# Binary Flower Pollination Algorithm and Its Application to Feature Selection

**Douglas Rodrigues, Xin-She Yang, André Nunes de Souza and João Paulo Papa**

**Abstract** The problem of feature selection has been paramount in the last years, since it can be as important as the classification step itself. The main goal of feature selection is to find out the subset of features that optimize some fitness function, often in terms of a classifier's accuracy or even the computational burden for extracting each feature. Therefore, the approaches to feature selection can be modeled as optimization tasks. In this chapter, we evaluate a binary-constrained version of the Flower Pollination Algorithm (FPA) for feature selection, in which the search space is a boolean lattice where each possible solution, or a string of bits, denotes whether a feature will be used to compose the final set. Numerical experiments over some public and private datasets have been carried out and comparison with Particle Swarm Optimization, Harmony Search and Firefly Algorithm has demonstrated the suitability of the FPA for feature selection.

**Keywords** Feature selection · Flower pollination algorithm · Optimum-path forest

D. Rodrigues · J.P. Papa (✉)
Department of Computing, UNESP, Bauru, SP, Brazil
e-mail: papa@fc.unesp.br

D. Rodrigues
e-mail: douglasrodrigues.dr@gmail.com

X.-S. Yang
School of Science and Technology, Middlesex University, London NW4 4BT, UK
e-mail: x.yang@mdx.ac.uk

A.N. de Souza
Department of Electrical Engineering, UNESP, Bauru, SP, Brazil
e-mail: andrejau@feb.unesp.br



X.-S. Yang (ed.), *Recent Advances in Swarm Intelligence and Evolutionary Computation*,
Studies in Computational Intelligence 585, DOI 10.1007/978-3-319-13826-8_5

## 1 Introduction

Machine learning techniques have been actively studied in recent years with an increasing number of applications that make use of the so-called intelligence-based decision processes. Roughly speaking, a standard workflow for tackling such problems can be divided in four phases: (i) to preprocess the data (signal or image filtering, for instance); (ii) to extract features; (ii) to train a machine learning technique, and finally (iv) to evaluate its effectiveness over an unseen (test) data [3].

One of the most important steps concerns feature extraction is to find the most important subset of features that leads to the best recognition rates. There are situations we may obtain the same accuracy as before (with the original set of features) even after feature selection, but we can save computational efforts by avoiding extracting some features that are too costly.

Several studies have modeled the problem of feature selection as an optimization task, since the idea is to find out the subset of features that maximizes the accuracy of a given classifier, or minimizes its error over some validating set, for instance. Such approaches can be useful to the application of evolutionary optimization techniques to solve complex tasks. The readers can refer to some recent literature such as the Binary Particle Swarm Optimization (BPSO) [5], Binary Firefly Algorithm (BFA) [4], Binary Harmony Search (BHS) [14], Binary Gravitational Search Algorithm (BGSA) [16], Binary Cuckoo Search (BCS) [17], Binary Charged System Search (BCSS) [19], and Binary Bat Algorithm (BBA) [18].

Yang and Honavar [23] presented a multicriteria Genetic Algorithm (GA) to deal with feature selection, in which the main idea was to optimize both the accuracy and the feature extraction computational costs. Later on, Oh et al. [10] proposed a hybrid GA to tackle the same problem with seemingly better final performance. In addition, there are many papers that address feature selection using other methods such as the ant colonization [2, 7, 21]. The main idea consists of reducing the number of possible paths visited by ants in some works, as well as modified pheromone update rules. Other approaches such as Artificial Bee Colony [9, 20] and Gravitational Search Algorithm [1, 15] have been also employed to the same context.

Basically, the main idea of these methods is to convert the position of the agents (bats, particles, harmonies, etc.) into binary-valued coordinates, which are represented by a string of bits, each denoting the presence or absence of a feature. The problem of feature selection can also be considered as a search task in a boolean lattice, in which the number of dimensions stands for the number of features. As the original versions of most evolutionary optimization techniques were proposed to handle continuous-valued problems, the idea is to apply a discretization function (usually a constrained sigmoid function) to map the agent locations to the boolean lattice.

Very recently, Yang [25, 26] proposed the Flower Pollination Algorithm (FPA), which is inspired by the flower pollination process of flowering plants. This approach has demonstrated interesting results for traditional (continuous-valued) optimization problems, which motivated us to extend it to solve binary optimization tasks. In this case, we now to tackle the problem of feature selection and propose

approach the Binary Flower Pollination Algorithm (BFPA). In regard to the fitness function, we have used a classifier's effectiveness over a validating set: as we need to train a classifier every time an agent (pollen) changes its position, we need a fast classifier. For this purpose, we use the Optimum-Path Forest (OPF) [12, 13], which has demonstrated very promising results in several applications, and this approach is also parameter-independent. The proposed approach has been compared with other methods such as BPSO, BFA and BHS to evaluate several datasets. The results are also analyzed by using statistical tools.

The remainder of this chapter is organized as follows. Section 2 introduces the theory background about FPA and OPF techniques. Sections 3 and 4 present the methodology and the experimental results, respectively. Finally, Sect. 5 draws some conclusions and future works.

## 2 Theoretical Background

In this section, we first briefly review some of the main concepts and techniques to be used in this chapter, as well as the proposed Binary Flower Pollination Algorithm.

### *2.1 Flower Pollination Algorithm*

The Flower Pollination Algorithm was proposed by Yang [25], inspired by the pollination process of flowering plants. The FPA is governed by four basic rules:

1. Biotic cross-pollination can be considered as a process of global pollination, and pollen-carrying pollinators move in a way that obeys Lévy flights.
2. For local pollination, abiotic pollination and self-pollination are used.
3. Pollinators such as insects can develop flower constancy, which is equivalent to a reproduction probability that is proportional to the similarity of two flowers involved.
4. The interaction or switching of local pollination and global pollination can be controlled by a switch probability $p \in [0, 1]$, slightly biased towards local pollination.

