measuring and assessing human typing rhythms by using keyboards, mobile phones, or touch screen panels. The keystroke dynamics can be considered as a type of biometric such as other biometrics including fingerprint, face, and iris. Additionally, the keystroke data may be used to predict some important information about the typist such as the age of the person [1,2]. Impostor attempts to authenticate using a compromised password could be detected and rejected as their typing rhythms differ significantly from those of the genuine user [3].

The keystroke dynamics biometric is economical and can be easily integrated into existing computer security systems with minimal user intervention in order to increase passwords strength. The passwords, when being combined with keystroke timing, add a new difficulty for an attacker who possesses users plaintext password. The advantages of using keystroke dynamics are uniqueness, low implementation, deployment cost, and noninvasiveness. Moreover, using keystroke dynamics doesn't need any extra cost for new hardware. Many of the existing biometric techniques like iris or finger-print recognition are effective only when the person to be authenticated or verified is physically accessible. However, the keystroke dynamics are not depending on the physical existence of the person to be verified.

However, using keystroke rhythms suffers from low accuracy, and low performance [1,4,5]. The European standard for access-control systems specifies a false-alarm rate of less than 1%, with a miss rate of no more than 0.001%. At present, these

* Corresponding author.
E-mail addresses: Tahamahdy3000@yahoo.com (T.M. Mohamed), hossamm@gmail.com (H.M. Moftah).
Peer review under responsibility of Faculty of Computers and Information Technology, Future University in Egypt.

https://doi.org/10.1016/j.fcij.2017.11.005

accuracies are not achieved [3]. Some possible reasons for these deficiencies may be related to the classifiers used, the features used, measurements inaccuracies, and many other factors. We believe that, determining the most important features of the keystroke dynamics is considered as an important step for conforming with the security standards constraints. Compared with the other mature biometrics, keystroke dynamics is still at its very early stages [4]. The feature reduction process is a necessary step to avoid noisy, misleading or irrelevant features. In machine learning, the main objective of attribute reduction is to decrease the dimensionality of feature space, and to enhance the predictive accuracy of classification algorithms [6].

The bat algorithm is a new meta-heuristic algorithm that could be used in feature selection process [8]. This paper presents a novel multi-objective binary bat algorithm used to simultaneous ranking and selection of keystroke dynamics features. The rest of this paper is organized as follows; in section 2, the necessary background is explained. The literature is reviewed in section 3. In section 4, the proposed algorithm is introduced and explained. In section 5, the experimental results are shown and discussed. The paper is concluded in section 6.

2. Background

In this section, we will review the necessary background to the keystroke dynamics data set that we will use throughout the paper, followed by the explanation of the original bat algorithm.

2.1. The data set

One of the most famous, and reliable, keystroke dynamics data sets is the DSN data set [3]. This dataset is considered as a good benchmark to objectively assess the keystroke biometric algorithms [4]. The authors in Ref. [3] collected the data and evaluated 14 different classifiers on the data set. The dataset is available online [7]. In this paper, the proposed algorithm is applied on this dataset. The dataset consists of 51 subjects (typists) typing a certain password. Each subject types the password 400 times in different testing conditions. So, the total number of cases is 20,400 case. The data set includes 31 features that describes the password typed (".tie5Roanl"). This password contains capital letters, small letters, numbers, and special characters. The return key is pressed after typing the password [7]. For each character in the password, the key down hold time (**H Time**), the keydown−keydown time (**DD Time**), the time from when a key was released to when another key was pressed (**UD time**) is measured in seconds. In this paper, we will name these features as H-features, DD features, and UD features respectively. It is noted that, the values of all these features are very small (fractions of seconds). Table 1 shows the different features of the DSN keystroke data set. We give an abbreviation for each feature for later use. For example, the fourth feature, abbreviated as F4, represents the hold time of the letter "t". As noted from the table, all features are either *H, UD, or DD* feature types.

Table 1
Keystroke features notation used.

Abbr.	Feature	Abbr.	Feature
F1	H.period	F17	DD.Shift.r.o
F2	DD.period.t	F18	UD.Shift.r.o
F3	UD.period.t	F19	H.o
F4	H.t	F20	DD.o.a
F5	DD.t.i	F21	UD.o.a
F6	UD.t.i	F22	H.a
F7	H.i	F23	DD.a.n
F8	DD.i.e	F24	UD.a.n
F9	UD.i.e	F25	H.n
F10	H.e	F26	DD.n.l
F11	DD.e.five	F27	UD.n.l
F12	UD.e.five	F28	H.l
F13	H.five	F29	DD.l.Return
F14	DD.five.Shift.r	F30	UD.l.Return
F15	UD.five.Shift.r	F31	H.Return
F16	H.Shift.r		

Till now, most of the work performed on this data set, and other keystroke datasets, is concentrated on evaluation of different classifiers (detectors) rather than evaluating the dataset itself. That is, the most important and relevant features are not identified. To the best of our knowledge, no research is performed to evaluate, and rank, the features of the keystroke dynamics datasets.

2.2. The original bat algorithm

Bats are the only mammals with wings. They are fascinating animals. They also have advanced capability of echolocation. Most bats uses echolocation to a certain degree; among all species, micro-bats use echolocation extensively. Micro-bats use a type of sonar, called, echolocation, to detect prey, avoid obstacles in the dark. These bats emit a very loud sound pulse and listen for the echo that bounces back from the surrounding objects [8].

The bat algorithm may be described as follows [8]:

- *All bats positions are initialized randomly.*
- *Bats randomly fly with velocity v_i at position x_i with a fixed frequency f_{min}, varying wavelength λ and loudness A_0 to search for prey (the optimal solution).*
- *The frequency and the rate of pulse emission r ε [0, 1] are adjusted depending on the proximity of the target.*
- *The loudness varies from a large (positive) A_0 to a minimum constant value A_{min}.*
- *The process is repeated until the optimal solution found.*

The bat algorithm outperforms many other meta-heuristic algorithms, such as PSO and GA, in many optimization fields. So, the bat algorithm is used, and modified, in this paper for selecting, and ranking, the features of the DSN dataset.

