International Journal of Computational Intelligence Systems, Vol. 11 (2018) 1–14

Bare bones particle swarm optimization with adaptive chaotic jump for feature selection in classification

Chenye Qiu

School of Internet of Things, Nanjing University of Posts and Telecommunications
No.66 Xinmofan Road
Nanjing, Jiangsu, 210003, China
E-mail: qiuchenye@njupt.edu.cn

Received 11 April 2017

Accepted 13 September 2017

Abstract

Feature selection (FS) is a crucial data pre-processing process in classification problems. It aims to reduce the dimensionality of the problem by eliminating irrelevant or redundant features while achieve similar or even higher classification accuracy than using all the features. As a variant of particle swarm optimization (PSO), Bare bones particle swarm optimization (BBPSO) is a simple but very powerful optimizer. However, it also suffers from premature convergence like other PSO algorithms, especially in high-dimensional optimization problems. In order to improve its performance in FS problems, this paper proposes a novel BBPSO based FS method called BBPSO-ACJ. An adaptive chaotic jump strategy is designed to help the stagnated particles make a large change in their searching trajectory. It can enrich the search behavior of BBPSO and prevent the particles from being trapped into local attractors. A new global best updating mechanism is employed to reduce the size of obtained feature subset. The proposed BBPSO-ACJ is compared with eight evolutionary computation (EC) based wrapper methods and two filter methods on nine benchmark datasets with different number of dimensions and instances. The experimental results indicate that the proposed method can select the most discriminative features from the entire feature set and achieve significantly better classification performance than other comparative methods.

Keywords: feature selection, bare bones particle swarm, adaptive chaotic jump, global best updating mechanism.

1. Introduction

Feature selection (FS) is an effective and important data pre-processing step for data mining and pattern recognition[1]. In high-dimensional classification problem, a large number of features may significantly degrade the classification performance of learning algorithm and increase the computational cost, which causes “the curse of dimensionality”. FS aims to select the most informative and discriminative features from the original entire feature set to train a classification model[2]. By discarding irrelevant or redundant features, a smaller feature subset has three main advantages: 1) reduce the computational cost; 2) improve the performance of classifier and avoid over-fitting; 3) enhance the interpretation ability of the classification model.

According to the method used to evaluate the feature subsets, FS approaches can be divided into two categories: wrapper[3,4] and filter[5] approaches. The wrapper approach uses a given learning algorithm to evaluate the feature subsets while the filter approach utilizes the inherent characteristics of the dataset to evaluate the feature subsets, such as the correlation, redundancy, and statistical dependence[6,7]. The wrapper approach usually gets better classification performance since the feature subsets are directly chosen according to their classification accuracies, but it also needs considerable computational time due to the learning algorithm in the evaluation process[8,9].

The goal of FS is to find optimal feature subsets according to some evaluation criteria. Therefore, FS can be modelled as a combinatorial optimization problem. The main difficulty of FS lies in the large search space because the number of features would make an exponentially increase for the search space[10,11]. For a dataset with n features, there are 2^n possible feature subsets. In this situation, an exhaustive search method which considers all the possible feature subsets is not suitable for solving FS problem due to very high computational cost[12]. In order to solve the complicated combinatorial optimization problem accurately and efficiently, a powerful global search algorithm is an essential requirement. Particle swarm optimization is a population based optimization algorithm which is inspired by the social behavior of bird flocking[13] and it shows strong global search capacity due to its exquisite design of algorithm structure. It has been successfully used in many real-world applications due to its fast convergence speed and ease of implementation. PSO has been extended to FS problems recently and many PSO-based approaches have shown promising results. Wang *et al.*[14] proposed a FS model based on PSO and rough sets theory and the experimental results shows the proposed approach outperforms GA based FS methods. In order to prevent premature convergence in FS problem, Chuang *et al.*[15] introduced the catfish effect to binary PSO to strength its search ability. In Ref. 16, the inertia weight of binary PSO was generated with chaotic sequences in order to improve the performance of PSO in FS problem[16]. Jiang *et al.*[17] applied artificial fish swarm algorithm to PSO in order to improve its local search capacity. In Ref.18, PSO with novel initialization methods and global best updating strategies was applied to FS problem. Butler-Yeoman *et al.*[19] proposed two versions of hybrid PSO based FS methods which take advantages of both filter and wrapper evaluations. Moradi *et al.*[20] improved the performance of PSO by introducing a local search operator to reduce the size of obtained feature subsets.

These studies demonstrate the capability of PSO in solving FS problems. But there is one problem when applying PSO to FS which would largely affect the quality of the obtained feature subsets. In PSO, there are several important control parameters which would strongly affect the search behavior and the optimization ability of the algorithm, such as the inertia weight, cognitive weight, and social weight. Many studies have indicated that the proper setting of these parameters is crucial for PSO[21,22]. Inappropriate setting of these problems may lead to premature convergence or low convergence speed.

However, there exist no simple principles about how to select these parameters appropriately in different application scenarios. It is still an open question about how to set and adjust these parameters in different optimization problems.

In order to deal with the disadvantages of PSO, Kennedy[23] proposed a variant of PSO, called bare bones PSO (BBPSO). Unlike the traditional PSO, BBPSO eliminates the velocity term and uses the Gaussian sampling to update the positions of particles based on social and personal flying experience. In the standard BBPSO, those important parameters in PSO do not exist anymore and only the position term is considered in the evolutionary process. Therefore, BBPSO is almost a parameter free algorithm which makes the structure of the algorithm very simple but it also shows powerful optimization ability.

However, BBPSO also suffers from premature convergence in the evolutionary process due to its special search mechanism. In BBPSO, if a particle's personal best is also the global best, it would stay in its present position until some other particles finds better solutions. Therefore, BBPSO may get stuck into local optimal, especially in high-dimensional optimization problems. In order to overcome the premature convergence problem, some jump or disruption strategies were introduced to strengthen the search ability of the algorithm. Krohling and Mendel[24] performed Gaussian or Cauchy jump on the stagnated particles to help them escape from local optimal. Liu *et al.*[25] introduced a new disruption strategy to keep the balance between exploration and exploitation ability of the algorithm. Lee *et al.*[26] introduced a heterogeneous cooperation and jump strategy to strength the exploration ability of the BBPSO. Blackwell[27] analyzed BBPSO theoretically and a series of experimental results showed that an adaptive distribution can effectively improve the performance of the algorithm.

Some researchers have adopted BBPSO for FS problem. Zhang *et al.*[28] proposed a BBPSO with a new local leader updating strategy and uniform combination for FS problem. In Ref.29, BBPSO with a chaotic initialization strategy was applied to solve FS problem[29]. However, there are some problems need to be further investigated in order to improve the performance of BBPSO in FS.

