Applications of Flower Pollination Algorithm in Feature Selection and Knapsack Problems

Hossam M. Zawbaa and E. Emary

Abstract This chapter presents one of the recently proposed bio-inspired optimization methods, namely, flower pollination algorithm (FPA). FPA for its capability to adaptively search a large search space with maybe many local optima has been employed to solve many real problems. FPA is used to handle the feature selection problem in wrapper-based approach where it is used to search the space of feature for an optimal feature set maximizing a given criteria. The used feature selection methodology was applied in classification and regression data sets and was found to be successful. Moreover, FPA was applied to handle the knapsack problem where different data sets with different dimensions were adopted to assess FPA performance. On all the mentioned problems FPA was benchmarked against bat algorithm (BA), genetic algorithm (GA), particle swarm optimization (PSO) and is found to be very competitive.

Keywords Flower pollination algorithm · Bio-inspired optimization · Evolutionary computation · Feature selection · Knapsack problem

1 Introduction

This chapter presents the importance of flower pollination algorithm (FPA) for feature selection for regression and classification data and knapsack. In the current applications of machine learning and pattern recognition techniques, there are thousands of such features. The vast amounts of data generated today in biology offer more detailed and useful information on one hand; on the contrary, it makes the data analyzing process more difficult because not all the information is relevant. Selecting the important features of a given dataset is a complex problem. *Feature selection* is a technique for solving classification and regression problems, and it identifies

H.M. Zawbaa (✉)
Faculty of Computers and Information, Beni-Suef University, Beni Suef, Egypt
e-mail: hossam.zawbaa@gmail.com

E. Emary
Faculty of Computers and Information, Cairo University, Giza, Egypt

X.-S. Yang (ed.), *Nature-Inspired Algorithms and Applied Optimization*,
Studies in Computational Intelligence 744, https://doi.org/10.1007/978-3-319-67669-2_10

the significant feature subset and removes the unnecessary ones. This mechanism is particularly useful when the size of feature subset is large, and not all of them are required for describing the data features in experiments [1]. Hence, the use of feature selection method is crucial to reduce the enormous number of features. Feature selection helps in understanding data, decreasing the computation time, reducing the effect of the curse of dimensionality and enhancing the performance of prediction model [2]. Furthermore, the feature selection process enhances the visualization and the comprehensibility of the selected feature subset [3].

In real-world applications, due to different reasons not discussed here, many features introduce noise, while others can be totally irrelevant or even misleading, affecting prediction performance. In these cases, feature selection is a must [4]. Two main criteria are employed to differentiate between the feature selection algorithms as follows:

1. *Search strategy*: the method employed to generate feature subsets or feature combinations.
2. *Subset quality (fitness)*: the criteria used to judge the quality of a feature subset.

There are two major approaches of feature selection methods: wrapper-based approach (applying machine learning algorithms) and filter-based approach (using statistical methods) [5]. The *wrapper-based approach* employs a machine learning technique as part of the assessment operation that helps to obtain better results than the filter-based [6], but it has a risk of over-fitting the model and can be computationally costly, and hence, a brilliant search method is required to minimize the computational time [7]. In contrast, the *filter-based approach* explores for a feature subset that optimizes a given data-dependent criterion rather than using classification-dependent criteria as in the wrapper methods [8].

In general, the feature selection is expressed as *multi-objective* with these two goals: (1) *minimize* the selected feature subset and (2) *maximize* the classification precision (*minimize* the prediction error in the regression problems). Commonly, these two goals are contradictory, and the optimal solution is a trade-off between them. Several search methods have been employed, based mainly on greedy search; however, these techniques have at least two drawbacks: stagnation in local optima and big computational time [9]. Evolutionary computing (EC) and population-based algorithms adaptively search the feature space by using a set of search agents that interact in a social manner to reach the optimal solution [10]. EC methods are inspired by the animal social and biological behavior in nature like (wolves, antlions, dragonflies, spiders, and so on) in a group [11].

Most of the recent optimization techniques are *nature-inspired*, i.e. they have been inspired from nature [12].

2 Related Work

Feature selection methods are composed of two elements: the search strategy and the evaluation technique (subset goodness). In the *wrapper-based* approach (alternative to the filter-based approach), the term wrapper refers to the assessment method. Learning boolean is a filter feature selection method that exhaustively explores all potential feature combinations and chooses the minimum feature subset [6].

Various heuristic techniques mimic the biological and physical conducts in nature, and they have been introduced as robust techniques for the global optimization. GA was the earliest evolutionary based technique proposed in the literature, later enhanced relying on the evolution operator during the reproduction [13]. GA feature selection method using a fuzzy set as the fitness function has been introduced in [14]. Wrapper-filter based feature selection methods combine GA with local search methods [15].

In particle swarm optimization (PSO) methods, a solution is represented by a particle with specific properties like position, fitness, and speed [16]. A binary version of PSO (BPSO) modifies the native PSO algorithm to deal with the binary optimization problems [17]. Moreover, an expanded version of BPSO is implemented to deal with feature selection [18]. The binary variant of bat algorithm (BBA) is employed to feature selection, where the search area is described as an n-cube [19].

Ant colony optimization (ACO) uses Fisher discrimination rate to adopt the heuristic information and rough set approach employed for feature selection [20]. Artificial fish swarm (AFS) algorithm mimics the stimulant reaction by controlling the tail and fin [21]. Artificial bee colony (ABC) relies on the natural conduct of honeybees that randomly produced employer bees are moved in the elite bee direction [22]. The elite bee represents the optimal (near to optimal) solution [23]. Antlion optimization algorithm (ALO) is a comparatively recent EC method, which simulates the antlions hunting in nature [24].

Artificial neural networks (ANN) particularly single hidden layer feed-forward neural networks (SLFN) are viewed as a standout amongst the most conventional machine learning models used in regression and classification domains [25]. The learning algorithm is considered the cornerstone of any neural network. Classical gradient-based learning algorithms are suffering from over-fitting, local minima, and they consume a long time to learn [26]. The back-propagation artificial neural network (BP-ANN) has average learning velocity and is likely to get caught in the local minima, leading to miserable performance and efficiency. The revised back propagation artificial neural network (RBP-ANN) is applied to defeat the constraints of BP-ANN and RBP-ANN [27].

In extreme learning machine (ELM) techniques, the output connections are tuned by solving an optimization problem, i.e. finding the minimum of the cost function by linearization [28]. Huang [29] introduced ELM in order to avoid some of the difficulties observed in gradient-based learning methods. ELM is used as a supervised

learning method for SLFN neural networks [30, 31]. ELM is choosing the weights of the input and hidden layers randomly rather than completely adapting all the internal parameters. Moreover, ELM could analytically define the output layer weights [32].