However, it is necessary that the aforementioned basic rules be converted into appropriate updating equations. For example, in the global pollination step, flower pollen gametes are carried by pollinators such as insects, and pollen can travel over a long distance because insects can often fly and move over a much longer range [25]. Therefore, Rules 1 and 3 can be represented mathematically as:

$$x_i^{(t+1)} = x_i^t + \alpha L(\lambda)(g_* - x_i^t), \tag{1}$$

where

$$L(\lambda) = \frac{\lambda \cdot \Gamma(\lambda) \cdot \sin(\lambda)}{\pi} \cdot \frac{1}{s^{1+\lambda}}, \quad s > 0 \tag{2}$$

where $x_i^t$ is the pollen $i$ (solution vector) at iteration $t$, $g_*$ is the current best solution found among all solutions at the current generation, and $\alpha$ is a scaling factor to control the step size, $L(\lambda)$ is the Lévy flights step size corresponding to the strength of the pollination. In addition, $\Gamma(\lambda)$ stands for the gamma function, and $s$ is the step size. Since insects may move over a long distance with various distance steps, Lévy flights can be used to mimic this characteristic efficiently.

For local pollination, both Rules 2 and 3 can be represented as:

$$x_i^{(t+1)} = x_i^t + \varepsilon(x_j^t - x_k^t), \tag{3}$$

where $x_j^t$ and $x_k^t$ stand for the pollen from different flowers $j$ and $k$ of the same plant species, respectively. This mimics flower constancy in a limited neighbourhood. Mathematically, if $x_j^t$ and $x_k^t$ come from the same species or are selected from the same population, it equivalently becomes a local random walk if $\varepsilon$ is drawn from a uniform distribution in [0,1]. In order to mimic the local and global flower pollination, a switch probability (Rule 4) or proximity probability $p$ is used.

### 2.1.1 Binary Flower Pollination Algorithm

In the standard FPA, the solutions are updated in the search space towards continuous-valued positions. However, in the proposed Binary Flower Pollination Algorithm the search space is modelled as a $d$-dimensional boolean lattice, in which the solutions are updated across the corners of a hypercube. In addition, as the problem is to select or not a given feature, a binary solution vector is used, where 1 corresponds to that a feature will be selected to compose the new dataset with 0 being otherwise. In order to build this binary vector, we have to use Eq. (5) just after Eq. (3), which can restrict the new solutions to only binary values:

$$S(x_i^j(t)) = \frac{1}{1 + e^{-x_i^j(t)}}, \tag{4}$$

$$x_i^j(t) = \begin{cases} 1 & \text{if } S(x_i^j(t)) > \sigma, \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where $\sigma \sim U(0, 1)$. Algorithm 1 presents the proposed BFPA for feature selection using the recognition rate of the OPF classifier as the objective function. Note the proposed approach can be used with any other supervised classification technique.

Lines 1–4 initialize each pollen's position as being a binary string with random values, as well as the fitness value $f_i$ of each individual $i$. The main loop in Lines

6–27 is the core of the proposed algorithm, in which the inner loop in Lines 7–13 is responsible for creating the new training $Z'_1$ and evaluating sets $Z'_2$, and then OPF is trained over $Z'_1$ and it is used to classify $Z'_2$. The recognition accuracy over $Z'_2$ is stored in *acc* and then compared with the fitness value $f_i$ (accuracy) of individual $i$: if the latter is worse than *acc*, the old fitness value is kept; otherwise, the fitness value is then updated. Lines 12–13 update the best local position of the current pollen. Lines 14–18 update the global optimum, and the last loop (Lines 19–27) moves each pollen to a new binary position restricted by Eq. (5) (Lines 25–27).

**Algorithm 1:** BFPA - Binary Flower Pollination Algorithm.

**input** : Training set $Z_1$ and evaluating set $Z_2$, $\alpha$, number of flowers $m$, dimension $d$ and iterations $T$.
**output** : Global best position $\widehat{g}$.
**auxiliaries**: Fitness vector $f$ with size $m$ and variables *acc*, $maxfit$, $globalfit$ and *maxindex*.

1 **for** *each flower* $i$ $(\forall i = 1,\ldots,m)$ **do**
2  **for** *each dimension* $j$ $(\forall j = 1,\ldots,d)$ **do**
3  $x_i^j(0) \leftarrow \text{Random}\{0,1\}$;
4  $f_i \leftarrow -\infty$;
5 $globalfit \leftarrow -\infty$;
6 **for** *each iteration* $t$ $(t = 1,\ldots,T)$ **do**
7  **for** *each flower* $i$ $(\forall i = 1,\ldots,m)$ **do**
8  Create $Z'_1$ and $Z'_2$ from $Z_1$ and $Z_2$, respectively, such that both contains only features such that $x_i^j(t) \neq 0, \forall j = 1,\ldots,d$;
9  Train OPF over $Z'_1$, evaluate its over $Z'_2$ and stores the accuracy in *acc*;
10  **if** $(acc > f_i)$ **then**
11  $f_i \leftarrow acc$;
12  **for** *each dimension* $j$ $(\forall j = 1,\ldots,d)$ **do**
13  $\widehat{x}_i^j \leftarrow x_i^j(t)$;
14  $[maxfit, maxindex] \leftarrow max(f)$;
15  **if** $(maxfit > globalfit)$ **then**
16  $globalfit \leftarrow maxfit$;
17  **for** *each dimension* $j$ $(\forall j = 1,\ldots,d)$ **do**
18  $\widehat{g}^j \leftarrow x^j_{maxindex}(t)$;
19  **for** *each flower* $i$ $(\forall i = 1,\ldots,m)$ **do**
20  **for** *each dimension* $j$ $(\forall j = 1,\ldots,d)$ **do**
21  $rand \leftarrow \text{Random}\{0,1\}$;
22  **if** $rand < p$ **then**
23  $x_i^j(t) \leftarrow x_i^j(t-1) + \alpha \oplus \text{Lévy}(\lambda)$; **else**
24  $x_i^j(t) \leftarrow x_i^j(t-1) + \varepsilon(x_i^j(t-1) - x_i^k(t-1))$;
25  **if** $(\sigma < \frac{1}{1+e^{x_i^j(t)}})$ **then**
26  $x_i^j(t) \leftarrow 1$; **else**
27  $x_i^j(t) \leftarrow 0$;