3. Literature review

The authors in Refs. [3,7] introduce the DSN, keystroke dynamics, benchmark dataset. They use 14 detectors to evaluate the classification accuracy. However, no ranking information is given to the different features. In Ref. [9], the author presents an anomaly detector for keystroke dynamics authentication, applied on the DSN dataset. They claim that the new detector outperforms the fourteen detectors used in Ref. [3]. The authors in Ref. [10], try to determine the important factors that influence the keystroke-dynamics error rates. They use the DSN benchmark dataset combined with a statistical analysis to validate their model. The authors state that, if the hold times (H-features) are combined with either keydown−keydown times (DD features) or keyup−keydown times (UD features), the classifier output is the same in this case. However, this result is partially agree with the results obtained in our work. Moreover, the authors state that, there are no differences between the three feature types on the detector performance. The results of our work show the incorrectness of their claim.

In Ref. [11], the authors present a new distance metric evaluated on the dataset. They claim that, their proposed distance metric outperforms other algorithms that use traditional distance metrics. In Ref. [4], the authors proposed new classifiers for the data set. They claim that the proposed algorithms outperform a spectrum of top performing keystroke dynamics classifiers. In Ref. [12], the authors investigate the problem of user authentication based on keystroke timing pattern. They propose a nearest neighbor based regression algorithm for anomaly detection applied on the DSN dataset. The authors add two additional features to the existing data set. However, the effect of the new added features is not investigated. In Ref. [13], the authors use the decision trees and support vector machines (SVMs) combined with ants colony optimization (ACO) for feature selection. In Ref. [14], the authors present an extensive literature review on existing benchmark keystroke dynamics datasets. They present several criteria and tests in order to characterize them. The review analysis shows a great disparity in the acquisition protocols, the population involved, the complexity of the passwords, and the expected performance. As noted from the previous review, all research on the keystroke dynamics is concentrated on classifiers rather than the used features.

Regarding the bat algorithm, Yang [8] proposes a new meta-heuristic method called, the bat algorithm (BA), based on the echolocation behavior of bats combining the advantages of existing optimization algorithms. Many research is conducted to evaluate the efficiency of the bat algorithm. Results show that, the bat algorithm outperforms other optimization algorithms such as genetic algorithm and particle swarm optimization [8]. The original bat algorithm proposed by Yang [8] is continuous. However, the feature selection process is a discrete process, in which, a certain feature is chosen or not. Some research is performed to introduce the binary bat algorithm. For example, in Ref. [15], the authors proposed the binary bat algorithm (BBA) used for feature selection. The binarization process uses the sigmoid function for binary bats positions. In Ref. [16], the authors proposed the *V shaped* binary bat algorithm (BBA). The authors claim that, the *V shaped* binarization is more efficient than the sigmoid binarization. The algorithm outperforms others on many benchmark functions. The authors present a real application of the proposed BBA in optical engineering field. To the best of our knowledge, this algorithm is not applied on the feature selection process. In Ref. [6], the authors use the bat algorithm for attribute reduction (BAAR) based on rough sets. However, the algorithm suffers from binarization violation when updating the velocities. In Ref. [17], the authors state that, the classification accuracy of the fuzzy classifiers is better, when combined with the bat algorithm, compared to other meta-heuristic optimization techniques such as PSO, Bee and GA.

From the previous literature review, it is noted that, the bat algorithm outperforms many other meta-heuristic optimization techniques such as PSO and genetic algorithms. Also, there is no known research regarding the selection of the important keystroke dynamics features. Additionally, the *V shaped* binary bat algorithm is not investigated in the feature selection process. So, we propose a new multi-objective binary bat algorithm, based on the *V shaped* function, to select the most relevant features of the DSN data set and also rank them.

4. The proposed algorithm

4.1. The V shaped binary bat algorithm

In the proposed algorithm, we modify the *V shaped* binary bat algorithm (BBA) proposed in Ref. [16]. The original algorithm proposed in Ref. [16] is shown in Fig. 1.

Velocities and positions updating of the bats are given by equations (1) and (2) [16].

$$\mathrm{V}\left(v_i^k(\mathrm{t})\right) = \left|\frac{2}{\pi}\arctan\left(\frac{\pi}{2}v_i^k(t)\right)\right| \tag{1}$$

$$x_i^k(t+1) = \begin{cases} \left(x_i^k(t)\right)^{-1} & \text{If } rand < V\left(v_i^k(t+1)\right) \\ x_i^k(t) & rand \geq V\left(v_i^k(t+1)\right) \end{cases} \tag{2}$$

where $x_i^k(t)$ and $v_i^k(t)$ are the position and the velocity of i-th bat at iteration t in k-th dimension. Here, $\left(x_i^k(t)\right)^{-1}$ is the complement of $x_i^k(t)$. The features are modeled as the dimensions. The V shaped binarization function is shown in equation (1) [16].

4.2. Parameters initialization

In the proposed algorithm, we initialize the bats positions and velocities randomly, the loudness is initialized to a large value. We choose to initialize the loudness to 4. The loudness is decreased when the bats are more nearer from the optimal solution. Typically, we decrease the loudness by a rate of 0.1 in each iteration. The pulse rate is initialized to a small value, typically 0.2. Then, the pulse rate is increased by a rate of 0.1 as the bats are more nearer from the solutions.

Initialize the bat population $X_i (i = 1, 2, \ldots, n)$ and V_i
Define pulse frequency F_i
Initialize pulse rates r_i and the loudness A_i
while (t < Max number of iterations)
  Generate new solutions by adjusting frequency, updating velocities and positions
  if $(rand > r_i)$
    Select a solution among the best solutions randomly
    Generate a local solution around the selected best solution
  end if
  Generate a new solution by flying randomly
  if $(rand < A_i \,\&\, f(x_i) < f(Gbest))$
    Accept the new solutions
    Increase r_i and reduce A_i
  end if
  Rank the bats and find the current $Gbest$
end while

Fig. 1. Pseudo-code of the *V shaped* BBA [16].

4.3. The objective function

The bat algorithm is an optimization algorithm used to maximize or minimize an objective function. In this paper, we use an objective function for minimizing the classification error, or the mean square error (MSE). MSE is given by equation (3) as:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(\widehat{Y}_i - Y_i\right)^2 \tag{3}$$

where Y_i is the class label of pattern i, $\widehat{Y}_i$ is the output of the classifier. For guaranteeing the stability of the classification process, we used the k fold cross validation with $k = 10$. In 10 fold cross validation, the data is divided into 10 subsets each of size $n/10$ such that n is the total number of patterns (here 20,400 patterns). The training process is performed on nine subsets and the testing process is performed on the remaining subset. Then, the MSE is computed identifying the percentage of misclassified patterns relative to all patterns. This process is repeated 10 times and the MSE is calculated. The average MSE is computed for these ten trials. Computing the MSE using cross validation is more robust than computing MSE without folding. The target of the binary bat algorithm is to search for the optimal number of features that are less than or equal to the total number of features in the original dataset. The position of each bat represents a string of binary bits indicating the existence or absence of the features. The objective function is evaluated for each bat to find the optimal solution. Here the optimal solution is the optimal number of features leads to minimizing the classification error.