3 Flower Pollination Algorithm (FPA) with Selected Applications

FPA is metaheuristic optimization technique relying on the pollination operation of flowering plants that introduced by Yang in 2012 [33]. Pollination is carried out in two modes *self pollination* (local search) and *cross pollination* (global search). Detailed information about the two ways of pollination as follow [34]:

1. *Cross pollination* happens from the pollen of a flower of a different plant at long distance via pollinators that can fly a big distance (global pollination) [34]. In the cross pollination, the pollinators convey the flower pollens and can fly long distance to assure the pollination and proliferation of the optimal solution g_*. The initial rule may be formulated as in Eq. (1):

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

   where X_i^t represents the vector of a i solution at t iteration, g_* demonstrates the present best solution, and L describes the pollination strength that randomly pulled from the Lèvy distribution.

2. *Self pollination* is implantation of one flower from the pollen of identical flower or different flowers of the identical plant that usually happens when there is no pollinator possible. The local pollination and flower constancy is expressed as in the Eq. (2):

$$X_i^{t+1} = X_i^t + \varepsilon(X_j^t - X_k^t), \tag{2}$$

   where X_j^t and X_k^t demonstrate two random solutions, and ε drawn from the uniform distribution.

Because of local pollination may have substantial fraction (p) in the aggregate pollination actions (in our experiments, we used $p = 0.5$). A switching probability $p\varepsilon[0, 1]$ manages the local and global pollination. FPA search methodology can be outlined as in the algorithm (1).

Inputs: N Total flower agents,
$IterMax$ Total iterations number,
p Switch probability,
Outputs: The best solution (g_*) and its fitness value.
Initialize the N flowers population randomly.
Choose the best solution (g_*).
while Stopping criteria do not meet **do**
  for all Flower i in the solution set **do**
    if $rand < p$ **then**
      Design the L d-dimensional vector based on Lèvy distribution.
      Employ the global pollination on i solution as in the equation (1).
    else
      Pull ϵ from the uniform distribution.
      Select the j and k solutions randomly.
      Execute the local pollination on the i solution by employing the j and k solutions as in the equation (2).
    end if
    Assess the new solution.
    if the new solution is better than the current one **then**
      Substitute the current solution i by the new solution.
    end if
  end for
  Upgrade the optimal solution g_*.
end while
Select the optimal solution and its fitness.

Algorithm 1: Flower pollination algorithm (FPA)

3.1 FPA Applied for Feature Selection

FPA is adopted here for exploiting the capabilities of filter and wrapper approaches for feature selection. The *filter approach* can be described as data-oriented methods that not directly related to classification performance. The *wrapper-based* approach is more related to prediction performance, but it does not face redundancy and dependency among the selected feature set.

We are seeking to find similarities and differences based on some evaluation criteria that may help in finding weak and strength features of each. All swarm intelligence methods regularly share the data between their multiple agents. Therefore, at every iteration, all/some agents upgrade/modify their position relied on the data of their own position and the other positions.

FPA is applied for feature selection in both classification and regression problems. For a vector with N features, the various feature selection would be 2^N that is the vast space of features to be searched *exhaustively*. Therefore, intelligent optimization is applied to explore the search area adaptively for best feature subset. The optimal feature subset is the one with *least prediction error* and a *less number of selected features* as a common objective in literature. In classification problems, the general fitness function for the proposed optimization algorithms is to maximize the

classification accuracy over the validation set given the training set, as shown in Eq. (3) while keeping the minimum number of features selected:

$$\downarrow Fitness = \alpha(1 - P) + \beta\frac{|R|}{|C|}, \tag{3}$$

where R indicates the size of chosen feature set, C demonstrates the total number of features in the dataset, α and β depict the significance of classification performance and the chosen feature set length, $\alpha \in [0, 1]$ and $\beta = 1 - \alpha$, P is the classification performance measured as in Eq. (4):

$$P = \frac{N_c}{N}, \tag{4}$$

where N_c indicates the number of correctly classified instances, and N is the total number of instances.

In the case of regression problems, the general fitness function for the proposed optimization algorithms is to minimize the prediction error over the validation set given the training set as in Eq. (5) while keeping a minimum number of features selected.

$$\downarrow Fitness = \alpha * E + \beta\frac{|R|}{|C|}, \tag{5}$$

where E indicates the prediction error, α and β show the importance of prediction error and selected feature subset respectively. E is defined as:

$$E = \sum_{i=1}^{M} |a_i - t_i|, \tag{6}$$

where a_i and t_i are the actual model prediction value and target value for point i in the validation set.

The used features are the same as the number of features in a given dataset. All features are limited in the range [0, 1], where the feature value approaches to 1; its corresponding feature is a candidate to be selected in classification. In individual fitness calculation, the feature is a threshold to decide whether a feature will be selected at the evaluation stage. Therefore, a static threshold of 0.5 is used as in the Eq. (7):

$$y_{ij} = \begin{cases} 0 & \text{if } x_{ij} < 0.5 \\ 1 & \text{Otherwise,} \end{cases} \tag{7}$$

where x_{ij} is a D—dimensional point in the search space of features and y_{ij} is the binary value $\in 0, 1$ corresponding to selecting/unselecting feature j in solution i from the solution set.

3.2 FPA Applied for Knapsack Problem

Given a set of n elements with each element has a profit p_j and a weight w_j and a Knapsack of capacity C the objective is to find the most profitable solution without violating knapsack weight capacity [35]. A vector describing whether an element is selected or not can be represented in binary form with an n-dimensional vector with individual elements $x_i \in 0, 1$. So, the problem can be mathematically formulated as:

$$Maximimize \sum_{j=1}^{n} p_j x_j, \tag{8}$$

subject to

$$\sum_{j=1}^{n} w_j x_j \leqslant C. \tag{9}$$

The knapsack problem is an NP-hard problem which requires a very intelligent optimization to search the huge search space of possibilities. FPA is adopted in this work to solve a set of Knapsack problems with variant dimensions to prove the searching capability of the FPA. Death penalty [36] is adopted to handle the constraint of the knapsack while the total fitness is calculated as in Eq. (8) but with using negative sign to standardize the maximization into minimization.

4 Experimental Results and Discussion

The global and optimizer-specific parameter setting is outlined in Table 1. All the parameters are set either according to domain-specific knowledge as the α and β parameters of the used fitness function, or based on trial and error on small simulations and common in literature such as the rest of parameters.

In this study, the wrapper approach is used to find a feature subset supervised by the prediction performance. Hence, an intelligent search method is necessary for searching the feature space. In the case of classification datasets, the used classifier in the fitness function as given in Eq. (3) is KNN [37]. KNN is utilized in the experiments based on trial and error basis where the best choice of K is selected ($K = 5$) as the best performing on all the datasets.