## 2.2 Optimum-Path Forest Classifier

The Optimum-Path Classifier [12, 13] models the samples as graph nodes, whose arcs are defined by an adjacency relation and weighted by some distance function. Further, a role competition process between some key nodes (prototypes) is carried out in order to partition the graph into optimum-path trees (OPTs) according to some path-cost function. Therefore, to design an Optimum-Path Forest-based classifier, one needs to define: (i) an adjacency relation, (ii) a path-cost function and (iii) a methodology to estimate prototypes.

Suppose we have a fully labeled dataset $Z = Z_1 \cup Z_2$, in which $Z_1$ and $Z_2$ stand for training and test sets, respectively. Let $S \subset Z_1$ be a set of prototypes of all classes (i.e., key samples that best represent the classes). Let $(Z_1, A)$ be a complete graph whose nodes are the samples in $Z_1$ and any pair of samples defines an arc in $A = Z_1 \times Z_1$. Let $\pi_s$ be a path in the graph that ends in sample $s \in Z_1$, and $\langle \pi_s \cdot (s,t) \rangle$ the concatenation between $\pi_s$ and the arc $(s, t)$, $t \in Z_1$. In this chapter, we employ a path-cost function that returns the maximum arc-weight along a path in order to avoid chains, and also to show the idea of connectivity between samples. This path-cost function is denoted here as $\Psi$, and it can be computed as follows:

$$\begin{aligned} \Psi(\langle s \rangle) &= \begin{cases} 0, & \text{if } s \in S, \\ +\infty, & \text{otherwise,} \end{cases} \\ \Psi(\pi_s \cdot \langle s,t \rangle) &= \max\{\Psi(\pi_s), d(s,t)\}, \end{aligned} \tag{6}$$

in which $d(s, t)$ means the distance between nodes $s$ and $t$. Thus, the objective of the Optimum-Path Forest algorithm (supervised version) is to minimize $\Psi(\pi_t)$, $\forall t \in Z_1$.

An optimal set of prototypes $S^*$ can be found by exploiting the theoretical relation between the minimum-spanning tree and optimum-path tree for $\Psi$. By computing a minimum-spanning tree in the complete graph $(Z_1, A)$, we obtain a connected acyclic graph whose nodes are all samples of $Z_1$ and the arcs are undirected and weighted by the distances $d$ between adjacent samples. The spanning tree is optimum in the sense that the sum of its arc weights is the minimum as compared to any other spanning tree in the complete graph. In the minimum-spanning tree, every pair of samples is connected by a single path, which is optimum according to $\Psi$. Thus, the minimum-spanning tree contains one optimum-path tree for any selected root node. The optimum prototypes are the closest elements of the minimum-spanning tree with different labels in $Z_1$.

The Optimum-Path Forest training phase consists, essentially, of starting the competition process between prototypes in order to minimize the cost of each training sample. At the final of such procedure, we obtain an optimum-path forest, which is a collection of optimum-path trees rooted at each prototype. A sample connected to an OPT means that it is more strongly connected to the root of that tree than to any other root in the forest.

Furthermore, in the classification phase, for any sample $t \in Z_2$, we consider all arcs connecting $t$ with samples $s \in Z_1$, as though $t$ were part of the training graph. Considering all possible paths from $S^*$ to $t$, we find the optimum path $P^*(t)$ from $S^*$ and label $t$ with the class $\lambda(R(t))$ of its most strongly connected prototype $R(t) \in S^*$. This path can be identified incrementally, by evaluating the optimum cost $C(t)$ as:

$$C(t) = \min\{\max\{C(s), d(s,t)\}\}, \ \forall s \in Z_1. \tag{7}$$

Let the node $s^* \in Z_1$ be the one that satisfies (Eq. 7) (i.e., the predecessor $P(t)$ in the optimum path $P^*(t)$). Given that $L(s^*) = \lambda(R(t))$, the classification simply assigns $L(s^*)$ as the class of $t$.

## 3 Methodology

In this section, we present the methodology used to evaluate the performance of BFPA. Details about the dataset used, experimental setup and the compared techniques are also provided.

### 3.1 Datasets

Table 1 presents the datasets used in this work.[1] Such datasets differ on the number of samples, features and also classes. Therefore, the idea is to evaluate the proposed approach in different contexts.

The last two datasets, i.e., $NTL_c$ and $NTL_i$, are related to non-technical losses detection in comercial and industrial profiles, respectively. These are private datasets obtained by a Brazilian electrical power company. Such sort of problem is of great interest to electrical power companies, mainly in Brazil, in which the amount of losses in energy thefts can reach up to 20 % in some regions. Therefore, the characterization of illegal consumers, i.e., to find out the most important features that allow us to identify them, is so important as to effective recognize them.

### 3.2 Nature-Inspired Metaheuristic Algorithms

In this work, we have also employed three others evolutionary optimization techniques for comparison purposes. A brief detail about each of them is given below.

[1] The first four datasets can be found on http://featureselection.asu.edu/datasets.php.

**Table 1** Description of the benchmarking datasets

| Dataset | # samples | # features | # classes |
|---|---|---|---|
| GLI-85 | 85 | 22,283 | 2 |
| SMK-CAN-187 | 187 | 19,993 | 2 |
| TOX-171 | 171 | 5,748 | 4 |
| AR10P | 130 | 2,400 | 10 |
| $NTL_c$ | 4,952 | 8 | 2 |
| $NTL_i$ | 3,182 | 8 | 2 |

*Particle Swarm Optimization* (PSO): PSO was inspired by the social behavior of bird flocking or fish schooling [8]. The fundamental idea is that each particle represents a potential solution which is updated according to its own experience and from its neighbors' knowledge. The motion of an individual particle for the optimal solution is governed through its position and velocity interactions, and also by its own previous best performance and the best performance of their neighbors.