(i) In high-dimensional FS problem with a large number of candidate feature subsets, BBPSO may fall into local optimal during the search process. Therefore, the optimization ability of BBPSO needs to be enhanced to overcome the premature convergence problem.

(ii) Various real-world FS problems show great diversity over the data characteristics and the number of features and instances. In order to obtain optimal feature subsets in different application scenarios, BBPSO should adjust its search behavior according to the evolutionary status of population adaptively.

Available researches on applying BBPSO for FS problems are relatively few and they cannot fully exploit the potential of BBPSO in this area. In order to solve above issues and fully validate the ability of BBPSO in FS problems, a novel BBPSO with adaptive chaotic jump strategy and new global best updating mechanism (BBPSO-ACJ) is proposed. By employing the adaptive chaotic jump strategy, each particle will choose its position updating mechanism adaptively according to its own condition where the stagnated particles have more opportunity to perform a chaotic jump to escape from local attractor and good particles would update its position in the canonical way. Therefore, the novel operator not only prevents particles from falling into local optimal, but also maintains balance between global exploration and local exploitation. In addition, the new global best updating mechanism can effectively reduce the number of selected features.

This paper is organized as follows. Section 2 introduces PSO, BBPSO, and KNN briefly. In section 3, we propose a BBPSO with adaptive chaotic jump strategy for FS problem. Section 4 includes experiments design, results and analyses of the results, respectively. The conclusions are given in Section 5.

2. Background

2.1. Particle swarm optimization

PSO is a swarm intelligence based optimization algorithm which simulates the behavior of bird flying or fish schooling[13]. The whole population is called swarm which concludes a set of particles. Each particle represents a candidate solution of the optimization problem. Let $x_i = (x_{i1}, x_{i2}, ..., x_{iD})$ be the ith particle in the swarm. D denotes the dimension of the search space. The velocity of particle i is $v_i = (v_{i1}, v_{i2}, ..., v_{iD})$ which indicates the speed and direction that the particle should move in the next cycle. The initial positions are randomly generated in the multi-dimensional search space and the initial velocities are generally set to 0. In the iteration process, all the particles are evaluated with a fitness function. The best fitness value of each particle is its own personal best (pbest) and the best fitness value of the whole swarm is recorded as the global best (gbest). Each particle adjusts its speed and direction according to its own flying experience and the experience of other particles in the swarm. The information exchange in the swarm is realized in this way. In the basic PSO algorithm, the velocity and position of each particle are updated by the following equations:

$$v_{id}^{t+1} = w \times v_{id}^t + c_1 \times r_1^t \times (pbest_{id}^t - x_{id}^t) + c_2 \times r_2^t \times (gbest_d^t - x_{id}^t) \quad (1)$$

$$x_{id}^{t+1} = x_{id}^t + v_{id}^{t+1} \quad (2)$$

where v_{id}^t is the dth dimension of the velocity of particle i in cycle t; x_{id}^t the dth dimension of the position of particle i in cycle t; $pbest_{id}^t$ is the dth dimension of the position of personal best of particle i in cycle t; $gbest_d^t$ is the dth dimension of the position of gbest in cycle t; w is the inertia weight which can be used to balance global and local search. The value is typically set between 0 and 1. When a particle converges to a local optimal and moves very slowly, a relatively large inertia weight can help the particle escape from the local attractor. A relatively small inertia weight is more appropriate for performing local search. c_1 is the cognitive weight and c_2 is the social weight; r_1^t and r_2^t are two random numbers.

The original PSO was proposed for optimization problems in continuous space. FS is a discrete optimization problem. In order to extend PSO to FS, two methods have been proposed:

(i) Binary PSO (BPSO) was developed for discrete problems by Kennedy and Eberhart[30]. In BPSO, the position of particle can take value 1 or 0. The velocity denotes the probability of the position taking value 1. In BPSO, the positions and velocities of the particles are initialized by:

$$x_{id}^t = \begin{cases} 1, \text{ if } rand() > 0.5 \\ 0, otherwise \end{cases} \quad (3)$$

$$v_{id}^{t} = -v_{\max} + 2 \times rand() \times v_{\max} \tag{4}$$

where *rand*() is a random number between [0,1]. $v_{\max}$ is the maximum speed of particle which is very important in BPSO. The maximum speed should be set properly in order to prevent premature convergence. If the value is too large, the position would always be 1 and it cannot search more spaces.

The velocities of particles are updated by Eq.(1). The positions of particles are updated by the following equations:

$$x_{id}^{t} = \begin{cases} 1, \text{ if } S(v_{id}^{t}) > rand() \\ 0, otherwise \end{cases} \tag{5}$$

where *rand*() is a random number between [0,1]. *S*() is a sigmoid function which is used to transform v_{id}^{t} to the range of (0,1).

$$S(v_{id}^{t}) = \frac{1}{1+\exp(-v_{id}^{t})} \tag{6}$$

(ii) Canonical PSO: In this method, the positions are restricted between [0,1]. The velocities and positions are updated as the canonical PSO, with Eq.(1) and Eq.(2). Before evaluating the population, a decoding procedure is needed. In Ref. 31, the position of a particle can be decoded into a feature subset. A threshold (e.g. 0.6) is chosen to decide if the feature is chosen. If a dimension of a particle's position is larger than the threshold, the corresponding feature is chosen. Otherwise, the feature does not exist in this feature subset.

2.2. Bare bones Particle Swarm Optimization

The BBPSO is the simplest variant of PSO. It eliminates the velocity term and only the positions of the particles are used in the search process. Therefore, users do not need to consider those control parameters in PSO which would largely affect the performance of PSO. Due to the simplicity and powerful optimization ability of BBPSO, it has aroused a lot of attentions from researchers. When BBPSO updates the positions of particles, all the particles learn from its own flying experience and the flying experience of the best particle of the swarm. The positions in BBPSO are updated according to the following equation:

$$x_{id}^{t+1} = N(\frac{pbest_{id}^{t} + gbest_{d}^{t}}{2}, \left|pbest_{id}^{t} - gbest_{d}^{t}\right|) \tag{7}$$

As is shown in Eq.(7), the positions are randomly generated by the Gaussian distribution with the mean of (pbest+gbest)/2 and the variance of |pbest-gbest|. Kennedy also proposed another version called BBPSO-Exp[23] which promotes local search around the pbest position. The positions are updated by:

$$x_{id}^{t+1} = \begin{cases} N(\frac{pbest_{id}^{t} + gbest_{d}^{t}}{2}, \left|pbest_{id}^{t} - gbest_{d}^{t}\right|), & R<0.5 \\ pbest_{id}^{t}\ , & \text{otherwise} \end{cases} \tag{8}$$

where *R* is a random number between [0,1]. It can be included from Eq.(8) that the particle has 50% chance to jump to its own pbest position in BBPSO-Exp.