4.4. Block diagram of the proposed algorithm

The block diagram of the proposed algorithm is shown in Fig. 2. In the beginning, the parameters of the bats are initialized such as the population number (number of bats), bats positions, pulse rate, and loudness. Then, the next step is to evaluate the initial solutions obtained from each bat. These solutions are evaluated using the objective function. The next step is to update the positions and velocities of each bat. Then, the effect of different combinations of population size and max iterations on the performance of the algorithm is studied. Finding the optimal combination of population size and the maximum number of iterations is evaluated. Then, the parameters of the multi-objective function are identified and tested. The multi-objective function allows to identify the most important and relevant features of the dataset. Finally, these features are ranked and evaluated using the multi-objective function.

4.5. The multi-objective function

The proposed algorithm includes a minimization of the multi-objective function. The proposed multi-objective function is defined by equation (4) as:

$$\text{minimize} \sum_{i=1}^{2} w_i f_i(x) \tag{4}$$

Here, the proposed multi-objective function is a linear combination of two functions f_i applied on feature vector x. The first function is the mean square error (MSE) of the classification defined in equation (3). The second function is a function depending on the number of features selected. This function is shown by equation (5).

$$F = \frac{N}{T} \tag{5}$$

where N is the number of the selected features and T is the total number of features. The two functions are weighted with weights w_i. In our experiments, we used $\sum w_i = 1$. The multi-

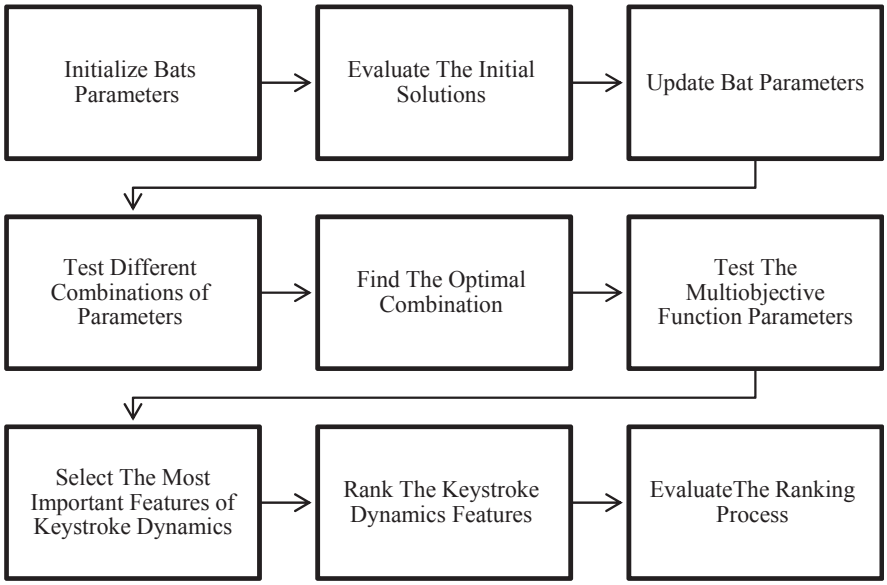


Fig. 2. The block diagram of the proposed algorithm.

objective function works as follows; when w_1 is larger than w_2, then our interest is to minimize the mean square error of the classification rather than minimizing the number of features used. So, this process can identify the most important features in the data set that minimize the MSE. However, in this case, we cannot rank the features itself. On the other hand, if w_1 is smaller than w_2, then, in this case, our interest is to decrease the number of features used rather than decrease the MSE. In this case, the most important features will survive, and appear in different runs of the algorithm. These important features may be viewed as a rich food for bats. The power of combining the two functions, and minimize the multi-objective function, is to achieve the two objectives simultaneously. The bats tries to find an optimal solution that satisfy the minimum *MSE* while minimizing the number of features used. In our experiments, we use different values to the two weights showing the effects on the ranking process. The algorithm of the multi-objective function is shown in Fig. 3.

4.6. The classifiers used

In our experiments, we used two classifiers for comparisons, the linear discriminant classifier (LDC) and the quadratic discriminant classifier (QDC) [18]. The LDC is given by Equation (6) as:

$$D_i^2(y) = \left(y - y_i^-\right)' S_{pl}^{-1} \left(y - y_i^-\right) \tag{6}$$

where S_{pl}^{-1} is the spooled matrix. The quadratic classification function (QDC) is more generic than the linear one. The distance between the test pattern and a certain class, in QDC, is given by Equation (7) as:

$$D_i^2(y) = \left(y - y_i^-\right)' S_i^{-1} \left(y - y_i^-\right) \tag{7}$$

Such that S_i^{-1} is the covariance matrix of class i. In our experiments, we evaluate the *V shaped* binary bat algorithm and the multi-objective function using the two classifiers with respect to the mean square error (MSE) and the classification time using different bat algorithm parameters.

5. Experimental results

In our experiments, we used core i5 machine having 4 GB Ram. We use Matlab R2012a for results computation. The experimental results begin by testing the exhaustive search algorithm for measuring the possibility of searching for the best features with the simple exhaustive search. The data set consists of 31 features. So, the total number of combinations of this features is 2^{31} combinations. To estimate the total time needed, we run the exhaustive search algorithm on all features for 1000 iterations only. The total time consumed to run these 1000 trials is 395 s. So, the total time required to run the complete iterations (2^{31}) is approximately *27* years, which is impossible to be performed in exhaustive search. This complexity of computation is due to the large number of patterns, features, and the time consumed in the cross validation process. So, the exhaustive search is not practical. Thanks to the bat algorithm.