*Firefly Algorithm* (FA): FA was also proposed by Yang [24], based on the flashing behaviour and attractiveness of fireflies. The brightness of a firefly at a given position is determined by the value of the objective function in that position. Each firefly is attracted by a brighter firefly through the attraction factor that vary with their distance.

*Harmony Search* (HS): HS was a meta-heuristic algorithm inspired by the improvisation process of music players [6]. Musicians often improvise in searching for a perfect state of harmony. The main idea is to use the same process adopted by musicians to create new songs to obtain a near-optimal solution according to some fitness function. Each possible solution is modelled as a harmony, and each musical note corresponds to one decision variable.

In this present work, we have used all the binary optimization versions of each aforementioned technique, i.e., Binary PSO (BPSO) [5], Binary Firefly (BFA) [4, 11], as well as Binary HS (BHS) [14].

## 3.3 Experimental Setup

Firstly, the dataset $Z$ is randomly partitioned in $N$ folds, i.e., $Z = F_1 \cup F_2 \cup \cdots \cup F_N$. For each fold $F_i$, we train a given instance of the OPF classifier over it, for further evaluation of another fold $F_j$, $i \neq j$. Therefore, the classification accuracy over $F_j$ is then used as the fitness function to guide the optimization algorithms for selecting the most representative set of features. Each agent of the population (pollen, particle, firefly, harmony) in these meta-heuristic algorithms is associated with a string of bits denoting the presence or absence of a feature. Thus, for each agent, we construct a classifier from the training set $F_i$ only with the selected features, say $F_i^*$, and assigns the accuracy over $F_j$ as the fitness function. As long as the procedure converges, i.e., all generations of a population were computed, the agent with the highest fitness value encodes a solution with the best compacted set of features.

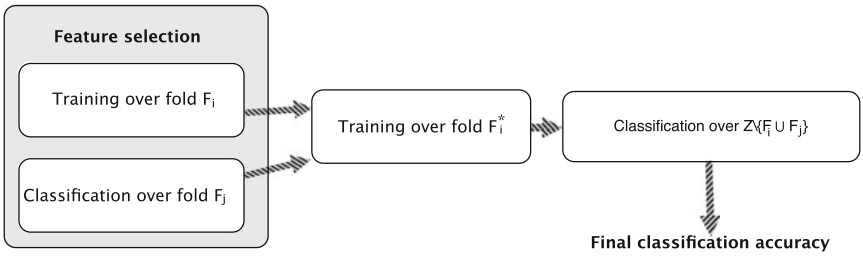


**Fig. 1** Proposed methodology to evaluate the compared techniques

After that, we build a classification model using the training set with the selected features $(F_i^*)$, and we also evaluate the quality of the solution through a classification process over the test set, which is built over the remaining folds in $Z \backslash \{F_i \cup F_j\}$. This procedure is conducted for each fold $F_i$ in the dataset to be part of the training set, and thus we have $N(N-1)$ combinations in the final of the process, which will be averaged for comparison purposes. Figure 1 illustrates the methodology described above.

In regard to the recognition rate, we used an accuracy measure proposed by Papa et al. [12]. If there are two classes, for example, with very different sizes and a classifier always assigns the label of the largest class, its accuracy will fall drastically due to the high error rate on the smallest class. The accuracy is measured by taking into account that the classes may have different sizes in a testing set $F_j$. Let us define:

$$e_{i,1} = \frac{FP_i}{|F_j| - |F_j^i|}, \tag{8}$$

and

$$e_{i,2} = \frac{FN_i}{|F_j^i|}, \quad i = 1, 2, \ldots, C, \tag{9}$$

where $C$ stands for the number of classes, $|F_j^i|$ concerns with the number of samples in $F_j$ that come from class $i$, and $FP_i$ and $FN_i$ stand for the false positives and false negatives for class $i$, respectively. That is, $FP_i$ is the number of samples from other classes that were classified as being from the class $i$ in $F_j$, and $FN_i$ is the number of samples from the class $i$ that were incorrectly classified as being from other classes in $F_j$. The error terms $e_{i,1}$ and $e_{i,2}$ are then used to define the total error from class $i$:

$$E_i = e_{i,1} + e_{i,2}. \tag{10}$$

Finally, the accuracy $Acc$ is then defined as follows:

$$Acc = 1 - \frac{\sum_{i=1}^{C} E_i}{2C}. \quad (11)$$

## 4 Experimental Results

In this section, we summarize and discuss the experimental results regarding the proposed approach for feature selection. The results presented in this section stand for the mean accuracy and standard deviation over 25 independent runs using the methodology presented in Sect. 3.3. Since the evolutionary optimization algorithms are non-deterministic, such approaches seem to be robust to avoid biased results. The optimization algorithms (BFPA, BPSO, BHS and BFA) were implemented in C language following the guidelines provided by their references. The experiments were executed on a computer with a Pentium Intel Core *i5*® 3.20 Ghz processor, 4 GB of RAM and Linux Ubuntu Desktop LTS 10.04 as the operational system.

Table 2 presents the parameters used for each optimization technique employed in this work. The $c_1$ and $c_2$ parameters of PSO control the pace during the particles' movement, and the "Harmony Memory Considering Rate" (HMCR) of BHS stands for the amount of information that will be used from the artist's memory (songs that have been already composed) in order to compose a new harmony. In regard to BFA, $\alpha$ and $\beta_0$ are related to the step size of a firefly, and $\gamma$ stands for the light absorption coefficient. In addition, we have used a population of 30 agents and 100 iterations for all techniques, with such values being an empirical set.

