BBA: A Binary Bat Algorithm for Feature Selection

R. Y. M. Nakamura, L. A. M. Pereira, K. A. Costa, D. Rodrigues, J. P. Papa
Department of Computing
São Paulo State University
Bauru, Brazil

X.-S. Yang
National Physical Laboratory
London, UK

***Abstract*—Feature selection aims to find the most important information from a given set of features. As this task can be seen as an optimization problem, the combinatorial growth of the possible solutions may be in-viable for a exhaustive search. In this paper we propose a new nature-inspired feature selection technique based on the bats behaviour, which has never been applied to this context so far. The wrapper approach combines the power of exploration of the bats together with the speed of the Optimum-Path Forest classifier to find the set of features that maximizes the accuracy in a validating set. Experiments conducted in five public datasets have demonstrated that the proposed approach can outperform some well-known swarm-based techniques.**

***Keywords*-Feature Selection, Bat Algorithm, Optimum-Path Forest**

I. Introduction

Feature selection attempts to find the most discriminative information in several application domains. It is often desirable to find features that are simple to extract, invariant to geometric and affine transformations, insensitive to noise and also useful for characterizing patterns in different categories [1]. The choice of such features is a critical step and strongly depends on the nature of the problem.

Once an exhaustive search for the optimal set of features in a high dimensional space may be impracticable, several works attempt to model the feature selection as a combinatorial optimization problem, in which the set of features that leads to the best feature space separability is then employed to map the original dataset to a new one. The objective function can be the accuracy of a given classifier or some other criterion that may consider the best trade-off between feature extraction computational burden and effectiveness.

Several heuristic algorithms derived from the behaviour of biological and/or physical systems in the nature have been proposed as powerful methods for global optimizations. Kennedy and Eberhart [2] proposed a binary version of the well-known Particle Swarm Optimization (PSO) [3] called BPSO, in which the traditional PSO algorithm was modified in order to handle binary optimization problems. Further, Firpi and Goodman [4] extended BPSO to the context of feature selection.

Further, Rashedi et al. [5] proposed a binary version of the Gravitational Search Algorithm (GSA) [6] called BGSA for feature selection, and Ramos et al. [7] presented their version of the Harmony Search (HS) [8] for the same purpose in the context of theft detection in power distribution systems. In addition, several works have addressed the same problem with a wide range of nature-inspired techniques [9], [10], [11].

Very recently, Yang [12] proposed a new meta-heuristic method for continuous optimization namely Bat Algorithm (BA), which is based on the fascinating capability of micro-bats in to find their prey and discriminate different types of insects even in complete darkness. Such approach has demonstrated to outperform some well-known nature-inspired optimization techniques. In this paper, we propose a binary version of the Bat Algorithm for feature selection purposes called BBA, in which the search space is modelled as a n-cube, where n stands for the number of features. In such case, the optimal (near-optimal) solution is chosen among the 2^n possibilities, and it corresponds to one hypercube's corner.

The main idea is to associate for each bat a set of binary coordinates that denote whether a feature will belong to the final set of features or not. The function to be maximized is the one given by a supervised classifier's accuracy. As the quality of the solution is related with the number of bats, we need to evaluate each one of them by training a classifier with the selected features encoded by the bat's position and also to classifying an evaluating set. Thus, we need a fast and robust classifier, since we have one instance of it for each bat. As such, we opted to use the Optimum-Path Forest (OPF) classifier [13], [14], which has been demonstrated to be so effective as Support Vector Machines, but faster for training. The experiments have been performed in five public datasets against Particle Swarm Optimization, Harmony Search, Gravitational Search Algorithm and Firefly Algorithm in order to evaluate the robustness of the proposed technique.

The remainder of the paper is organized as follows. In Section II we revisit the Bat Algorithm and the Optimum-Path Forest theory background. Section III presents the BBA for feature selection and experimental results are discussed in Section IV. Finally, conclusions are stated in Section V.

II. Feature Selection using Bat Algorithm

In this section we describe the BA and BBA approaches as well as the OPF classifier.

A. Bat Algorithm

Bats are fascinating animals and their advanced capability of echolocation have attracted attention of researchers from different fields. Echolocation works as a type of sonar: bats, mainly micro-bats, emit a loud and short pulse of sound, wait

it hits into an object and, after a fraction of time, the echo returns back to their ears [15]. Thus, bats can compute how far they are from an object [16]. In addition, this amazing orientation mechanism makes bats being able to distinguish the difference between an obstacle and a prey, allowing them to hunt even in complete darkness [17].