5.1. Using V shaped BBA in feature selection

In this subsection, we will show the results of applying the binary *V shaped* bat algorithm on the data set. We used two classifiers for testing; the linear classifier and the quadratic classifier. There are two important factors that should be considered when using the bat algorithm; The first factor is the population size (i.e. number of bats) used. The second factor is the number of iterations that guarantees convergence. Firstly,

For k =1 to 50

Initialize the bat population $X_i (i = 1, 2, \ldots, n)$ and V_i
Define pulse frequency F_i
Initialize the pulse rates r_i and the loudness A_i
while (t < Max number of iterations)
fitness = w_1*fn_1+w_2*fn_2

Generate new solutions by adjusting frequency, updating velocities and positions
if $(rand > r_i)$
Select a solution among the best solutions randomly
Generate a local solution around the selected best solution
end if
Generate a new solution by flying randomly
if $(rand < A_i \ \& \ f(x_i) < f(Gbest))$
Accept the new solutions
Increase r_i and reduce A_i
end if
Rank the bats and find the current $Gbest$

end while

count f_t = count of feature f_t

end for

probability of occurrence = count (I) /50

Rank features according to probability of occurrence

Fig. 3. The multi-objective binary bat algorithm.

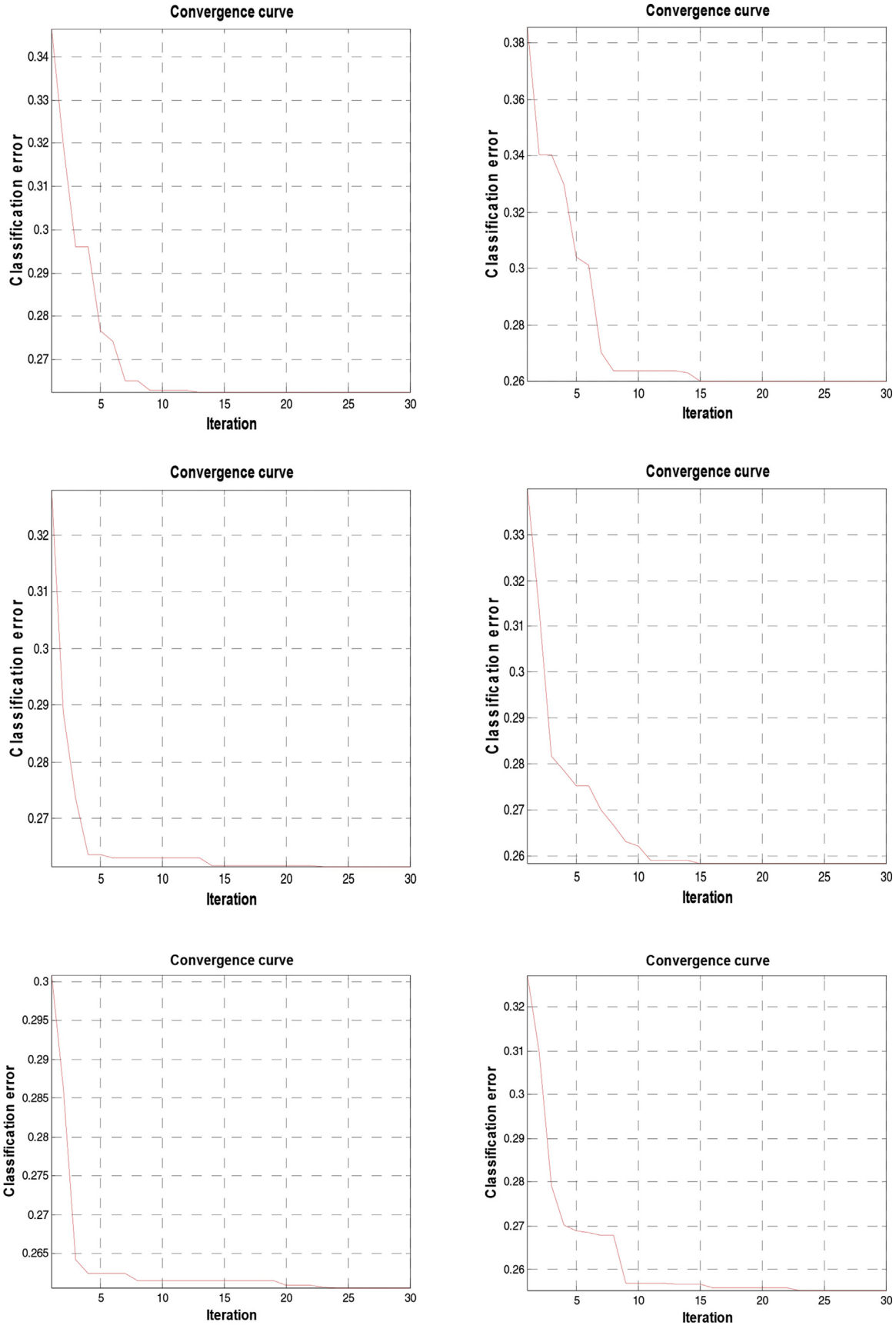
we test the effect of the number of bats (population size) on the convergence of the algorithm.

Table 2 shows the results of using different population sizes (varying from 10 to 50) using two used classifiers. In Table 2, the number of iterations is always fixed to 30 for all trials. It is noted from the table that, the quadratic classifier is slightly better in performance than the linear classifier with respect to the mean square error (MSE). It is also noted that, the average MSE results of the two classifiers in all trials do not change with different parameters (0.26 for the linear classifier and 0.25 for the quadratic classifier). This is due to the use of cross validation which makes the results more stable. Moreover, this also ensures the convergence of the algorithm. For the classification time, it is noted that, increasing the population size (total number of bats) increases the time complexity of the algorithm. Moreover, the total time required to run the algorithm using the quadratic classifier is less than the time required to run the linear classifier. For the features selected, it is noted that the average number of selected features using the quadratic classifier is less than that of the linear classifier. So, the conclusions from Table 2 are that, increasing the total number of bats increases the total execution time of the algorithm. The quadratic classifier when applied to this data set is more better than the linear classifier with respect to classification accuracy and classification time. Additionally, the features selected by the quadratic classifier are lower than the features selected by the linear classifier. The average of all trials is shown in the last row of Table 2.

Table 2
Effect of changing population size.