2.3. K nearest neighbor algorithm

The *K* nearest neighbor algorithm (KNN) is a non-parametric method which can be used for classification and regression tasks. The original dataset is divided into training set and test set. In classification problems, each training sample is a vector in a multi-dimensional feature space and contains a class label. Each test sample is a vector without a class label. For a new test sample to be classified, it is first assigned *K* closest training samples according to some distance or similarity function. Some frequently-used distance metrics include the Euclidean distance, the Hamming distance and etc. Then the test sample is classified by a majority vote of its *K* closest neighbors. KNN is a simple machine learning algorithm as the only control parameter in KNN is the number of neighbors. The main drawback of KNN is that it cannot perform well in unbalanced datasets due to its "majority vote" mechanism. This mechanism would easily make the test samples which belong to the minority class being wrongly classified. Besides, the local structure of the dataset would also affect the performance of KNN. In order to achieve good classification performance, many variants of KNN have been proposed. Until now, KNN is still widely used in different machine learning tasks.

3. The proposed approach

BBPSO is a popular variant of PSO due to simplicity and efficiency. Its parameter-free nature makes it possible to show good performance in different application scenarios.

The goal of this study is to propose a BBPSO based FS method. Due to the large search space and many local optimal in FS problems, the search ability of BBPSO needs to be enhanced in order to obtain feature subsets with high quality. An adaptive chaotic jump strategy is introduced into BBPSO to prevent the algorithm from falling into local optimal. Moreover, the updating strategy of gbest is modified to encourage the algorithm to generate smaller feature subsets. This section describes the proposed FS algorithm, called BBPSO with adaptive chaotic jump (BBPSO-ACJ).

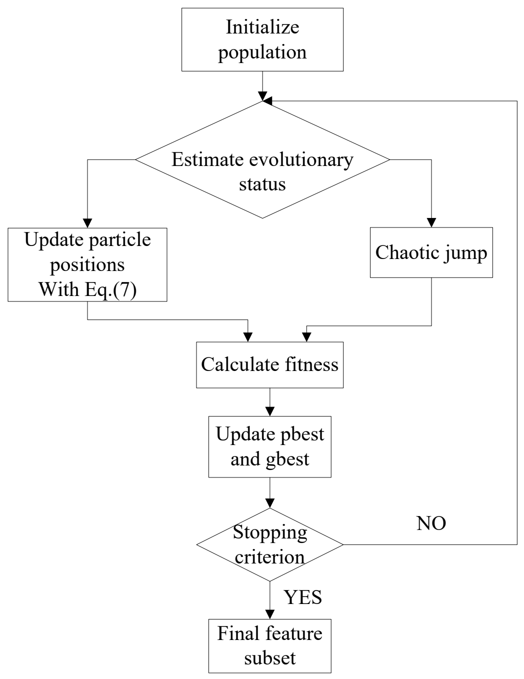

Fig. 1. Flowchart of the proposed algorithm.

Fig. 1 shows the flowchart of the proposed algorithm. First the population is randomly initialized in the search space. Each particle position represents a feature subset and its pbest is the initial value. After initialization, the following steps are repeated until the maximum iteration is reached. The adaptive chaotic jump strategy is used to decide the updating mechanism of each particle. Update the positions with Gaussian sampling or chaotic jump. All the new positions are evaluated with the fitness function. If the new position is out of the search space, the particle will be assigned the value of its corresponding lower or upper bound. Update the pbest and gbest according to the fitness value. When the maximum iteration is reached, the optimal feature subset is reported.

3.1. Feature subset representation

In evolutionary computation (EC) based FS approaches, a chromosome represents a candidate feature subset which is evaluated with some criteria, such as the classification accuracy, the mutual information of the feature subsets and etc. According to the type of evaluation criteria which is employed, the EC based FS approaches can be divided into filter approach and wrapper approach.

In PSO based FS methods, binary encoding is widely used in which position represents a feature subset and velocity denotes the possibility of the feature being selected. In BBPSO, due to the absence of the velocity term, the encoding scheme needs to be modified. Given the position of the *i*th paricle: $x_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$. D denotes the dimension of the problem, *i.e.*, the number of the original features. The encoding of a particle's position is shown in Fig.2. The position is restricted between [0,1].

x_{i1}	x_{i2}	x_{i3}	$\cdots$	$\cdots$	x_{iD}

Fig. 2. The encoding of a particle

In order to form a feature subset, a decoding process is needed before the evaluation. Position can be translated into a feature subset as follows:

$$A_{id} = \begin{cases} 1, & if\, x_{id} > 0.5 \\ 0, & otherwise \end{cases} \tag{9}$$

where A_{id} means the feature subset decoded from the position of particle x_{id}. A_{id} can take the value of 1 or 0 depending on the value of x_{id}. A_{id}=1 means the *d*th feature is chosen. Otherwise, this feature is excluded. Fig. 3 illustrates a 7-dimensional problem which means the complete dataset has 7 features. The decoded particle position in the figure indicates that the 1th, 3rd, and 5th features are selected while the other four features are not chosen in this feature subset.

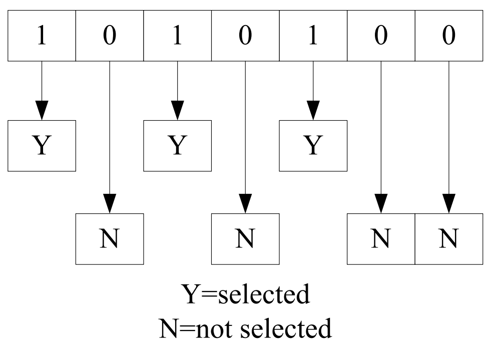

Fig. 3. A 7 dimensional encoding problem

3.2. Feature subset evaluation

In this paper, the wrapper approach is employed to evaluate the feature subsets. Therefore, the classification accuracy is used as the fitness function and it can be defined as:

$$CA = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

where *TP* (True Positive) is the number of positive instances that are correctly classified. *FP* (False Positive) is the number of negative instances that are wrongly classified as positive. *TN* (True Negative) is the number of negative instances that are correctly classified. *FN* (False Negative) is the number of positive instances that are wrongly classified as negative. The larger *CA* is, the more accurate the feature subset is. The aim of the FS method is to find the feature subset that can maximize the *CA* metric.

3.3. Adaptive chaotic jump strategy

BBPSO is known for its fast convergence speed which may leads to the rapid loss of population diversity and increases the probability of falling into local optimal. In high-dimensional FS problems, the search space grows exponentially with the number of features and there are many local optimal in the large search space. BBPSO is prone to premature convergence in this situation. However, if the algorithm focuses on promoting the population diversity, the convergence speed would be decreased. To keep the balance between convergence speed and population diversity during the optimization process is very crucial for BBPSO.