Based on the behaviour of the bats, Yang [12] has developed a new and interesting meta-heuristic optimization technique called Bat Algorithm. Such technique has been developed to behave as a band of bats tracking prey/foods using their capability of echolocation. In order to model this algorithm, Yang [12] has idealized some rules, as follows:

1) All bats use echolocation to sense distance, and they also "know" the difference between food/prey and background barriers in some magical way;
2) A bat b_i fly randomly 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. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission $r \in [0, 1]$, depending on the proximity of their target;
3) Although the loudness can vary in many ways, Yang [12] assume that the loudness varies from a large (positive) A_0 to a minimum constant value A_{min}.

Algorithm 1 presents the Bat Algorithm (adapted from [12]):

Algorithm 1. – BAT ALGORITHM

Objective function $f(x)$, $x = (x^1, ..., x^n)$.
Initialize the bat population x_i *and* v_i, $i = 1, 2, ..., m$.
Define pulse frequency f_i *at* x_i, $\forall i = 1, 2, \ldots, m$.
Initialize pulse rates r_i *and the loudness* A_i, $i = 1, 2, \ldots, m$.

1. *While* $t < T$
2. *For each bat* b_i, *do*
3. *Generate new solutions through Equations (1),*
4. *(2) and (3).*
5. *If* $rand > r_i$, *then*
6. *Select a solution among the best solutions.*
7. *Generate a local solution around the*
8. *best solution.*
9. *If* $rand < A_i$ *and* $f(x_i) < f(\hat{x})$, *then*
10. *Accept the new solutions.*
11. *Increase* r_i *and reduce* A_i.
12. *Rank the bats and find the current best* $\hat{x}$.

Firstly, the initial position x_i, velocity v_i and frequency f_i are initialized for each bat b_i. For each time step t, being T the maximum number of iterations, the movement of the virtual bats is given by updating their velocity and position using Equations 1, 2 and 3, as follows:

$$f_i = f_{\min} + (f_{\min} - f_{\max})\beta, \tag{1}$$

$$v_i^j(t) = v_i^j(t-1) + [\hat{x}^j - x_i^j(t-1)]f_i, \tag{2}$$

$$x_i^j(t) = x_i^j(t-1) + v_i^j(t), \tag{3}$$

where β denotes a randomly generated number within the interval $[0, 1]$. Recall that $x_i^j(t)$ denotes the value of decision variable j for bat i at time step t. The result of f_i (Equation 1) is used to control the pace and range of the movement of the bats. The variable $\hat{x}^j$ represents the current global best location (solution) for decision variable j, which is achieved comparing all the solutions provided by the m bats.

In order to improve the variability of the possible solutions, Yang [12] has proposed to employ random walks. Primarily, one solution is selected among the current best solutions, and then the random walk is applied in order to generate a new solution for each bat that accepts the condition in Line 5 of Algorithm 1:

$$x_{new} = x_{old} + \epsilon\overline{A}(t), \tag{4}$$

in which $\overline{A}(t)$ stands for the average loudness of all the bats at time t, and $\epsilon \in [-1, 1]$ attempts to the direction and strength of the random walk. For each iteration of the algorithm, the loudness A_i and the emission pulse rate r_i are updated, as follows:

$$A_i(t+1) = \alpha A_i(t) \tag{5}$$

and

$$r_i(t+1) = r_i(0)[1 - exp(-\gamma t)], \tag{6}$$

where α and γ are ad-hoc constants. At the first step of the algorithm, the emission rate $r_i(0)$ and the loudness $A_i(0)$ are often randomly chosen. Generally, $A_i(0) \in [1, 2]$ and $r_i(0) \in [0, 1]$ [12].

B. BBA: Binary Bat Algorithm

As the reader can observe, each bat moves in the search space towards continuous-valued positions. However, in case of feature selection, the search space is modelled as a n-dimensional boolean lattice, in which the bat moves across the corners of a hypercube. Since the problem is to select or not a given feature, the bat's position is then represented by binary vectors.

This paper proposes a binary version of the Bat Algorithm restricting the new bat's position to only binary values using a sigmoid function:

$$S(v_i^j) = \frac{1}{1 + e^{-v_i^j}}. \tag{7}$$

Therefore, Equation 3 can be replaced by:

$$x_i^j = \begin{cases} 1 & \text{if } S(v_i^j) > \sigma, \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

in which $\sigma \sim U(0, 1)$. Therefore, Equation 8 can provide only binary values for each bat's coordinates in the boolean lattice, which stand for the presence of absence of the features.