4.1 Assessment Indicators

Each algorithm has been applied $K * M$ times with random positioning of the search agents except for the full features selected solution that was compelled to be a posi-

Table 1 The parameter setting for experiments

Parameter	Value(s)
K for cross validation	10
M total number of runs	20
Number of search agents	8
Number of iterations (dimension $<$ 100)	100
Number of iterations (dimension $\geq$ 100)	200
Problem dimension	Number of features in the dataset
Search range in binary methods	{0, 1}
Search range in continuous methods	[0, 1]
α in the fitness function	0.99
β in the fitness function	0.01

tion for one of the search agents. Compelling the full features solution ensures that all consequent feature subsets; if selected as the global best solution, are fitter than it. Repeated runs of the optimization algorithms were applied to test their convergence capability. We have applied two types of indicators (measures) to compare the various algorithms.

1. Firstly, this group of indicators is applied directly to the fitness function obtained based on the validation set and used to characterize the algorithm performance as follows:

   - *Mean fitness*: is an average value of all the solutions in the final sets obtained by an optimizer in a number of individual runs [38].
   - *Median fitness*: is used to assess the average performance tolerating noise performance of the optimizer over all the *M* runs [38].
   - *Best fitness*: is the minimum value of the fitness function that acquired by the optimizer in *M* independent applications [38].
   - *Worst fitness*: is the maximum fitness function value (or worst obtained fitness value) acquired by an optimization method in *M* independent applications [38].
   - *Statistical standard deviation (std)*: is a representation of the variation of the obtained best solutions found for running a stochastic optimizer for *M* different runs. *Std* is used as an indicator for the optimizer capability to converge to same/similar optimal solution [38].

2. The second group of indicators is applied to assess the performance of the entire prediction model as follows:

   - *Average classification error*: depicts how precise the classifier of the chosen feature subset, as shown in the Eq. (10):

$$Perf = \frac{1}{M}\sum_{j=1}^{M}\frac{1}{N}\sum_{i=1}^{N} Unmatch(C_i, L_i), \tag{10}$$

where M represents the total number of runs for the optimization method, N describes the total instances in the test subset; C_i depicts the classifier output label of the i data instance. L_i denotes the source class label of the i data instance, and *Unmatch* specifies the function that yields 0 if the two labels are equivalent and yields 1 otherwise.

- *Mean square error (MSE)*: is measuring the mean square error of the difference between actual output and the predicted one as given in Eq. (11):

$$MSE = \frac{\sum_{i=1}^{n}(pred_i - obs_i)^2}{n}, \tag{11}$$

- *Root mean square error (RMSE)*: is measuring the difference among actual output and the predicted ones as given in Eq. (12):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(obs_i - pred_i)^2}{n}}, \tag{12}$$

where obs_i and $pred_i$ are the observed and predicted values respectively. μ represents the mean of the noticed values, n demonstrates the total of examples, and i depicts the example number in a given dataset.

- *Average selection size*: demonstrates the average size of the chosen feature subset to the aggregate amount of features as in the Eq. (13):

$$Selection\_Size = \frac{1}{M}\sum_{i=1}^{M}\frac{size(g_*^i)}{N_t}, \tag{13}$$

where N_t represents the total number of features in a given dataset.

- *Average feature reduction*: demonstrates the mean size of the reduced features to the aggregate amount of features as in the Eq. (14):

$$Reduction = 1 - \frac{1}{M}\sum_{i=1}^{M}\frac{size(g_*^i)}{N_t}, \tag{14}$$

- *Average Fisher score (F-score)*: assesses the feature subset that has large distances between the data samples in various classes, while the distances among data instances in the same class are as minimum as possible [39]. *F-score* is computed for individual features given the class labels and for M independent applications of an algorithm; as shown in Eq. (15):

$$F_j = \frac{\sum_{k=1}^{c} n_k(\mu_k^j - \mu^j)^2}{(\sigma^j)^2}, \tag{15}$$

where F_j is the Fisher score for feature j, μ^j is the mean of the entire dataset. $(\sigma^j)^2$ is the standard deviation of the whole dataset, n_k denotes the size of the k class, and μ_k^j indicates the mean of k class.

- *Wilcoxon*: introduced by Wilcoxon [40] as a non-parametric test. The test allocates rank to all the scores considered as one group and afterward sums the ranks of every group. The null hypothesis originates from the same population, so any difference in the two rank sums come only from the testing error. The rank sum test is regularly depicted as the non-parametric version of the *T-test* for two independent groups.
- *T-test*: is a statistical significance that decides whether or not the difference between two classes' averages most likely reflects a real difference in the population from which the groups were sampled; as in the Eq. (16) [41].

$$t = \frac{\bar{x} - \mu_0}{\frac{S}{\sqrt{n}}} \tag{16}$$

where μ_0 is the average of the t-distribution and $\frac{S}{\sqrt{n}}$ is its standard deviation.

- *Average computational time*: is the run time for a given optimization algorithm in millisecond that calculated over the different runs as given in Eq. (17):

$$T_o = \frac{1}{M} \sum_{i=1}^{M} RunTime_{o,i}, \tag{17}$$

where M demonstrates the total number of runs for the optimizer O, and $RunTime_{o,i}$ is the computational time in millisecond for optimizer o at run number i.

4.2 Datasets

All datasets were collected to have a variety of *features* and *instances* as delegates of various problem types, which the introduced methods will be examined on. Besides, we selected a set of respectively high dimensional data to ensure the performance of optimization algorithms in huge search spaces. Each dataset is split by cross-validation [42] mode for evaluation, which $K-1$ folds are employed for the training, validation, and testing sets. Each set is repeated M times, hence, each optimizer is

estimated $K * M$ times for individual dataset. Each dataset is equally sized into training, validation, and testing. *Training* part is used to train the used classifier through optimization and at the final evaluation. *Validation* part used to assess the performance of the classifier at the optimization time. *Testing* part is employed to determine the finally selected features given the trained classifier. The classification and regression models are used to ensure the quality of the selected features and are assessed on the validation set inside the fitness function during the optimization process [6]. In the case of regression datasets, the regression model used in the fitness function as in Eq. (5) is extreme learning machine (ELM) with a different number of hidden layers and sigmoid basis function. ELM used for regression purposes and is adopted to evaluate the fitness function. ELM has seven nodes in input layer representation and one hundred hidden nodes (based on trial and error basis); because ELM needs more hidden nodes than the classical gradient training algorithms [28].

Table 2 outlines twenty-one datasets used in classification problems. The datasets are acquired from the UCI machine learning repository [43, 44]. Table 3 displays the ten datasets applied in the regression experiments. The used datasets are picked from the UCI machine learning repository [43].