Figure 2 displays the mean accuracy results using the proposed methodology (Sect. 3.3). It can be observed that the feature selection techniques can slightly improve the results obtained using the original datasets, i.e., without feature selection. The second point is that all techniques achieved quite similar results. Therefore, the results showed BFPA is suitable for feature selection tasks. We have also performed the statistical Wilcoxon Signed-Rank Test [22] to verify whether there is a significant difference between BFPA and the other techniques used in this work (considering the OPF recognition rate). Table 3 displays the $p$-values, being the bold ones the situations in which BFPA and the respective technique have obtained different performances, i.e., when the $p$-values are lower than a significance level of $\alpha = 0.05$.

**Table 2** Parameters used for each meta-heuristic optimization technique. Notice the inertia weight $w$ for PSO was linearly decreased from 0.9 to 0.4 during the convergence process

| Technique | Parameters |
|---|---|
| BPSO | $c_1 = c_2 = 2$ |
| BFA | $\gamma = 0.8$, $\beta_0 = 1.0$, $\alpha = 0.01$ |
| BHS | HMCR = 0.9 |
| BFPA | $a = 1.0$, $p = 0.8$ |

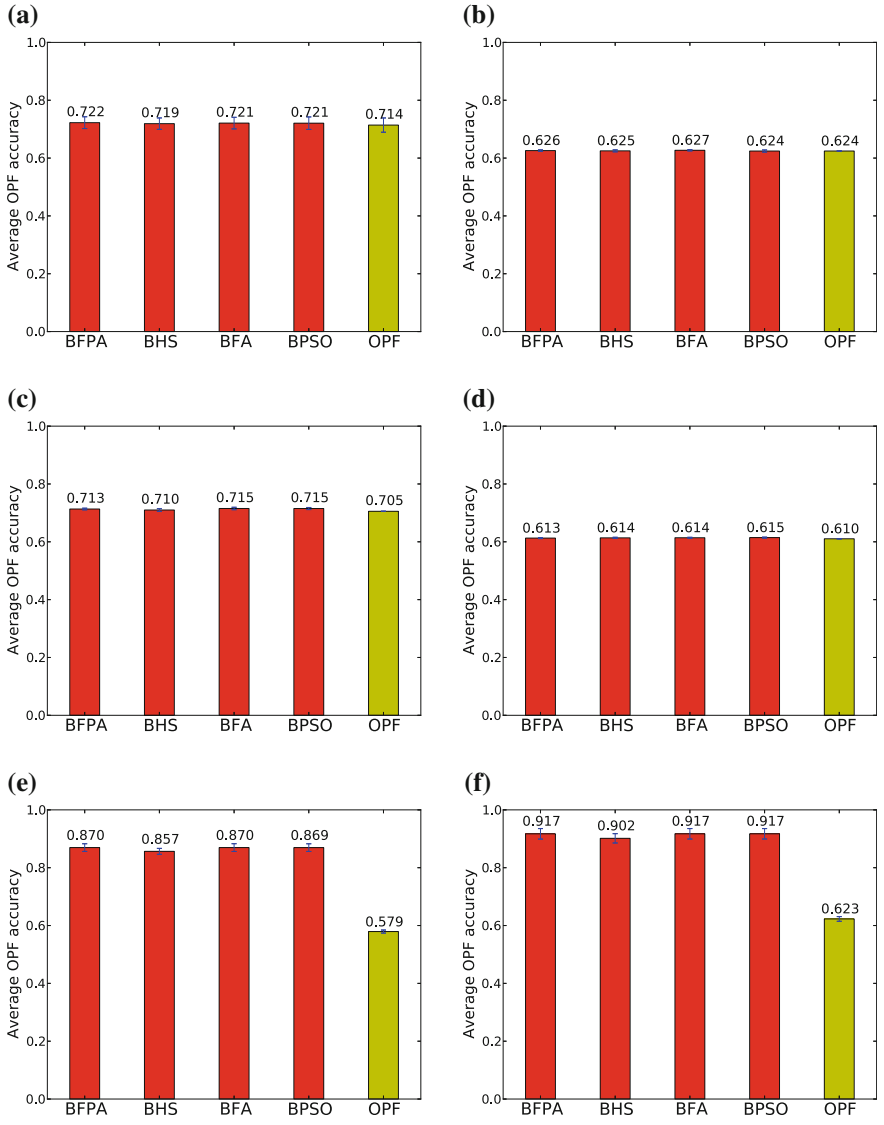


**Fig. 2** Average OPF accuracy over **a** GLI-85, **b** SMK-CAN-187, **c** TOX-171, **d** AR10P, **e** $NTL_c$ and **f** $NTL_i$ datasets

It can also be seen that there is a statistical difference between BFPA and BPSO, and BFPA and BFA for AR10P dataset, and also a difference between BFPA and BHS considering TOX-171 dataset. Figure 3 displays the convergence rates of all techniques considering the datasets employed in this work. Such recognition rates are the ones obtained over the validating set $F_j$, as depicted in Fig. 1.

**Table 3** Wilcoxon Signed-Rank Test evaluation: *p*-values computed between BFPA, BPSO, BFA and BHS

| Dataset | BPSO | BFA | BHS |
|---|---|---|---|
| GLI-85 | 0.3130 | 0.5629 | 0.1425 |
| SMK-CAN-187 | 0.1829 | 0.2012 | 0.4432 |
| TOX-171 | 0.1742 | 0.1500 | **0.0112** |
| AR10P | **0.0023** | **0.0197** | 0.0573 |
| $NTL_c$ | 0.3281 | 0.4769 | 1.2290 |
| $NTL_i$ | 0.1957 | 0.3318 | 1.2290 |