C. Supervised Classification Through Optimum-Path Forest

The OPF classifier works by modelling the problem of pattern recognition as a graph partition in a given feature space. The nodes are represented by the feature vectors and the edges connect all pairs of them, defining a full connectedness graph. This kind of representation is straightforward, given that the graph does not need to be explicitly represented, allowing

us to save memory. The partition of the graph is carried out by a competition process between some key samples (prototypes), which offer optimum paths to the remaining nodes of the graph. Each prototype sample defines its optimum-path tree (OPT), and the collection of all OPTs defines an optimum-path forest, which gives the name to the classifier [13], [14].

The OPF can be seen as a generalization of the well known Dijkstra's algorithm to compute optimum paths from a source node to the remaining ones [18]. The main difference relies on the fact that OPF uses a set of source nodes (prototypes) with any smooth path-cost function [19]. In case of Dijkstra's algorithm, a function that summed the arc-weights along a path was applied. In regard to the supervised OPF version addressed here, we have used a function that gives the maximum arc-weight along a path, as explained below.

Let $Z = Z_1 \cup Z_2 \cup Z_3 \cup Z_4$ be a dataset labelled with a function λ, in which Z_1, Z_2, Z_3 and Z_4 are, respectively, a training, learning, evaluating and test sets. Let $S \subseteq Z_1$ a set of prototype samples. Essentially, the OPF classifier creates a discrete optimal partition of the feature space such that any sample $s \in Z_2 \cup Z_3 \cup Z_4$ can be classified according to this partition. This partition is an optimum path forest (OPF) computed in $\Re^n$ by the Image Foresting Transform (IFT) algorithm [19].

The OPF algorithm may be used with any *smooth* path-cost function which can group samples with similar properties [19]. Particularly, we used the path-cost function $f_{\max}$, which is computed as follows:

$$\begin{aligned} f_{\max}(\langle s \rangle) &= \begin{cases} 0 & \text{if } s \in S, \\ +\infty & \text{otherwise} \end{cases} \\ f_{\max}(\pi \cdot \langle s,t \rangle) &= \max\{f_{\max}(\pi), d(s,t)\}, \end{aligned} \quad (9)$$

in which $d(s,t)$ means the distance between samples s and t, and a path π is defined as a sequence of adjacent samples. In such a way, we have that $f_{\max}(\pi)$ computes the maximum distance between adjacent samples in π, when π is not a trivial path.

The OPF algorithm works with a training and a testing phase. In the former step, the competition process begins with the prototypes computation. We are interested into finding the elements that fall on the boundary of the classes with different labels. For that purpose, we can compute a Minimum Spanning Tree (MST) over the original graph and then mark as prototypes the connected elements with different labels. Figure 1b displays the MST with the prototypes at the boundary. After that, we can begin the competition process between prototypes in order to build the optimum-path forest, as displayed in Figure 1c. The classification phase is conducted by taking a sample from the test set (blue triangle in Figure 1d) and connecting it to all training samples. The distance to all training nodes are computed and used to weight the edges. Finally, each training node offers to the test sample a cost given by a path-cost function (maximum arc-weight along a path - Equation 9), and the training node that has offered the minimum path-cost will conquer the test sample. This procedure is shown in Figure 1e.

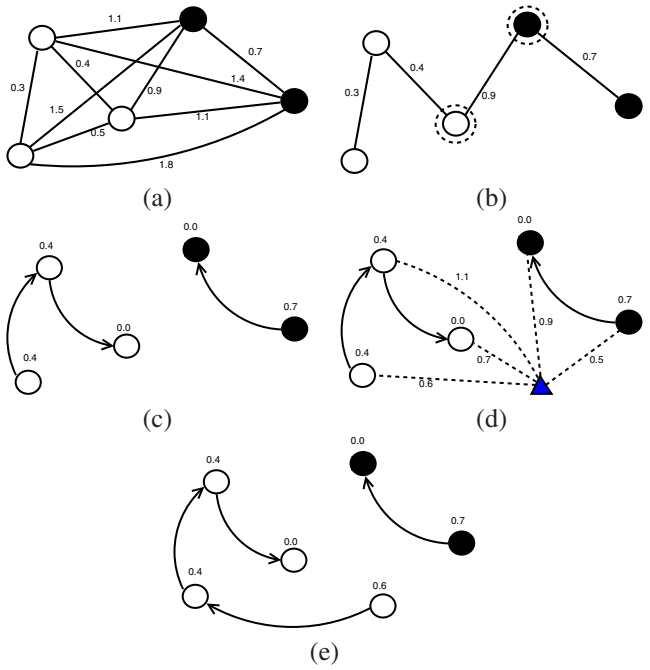


Fig. 1. OPF pipeline: (a) complete graph, (b) MST and prototypes bounded, (c) optimum-path forest generated at the final of training step, (d) classification process and (e) the triangle sample is associated to the white circle class. The values above the nodes are their costs after training, and the values above the edges stand for the distance between their corresponding nodes.