Pop. size	MSE		Running time (s)		No. of features	
	Linear	Quad.	Linear	Quad.	Linear	Quad.
10	0.26	0.25	681	668	27	19
20	0.26	0.25	1364	1318	25	17
30	0.26	0.25	2052	2004	26	18
40	0.26	0.25	2893	2876	25	17
50	0.26	0.25	3588	3507	27	18
Avg.	**0.26**	**0.25**	**2115**	**2074**	**26**	**17.8**

Fig. 4 show the convergence curves of some of the experiments of Table 2 using both classifiers. The top graphs in the figure represent the minimum population size. The bottom graphs in the figure represent the maximum population size. It is noted from Fig. 4 that, the convergence occurred in all cases when the maximum number of iterations equals 30 (because the maximum number of iterations is fixed in this test).



(a) Linear classifier (b) Quadratic classifier

Fig. 4. The convergence of the iterations of Table 2.

However, increasing the population size speed up the convergence process at the cost of increasing the running time of the algorithm.

Table 3 shows the results of changing the number of iterations keeping the number of bats fixed (the number of bats is fixed to 30). It is noted from Table 3 that, the convergence occurs in all cases as well. The MSE resulted of the two classifiers are the same as those obtained in Table 2 ensuring the convergence of the algorithm. However, using a small number of iterations decreases the running time of the algorithm. The average number of the selected features is approximately the same as these shown in Table 2. Additionally, the average running time of the algorithm is approximately the same as these shown in Table 2.

Fig. 5 shows the convergence curves of some of the experiments described in Table 3 using both classifiers. Again, the top graphs in the figure represent lower number of iterations. The bottom graphs in the figure represent the maximum number of iterations. It is noted from Fig. 5 that, the convergence occurred in all cases when the maximum number of iterations is less than or equal to 30 regardless of the population size (because the maximum number of iterations is fixed in this test). In some cases, the convergence is occurred when the number of iterations is less than 10 indicating the efficient convergence of the algorithm. The conclusions from Tables 2 and 3 are that, the algorithm can converge using a small number of iterations and a small population size; typically (20, 30) respectively. Another important conclusion is that, the quadratic classifier is better than the linear classifier with respect to the classification error and the number of features selected.

5.2. The multi-objective binary bat algorithm

Results from the previous subsection show that, the selected features vary according to the different parameters of the bat algorithm. The results obtained so far determines the best feature sets leading to best classification results (i.e. least MSE). However, the rank of the features itself is not identified. Moreover, the relationship between these features is not discovered. The proposed multi-objective algorithm is used to optimize the feature selection problem by decreasing the number of features while decreasing the cost function (MSE). Another important target of the multi-objective function is to rank the features and discover the most important and relevant features in the dataset. To test the multi-objective function, we performed 50 experiments using different values for the weighting factors w_1, w_2 to determine the most important features.

Table 3
Effect of changing No. of iterations.

Max iterations	MSE		Running time (s)		No. of features	
	Linear	Quad.	Linear	Quad.	Linear	Quad.
10	0.26	0.25	713	699	25	17
20	0.26	0.25	1393	1340	27	18
30	0.26	0.25	2052	2004	26	18
40	0.26	0.26	2822	2800	26	18
50	0.26	0.25	3578	3483	26	17
Avg.	**0.26**	**0.25**	**2111**	**2065**	**26**	**17.6**

The experiments of the multi-objective function are performed using the quadratic classifier due to its superiority over the linear classifier as previously shown. In these experiments, we used the population number equal 30. The algorithm is conducted by using 20 iterations to ensure convergence and covering of the search space while maintaining the classification time minimized. The fifty experiments are grouped into 5 groups labeled from 1 to 5. Each group uses certain weights w_1, w_2. Ten experiments are conducted for each group, and the averages are computed for each group individually. As expected, the number of the selected features increases as d increases, where $d = w_1 - w_2$. However, the classification error (MSE) decreases in this case.

Table 4 shows the different values of both w_1, w_2 and the average number of features resulted. We choose $w_1 + w_2 = 1$ in all cases to explore the relationship between the two functions that participating in the multi-objective function. The table shows the **MOCost** which is the cost of the multi-objective function, and the **MSEClass** which is the classification MSE. It is noted that, when w_1, w_2 are 0.9, 0.1 respectively, the MSE is 0.26 which equal the MSE resulted from the linear classifier. However, the benefit in this case is that, the average number of features selected, *nf*, is 15 rather than 26 in the case of linear classifier.

The convergence of a sample experiment of the multi-objective function is shown in Fig. 6. It is noted that, the convergence is approximately exponentially decayed. The convergence occurred after 14 iterations only. In these experiments, the population size is fixed to 30.

Fig. 7 shows the features selected using the multi-objective experiments that are shown in Table 4. The figure shows the relationship between the features selected and the group. For example, it is noted that, feature F31 is always selected by both groups 4 and 5. The same feature is partially selected by group 2 and group 1. So, this feature may be considered as a somewhat important feature. This process could be viewed as a voting process on the features. Feature 30 (F30) is not selected by any group. So, this feature can be viewed as a non important feature. It is interesting to note that, the most selected features are the H-features as illustrated in Table 5. Fig. 8 shows the average of the features selected by all groups for more clarification. Again, if a feature is always selected by all groups then this feature is important. If the feature is partially selected, then the feature is somewhat important. However, if the feature is not selected at all, then this feature is not important.

Table 5 shows the rank of the features according to the average of all groups. The features are ranked to either *essential*, *strong*, *moderate*, *weak*, or *rare* features. The ranking is based on the frequency of each feature over all groups. The essential features are the features appear in all trials; that appears 50 times in the 50 experiments. The strong features are features having the repetitions from 40 to 49. The moderate features are the features with the repetitions between