4.3 FPA for Feature Selection Using Classification Data

In classification data category, the classifier used in fitness function as in Eq. (3) is KNN [37]. KNN is applied in the experiments based on trial and error basis where the best choice of K is selected ($K = 5$) as the best performing on all the datasets. The aggregate purpose of this part is to declare the bio-inspired optimization methods for feature selection approaches that minimize the selected feature set and maximize the classification performance from applying the whole features and conventional feature selection methods in the classification problem.

Table 4 outlines the average statistical mean fitness of FPA [45], BA [46], GA, and PSO optimization algorithms for all 21 classification datasets that calculated over the 20 runs. We can observe that all used optimization methods outperform the full features selected that proves the capability of *wrapper-based* method in feature selection problem. We can also highlight that the CS performs in general better than the other optimizers that demonstrate the ability of CS adaptively to explore the area for the optimal feature combination. For evaluating the stability of the stochastic algorithms in the study and converge to the same optimal solution. We measure the standard deviation, and the results are depicted in the Table 5. We can see that, although the FPA depends on Lèvy distribution that has infinite variance it still keeps comparable std measure.

Table 2 List of datasets used in classification data

DS	Name	No. of features	No. of samples
1	Breastcancer	9	699
2	BreastEW	30	569
3	Clean1	166	476
4	Clean2	166	6598
5	CongressEW	16	435
6	Exactly	13	1000
7	Exactly2	13	1000
8	HeartEW	13	270
9	IonosphereEW	34	351
10	KrvskpEW	36	3196
11	Lymphography	18	148
12	M-of-n	13	1000
13	PenglungEW	325	73
14	Semeion	265	1593
15	SonarEW	60	208
16	SpectEW	22	267
17	Tic-tac-toe	9	958
18	Vote	16	300
19	WaveformEW	40	5000
20	WineEW	13	178
21	Zoo	16	101

Table 3 List of datasets used in regression data

DS	Name	No. of features	No. of samples
1	CASP	9	45730
2	CBM	17	11934
3	CCPP	4	47840
4	ENB2012_Y1	8	768
5	ENB2012_Y2	8	768
6	ForestFire	12	517
7	Housing	13	506
8	RelationNetwork	22	53413
9	Slump_test	10	103
10	Yacht_hydrodynamics	6	308

Table 4 Mean fitness of 20 runs

DS	Full	BA	FPA	GA	PSO
1	0.026	0.022	**0.021**	0.022	0.027
2	0.053	0.024	**0.022**	0.025	0.030
3	0.214	0.140	**0.136**	0.150	0.148
4	0.048	0.038	**0.037**	0.038	0.038
5	0.090	0.036	**0.033**	0.043	0.048
6	0.336	0.161	**0.072**	0.219	0.296
7	0.284	0.237	**0.234**	0.240	0.242
8	0.196	0.132	**0.123**	0.138	0.133
9	0.160	0.109	**0.105**	0.111	0.127
10	0.091	0.037	**0.031**	0.036	0.050
11	0.281	0.136	**0.116**	0.161	0.165
12	0.155	0.037	**0.025**	0.081	0.114
13	0.203	0.175	**0.152**	0.193	0.180
14	0.044	0.030	0.030	0.034	**0.029**
15	0.338	**0.128**	0.132	0.136	0.164
16	0.161	0.136	**0.126**	0.141	0.136
17	0.259	0.222	**0.219**	0.224	0.229
18	0.087	0.033	**0.029**	0.034	0.041
19	0.231	0.202	**0.200**	0.206	0.223
20	0.067	0.015	**0.007**	0.015	0.019
21	0.265	0.102	**0.076**	0.132	0.125
Avg.	0.171	0.102	**0.092**	0.113	0.122

Table 6 outlines the average classification error of the selected feature subset from the optimization methods of test set averaged over the 20 runs. From the table, FPA obtains the best results on average, thus demonstrating the capability of FPA to find optimal feature combinations ensuring proper test performance. Regarding the size of selected features on the original size, Table 7 outlines the kept feature ratio to the total number of features. We can notice that FFA gets the best selection feature subset results in general. The performance over the test data is to some extent compatible with the results from the F-score calculated over the selected features by the different optimizers; as shown in the Table 8. GA has obtained the best F-score values overall. Table 9 outlines the average computational time of different optimization algorithms. From the table, FPA has the best computational time in comparison to all other algorithms.

Table 5 Std of fitness values for 20 runs

DS	Full	BA	FPA	GA	PSO
1	0.013	0.008	0.009	0.009	**0.007**
2	0.018	**0.006**	0.008	0.008	0.010
3	0.047	**0.027**	**0.027**	0.029	0.046
4	0.003	0.003	0.004	**0.002**	0.003
5	0.028	0.016	**0.009**	0.014	0.016
6	0.033	0.118	0.034	0.103	**0.029**
7	0.040	0.017	0.016	**0.012**	0.013
8	0.039	0.024	**0.015**	0.019	0.020
9	**0.005**	0.027	0.025	0.027	0.025
10	0.012	0.008	**0.005**	0.009	0.007
11	0.067	0.044	0.035	**0.028**	0.045
12	**0.019**	0.036	0.035	0.046	0.054
13	**0.005**	0.126	0.102	0.112	0.108
14	0.008	0.003	0.006	0.006	**0.005**
15	0.041	**0.030**	0.041	0.036	0.048
16	0.045	0.027	0.027	0.037	**0.024**
17	0.030	**0.011**	0.012	0.017	0.021
18	0.031	**0.013**	0.015	0.015	0.015
19	0.013	0.013	0.011	**0.008**	0.010
20	**0.000**	0.019	0.012	0.015	0.021
21	**0.029**	0.077	0.059	0.052	0.065
Avg.	0.025	0.031	**0.024**	0.029	0.028

4.4 FPA for Feature Selection Using Regression Data

In regression data, the regression model used in fitness function as in Eq. (5) is extreme learning machine (ELM). The aggregate purpose of this section is to introduce bio-inspired optimization algorithms for feature selection approach that reduce the number of selected feature subset and reduce the prediction error from applying the whole feature set and conventional feature selection techniques in regression problems.