In order to improve the diversity of population while maintaining the high convergence speed, a novel adaptive chaotic jump strategy (ACJ) is proposed. It allows each particle to choose its updating mechanism according to its own accumulated experience. In this way, good particles update their positions in normal way, *i.e.* by the Gaussian sampling, while bad particles can make a large modification of their search previous trajectory, *i.e.* the chaotic jump. This strategy can promote population diversity without sacrificing the convergence speed of the algorithm. This section will first introduce the chaotic jump operator, and then the adaptive strategy will be described.

3.3.1. Chaotic jump operator

Chaos is a non-linear system which shows a great sensitivity to initial conditions[33]. A small variation of the initial parameter will lead to large differences in its long-term behavior. Therefore, it is impossible to predict the long term behavior of the chaotic system. Although chaos is random and unpredictable, it also shows some kind of regularity[34]. Due to the unpredictable characteristic of chaotic system, it can reach any possible region in the whole search space which make it possesses the special merit of escaping from local optimal[35]. Moreover, it is very easy and convenient to generate and store a chaotic sequence. Therefore, researchers have paid much attention to chaotic system to make use of its special advantages. It has been introduce to evolutionary algorithms to improve their global search ability[36,37,38]. Recently, chaotic systems have been embedded into PSO to promote population diversity and enhance the optimization performance. These approaches include chaotic control parameters, chaotic local search and etc[39].

In this paper, chaotic sequences are employed to BBPSO for solving FS problems. A chaotic jump strategy is proposed to enrich the searching behavior of BBPSO and strengthen the ability of jumping out of local attractor. The logistic map is polynomial map and it can be obtained from very simple non-linear dynamical equations. Due to its simplicity, the logistic map is the most frequently used chaotic sequence and it is defined by:

$$z_{i+1} = 4z_i(1 - z_i), \quad z_i \in (0,1). \tag{11}$$

where z_i is the chaotic variable and i denotes the iteration number. Fig. 4 shows the chaotic dynamics of the logistic map, where $z_1 = 0.13$ and the maximum of iteration is set as 150.

For the *i*th particle to perform a chaotic local jump, the position is updated using the following equation:

$$x_{id}^t = pbest_{id}^t (1.0 + (2z_k - 1.0)) \tag{12}$$

where z_k is the chaotic variable generated with Eq.(11). For the stagnated particles, the chaotic local search can offer a wider search capability due to the non-periodicity of the logistic map. It can improve the performance of BBPSO in terms of preventing premature convergence.

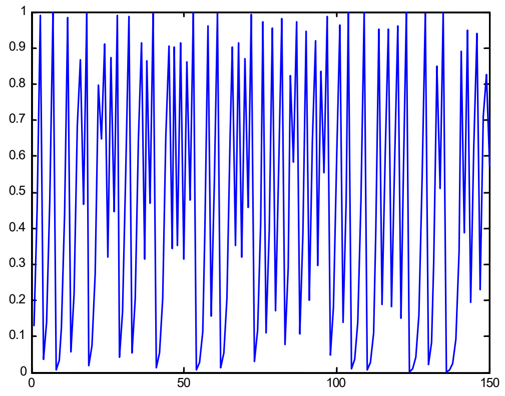

Fig. 4. Dynamics of logistic map

3.3.2. *Adaptive strategy*

Based on the chaotic jump strategy proposed in the above section, this section will introduce the adaptive strategy. The proposed approach can decide when to perform the chaotic jump adaptively according to the evolutionary status of each particle. An array named *stagnation* is introduced to monitor the evolutionary status of each particle. For particle *i*, *stagnation*[*i*] denotes how many iterations that particle *i* does not get fitness improvement.

If there is no fitness improvement for particle *i* in one iteration, then *stagnation*[*i*] is increased by one. A large *stagnation*[*i*] concludes that particle *i* is stagnated and the particle should perform the chaotic jump which may help the particle jump out of the local attractor. The adaptive chaotic jump strategy (ACJ) is described in Algorithm 1 in detail. From the procedure of ACJ, we can see that the probability of particle *i* to perform the chaotic jump is computed as follows:

$$p_{cj,i} = \frac{1}{1+\mathrm{e}^{2.0-stagnation[i]}} \tag{13}$$

where $p_{cj,i}$ is the probability of particle *i* performing the chaotic jump. The probability increases with the number of iterations that a particle does not get fitness improvement. Fig. 5 shows the change of the jump probability with respect to the iterations without fitness improvement. As is shown in Fig. 5, if a particle does not update its pbest in 3 iterations, the probability of chaotic jump is larger than 0.7.

```
Algorithm 1. Adaptive Chaotic Jump Strategy
For each particle $i$
  $stagnation[i] = 0$
End For
DO
For each particle $i$
IF $\frac{1}{1+\mathrm{e}^{2.0-stagnation[i]}} < rand()$
THEN Update the position of $x_i$ according to Eq.(7)
ELSE
Update the position of $x_i$ according to Eq.(12)
 $stagnation[i] = 0$
END IF
IF $f(x_i) > f(pbest_i)$
THEN Update $pbest_i$
 $stagnation[i] = 0$
ELSE
 $stagnation[i] = stagnation[i] + 1$
END IF
WHILE termination condition not met.
Output: $gbest$
```

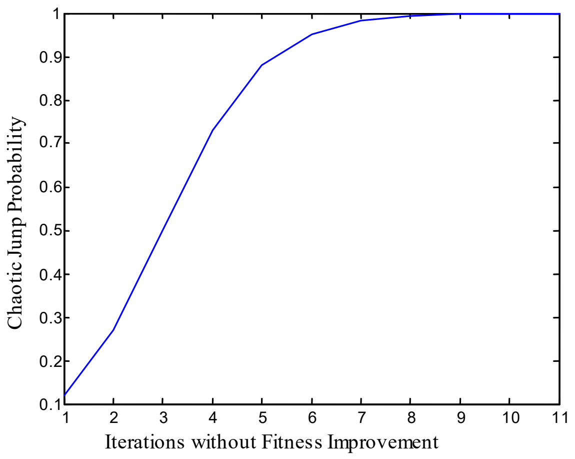

Fig. 5. An illustration of jump probability

3.4. Update of gbest

In BBPSO, gbest plays a crucial role in guiding the whole population searching for better solutions. The FS algorithm aims to select a reduced set of features from the original features without losing informative information. In the evolutionary process, the algorithm evaluates particles according to classification accuracy and the feature subset with the highest classification accuracy is defined the gbest. However, the case that different feature subsets may have the same classification performance would always appear in FS problem. For example, subset {1, 2, 4, 7, 9} and subset {2, 5, 7} achieve the same classification accuracy. In this situation, the latter feature subset which achieves the same classification accuracy with fewer features is a better choice. Therefore, the gbest updating mechanism first considers the classification accuracy. When two feature subsets tie in terms of classification performance, the feature subset with fewer features is chosen as the new gbest. The equation shows our proposed gbest updating strategy:

$$gbest^{t+1} = \begin{cases} x_i^{t+1} & if\ f(x_i^{t+1}) > f(gbest^t) \\ x_i^{t+1} & if\ f(x_i^{t+1}) = f(gbest^t)\ \&\ Z(x_i^{t+1}) < Z(gbest^t) \\ gbest^t & otherwise \end{cases} \quad (14)$$

where $Z()$ means the number of the features. The new gbest updating mechanism can encourage the population to search for feature subsets with higher classification and fewer features.