III. Bats for Feature Selection

In this section, we present our proposed approach using Bat Algorithm (Sections II-A and II-B) together with OPF classifier (Section II-C) to find the best combination of features.

Ramos et al. [7], [20] have proposed a wrapper approach with HS [8] and GSA [6] techniques in order to find a subset of features that maximize the OPF accuracy over a validation set. In such case, the classification accuracy of OPF classifier has been used as the fitness function of the optimization problem.

In this paper, we present two main contributions: (i) the first one is the BBA algorithm, which can be used with any other classification technique, and the latter contribution is (ii) to use BBA as the heuristic algorithm to improve OPF effectiveness. We have that each bat's position in the search space encodes the subset of features that it represents. Thus, for each one of them, an OPF classifier is trained in Z_1 and evaluated over Z_2 in order to assess the fitness value of each bat. Note that the training and evaluating sets may be different among the bats, since each one of them may encode a different set of features. Algorithm 2 presents in details the proposed technique.

The first loop in Lines $1-7$ initializes the population of bats. The bats' position are then initialized with randomly chosen binary values in Lines $2-3$, which corresponds whether a feature will be selected or not to compose the new dataset. Lines $11-15$ compose the new training and evaluating sets with the selected features, and Lines $17-22$ evaluate each bat in order to update its fitness value. Furthermore, the loudness A_i and the rate of pulse emission r_i are updated if a new solution has been accepted. While the loudness usually

decreases once a bat has found its prey, the rate pulse emission increases (Equations 5 and 6).

Algorithm 2. – FEATURE SELECTION THROUGH BA

INPUT: Labelled training Z_1 and evaluating set Z_2, population size m, number of features n, number of iterations T, loudness A, pulse emission rate r, ϵ, α and γ values.

OUTPUT: Subset of features F that gives the maximum accuracy over Z_2.

AUXILIARY: Fitness vector fit of size m, initial frequency vector r^0 of size m, global best position vector $\vec{x}$ of size n, and variables acc, $rand$, $maxfit$, $maxindex$, $globalfit$, β, $f_{\max}$, $f_{\min}$.

1. *For each bat* b_i $(\forall i = 1, \ldots, m)$, *do*
2. | *For each feature* j $(\forall j = 1, \ldots, n)$, *do*
3. | | $x_i^j \leftarrow$ *Random*$\{0, 1\}$
4. | $v_i^j \leftarrow 0$
5. | $A_i \leftarrow Random[1, 2]$
6. | $r_i \leftarrow Random[0, 1]$
7. | $fit_i \leftarrow -\infty$
8. $globalfit \leftarrow -\infty$
9. *For each iteration* t $(t = 1, \ldots, T)$, *do*
10. | *For each bat* b_i $(\forall i = 1, \ldots, m)$, *do*
11. | | *Create* Z_1' *and* Z_2' *from* Z_1 *and* Z_2, *respectively,*
12. | | *such that both contains only features in* b_i *in*
13. | | *which* $x_i^j \neq 0$, $\forall j = 1, \ldots, n$.
14. | | *Train classifier over* Z_1' , *evaluate its over* Z_2' *and*
15. | | *stores the accuracy in* acc.
16. | | $rand \leftarrow$ *Random*$[0, 1]$
17. | | *If* $(rand < A_i$ *and* $acc > fit_i)$, *then*
18. | | | $fit_i \leftarrow acc$
19. | | | $A_i \leftarrow \alpha A_i$
20. | | | $r_i \leftarrow r_i^0[1 - exp(-\gamma t)]$
21. | $[maxfit, maxindex] \leftarrow max(fit)$
22. | *If*$(maxfit > globalfit)$, *then*
23. | | $globalfit \leftarrow maxfit$
24. | | *For each dimension* j $(\forall j = 1, \ldots, m)$, *do*
25. | | | $\hat{x}^j \leftarrow x_{maxindex}^j$
26. | *For each bat* b_i $(\forall i = 1, \ldots, m)$, *do*
27. | | $\beta \leftarrow$ *Random*$[0, 1]$
28. | | $rand \leftarrow$ *Random*$[0, 1]$
29. | | *If* $(rand > r_i)$
30. | | | *For each feature* j $(\forall j = 1, \ldots, n)$, *do*
31. | | | | $x_i^j = x_i^j + \epsilon\overline{A}$
32. | | | | $\sigma \leftarrow$ *Random*$\{0, 1\}$
33. | | | | *If* $(\sigma < 1\frac{1}{1+e^{-x_i^j}})$, *then*
34. | | | | | $x_i^j \leftarrow 1$
35. | | | | *else* $x_i^j \leftarrow 0$
36. | | $rand \leftarrow$ *Random*$[0, 1]$
37. | | *If* $(rand < A_i$ *and* $fit_i < globalfit)$, *then*
38. | | | *For each feature* j $(\forall j = 1, \ldots, n)$, *do*
39. | | | | $f_i \leftarrow f_{\min} + (f_{\max} - f_{\min})\beta$
40. | | | | $v_i^j \leftarrow v_i^j + (\hat{x}^j - x_i^j)f_i$
41. | | | | $x_i^j \leftarrow x_i^j + v_i^j$
42. | | | | $\sigma \leftarrow$ *Random*$\{0, 1\}$
43. | | | | *If* $(\sigma < \frac{1}{1+e^{-x_i^j}})$, *then*
44. | | | | | $x_i^j \leftarrow 1$
45. | | | | *else* $x_i^j \leftarrow 0$
46. *For each feature* j $(\forall j = 1, \ldots, n)$, *do*
47. | $F^j \leftarrow \hat{x}^j$
48. *Return* F.