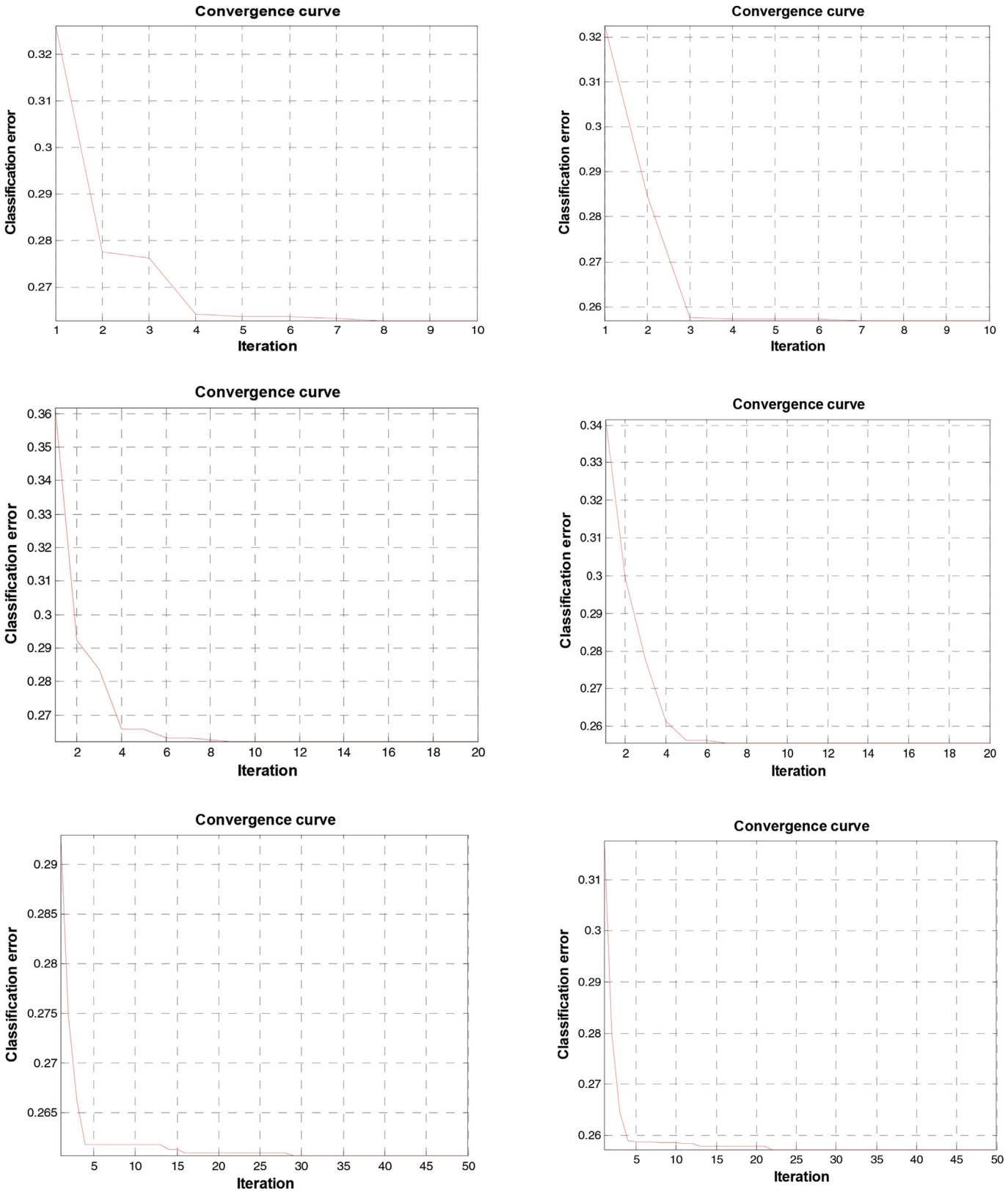


Fig. 5. Convergence of iterations of Table 3.

Table 4
Multi-objective experiments.

Group	w_1	w_2	MOCost	nf	MSEClass
1	0.5	0.5	0.31	7	0.40
2	0.6	0.4	0.32	9	0.34
3	0.7	0.3	0.31	11	0.28
4	0.8	0.2	0.30	13	0.27
5	0.9	0.1	0.28	15	0.26

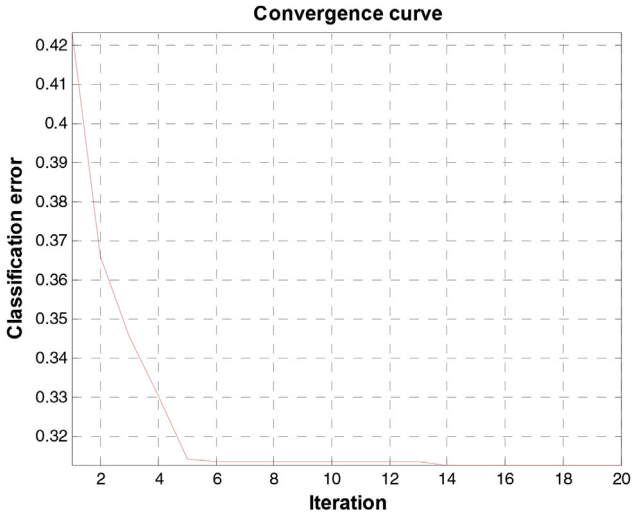


Fig. 6. Convergence of the multi-objective function.

30 and 39. The weak features are the features with the repetitions between 1 and 30. Finally, the rare features are the features with zero repetitions (i.e. not being selected in any experiments). From Table 5, all essential, strong, and moderate features are of H features types. However, the weak and rare features are all of type UD, and DD. Interestingly, UD and DD are not ranked with the same rank. The UD features are more frequent in the weak features than the DD features as shown in the table. The share of rare features are equally divided between UD and DD features.

Table 6 shows the effect of the different feature sets on the classification. When using H-features only, the MSE is 0.30. In this case, the number of features (nf) is 11 as defined by the data set. When using both UD and DD features, the MSE increases to 0.66. When using UD features only, the MSE decreases to 0.65. However, by using DD features only, the MSE increases to 0.71. However, when using all features, the MSE increases to 0.32. The important note here is that, using the H features only achieves a reduction of 0.02 in the MSE compared to using all features in the classification. Moreover, the total number of the H features is approximately one third of the total number of features. So, the H features are the more dominating and relevant features. Suggesting that, using H-features only is more efficient than using all features of the data set. The classification accuracy may be enhanced by adding other new features that may be combined with the H-features to enhance the classification accuracy.