4. Experimental results

4.1. Dataset

In order to test the effectiveness of the proposed method, nine datasets taken from the UCI machine learning repository are used for experiments and the general cases of these datasets are shown in Table 1. These datasets show a large diversity over number of features, classes, and instances and they are widely used in research areas such as feature selection and classification.

Table 1. Datasets

Dataset	# Features	# Instances	# Classes
Glass	9	214	2
Wine	13	178	3
Heart	13	270	2
Australia	16	690	2
Segment	19	2310	7
Germany	24	1000	2
Ionosphere	34	351	2
Sonar	60	208	2
Musk1	166	476	2

For each dataset, all the instances are randomly divided into two parts: 70% as the training set and 30% as the test set. All the datasets are normalized to [0,1] before the FS process. An individual of an EC based algorithm represents a candidate feature subset. During the training process, KNN is employed as the learning algorithm to evaluate the individuals. In this paper, K is set as 5. The fitness value (classification accuracy) of each individual is calculated through a 10-fold cross validation on the training set. The whole training set is randomly divided into 10 folds. 9 folds are used as the sub-training data and the remaining 1 fold is used as the sub-test data. The process is repeated 10 times and the average value is the fitness value of the individual. The 10-fold cross validation is run on the training set, so it is independent from the test set. After the training process, the best feature subset is recorded as the final result and it is testified on the test set with 5NN.

4.2. Comparative algorithms

To verify the performance of the proposed BBPSO-ACJ, the following 8 EC based wrappers are employed: Genetic algorithm (GA)[40], PSO[31], Binary PSO (BPSO)[30],Binary PSO with chaotic inertia weight (BPSO-CI)[35], BBPSO[23], Quantum inspired PSO (QBPSO)[41], Binary PSO with catfish effect (BPSO-CE) [15], PSO (4-2)[18].

Furthermore, two filter based methods, linear forward selection (LFS) and greedy stepwise based selection (GSBS), are also employed for comparison. LFS starts from an empty feature subset and selects features into the subset step by step while GSBS gradually eliminates features from the original entire feature subset[18]. LFS and GSBS are run on WEKA with the default settings[42].

4.3. Parameters setting

The experiments are performed on a machine with Intel(R) Core(TM) i5-6500 at 3.2 GHz and 8.00 GB of RAM using MATLAB and the operating system is MS Windows 10.

The maximum number of iterations is empirically set to 50 and the population size for EC based FS method is set to 20. The crossover and mutation rates of GA are set to 0.8 and 0.2, respectively. For PSO, BPSO, BPSO-CI, BPSO-CE, and PSO(4-2) the cognitive weight c_1 and social weight c_2 are both set to 2. The upper and lower bounds of velocity for binary PSO based methods are all set to 6 and -6, respectively. The time decreasing inertia weight is used with $w_{max} = 0.9$ and $w_{min} = 0.4$. The parameter β in QBPSO which is used to control the convergence speed is set as 0.55. For each dataset, all the EC based FS methods are repeated 20 independent times to avoid the impact of random factors.

4.4. Results

4.4.1. Comparison between BBPSO-ACJ with EC based FS methods

The performance of the proposed BBPSO-ACJ is compared with 8 other EC based FS methods. For each dataset, the obtained feature subsets by the 9 algorithms in the training set are testified with 5NN in the test set. Table 2 shows the means and standard deviations of classification accuracy in the test set in 20 independent runs. The best mean classification accuracy in each dataset is shown in **boldface**. Moreover, the classification accuracy of 5NN on the original entire features is also shown in Table 2(*i.e.* Without FS).

From Table 2, we can find that in most cases, the EC based FS methods can achieve higher classification accuracy than using the original entire feature set. Taking the Ionosphere dataset for example, while 5NN is able to obtain the 79.25% accuracy using all the 34 features, GA based FS method obtains 83.62% accuracy and other EC based algorithms achieve even higher accuracy than GA. Hence, it can be inferred that FS is an effective and crucial data prepossessing step for classification problem as it is able to improve the classification performance with fewer features.

Table 2 Classification accuracy of different algorithm. The average of results over 20 independent runs is reported.

Dataset	Without FS	GA	PSO	BPSO	BPSO-CI	BBPSO	QBPSO	PSO(4-2)	BPSO-CE	BBPSO-ACJ
Glass	63.08	71.9+8.86	71.08+7.84	76.92+4.87	75.38+6.49	75.05+5.81	72.31+7.94	73.21+6	74.07+7.51	**77.18+4.44**
Wine	94.44	96.29+2.33	96.67+0.78	97.04+1.66	96.3+0.53	97.04+1.29	97.04+1.29	95.26+1.75	96.67+0.78	**97.96+1.37**
Heart	81.48	82.46+6.73	83.44+2.53	82.96+2.03	81.98+3.09	83.21+4.12	82.1+3.41	79.78+4.13	84.44+2.68	**87.45+2.09**
Australia	83.62	83.59+2.22	83.03+1.52	84.23+0.22	84.04+0.87	84+1.06	83.17+1.63	83.34+3.65	83.86+0.96	**84.38+0.41**
Segment	90.05	83.11+5.08	91.3+0.72	91.58+0.48	91.61+0.46	91.52+0.44	91.64+0.48	85.42+4.81	91.7+0.32	**92.07+0.13**
German	68	70.94+1.84	71.07+1.79	72.07+2.88	72.6+2.03	73.19+2.9	73+3.6	68.47+1.45	72.29+1.79	**75.39+2.15**
Ionospher	79.25	83.62+3.12	84.25+2.67	84.91+2.038	84.81+2.2	84.72+2.26	87.45+2.63	86.89+1.64	83.68+2.09	**87.55+0.74**
Sonar	73.02	73.28+5.36	77.25+3.27	72.06+3.82	76.03+2.3	76.67+3	76.51+3.16	77.94+3.21	77.1+4.19	**79.05+2.97**
Musk1	80.42	82.81+2.36	83.36+2.67	83.92+2.08	84.13+3.08	84.42+2.13	84.13+3.26	84.87+2.7	84.92+1.61	**87.2+1.28**
Average	79.8	80.89+4.21	82.38+2.64	82.85+2.23	82.99+2.34	83.31+2.56	83.04+3.04	81.69+3.26	83.19+2.44	**85.36+1.73**

Table 3 Selected feature number of different algorithms. The average of results over 20 independent runs is reported.