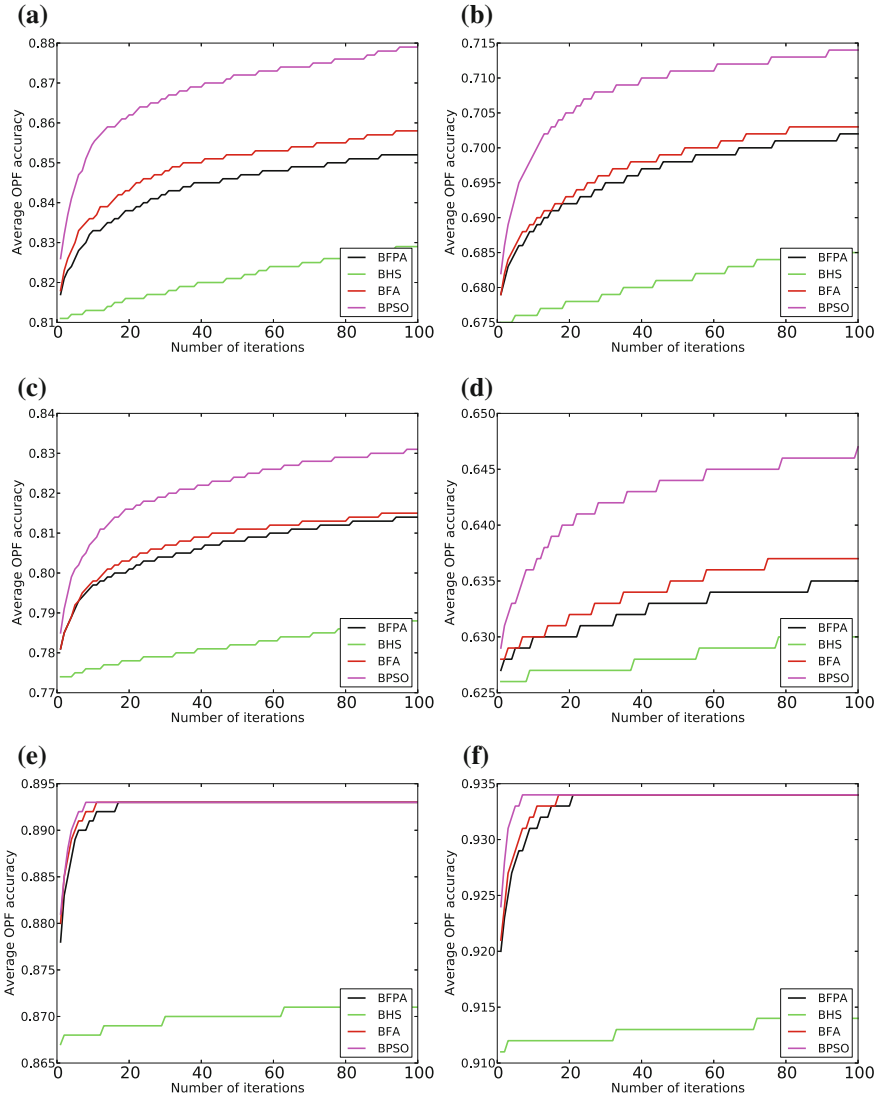


**Fig. 3** Average convergence rate considering the OPF accuracy over the feature selection step for **a** GLI-85, **b** SMK-CAN-187, **c** TOX-171, **d** AR10P, **e** $NTL_c$ and **f** $NTL_i$ datasets

From Fig. 3, it is possible to observe that BPSO has been the technique with the fastest convergence rate, followed by BFA and BFPA. However, the good BPSO performance over the feature selection process does not seem to enhance a lot its final accuracy over the test set, as displayed in Fig. 2. In addition, BHS has the slowest convergence process, since it updates only one agent (harmony) per iteration, which turns it fast considering the execution time, but it tends to be slower for