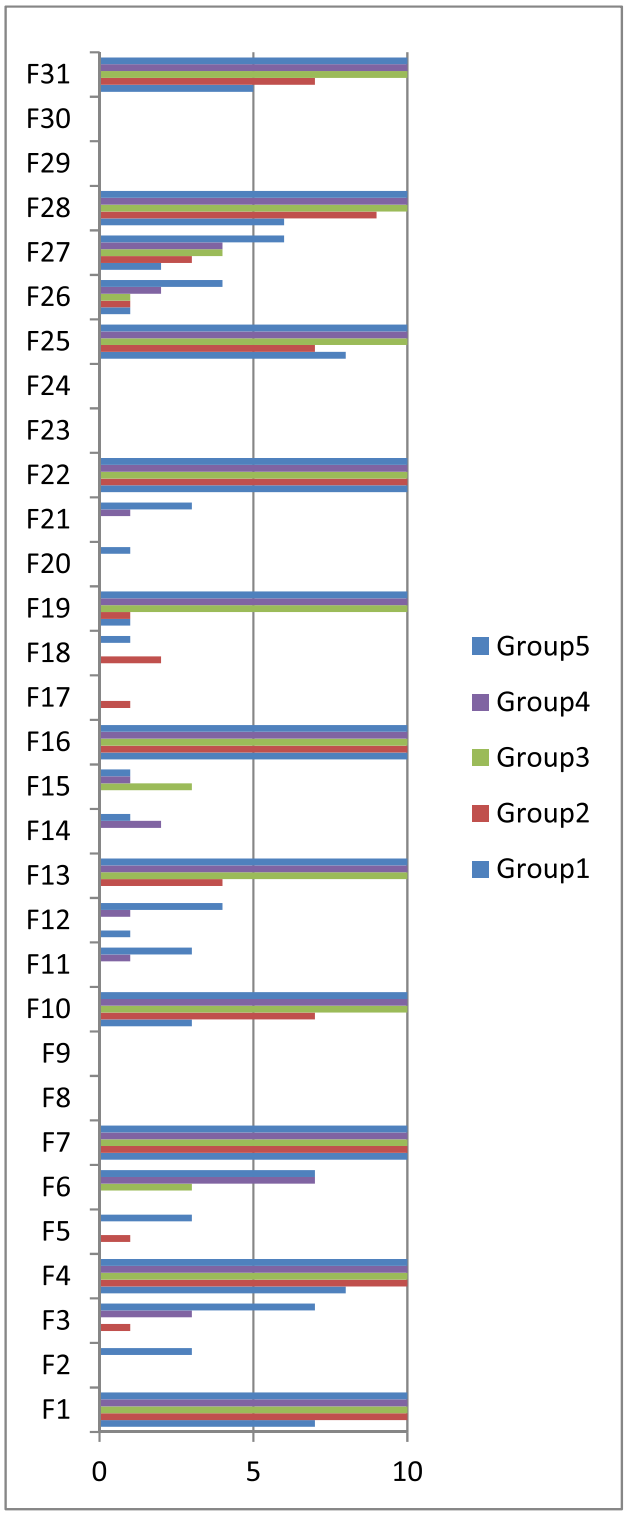


Fig. 7. Multi-objective experiments.

5.3. Results discussion

The exhaustive search process for finding the best features of the keystroke dynamics is time consuming process. The solution is to use a meta-heuristic algorithm such as the bat algorithm. In our experiments, we use the MSE as an objective function. Two classifiers are used for testing the validity of the

Table 5
Ranking of features.

	Essential	Strong	Moderate	Weak	Rare
Features	F7, F16, F22	F1, F4, F10, F25, F28, F31	F13, F19	F2, F3, F5, F6, F11, F12, F14, F15, F17, F18, F20, F21, F26, F27	F8, F9, F23, F24, F29, F30
Comment	All are H	All are H	All are H	All are DD, UD	All are DD, UD

algorithm; the linear classifier and the quadratic classifiers. The data set of the keystroke dynamics consists of 31 features. The bat algorithm has two important parameters; the population size (i.e. the number of bats) and the number of iterations. It is noted that, the convergence occurs in all experiments using small number of iterations (typically less than 30 iterations). It is also noted that, the larger the population size, the more rapid the convergence of the algorithm. As expected, increasing the population size and the number of iterations will increase the execution time of the algorithm. It is also noted that, the MSE of the quadratic classifier is less than that of the linear classifier. Moreover, the average number of features selected by the quadratic classifier is less than that of the linear classifier. We choose the number of iterations and the population size to be (20, 30) respectively. The main disadvantage of the single objective function (i.e. MSE) is that; the important features cannot be identified or ranked.

The proposed multi objective algorithm is used to optimize the feature selection problem by decreasing the number of features while decreasing the cost function (MSE) simultaneously. Another important target of the multi-objective function is to rank the features and discover the most important and relevant features. The experiments of the multi-objective function are performed using the quadratic classifier due to its superiority over the linear classifier. Fifty experiments are performed and grouped into 5 groups labeled from 1 to 5 based on the multi-objective parameters. When the weighting factor of the MSE cost function is small, and the weighting factor of the number of features function is large, in this case, our concern is to identify the most important features of the data set. Changing these weighting factors may be seen as a voting process for different features.

The multi-objective function gives the possibility of ranking the features according to the average frequency of occurrence in all groups. The features are ranked to either essential, strong, moderate, weak, and rare features. It is noted that, all essential, strong, and moderate features are all of key down hold time features (H-features) type. However, the weak and rare features are all of type UD, and DD. The UD features are more frequent than the DD features. Regarding the MSE, when using H-features only, the MSE is lower than the MSE resulted from using all features. In this case, the total number of the H-features is approximately one third of the total number of features in the data set. The feature reduction in this case helps in saving the computation and simplifying algorithm complexity.