Dataset	All	GA	PSO	BPSO	BPSO-CI	BBPSO	QBPSO	PSO(4-2)	BPSO-CE	BBPSO-ACJ
Glass	9	4.5	3.8	4.6	4.3	4.1429	**3.8**	4.1	4	4
Wine	13	**6.35**	7.4	8.6	8	7.8	7.7	6.84	7.4	8.2
Heart	13	6.8	6.8	8	7.4	7.3	6.7	6.6	7.3	**6.2**
Australia	16	5.95	5.9	6.88	6.4	6.14	5.3	5.79	6.29	**5.4**
Segment	19	10.4	10.9	10.6	11.2	11.5	10.9	10.78	10.4	**9.9**
German	24	10.7	12.4	10.67	10.6	11.14	10.43	12.76	10.29	**8.83**
Ionosphere	34	10.93	13.2	10.7	10.9	9.4	4.4	**3.13**	10.6	6.3
Sonar	60	30.8	28.07	29.6	31.4	29.8	28.4	**11.24**	30.43	25.7
Musk1	166	81.83	82.2	82.14	81.3	80.43	**49.8**	77.3	85.29	72.5
Average	39.33	18.7	18.96	19.09	19.06	18.63	**14.16**	15.39	19.11	16.34

It can be seen from Table 2 that BBPSO-ACJ achieves the highest classification accuracy in all datasets compared with 8 other EC based wrappers. For instance, BBPSO-ACJ gets the mean classification accuracy 87.2% in Musk1 dataset, while the second best is 84.92% which is obtained by BPSO-CE. In terms of average result in all datasets, BBPSO-ACJ obtains the highest average value 85.36%, while BBPSO (83.31%) and PSOCE (83.19%) place second and third, respectively.

In addition, Table 2 also shows that BBPSO-ACJ obtains the lowest standard deviation. The average standard deviation is 1.73 for BBPSO-ACJ while the second best (*i.e.* BPSO) is 2.23. It can be concluded that BBPSO-ACJ shows the best robustness among all the EC based FS algorithms.

Table 3 shows the number of original entire feature set and the selected feature number of different FS methods. For each dataset, the algorithm which produces the smallest feature subset is shown in **boldface**. In terms of selected feature number, QBPSO obtains the optimal performance with the average feature subset size 14.16. Especially for the Musk1 dataset, QPSO only selects 49.8 features out of the original 166 features while the second best is 72.5(i.e. BBPSO-ACJ). PSO(4-2) places second with the average feature number 15.39. In Sonar dataset, it selects 11.24 features from the entire 60 features with rather good classification accuracy (rank 2 in all 9 EC based FS algorithms). BBPSO-ACJ ranks 3 in all the 9 algorithms, with the average feature number 16.34. It selects the smallest feature subsets in four datasets: Heart, Australia, Segment, and German. But it does not perform as well as QPSO in high dimensional datasets (*i.e.* Sonar) which affects its average performance. In general, it can be concluded that BBPSO-ACJ is superior to other methods in terms of classification accuracy and it also shows competitive performance in terms of selected feature number.

4.4.2. Comparisons with filter based methods

In this section, BBPSO-ACJ is compared with two filter based FS methods, LFS and GSBS. Tables 4 shows the classification accuracy (CA) and number of selected features (#features) of the three FS methods.

Table 4 Comparison between BBPSO-ACJ and two filter based methods

Dataset	Metric	LFS	GSBS	BBPSO-ACJ
Glass	CA	70.31	66.53	77.18
	#features	6	7	4
Wine	CA	74.07	85.19	97.96
	#features	7	8	8.2
Heart	CA	76.53	67.48	87.45
	#features	6	8	6.2
Australia	CA	70.05	69.57	84.38
	#features	4	12	5.4
Segment	CA	88.24	84.67	92.07
	#features	4	13	9.9
German	MA	68.67	64.33	75.39
	#features	3	18	8.83
Ionosphere	MA	86.67	78.1	87.55
	#features	4	30	6.3
Sonar	MA	77.78	68.25	79.05
	#features	3	48	25.7
Muskl	MA	85.31	76.22	87.2
	#features	10	122	72.5

It can be seen from Table 4 that LFS selects fewer features than GSBS in all the 9 datasets and achieves higher classification accuracy in 8 out of 9 datasets. BBPSO-ACJ achieves significantly higher classification accuracy than LFS in all the datasets. BBPSO-ACJ selects slightly larger or even smaller feature subsets than LFS in datasets with small number of features, such as Glass, Wine, and Heart. LFS shows much better performance in high-dimensional datasets than BBPSO-ACJ. This can be attributed to the forward selecting mechanism of LFS which make it tend to select very small number of features. Compared with GSBS, BBPSO-ACJ obtains much higher classification performance in all datasets and selects smaller feature subsets in 8 out of 9 datasets. Therefore, it can be concluded from Table 4 that BBPSO can effectively reduce the number of features and shows superior performance to the two deterministic FS methods in terms of classification accuracy.

4.4.3. Statistical analysis of BBPSO-ACJ

Statistics provides a powerful tool to investigate the difference between different algorithms. In this study, the Friedman test and the multiple comparison approach are employed to testify whether the difference of the 9 EC based FS methods in terms of classification accuracy is significant[20,28].

The Friedman test is a non-parametric method which is used to compare the classification performance of different classifiers over multiple datasets by ranking each algorithm on each dataset[43]. In this paper, the hypothesis is that all the 9 EC based FS methods have equal classification performance and the Friedman test is used to test the hypothesis.

When performing the Friedman test, first create a ranking matrix R in which each element R_{ij} means the rank (from 1 to 9) of the FS method j on dataset i. If an FS method obtains the highest classification accuracy, it gets rank 1 and the second best gets rank 2, and so on. The test statistic is computed by the following equations:

$$T_f = \frac{(n-1)\{B_f - \frac{nk(k+1)^2}{4}\}}{A_f - B_f} \tag{15}$$

$$R_j = \sum_{i=1}^{n} R_{ij}^2 \tag{16}$$

$$A_f = \sum_{i=1}^{n}\sum_{j=1}^{k} R_{ij}^2 \tag{17}$$

$$B_f = \frac{1}{n}\sum_{j=1}^{k} R_j^2 \tag{18}$$

The null hypothesis is rejected at the α significance level if the test statistic is greater the $1-\alpha$ quantile of the F-distribution with $k-1$ and $(k-1)(n-1)$ degrees of freedom. In this study, T_f is 7.56 and $F_{0.05}(8,64) = 2.09$. Since T_f is larger than $F_{0.05}(8,64)$, the null hypothesis that all 9 EC based FS methods have the same classification performance is rejected at 0.95 significance level.