In Line 21, the function max outputs the index and the fitness value of the bat that maximizes the fitness function. Lines $22 - 25$ update the global best position, i.e., $\hat{x}$, with the position of the bat that has achieved the highest fitness function. The loop in Lines $26 - 45$ is responsible for updating the bats' position and velocity.

The source code in Lines $29 - 35$ performs the local search as described in Lines $5 - 8$ of Algorithm 1. The source code in Lines $37 - 45$ corresponds to the Lines $9 - 11$ of of Algorithm 1. At Line 39, we update the frequency of each bat as described by Equation 1. In regard to $f_{\min}$ and $f_{\max}$ values, we have used $f_{\min} = 0$ and $f_{\max} = 1$. Finally, the output vector F with the selected features is generated in Lines $46 - 47$ and returned by the algorithm in Line 48.

IV. EXPERIMENTAL RESULTS

In this section we discuss the experiments conducted in order to assess the robustness of the Bat Algorithm for feature selection.

In regard to the datasets, we have employed five public datasets: Breast Cancer, Australian, German Numer, DNA and Mushrooms[1]. Table I presents the main characteristics of them.

TABLE I
DESCRIPTION OF THE DATASETS USED IN THIS WORK.

Dataset	# samples	# features	# classes
Australian	690	14	2
Breast Cancer	683	10	2
DNA	2,000	180	3
German Numer	1,000	24	2
Mushrooms	8,124	112	2

As aforementioned in Section II-C, we partitioned the datasets in four disjoint subsets Z_1, Z_2, Z_3 and Z_4, which correspond to the training, learning, validation and test sets, respectively. We conducted two rounds of experiments: (i) in the first one we have assessed the robustness of OPF classifier over the original datasets, i.e., without feature selection, and (ii) in the latter we have validated the proposed Binary BA against Binary PSO, Binary FFA, HS and Binary GSA.

For the first round of experiments, we have employed only the training and test sets, being 30% of the whole dataset for Z_1 and 30% for Z_4. In the second round, we have used the training and learning sets to find the subset of features that maximizes the accuracy of the learning set (Z_2). The main idea is, for each bat/particle/harmony/firefly/mass in case of BA/PSO/HS/FFA/GSA, to train OPF over Z_1 and classify Z_2. Therefore, the optimization techniques will be guided by the OPF accuracy over Z_2 in order to find the most informative set of features. Notice that the same training and test sets were used for both experiments.

In order to compute the mean accuracy and standard deviation, for the first round of experiments we conducted a cross-validation over 10 running. In regard to the second round, we have proposed an approach to retrieve the most

[1]http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

informative features: the idea is to establish a threshold that ranges from 10% to 90%, and for each value of this threshold we marked the features that were selected at least a minimum percentage of the running over this learning process in Z_1 and Z_2. For instance, a threshold of 40% means we choose the features that were selected by the technique at least 40% of the running. For each threshold, we computed the OPF accuracy over the validating Z_3 in order to evaluate the generalization capability of the selected features. Therefore, the final subset of features will be the one that maximizes the curve over the range of values, i.e., the features that maximizes the accuracy over Z_3. Finally, this subset of features will be used to train OPF over Z_1 and to classify the test set Z_4, which did not participate from any learning process. In this latter experiment, we employed 30% for training and 20% for learning the most informative set of features, 20% to compose the validating set and the remaining 30% for the test set. Table II displays the mean OPF classification accuracy over the original datasets, i.e., without feature selection. We have used an accuracy measure that considers unbalanced classes, as proposed by Papa et al. [13].