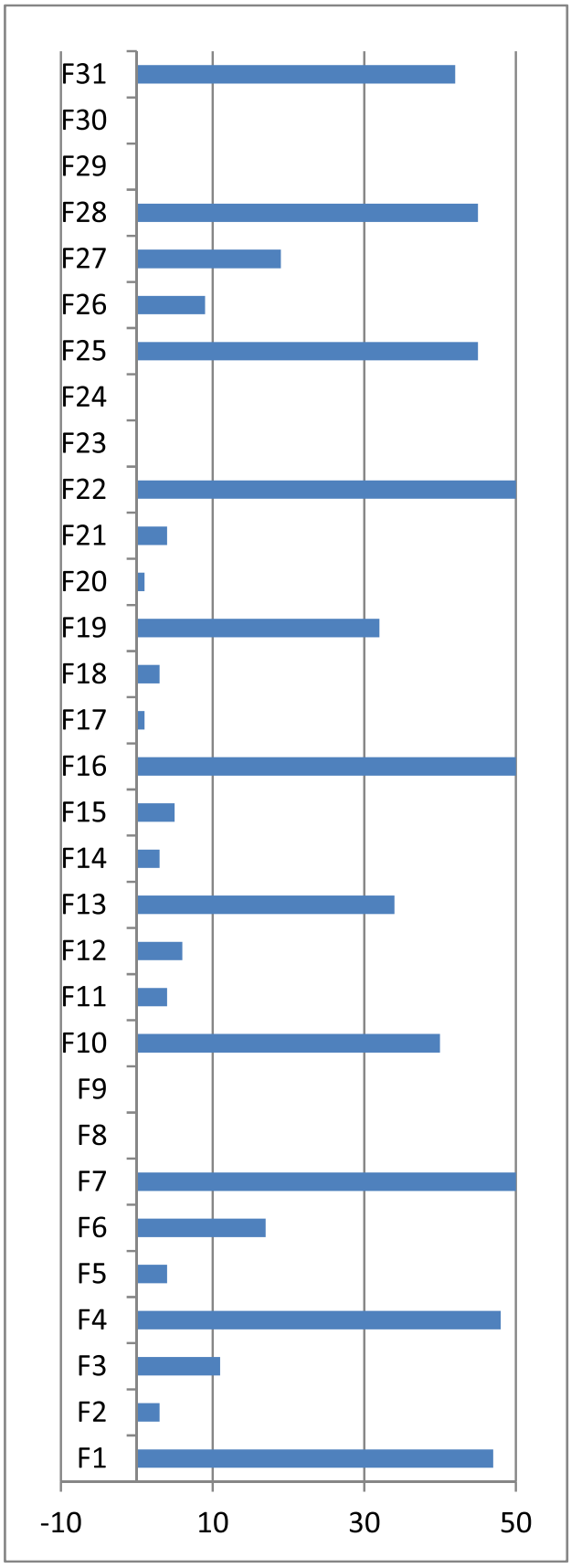


Fig. 8. The average of all groups.

Table 6
Ranking of features according to MSE.

H features only		UD and DD features		UD features only		DD features only		All features	
MSE	*nf*	MSE	*nf*	MSE	*nf*	MSE	*nf*	MSE	*nf*
0.30	11	0.66	20	0.65	10	0.71	10	0.32	33

6. Conclusions

In this paper, we propose a new multi-objective binary bat algorithm based on the *V shaped* binarization function applied on a keystroke rhythms data set. Results show that, the proposed algorithm could efficiently identify and rank the keystroke important features. The key down hold time features (H-features) give the highest rank. The second highest rank are the UD features. Finally, the DD features give the minimal rank. An important result is that, only H-features could be used for keystroke dynamics classification. In this case, the classification accuracy is more better than using all features of the dataset. Additionally, two thirds of dataset features could be ignored that decreases the execution time, and storage, of the algorithm. These results may represent a motivation for the researchers to search for other keystroke dynamics features to be combined with H-features, to enhance the classification accuracy replacing UD and DD features. Some future suggestions for new features may include the typing sequence difficulty, the frequency of typing error, the age of the typist, and the gender of the typist.

References

[1] Teh PS, Teoh ABJ, Yue S. A survey of keystroke dynamics biometrics. Scientific World Journal Nov 2013;2013. Hindawi Publishing Corporation.

[2] Tsimperidis I, Rostami S, Katos V. Age detection through keystroke dynamics from user authentication failures. Int J Digit Crime Forensics 2017;9(1):1–16.

[3] Killourhy KS, Maxion RA. Comparing anomaly-detection algorithms for keystroke dynamics. In: Dependable systems & networks, DSN'09. IEEE/IFIP international conference. IEEE; 2009. p. 125–34.

[4] Deng Y, Zhong Y. Keystroke dynamics user authentication based on Gaussian mixture model and deep belief nets. ISRN Signal Process 2013; 2013. Hindawi Publishing Corporation.

[5] Bhattasali T, Panasiuk P, Saeed K, Chaki N, Chaki R. Modular logic of authentication using dynamic keystroke pattern analysis. In: American Institute of Physics Conference, vol. 1738 (1); 2016.

[6] Taha AM, Tang AYC. Bat algorithm for rough set attribute reduction. J Theor Appl Inf Technol (JATIT) 2013;51(1):1–8.

[7] Keystroke dynamics benchmark dataset is available at http://www.cs.cmu.edu/~keystroke/, Last visit on 29 May 2017.

[8] Yang XS. A new metaheuristic bat-inspired algorithm. In: Nature inspired cooperative strategies for optimization (NICSO), vol. 284. J.R. González, Springer Berlin Heidelberg; 2010. p. 65–74.

[9] Al-Jarrah MM. An anomaly detector for keystroke dynamics based on medians vector proximity. Emerg Trends Comput Inf Sci 2012;3(6): 988–93.

[10] Killourhy KS, Maxion RA. Why did my detector do that?! Predicting keystroke-dynamics error rates. In: Recent advances in intrusion detection (RAID 2010). Berlin: Springer-Verlag; 2010. p. 256–76.

[11] Zhong Y, Deng Y, Jain AK. Keystroke dynamics for user authentication. In: Computer vision and pattern recognition workshops (CVPRW). IEEE; 2012. p. 117–23.

[12] Maheshwary S, Pudi V. Mining keystroke timing pattern for user authentication. In: 5th international workshop on new frontiers in mining complex patterns. Italy: Department of Computer Science, University of Bari "Aldo Moro"; 2016.

[13] Alsultan A, Warwick K, Wei H. Non-conventional keystroke dynamics for user authentication. Pattern Recognit Lett 2017;89(1):53–9. Elsevier.

[14] Giot R, Dorizzi B, Rosenberger C. A review on the public benchmark databases for static keystroke dynamics. Comput Secur 2015;55:46–61. Elsevier.

[15] Nakamura RYM, Pereira LAM, Costa KA, Rodrigues D, Papa JP, Yang X-S. BBA: a binary bat algorithm for feature selection. In: Graphics, patterns and images (SIBGRAPI), 25th SIBGRAPI conference. IEEE; 2012. p. 291–7.

[16] Mirjalili S, Mirjalili SM, X.S Yang. Binary bat algorithm. Neural Comput Appl 2014;25(3):663–81. Springer.

[17] Parashar S, Senthilnath J, Yang X-S. A novel bat algorithm fuzzy classifier approach for classification problems. Int J Artif Intell Soft Comput 2017.

[18] Rencher A. Methods of multivariate analysis. 2nd ed. Brigham Young University, John Wiley & Sons Publication; 2002.