The multiple comparison approach is used to confirm which method shows significantly better classification performance when the Friedman test is rejected. The following inequality is used to decide whether two methods i and j are significantly different.

$$\left|R_j - R_i\right| > t(\alpha/2)\sqrt{2n(A_f - B_f)/(n-1)(k-1)} \tag{19}$$

where $t(\alpha/2)$ is a value on the t-table using $(n-1)(k-1)$ degrees of freedom $(\alpha/2 = P(t > t(\alpha/2)))$.

Like the Friedman test, the 9 EC based FS methods are ranked according to their classification performance. The rank sums of GA, PSO, BPSO, BPSO-CI, BBPSO, QBPSO, BPSO-CE, PSO (4-2), and BBPSO-ACJ are 70, 59, 44, 48, 37, 44, 56, 38, and 9, respectively. In the multiple comparison approach, the classification performances of two methods are considered significantly different when the sum ranks of the two methods are greater than a stated unit apart. In this study, the stated unit at 0.05 significance level is 17.66. Obviously, BBPSO-ACJ obtains significantly better classification accuracy than other comparative algorithms.

4.4.4. Analysis on computational time

This section compares the computational time of the 9 EC based FS methods. Table 5 shows the mean computational time (in seconds) of 20 independent runs on each dataset. For each dataset, the algorithm with the best computational efficiency is shown in **boldface**. They are all wrapper-based methods which employ the 5NN classifier to evaluate the individuals in the optimization process. Hence, most of the computational time is spent on the evaluation process. First of all, it can be seen from Table 5 that the 9 methods are close in the running time. All the methods can generate feature subsets in relatively short time, less than one minute in 8 out of 9 datasets and about 2 minutes in the German dataset. Among all the 9 algorithms, PSO shows the best computational efficiency, with the average CPU time 26.67 seconds. This is due to the fact that PSO does not include any extra search strategies. The difference of the average CPU time between PSO and BBPSO-ACJ is less than 2 seconds. Moreover, the proposed BBPSO-ACJ shows superior performance to PSO in terms of classification accuracy and selected feature number according to Table 2 and Table 3. Therefore, there is a trade-off between the quality of feature subsets and the computational cost.

Table 5 The average time of algorithms on each dataset(in seconds). The last row of the table shows the average CPU time of each method over the datasets.

Dataset	GA	PSO	BPSO	BPSO-CI	BBPSO	QBPSO	PSO(4-2)	BPSO-CE	BBPSO-ACJ
Glass	6.11	**5.48**	6.63	7.13	6.19	5.93	5.27	7.16	5.55
Wine	8.75	8.01	**7.41**	7.44	7.69	6.94	8.55	7.93	7.55
Heart	15.23	13.81	14.30	14.12	15.64	**12.51**	13.90	14.94	13.33
Australia	31.18	**29.96**	31.88	33.02	32.30	35.17	33.46	35.74	38.47
Segment	16.23	15.64	17.89	15.32	13.80	**12.77**	14.69	16.01	17.61
German	105.98	**101.04**	103.95	105.87	109.65	104.04	105.93	111.99	106.10
Ionosphere	15.56	14.02	13.71	**11.94**	15.70	13.66	13.94	12.79	16.53
Sonar	7.07	**6.13**	6.30	7.07	6.74	8.19	7.66	6.98	7.02
Musk1	52.20	45.99	46.10	48.22	47.95	43.75	47.83	51.42	**43.34**
Average	28.70	**26.67**	27.57	27.79	28.41	27.00	27.91	29.44	28.39

4.4.5. Analysis of the two new operators

In this section, the effect of the two new operators (the adaptive chaotic jump strategy and the new gbest updating mechanism) upon FS will be discussed in detail. Table 2 shows that BBPSO-ACJ obtains obviously better classification accuracy than the original BBPSO. Take the Sonar dataset for example, the classification accuracies for BBPSO-ACJ and BBPSO are 87.2 and 84.42, respectively. Moreover, BBPSO-ACJ shows better robustness than BBPSO which can be illustrated by the lower standard deviation of BBPSO-ACJ. Table 3 indicates that BBPSO-ACJ obtains smaller feature subsets than BBPSO in 8 out 9 datasets. The average selected number of features of BBPSO is 18.63 while the average feature number of BBPSO-ACJ is 16.34. Table 5 shows that BBPSO-ACJ costs even slightly shorter CPU time than BBPSO. Consequently, it can be inferred from the abovementioned results that the adaptive chaotic jump strategy can help BBPSO to obtain more discriminative feature subsets with higher classification performance and the new gbest updating mechanism can help the algorithm reduce the number of selected features. Furthermore, the two operators would not bring additional computational burden for BBPSO which is demonstrated by the computational time.

5. Conclusions

In this paper, a novel feature selection method called BBPSO-ACJ is proposed to testify the potential of BBPSO in FS problems. BBPSO is the simplest variant of PSO and it also shows good global search ability. But it also suffers from premature convergence, especially in high dimensional optimization problems. In order to improve its performance in FS problem, an adaptive chaotic jump strategy is employed to improve local exploitation in order to help the stagnated particles to jump out of local attractors and keep the balance between convergence speed population diversity. A new gbest updating mechanism is proposed to reduce the number of selected features. To validate the performance of the proposed algorithm, it is compared with 8 EC based FS methods and two filter based methods on nine datasets from UCI machine learning repository. The experimental results show the proposed method can effectively eliminate irrelevant or redundant features, and it achieves the highest classification accuracy in all the datasets compared with other EC based wrappers. The comparison with two deterministic filter methods indicates that BBPSO-ACJ outperforms LFS and GSBS in terms of classification accuracy and it achieves smaller feature subsets than GSBS. Two statistical tests are employed to demonstrate that BBPSO-ACJ obtains significantly better classification performance than other EC based FS methods. The CPU time analysis shows that the BBPSO-ACJ can obtain feature subsets in a relatively short time. Moreover, the detailed analysis on the two new operators shows that they can effectively improve the performance of BBPSO in FS problems in terms of classification accuracy and number of selected features. Besides, these two operators would not bring any additional computational burden which can be proved by the computation time.

Future work will focus on introducing filter methods into BBPSO based FS model to improve the computational efficiency. Another research area is multi-objective FS method which aims at improving multiple objectives simultaneously, such as classification accuracy, number of selected features, relevance, and redundancy between features and class label.

Acknowledgements

This work was supported by the NUPTSF under Grant No.NY214186 and the Natural Science Foundation of Jiangsu Province under Grant No.BK20160898.