Table 10 outlines the average statistical mean fitness of BA, CS, DA, FFA, FPA, MAKHA, GA, and PSO optimization algorithms for all ten regression datasets that calculated over the 20 runs. We can highlight that the FPA performs in general better than the other optimizers that prove the capability of FPA adaptively to explore the search area for best feature subset. For evaluating the stability of the stochastic algorithms in the study and converge to the same optimal solution. The standard

Table 6 Average classification error of 20 runs

DS	Full	BA	FPA	GA	PSO
1	0.043	0.043	0.044	**0.042**	0.045
2	**0.046**	0.056	0.059	0.063	0.067
3	0.212	0.204	0.212	0.205	**0.197**
4	0.050	0.046	**0.044**	0.048	0.048
5	0.080	0.079	**0.064**	0.068	0.074
6	0.311	0.185	**0.069**	0.253	0.314
7	0.261	0.252	**0.247**	0.253	0.252
8	**0.189**	0.212	0.214	0.210	0.220
9	0.194	0.177	0.177	**0.167**	0.170
10	0.091	0.047	**0.033**	0.041	0.058
11	0.231	0.241	0.225	0.257	**0.223**
12	0.153	0.045	**0.027**	0.092	0.124
13	0.289	0.275	**0.274**	0.311	0.297
14	0.050	0.042	**0.039**	0.042	0.042
15	0.324	0.266	0.269	**0.261**	0.282
16	0.236	0.201	0.191	**0.185**	0.187
17	0.263	0.261	0.260	**0.257**	0.275
18	0.113	0.079	0.070	0.067	**0.066**
19	0.239	0.223	**0.221**	0.224	0.236
20	0.079	0.**071**	0.077	**0.071**	0.086
21	0.286	**0.133**	0.144	0.178	0.144
Avg.	0.178	0.149	**0.141**	0.157	0.162

deviation results are depicted in the Table 11. We can see that, although the FPA depends on Lèvy distribution that has infinite variance it still keeps comparable std measure.

Table 12 describes the mean RMSE of the selected feature subset from the optimization algorithms of test data averaged over the 20 runs. From the table, FFA obtains the best results on average, thus demonstrating the capability of FFA to find optimal feature combinations ensuring proper test performance. Regarding the size of selected features on the original size, Table 13 outlines the kept feature ratio to the total number of features. We can highlight that GA obtains the best selection

Table 7 Average selection size of 20 runs

DS	BA	FPA	GA	PSO
1	**0.506**	**0.506**	0.556	**0.506**
2	**0.441**	0.448	0.456	0.493
3	**0.461**	0.482	0.483	0.468
4	**0.491**	0.515	0.502	0.516
5	0.424	**0.299**	0.396	0.403
6	0.513	**0.462**	0.556	0.556
7	0.308	0.299	0.256	**0.171**
8	0.521	0.496	**0.487**	0.556
9	0.399	**0.343**	0.395	0.422
10	**0.451**	0.454	0.488	**0.451**
11	0.401	0.481	**0.395**	0.451
12	0.513	**0.496**	0.624	0.581
13	**0.403**	0.444	0.426	0.416
14	**0.462**	0.488	0.478	0.481
15	**0.398**	0.420	0.415	0.409
16	**0.379**	0.434	0.414	0.429
17	0.654	0.605	0.605	**0.593**
18	0.340	0.299	**0.271**	0.347
19	0.542	0.533	0.583	**0.497**
20	0.393	**0.342**	0.470	0.410
21	**0.347**	0.424	0.375	0.382
Avg.	0.445	**0.441**	0.459	0.454

features size results overall. Table 14 outlines the average computational time of different optimization algorithms. From the table, DA has the best computational time in comparison to all other algorithms.

4.5 FPA for Knapsack Problem

In this section, FPA is used and benchmarked against BA, GA, and PSO on the binary Knapsack problem. A set of 20 benchmark problems were in the study having different dimensionality and capacities as in Table 15.

Table 8 Average F-score of 20 runs

DS	BA	FPA	GA	PSO
1	0.735	0.710	0.752	**0.680**
2	**0.218**	0.245	0.234	0.242
3	0.009	0.009	0.009	**0.008**
4	**0.008**	**0.008**	**0.008**	0.009
5	0.205	**0.178**	0.212	0.212
6	**0.001**	**0.001**	**0.001**	**0.001**
7	0.001	0.001	0.001	**0.000**
8	0.084	**0.078**	0.082	0.089
9	0.032	**0.027**	0.034	0.036
10	0.021	0.021	0.021	**0.020**
11	**0.132**	0.179	0.150	0.142
12	0.031	0.031	0.030	**0.028**
13	**0.310**	0.342	0.326	0.319
14	**0.009**	0.010	0.010	0.010
15	0.019	0.019	0.019	**0.018**
16	**0.021**	0.024	0.024	0.025
17	**0.005**	**0.005**	**0.005**	**0.005**
18	0.174	0.152	**0.145**	0.175
19	0.135	0.138	0.136	**0.117**
20	0.448	**0.425**	0.503	0.491
21	12.207	12.876	**10.636**	12.690
Avg.	0.705	0.737	**0.635**	0.729

Functions F1–F20 are expected to evaluate *the exploitation capability* of a given algorithm. We can see in Table 16 that the performance of the FPA optimization algorithm on the average outperforms the other methods. Such result proves the exploitation capability of the FPA algorithm. The same conclusion can be derived by remarking the median performance presented in Table 17 where the FPA still outperform the BA, GA, and PSO algorithms.

Table 18 depicts the best performance indicator for running individual optimizers over 20 runs. Such indicator targets the optimistic users. We can see from the tables that the FPA outperforms the GA and PSO. Table 19 depicts the worst fitness indicator for both simple and composite benchmark functions. Such indicator is expected to assess the worst performance of a given optimizer and hence target the pessimistic

Table 9 Average computational time (milliseconds) of 20 runs for other optimizers

DS	BA	FPA	GA	PSO
1	74.968	68.312	**48.753**	74.576
2	75.343	63.957	**55.524**	73.486
3	71.724	74.145	**39.352**	68.027
4	3081.594	2886.748	**1722.827**	3219.031
5	72.385	59.438	**32.316**	47.470
6	75.994	75.511	**57.433**	83.390
7	111.914	109.125	**59.681**	90.601
8	68.224	61.606	**47.014**	63.682
9	68.015	68.651	**61.015**	75.368
10	434.304	417.282	**395.025**	406.907
11	57.241	48.661	**33.788**	50.591
12	95.730	83.870	**61.417**	98.346
13	50.366	53.329	**23.921**	45.931
14	551.344	550.024	**305.656**	565.127
15	72.873	**71.880**	3979.521	72.161
16	69.113	68.056	**45.926**	61.559
17	103.918	99.622	**75.024**	99.064
18	70.820	71.250	**56.363**	67.409
19	914.644	885.766	**795.628**	966.532
20	41.435	33.260	**19.982**	37.951
21	48.605	41.365	**26.276**	39.603
Avg.	295.741	**280.565**	378.211	300.324