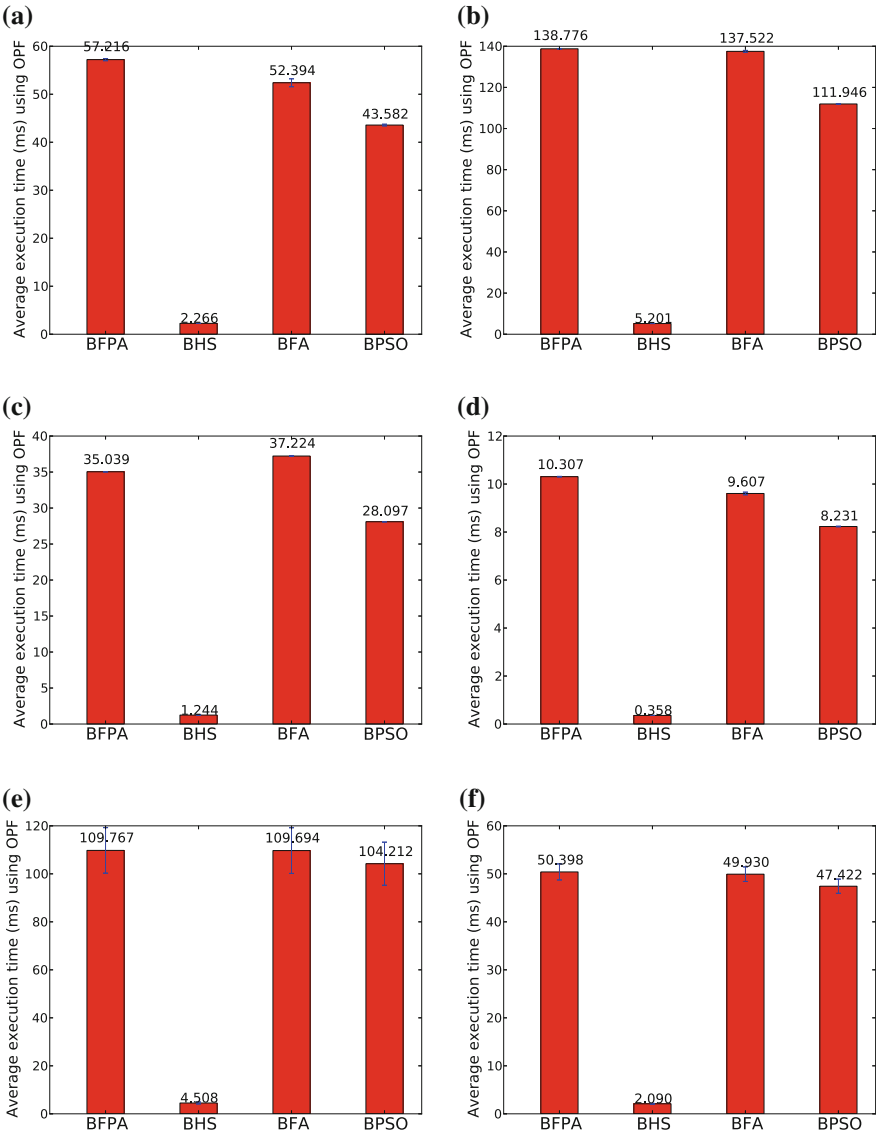


**Fig. 4** Average execution time (ms) for feature selection considering **a** GLI-85, **b** SMK-CAN-187, **c** TOX-171, **d** AR10P, **e** $NTL_c$ and **f** $NTL_i$ datasets

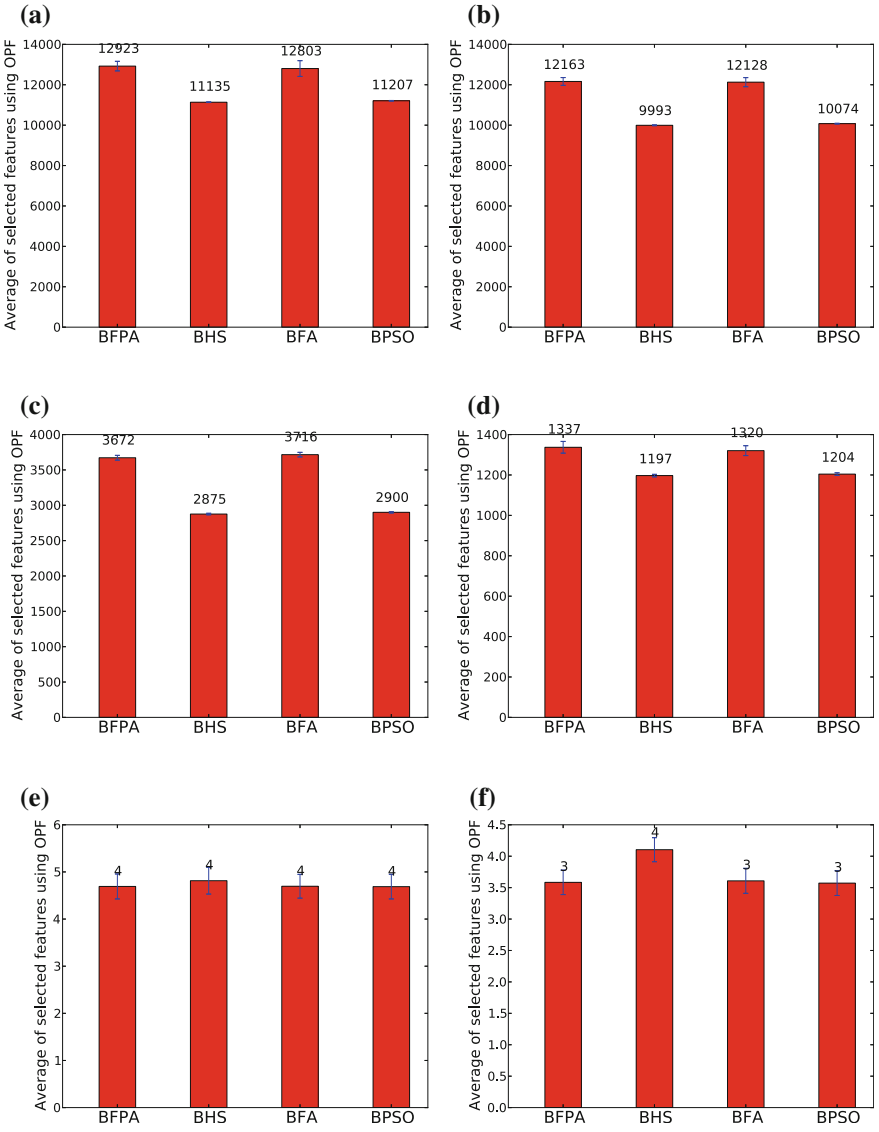


**Fig. 5** Mean number of selected features considering **a** GLI-85, **b** SMK-CAN-187, **c** TOX-171, **d** AR10P, **e** $NTL_c$ and **f** $NTL_i$ datasets. The numbers have been truncated for sake of presentation

convergence. Figure 4 displays the execution time for all considered optimization techniques. Though it is possible to observe that BFPA has been one of the slowest techniques, since it is the only one that employs Lévy flights to move pollens across the search space, which can increase the computational burden slightly. It is also observed that BFPA in general produces better results in terms of accuracy, as seen in Fig. 2.

In addition, Fig. 5 displays the mean number of selected features for each dataset. It is possible to observe BHS has selected the fewest number of features, followed by BPSO. However, as we have high dimensional datasets (in case of GLI-85, SMK-CAN-187, TOX-171 and AR10P), the absolute number do not differ a lot from all techniques. It seems that all techniques have obtained similar results considering the recognition rate, except for the computational load and convergence speed.

## 5 Conclusions

In this work, we have solve the problem of feature selection by considering feature selection as an evolutionary-based optimization task, constrained on a boolean lattice. The idea is to represent each possible solution as a string of bits, in which each of them denotes whether or not a feature will be used to compose the final set.

We have evaluated a recent nature-inspired approach, namely the Flower Pollination Algorithm, to tackle this task on six datasets. The proposed approach has been compared with other methods such as Particle Swarm Optimization, Harmony Search and Firefly Algorithm. The experimental results have been analyzed in terms of the recognition rates, convergence speed, number of selected features and computational loads. All techniques have obtained similar recognition rates, and it seems that PSO has the fastest convergence process, while HS has lowest computational cost. Therefore, we have showed that FPA is also suitable for feature selection tasks, since its results are comparable to the ones obtained by some state-of-the-art evolutionary techniques. Future research can focus on the parametric studies of the FPA as well as its extension and hybridization with other techniques.