TABLE II
CLASSIFICATION ACCURACY OVER THE ORIGINAL DATASETS, I.E., WITHOUT FEATURE SELECTION.

Dataset	Accuracy
Australian	76.16%±0.25
Breast Cancer	93.08%±1.55
DNA	75.98%±1.06
German Numer	57.65%±2.37
Mushrooms	98.61%±0.2

Figure 2 displays the curve of the second round of experiments for Breast Cancer dataset. We can see, for instance, that GSA with a threshold equals to 50% has selected the features that maximizes the accuracy over Z_3 (96.79%). Figure 3 shows the curve for Australian dataset, in which FFA with a threshold equals to 60% has maximized the fitness function (85.60%).

Fig. 2. OPF accuracy curve over Z_3 for Breast Cancer dataset.

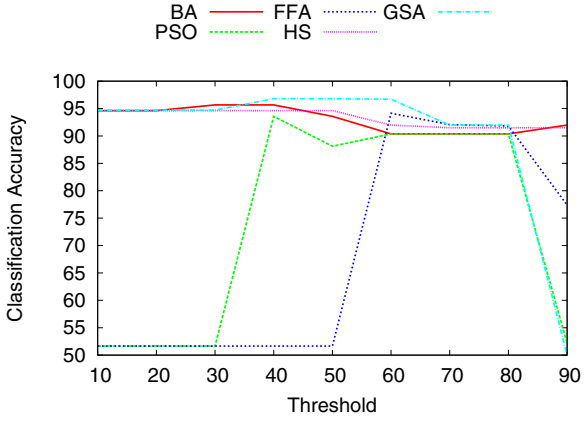


Figure 4, 5 and 6 display the OPF accuracy curve for German Numer, DNA and Mushrooms datasets, respectively. In respect to German Numer dataset, GSA has achieved the best

Fig. 3. OPF accuracy curve over Z_3 for Australian dataset.

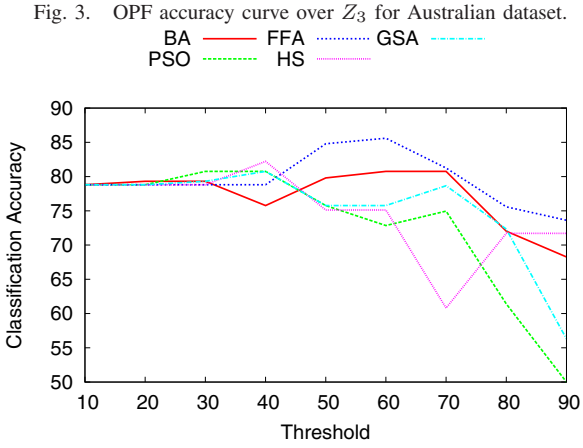
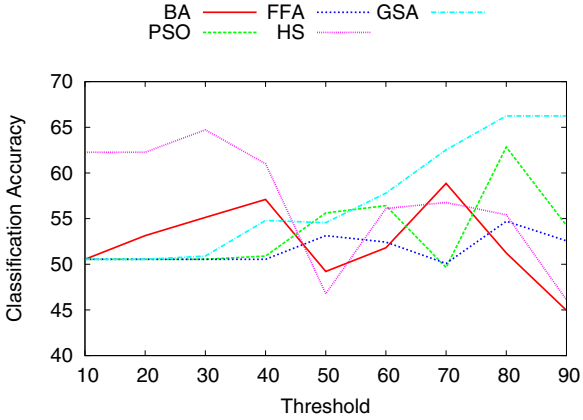


Fig. 4. OPF accuracy curve over Z_3 for German Numer dataset.



performance with threshold equals to 80% and classification accuracy of 66.25%. For DNA dataset, BA outperformed all techniques with threshold equals to 90% and recognition rate of 84.02%. In respect to Mushrooms dataset, BA, PSO and GSA have achieved 100% of classification rate with threshold equals to 60%, 70% and 60%, respectively.

Table III shows the OPF accuracy over Z_4 using the selected features that maximized the above curves. As one can see,

Fig. 5. OPF accuracy curve over Z_3 for DNA dataset.