Table 10 Mean fitness of 20 runs

DS	Full	BA	FPA	GA	PSO
1	6.104	**5.491**	5.495	5.692	5.605
2	0.008	0.004	**0.003**	0.007	0.006
3	17.090	**4.621**	4.763	5.156	5.073
4	5.143	3.041	**2.792**	3.399	3.389
5	7.906	3.287	**3.173**	3.211	3.208
6	131.740	57.736	**56.612**	58.683	57.327
7	8.828	4.122	**3.906**	4.617	4.912
8	0.189	**0.049**	0.052	0.056	0.051
9	7.566	**3.262**	3.411	4.071	3.872
10	10.431	1.766	**1.964**	3.884	3.122
Avg.	19.500	8.338	**8.217**	8.878	8.656

Table 11 Std of fitness values for 20 runs

DS	Full	BA	FPA	GA	PSO
1	0.009	0.168	**0.006**	0.084	0.091
2	**0.000**	0.003	0.002	**0.000**	**0.000**
3	**0.001**	0.136	0.100	0.058	0.047
4	**1.140**	0.433	0.143	0.868	0.235
5	0.742	0.306	0.094	0.100	**0.043**
6	96.463	28.073	**27.392**	28.576	28.008
7	1.034	0.327	0.321	**0.269**	0.580
8	0.174	**0.000**	0.002	0.003	0.003
9	0.332	0.326	0.161	**0.116**	0.178
10	0.553	**0.305**	0.320	1.277	2.372
Avg.	10.045	3.008	**2.854**	3.135	3.156

Table 12 Average RMSE of 20 runs

DS	Full	BA	FPA	GA	PSO
1	6.123	5.830	5.837	**5.824**	5.834
2	0.007	0.006	**0.005**	0.006	0.007
3	12.784	7.846	8.775	**5.298**	6.302
4	4.560	3.923	**2.933**	3.419	4.110
5	4.232	3.800	**3.207**	3.325	4.432
6	92.045	178.521	203.068	**59.166**	59.853
7	7.664	5.070	**4.672**	5.086	6.176
8	0.058	**0.054**	**0.054**	0.056	0.056
9	5.847	**5.485**	5.496	5.559	5.627
10	9.442	**3.371**	3.780	4.095	4.583
Avg.	14.276	21.391	23.783	**9.183**	9.698

users' satisfaction. We can see from the table that the worst performance of the FPA still outperform the other algorithms and proves the capability of using such FPA for pessimistic applications.

Table 20 depicts the standard deviation of individual optimizer's output best solution through the 30 runs. Such indicator is expected to assess the *repeatability* of the obtained solutions and the *convergence to same/similar optima*. We can see

Table 13 Average selection size of 20 runs

DS	BA	FPA	GA	PSO
1	**0.463**	0.500	0.500	0.500
2	0.471	**0.461**	0.471	0.539
3	0.417	0.500	**0.250**	0.333
4	0.438	**0.375**	0.438	0.542
5	0.458	**0.438**	**0.438**	0.521
6	0.458	**0.403**	0.417	0.528
7	0.410	0.436	**0.359**	0.410
8	0.568	**0.553**	0.689	0.561
9	0.567	0.533	**0.467**	0.567
10	0.250	**0.222**	0.250	0.361
Avg.	0.450	0.442	**0.428**	0.486

Table 14 Average computational time (in milliseconds) of 20 runs

DS	BA	FPA	GA	PSO
1	**1516.553**	1685.297	1625.241	1600.838
2	**1385.868**	1408.185	895.429	1465.862
3	1377.098	1395.069	1330.866	**1188.551**
4	774.341	787.606	750.926	**616.100**
5	778.917	860.789	753.356	**401.869**
6	847.260	847.406	**789.993**	866.437
7	551.635	574.881	469.265	**450.445**
8	**2181.821**	2254.263	2361.871	2330.990
9	**807.589**	825.084	838.978	813.069
10	730.932	**693.650**	705.527	746.888
Avg.	1095.201	1133.223	1052.145	**1048.105**

from Table 20 that the standard deviation for the FPA outperforms the other optimizers which proves that FAP has much exploration capability it can still converge to same/similar optimal and hence can be considered as a candidate optimizer for *repeatable* results.

Table 15 Used problem sets and the corresponding dimension of each problem

Function no.	No. Dims	Function no.	No. Dims
F1	10	F11	4
F2	5	F12	13
F3	6	F13	11
F4	7	F14	18
F5	8	F15	7
F6	7	F16	16
F7	15	F17	5
F8	24	F18	14
F9	4	F19	17
F10	18	F20	20

Table 16 Mean fitness for the different used optimizers on the different problems

Function no.	BA	FPA	GA	PSO
F1	−307.750	**−309**	−288.200	−300.200
F2	−49.800	**−51**	−50	−50.600
F3	−146.900	**−150**	−138.750	−142.250
F4	−105.450	**−107**	−103.200	−105.250
F5	−895.500	**−99800**	−892.600	−896.200
F6	**−1735**	**−1735**	−1728.800	−1733.550
F7	−1457.800	**−1458**	−1442.750	−1449
F8	−13519668.200	**−13535674.350**	−13138630.650	−13406262.400
F9	−1656.400	**−1663**	**−1663**	**−1663**
F10	−5957.550	**−5959**	−5891	−5950.600
F11	−1713.950	**−1719**	**−1719**	**−1719**
F12	**−6933**	**−6933**	−6863.100	−6932.600
F13	−5479.850	**−5486**	−5401.050	−5443.450
F14	−9002.750	**−9023**	−8908	−8925.600
F15	−3332.400	**−3345**	−3335	**−3345**
F16	−9760.100	**−9773**	−9577.700	−9729.750
F17	**−3573**	**−3573**	**−3573**	**−3573**
F18	−6628	**−6636**	−6580.600	−6613.450
F19	−6696.650	**−6701**	−6448.100	−6627.600
F20	−8715.600	**−8738**	−8339.500	−8674.300

Table 17 Median fitness for the different used optimizers on the different problems

Function No.	BA	FPA	GA	PSO
F1	−309	−309	−284	**−309**
F2	**−51**	**−51**	**−51**	**−51**
F3	**−150**	**−150**	**−150**	**−150**
F4	**−107**	**−107**	−105	**−107**
F5	**−900**	**−900**	−888	**−900**
F6	**−1735**	**−1735**	**−1735**	**−1735**
F7	**−1458**	**−1458**	−1443	−1449.500
F8	−13520148.500	**−13549094**	−13109204.500	−13421603
F9	**−1663**	**−1663**	**−1663**	**−1663**
F10	**−5959**	**−5959**	−5927	**−5959**
F11	**−1719**	**−1719**	**−1719**	**−1719**
F12	**−6933**	**−6933**	−6929	**−6933**
F13	**−5486**	**−5486**	**−5486**	**−5486**
F14	**−9023**	**−9023**	**−9023**	**−9023**
F15	**−3345**	**−3345**	**−3345**	**−3345**
F16	**−9773**	**−9773**	−9688	**−9773**
F17	**−3573**	**−3573**	**−3573**	**−3573**
F18	**−6636**	**−6636**	**−6636**	**−6636**
F19	**−6701**	**−6701**	−6481	**−6701**
F20	**−8738**	**−8738**	−8328	**−8738**

Tables 21 and 22 depict The P-value for two of the common significance tests that are expected to assess the significance of output enhance using the proposed variants. The used significance tests are two-sided Wilcoxon test and T-test. We can see that the P-value for Wilcoxon and T-test are around 0 and hence neglecting the null hypothesis and hence proves the significance of the proposed variant that it is found to be significant using FPA rather than BA, GA, and PSO algorithms.