## References

1. Bababdani, B.M., Mousavi, M.: Gravitational search algorithm: a new feature selection method for {QSAR} study of anticancer potency of imidazo[4,5-b]pyridine derivatives. Chemometr. Intell. Lab. Syst. **122**(15), 1–11 (2013)
2. Chen, B., Chen, L., Chen, Y.: Efficient ant colony optimization for image feature selection. Signal Process. **93**(6), 1566–1576 (2013). (special issue on Machine Learning in Intelligent Image Processing)
3. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern classification, 2nd edn. Wiley, New York (2001)
4. Falcon, R., Almeida, M., Nayak, A.: Fault identification with binary adaptive fireflies in parallel and distributed systems. In: IEEE Congress on Evolutionary Computation (CEC), pp. 1359–1366 (2011)
5. Firpi, HA., Goodman, E.: Swarmed feature selection. In: Proceedings of the 33rd Applied Imagery Pattern Recognition Workshop. IEEE Computer Society, Washington, DC, pp. 112–118 (2004)
6. Geem, Z.W.: Music-inspired harmony search algorithm: theory and applications, 1st edn. Springer, Berlin (2009)

7. Kabir, M., Shahjahan, M., Murase, K.: An efficient feature selection using ant colony optimization algorithm. In: Leung, C., Lee, M., Chan, J. (eds.) Neural Information Processing, Lecture Notes in Computer Science, vol. 5864, pp. 242–252, Springer, Berlin (2009)
8. Kennedy, J., Eberhart, R.C.: Swarm intelligence. Morgan Kaufman, Burlington (2001)
9. Marinakis, Y., Marinaki, M., Matsatsinis, N.: A hybrid discrete artificial bee colony—GRASP algorithm for clustering. In: Proceedings of the International Conference on Computers Industrial Engineering, pp. 548–553 (2009)
10. Oh, I.S., Lee, J.S., Moon, B.R.: Hybrid genetic algorithms for feature selection. IEEE Trans. Pattern Anal. Mach. Intell. **26**(11), 1424–1437 (2004)
11. Palit, S., Sinha, S.N., Molla, M.A., Khanra, A., Kule, M.: A cryptanalytic attack on the knapsack cryptosystem using binary firefly algorithm. In: 2nd International Conference on Computer and Communication Technology (ICCCT), pp. 428–432 (2011)
12. Papa, J.P., Falcão, A.X., Suzuki, C.T.N.: Supervised pattern classification based on optimum-path forest. Int. J. Imaging Syst. Technol. **19**(2), 120–131 (2009)
13. Papa, J.P., Falcão, A.X., Albuquerque, V.H.C., Tavares, J.M.R.S.: Efficient supervised optimum-path forest classification for large datasets. Pattern Recogn. **45**(1), 512–520 (2012)
14. Ramos, C.C.O., Souza, A.N., Chiachia, G., Falcão, A.X., Papa, J.P.: A novel algorithm for feature selection using harmony search and its application for non-technical losses detection. Comput. Electr. Eng. **37**(6), 886–894 (2011)
15. Ramos, C.C.O., De Souza, A., Falcão, A., Papa, J.: New insights on nontechnical losses characterization through evolutionary-based feature selection. Power Deliv. IEEE Trans. **27** (1), 140–146 (2012)
16. Ramos, C.C.O., de Souza, A.N., Falcão, A.X., Papa, J.P.: New insights on non-technical losses characterization through evolutionary-based feature selection. IEEE Trans. Power Deliv. **27**(1), 140–146 (2012)
17. Rodrigues, D., Pereira, L.A.M., Almeida, T.N.S., Papa, J.P., Souza, A.N., Ramos, C.C.O., Yang, X.S.: A binary cuckoo search algorithm for feature selection. In: Proceedings of IEEE International Symposium on Circuits and Systems, pp. 465–468 (2013a)
18. Rodrigues, D., Pereira, L.A.M., Nakamura, R.Y.M., Costa, K.A.P., Yang, X.S., Souza, A.N., Papa, J.P.: A wrapper approach for feature selection based on bat algorithm and optimum-path forest. Expert Syst. Appl. **41**(5), 2250–2258 (2013)
19. Rodrigues, D., Pereira, L.A.M., Papa, J.P., Ramos, C.C.O., Souza, A.N., Papa, L.P.: Optimizing feature selection through binary charged system search. In: Proceedings of 15th International Conference on Computer Analysis of Images and Patterns, pp. 377–384 (2013c)
20. Schiezaro, M., Pedrini, H.: Data feature selection based on artificial bee colony algorithm. EURASIP J. Image Video Process. **1**, 1–8 (2013)
21. Sivagaminathan, R.K., Ramakrishnan, S.: A hybrid approach for feature subset selection using neural networks and ant colony optimization. Expert Syst. Appl. **33**(1), 49–60 (2007)
22. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bull. **1**(6), 80–83 (1945)
23. Yang, J., Honavar, V.: Feature subset selection using a genetic algorithm. IEEE Intell. Syst. Appl. **13**(2), 44–49 (1998)
24. Yang, X.S.: Firefly algorithm, stochastic test functions and design optimisation. Int. J. Bio-Inspired Comput. **2**(2), 78–84 (2010)
25. Yang, X.S.: Flower pollination algorithm for global optimization. In: Proceedings of the 11th International Conference on Unconventional Computation and Natural Computation, pp. 240–249. Springer, Berlin (2012)
26. Yang, X.S., Karamanoglu, M., He, X.: Flower pollination algorithm: a novel approach for multiobjective optimization. Eng. Optim. **46**(9), 1222–1237 (2014)