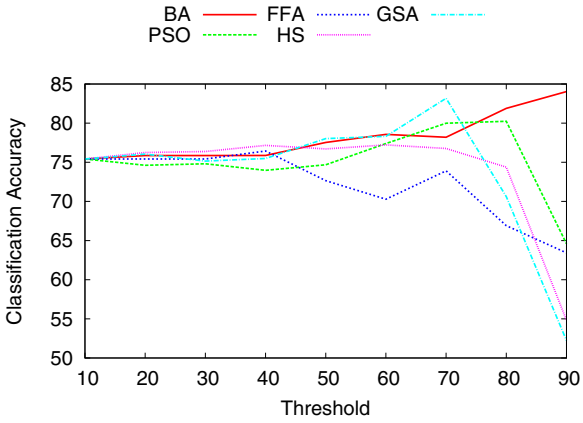
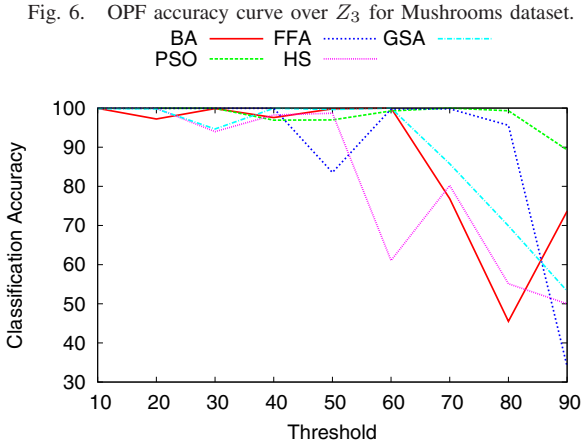


Fig. 6. OPF accuracy curve over Z_3 for Mushrooms dataset.



BA performs better than the other algorithms for Australian, Breast Cancer and Mushrooms datasets. In regard to DNA and German Numer datasets, BA has achieved the second best accuracy. Table IV displays the mean computational load in seconds for each optimization technique. As the Bat Algorithm has common properties with respect to PSO, it is expected that their execution times be similar to each other.

TABLE III
CLASSIFICATION ACCURACY WITH THE BEST SUBSET OF FEATURES OVER Z_4

Dataset	BA	PSO	FFA	HS	GSA
Australian	**77.25%**	76.12%	76.63%	76.53%	76.12%
Breast Cancer	**96.31%**	94.28%	92.85%	94.23%	95.74%
DNA	83.02%	79.31%	76.63%	83.23%	81.46%
German Numer	70.24%	59.52%	53.41%	55.24%	70.87%
Mushrooms	**100.0%**	99.96%	**100.0%**	99.95%	**100.0%**

TABLE IV
COMPUTATIONAL LOAD IN SECONDS REGARDING THE OPTIMIZATION ALGORITHMS FOR THE FEATURE SELECTION TASK.

Dataset	BA	PSO	FFA	HS	GSA
Australian	25.3	24.9	25.1	2.8	26.5
Breast Cancer	21.9	19.9	21.7	2.6	23.4
DNA	533.0	461.3	518.3	53.8	488.4
German Numer	55.6	53.7	56.2	6.4	57.6
Mushrooms	5369.3	5238.8	5431.1	579.5	5324.1

Table V states the values of the thresholds used for each optimization technique that maximized the classification accuracy over Z_3. We can see that there is not a consensus about the threshold values chosen by each optimization technique for some datasets, such as DNA and Mushrooms. In the opposite side, for Australian dataset BA and FFA have used the same threshold, while PSO, HS and GSA employed another one. We can conclude that there are similar features, and each group of optimization techniques have chosen a different set of them.

Finally, Table VI displays the number of selected features for each optimization technique. It is interesting to shed light over that BA has selected fewer samples than all techniques, except for German Numer dataset.

TABLE V
THRESHOLD VALUES WITH RESPECT TO THE HIGHEST CLASSIFICATION ACCURACIES OVER Z_3.

Dataset	BA	PSO	FFA	HS	GSA
Australian	60%	40%	60%	40%	40%
Breast Cancer	30%	40%	60%	50%	50%
DNA	90%	60%	30%	80%	80%
German Numer	40%	80%	80%	30%	90%
Mushrooms	60%	70%	70%	20%	60%

The information about the number of selected samples has a correlation with the thresholds shown in Table V. In case of Mushrooms dataset, for instance, HS has selected 106 features against only 53 features chosen by BA. We can observe in Table V that the thresholds for the same dataset have been 20% and 60% for HS and BA techniques, respectively. Thus, as HS has selected a high number of features, it is much easier to them to appear only 20% of the running than the 60% employed by BA, which has used a half of the features when compared with HS.

TABLE VI
NUMBER OF FEATURES SELECTED FOR EACH OPTIMIZATION TECHNIQUE.