Table 18 Best fitness for the different used optimizers on the different problems

Function no.	BA	FPA	GA	PSO
F1	**−309**	**−309**	**−309**	**−309**
F2	**−51**	**−51**	**−51**	**−51**
F3	**−150**	**−150**	**−150**	**−150**
F4	**−107**	**−107**	**−107**	**−107**
F5	**−900**	**−900**	**−900**	**−900**
F6	**−1735**	**−1735**	**−1735**	**−1735**
F7	**−1458**	**−1458**	−1456	−1456
F8	**−13549094**	**−13549094**	−13407977	−13518963
F9	**−1663**	**−1663**	**−1663**	**−1663**
F10	**−5959**	**−5959**	**−5959**	**−5959**
F11	**−1719**	**−1719**	**−1719**	**−1719**
F12	**−6933**	**−6933**	**−6933**	**−6933**
F13	**−5486**	**−5486**	**−5486**	**−5486**
F14	**−9023**	**−9023**	**−9023**	**−9023**
F15	**−3345**	**−3345**	**−3345**	**−3345**
F16	**−9773**	**−9773**	**−9773**	**−9773**
F17	**−3573**	**−3573**	**−3573**	**−3573**
F18	**−6636**	**−6636**	**−6636**	**−6636**
F19	**−6701**	**−6701**	**−6701**	**−6701**
F20	**−8738**	**−8738**	**−8738**	**−8738**

5 Conclusions

This work assesses the performance of FPA on two application domains namely feature selection and knapsack. For feature selection, FPA can overcome the performance of BA, GA, and PSO for its capability to adaptively search the search space with many local optima avoiding premature convergence. In the domain of knapsack also FPA is found to be very competitive to PSO, GA, and BA with the tolerable difference in run time and better optimization performance.

On the basis of future performance, we have five ideas that can be investigated in addition to the work presented here:

1. The proposed FPA method will be assessed using complex datasets that have a huge number (thousands) of input features.
2. Add more statistics evaluation measures such as (sensitivity, specificity, and F-measure).

Table 19 Worst fitness for the different used optimizers on the different problems

Function no.	BA	FPA	GA	PSO
F1	−284	**−309**	−239	−247
F2	−47	**−51**	−47	−47
F3	−119	**−150**	−119	−119
F4	−93	**−107**	−91	−93
F5	−858	**−900**	−883	−888
F6	**−1735**	**−1735**	−1682	−1706
F7	−1454	**−1458**	−1427	−1441
F8	−13482886	**−13494864**	−12914151	−13125716
F9	−1531	**−1663**	**−1663**	**−1663**
F10	−5930	**−5959**	−5729	−5797
F11	−1618	**−1719**	**−1719**	**−1719**
F12	**−6933**	**−6933**	−6350	−6925
F13	−5363	**−5486**	−5054	−5058
F14	−8618	**−9023**	−8448	−8338
F15	−3093	**−3345**	−3145	**−3345**
F16	−9515	**−9773**	−8633	−9565
F17	**−3573**	**−3573**	**−3573**	**−3573**
F18	−6476	**−6636**	−6436	−6185
F19	−6641	**−6701**	−5819	−6249
F20	−8442	**−8738**	−7823	−8355

3. Employ bio-inspired optimization methods for solving the challenging problems and in different applications like big data, bioinformatics, and biomedical.
4. Use more machine learning techniques for wrapper-based fitness evaluation such as support vector machine (SVM), random forest (RF), and support vector regression (SVR).
5. Propose a multi-objective fitness function that uses bio-inspired algorithms to the find optimal feature subset.

Table 20 Standard deviation of fitness for the different used optimizers on the different problems

Function no.	BA	FPA	GA	PSO
F1	5.590	**0**	19.718	17.307
F2	1.881	**0**	1.777	1.231
F3	9.542	**0**	14.917	13.772
F4	3.845	**0**	4.753	3.810
F5	10.092	**0**	6.451	5.540
F6	**0**	**0**	13.513	6.485
F7	0.894	**0**	7.873	4.645
F8	21138.405	**16426.210**	124654.021	95227.213
F9	29.516	**0**	**0**	**0**
F10	6.485	**0**	77.911	36.165
F11	22.584	**0**	**0**	**0**
F12	**0**	**0**	162.916	1.789
F13	27.504	**0**	146.101	101.379
F14	90.561	**0**	169.130	191.947
F15	56.349	**0**	44.721	**0**
F16	57.691	**0**	284.099	57.424
F17	**0**	**0**	**0**	**0**
F18	35.777	**0**	72.917	100.847
F19	14.420	**0**	240.613	152.194
F20	72.796	**0**	261.054	121.372

Table 21 P-value for T-test of FPA compared to other optimizers

Optimzer_1	Optimzer_2	P-value
FPA	BA	1.835600e-02
FPA	GA	0.000
FPA	PSO	0.000

Table 22 P-value for Wilcoxon of FPA compared to other optimizers

Optimzer_1	Optimzer_2	P-value
FPA	BA	2.000000e-06
FPA	GA	0.000
FPA	PSO	0.000

References

1. Chizi, B., Rokach, L., Maimon, O.: A survey of feature selection techniques, pp. 1888–1895. IGI Global (2009)
2. Chandrashekar, G., Sahin, F.: A survey on feature selection methods. Comput. Electr. Eng. **40**(1), 16–28 (2014)
3. Huang, C.L.: ACO-based hybrid classification system with feature subset selection and model parameters optimization. Neurocomputing **73**(1–3), 438–448 (2009)
4. Chen, Y., Miao, D., Wang, R.: A rough set approach to feature selection based on ant colony optimization. Pattern Recognit. Lett. **31**(3), 226–233 (2010)
5. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artif. Intell. **97**(1), 273–324 (1997)
6. Xue, B., Zhang, M., Browne, W.N.: Particle swarm optimisation for feature selection in classification: novel initialisation and updating mechanisms. Appl. Soft Comput. **18**, 261–276 (2014)
7. Guyon, I., Elisseeff, A.: An introduction to variable and attribute selection. Mach. Learn. Res. **3**, 1157–1182 (2003)
8. Chuang, L.Y., Tsai, S.W., Yang, C.H.: Improved binary particle swarm optimization using catfish effect for feature selection. Expert Syst. Appl. **38**(10), 12699–12707 (2011)
9. Xue, B., Zhang, M., Browne, W.N.: Particle swarm optimization for feature selection in classification: a multi-objective approach. IEEE Trans. Cybern. **43**(6), 1656–1671 (2013)
10. Shoghian, S., Kouzehgar, M.: A comparison among wolf pack search and four other optimization algorithms. Comput. Electr. Autom. Control Inf. Eng. **6**(12), 1619–1624 (2012)
11. Valdez, F.: Bio-Inspired Optimization Methods. Handbook of Computational Intelligence, pp. 1533–1538. Springer (2015)
12. Jr, I.F., Yang, X.S., Fister, I., Brest, J., Fister, D.: A brief review of nature-inspired algorithms for optimization. Elektrotehniski Vestnik **80**(3), 116–122 (2013)
13. Holland, J.H.: Adaptation in natural and artificial systems. MIT Press, Cambridge, MA, USA (1992)
14. Xue, X., Yao, M., Wu, Z., Yang, J.: Genetic ensemble of extreme learning machine. Neurocomputing **129**(1), 175–184 (2014)
15. Zhu, Z.X., Ong, Y.S., Dash, M.: Wrapper-filter feature selection algorithm using a memetic framework. IEEE Trans. Syst. Man Cybern. Part B: Cybern **37**, 70–76 (2007)
16. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory, pp. 39–43. International Symposium on Micro Machine and Human, Science (1995)
17. Kennedy, J., Eberhart, R.C.: A discrete binary version of the particle swarm algorithm. IEEE International Conference on System, Man and Cybernetics, vol. 5, pp. 4104–4108 (1997)
18. Firpi, H.A., Goodman, E.: Swarmed feature selection. In: 33rd Applied Imagery Pattern Recognition Workshop, USA, pp. 112–118 (2004)
19. Nakamura, R.Y.M., Pereira, L.A.M., Costa, K.A., Rodrigues, D., Papa, J.P., Yang, X.S.: BBA: a binary bat algorithm for feature selection. In: IEEE XXV Conference on Graphics, Patterns and Images, pp. 291–297 (2012)
20. Ming, H.: A rough set based hybrid method to feature selection. In: International Symposium on Knowledge Acquisition and Modeling, pp. 585–588 (2008)
21. Li, X.L., Shao, Z.J., Qian, J.X.: An optimizing method based on autonomous animates: Fish-swarm algorithm, pp. 32–38. Methods and practices of system, engineering (2002)
22. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Glob. Optim. **39**, 459–471 (2007)
23. Sundareswaran, K., Sreedevi, V.T.: Development of novel optimization procedure based on honey bee foraging behavior. In: International Conference on Systems, Man and Cybernetics, pp. 1220–1225 (2008)
24. Mirjalili, S.: The Ant Lion optimizer. Adv. Eng. Softw. **83**, 80–98 (2015)
25. Miche, Y., Sorjamaa, A., Bas, P., Simula, O., Jutten, C., Lendasse, A.: OP-ELM: optimally pruned extreme learning machine. IEEE Trans. Neural Netw. **21**(1), 158–162 (2010)

26. Han, F., Huang, D.S.: Improved extreme learning machine for function approximation by encoding a priori information. Neurocomputing **69**(1), 2369–2373 (2006)
27. Xu, H., Yu, B.: Automatic thesaurus construction for spam filtering using revised back propagation neural network. Expert Syst. Appl. **37**, 18–23 (2010)
28. Jiuwen, C., Zhiping, L.: Extreme Learning Machines on High Dimensional and Large Data Applications: A Survey. Mathematical Problems in Engineering, Hindawi Publishing Corporation, vol. 2015, no. 1, pp. 1–13 (2015)
29. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: a new learning scheme of feedforward neural networks. In: International Joint Conference on Neural Networks, pp. 985–990 (2004)
30. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: theory and applications. Neurocomputing **70**(1), 489–501 (2006)
31. Li, X., Xie, H., Wang, R., Cai, Y., Cao, J., Wang, F., Min, H., Deng, X.: Empirical analysis: stock market prediction via extreme learning machine. Neural Comput. Appl. **1**(3), 1–12 (2014)
32. Zhao, G.P., Hen, Z.Q., Miao, C.Y., Man, Z.H.: On improving the conditioning of extreme learning machine: a linear case. In: International Conference on Information, Communications and Signal Processing, pp. 1–5 (2009)
33. Yang, X.S.: Flower pollination algorithm for global optimization. Unconventional Computation and Natural Computation. Lecture Notes in Computer Science, vol. 7445, pp. 240–249 (2012)
34. Yang, X.S., karamanoglu, M., He, X.: Multi-objective Flower Algorithm for optimization. In: International Conference on Computational Science, Procedia Computer Science, vol. 18, pp. 861–868 (2013)
35. Ghosh, D., Goldengorin, B.: The binary knapsack problem: solutions with guaranteed quality. In: SOM-theme A Primary Processes within Firms (2001)
36. Yeniay, O.: Penalty function methods for constrained optimization with genetic algorithms. Math. Comput. Appl. **10**(1), 45–56 (2005)
37. Yang, C.S., Chuang, L.Y., Li, J.C., Yang, C.H.: Chaotic binary particle swarm optimization for feature selection using logistic map. In: IEEE Conference on Soft Computing in Industrial Applications, pp. 107–112 (2008)
38. Tilahun, S.L., Ong, H.C.: Prey-predator algorithm: a new metaheuristic algorithm for optimization problems. Inf. Technol. Decis. Mak. **14**(6), 1331–1352 (2015)
39. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley-Interscience (2000)
40. Wilcoxon, F.: Individual comparisons by ranking methods. Biom. Bull. **1**(6), 80–83 (1945)
41. Rice, J.A.: Mathematical Statistics and Data Analysis, 3rd edn. Duxbury Advanced (2006)
42. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Series in Statistics (2009)
43. Bache, K., Lichman, M.: UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, 2013, lastchecked on 15 May 2017. http://archive.ics.uci.edu/ml
44. Raman, B., Ioerger, T.R.: Instance-Based Filter for Feature Selection. Machine Learning Research, pp. 1–23 (2002)
45. Yang, X.S.: Nature-Inspired Metaheuristic Algorithms, 2nd edn. Luniver Press, UK (2010)
46. Yang, X.S.: A New Metaheuristic Bat-Inspired Algorithm. Nature Inspired Cooperative Strategies for Optimization, vol. 284, pp. 65–74. Springer (2010)