Dataset	BA	PSO	FFA	HS	GSA
Australian	8	12	12	9	12
Breast Cancer	7	8	7	7	8
DNA	18	25	75	67	28
German Numer	7	4	5	19	3
Mushrooms	53	76	106	106	57

V. CONCLUSIONS

In this paper we addressed the problem of the high dimensionality in object's description by means of finding the most informative features in a search space given by a boolean hypercube.

As the feature selection can be seen as an optimization problem, we have proposed a bat-inspired algorithm for this task. A binary version of the well-known continuous-valued Bat Algorithm was derived in order to position the bats in binary coordinates along the corners of the search space, which represents a string of bits that encodes whether a feature will be selected or not.

We conducted experiments against several meta-heuristic algorithms to show the robustness of the proposed technique and also the good generalization capability of the bat-inspired technique. We have employed five public datasets to accomplish this task, in which BA has been compared against PSO, FFA and GSA. The proposed algorithm has outperformed the compared techniques in 3 out of 5 datasets, being the second best in the remaining two datasets. For future works, we intend to apply BA for feature weighting.

ACKNOWLEDGEMENTS

The authors would like to thank FAPESP grants #2009/16206-1, #2011/14058-5 and #2011/14094-1, and also CNPq grant #303182/2011-3.

REFERENCES

[1] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[2] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 5, 1997, pp. 4104–4108.

[3] J. Kennedy and R. Eberhart, *Swarm Intelligence*. M. Kaufman, 2001.

[4] H. A. Firpi and E. Goodman, "Swarmed feature selection," in *Proceedings of the 33rd Applied Imagery Pattern Recognition Workshop*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 112–118.

[5] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "BGSA: binary gravitational search algorithm," *Natural Computing*, vol. 9, pp. 727–745, 2010.

[6] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: A gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[7] C. Ramos, A. Souza, G. Chiachia, A. Falcão, and J. Papa, "A novel algorithm for feature selection using harmony search and its application for non-technical losses detection," *Computers & Electrical Engineering*, vol. 37, no. 6, pp. 886–894, 2011.

[8] Z. W. Geem, *Music-Inspired Harmony Search Algorithm: Theory and Applications*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[9] H. R. Kanan, K. Faez, and S. M. Taheri, "Feature selection using ant colony optimization (ACO): a new method and comparative study in the application of face recognition system," in *Proceedings of the 7th industrial conference on Advances in data mining: theoretical aspects and applications*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 63–76.

[10] J. Huang, Y. Cai, and X. Xu, "A hybrid genetic algorithm for feature selection wrapper based on mutual information," *Pattern Recognition Letters*, vol. 28, no. 13, pp. 1825–1844, 2007.

[11] H. Banati and M. Bajaj, "Fire Fly Based Feature Selection Approach," *International Journal of Computer Science Issues*, vol. 8, no. 4, pp. 473–480, 2011.

[12] X.-S. Yang., "Bat algorithm for multi-objective optimisation," *International Journal of Bio-Inspired Computation*, vol. 3, no. 5, pp. 267–274, 2011.

[13] J. P. Papa, A. X. Falcão, and C. T. N. Suzuki, "Supervised pattern classification based on optimum-path forest," *International Journal of Imaging Systems and Technology*, vol. 19, no. 2, pp. 120–131, 2009.

[14] J. P. Papa, A. X. Falcão, V. H. C. Albuquerque, and J. M. R. S. Tavares, "Efficient supervised optimum-path forest classification for large datasets," *Pattern Recognition*, vol. 45, no. 1, pp. 512–520, 2012.

[15] D. R. Griffin, F. A. Webster, and C. R. Michael, "The echolocation of flying insects by bats," *Animal Behaviour*, vol. 8, no. 34, pp. 141 – 154, 1960.

[16] W. Metzner, "Echolocation behaviour in bats." *Science Progress Edinburgh*, vol. 75, no. 298, pp. 453–465, 1991.

[17] H.-U. Schnitzler and E. K. V. Kalko, "Echolocation by insect-eating bats," *BioScience*, vol. 51, no. 7, pp. 557–569, July 2001.

[18] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[19] A. Falcão, J. Stolfi, and R. Lotufo, "The image foresting transform theory, algorithms, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 1, pp. 19–29, 2004.

[20] C. C. O. Ramos, A. N. de Souza, A. X. Falcão, and J. P. Papa, "New insights on non-technical losses characterization through evolutionary-based feature selection," *IEEE Transactions on Power Delivery*, vol. 27, no. 1, pp. 140–146, 2012.