References

1. H. Liu, L. Yu, Toward integrating feature selection algorithms for classification and clustering, *IEEE Trans. Know. Data Eng*. 17 (2005) 491–502.
2. M. Dash and H. Liu, “Feature selection for classification,” *Intelligent Data Analysis*, 1(4), (1997) 131–156.
3. W. Lipo, Z. Nina, C. Feng, A general wrapper approach to selection of classdependent features, *Neural Networks, IEEE Trans.*, 19 (2008) 1267–1278.
4. A. Ghosh, A. Datta, S. Ghosh, Self-adaptive differential evolution for feature selection in hyperspectral image data, *Appl. Soft Comput*. 13 (2013), 1969–1977.
5. C.T. Su, H.C. Lin, Applying electromagnetism-like mechanism for feature selection, *Inf. Sci.* 181 (5) (2011) 972–986.
6. S. Ding, Feature selection based F-score and ACO algorithm in support vector machine, in *PROC. the 2nd International Symposium on Knowledge Acquisition and Modeling*, (Wuhan, China, 2009), pp.19-23.
7. L.T. Vinh, S. Lee, Y.-T. Park, B.J. d'Auriol, A novel feature selection method based on normalized mutual information, *Appl. Intell*. 37 (2010) 100–120.
8. M.H. Aghdam, N. Ghasem-Aghaee, M.E. Basiri, Text feature selection using ant colony optimization, E*xpert Syst. Appl.* 36 (2009) 6843–6853.
9. M. Kabir, Md. Shahjahan, K. Murase, A new local search based hybrid genetic algorithm for feature selection, *Neurocomputing* 74 (2011) 2914–2928.
10. R. Kohavi and G. H. John, Wrappers for feature subset selection, *Artificial Intelligence* 97 (1997) 273–324.
11. I.A. Gheyas, L.S. Smith, Feature subset selection in large dimensionality domains, *Pattern Recogn*. 43 (2010) 5–13.
12. H. Liu, H. Motoda, Computational Methods of Feature Selection, Chapman & Hall/CRC, 2007.
13. J. Kennedy, R.C. Eberhart, Particle swarm optimization, in *Proc. IEEE International Conference on Neural Networks,* 1995, pp. 1942–1948.
14. X. Wang, J. Yang, X. Teng, W. Xia, R. Jensen, Feature selection based on rough sets and particle swarm optimization, *Pattern Recogn. Lett*. 28 (2007) 459–471.
15. L.-Y. Chuang, S.-W. Tsai, C.-H. Yang, Improved binary particle swarm optimization using catfish effect for feature selection, *Expert Syst. Appl.* 38 (2011) 12699–12707.
16. L.Y. Chuang, C.H. Yang, J.C. Li, Chaotic maps based on binary particle swarm optimization for feature selection, *Appl. Soft Comput*. 11 (1) (2011) 239–248.
17. J. Jiang, Y. Bo, C. Song, L. Bao, Hybrid algorithm based on particle swarm optimization and artificial fish swarm algorithm, in *Proc. Advances in Neural Networks–ISNN 2012,* (Springer Berlin Heidelberg), (2012),pp. 607–614.
18. B. Xue, M. Zhang, W.N. Browne, Particle swarm optimisation for feature selection in classification: novel initialisation and updating mechanisms, *Appl. Soft Comput.* 18 (2014) 261–276.
19. Butler-Yeoman, T., Xue, B., Zhang, M.: Particle swarm optimisation for feature selection: A hybrid filter-wrapper approach, in *Proc. Evolutionary Computation (CEC), 2015 IEEE Congress on, IEEE (2015)*, pp. 2428–2435.
20. Parham Moradi, Mozhgan Gholampour. A hybrid particle swarm optimization for feature subset selection by integrating a novel local search strategy. *Applied Soft Computing* 43 (2016) 117–130.
21. M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, *IEEE Trans. Evol. Comput.* 6 (1) (2002) 58–72.
22. F. Van den Bergh, A. Engelbrecht, A study of particle swarm optimization particle trajectories, *Inf. Sci.* 176 (8) (2006) 937–971.
23. J. Kennedy, Bare bones particle swarms, in *Proc. 2003 IEEE Swarm Intelligence Symposium*, 2003, pp. 80–87.
24. R. A. Krohling, and E. Mendel, Bare Bones Particle Swarm Optimization with Gaussian or Cauchy Jumps, in *Proc. IEEE Congr. Evol. Comput.*, (Trondheim, Norway, 2009), pp. 3285-3291.
25. Hao Liu , Guiyan Ding , Bing Wang. Bare-bones particle swarm optimization with disruption operator. *Applied Mathematics and Computation* 238 (2014) 106–122.
26. Joon Woo Lee, Taeyong Choi, Hyunmin Do, Dongil Park, Chanhun Park, and Young-Su Son. Experimental results of heterogeneous cooperative Bare Bones Particle Swarm Optimization with Gaussian jump for large scale global optimization. in *Proc. 2015 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1979 – 1985.
27. T. Blackwell, A study of collapse in bare bones particle swarm optimization, *IEEE Trans. Evol. Comput.* 16 (3) (2012) 354–372.
28. Yong Zhang, Dunwei Gong, Ying Hu, Wanqiu Zhang. Feature selection algorithm based on bare bones particle swarm optimization, *Neurocomputing* 148 (2015) 150–157.
29. Ce Li, Haidong Hu, Hao Gao, Baoyun Wang. Adaptive Bare Bones Particle Swarm Optimization for Feature

Selection. in *Proc. 2016 Chinese Control and Decision Conference (CCDC)*.(Yinchuan, China), pp. 1594-1599.

30. J. Kennedy, R.C. Eberhart, A discrete binary version of the particle swarm algorithm, in *Proc.1997 Conference Systems Man and Cybernetics*, 1997, pp. 4104–4108.
31. Binh Tran, Mengjie Zhang, and Bing Xue. A PSO based hybrid feature selection algorithm for high-dimensional classification. in *Proc. 2016 IEEE Congress on Evolutionary Computation (CEC)*, pp.3801-3808.
32. M.G.H. Omran, A.P. Engelbrecht, A. Salman, Bare bones differential evolution, *Eur J Oper Res* 196 (2009) 128–139.
33. Alatas B, Akin E, Ozer A. Chaos embedded particle swarm optimization algorithms. *Chaos Solitons Fractals 40* (2009) 1715–1734.
34. Yang DX, Yang PX, Zhang CG. Chaotic characteristic analysis of strong earthquake ground motions, *Int J Bifurcation Chaos* 22(3) (2012) 125-145.
35. Li-Yeh Chuang, Cheng-Hong Yang, Jung-Chike Li. Chaotic maps based on binary particle swarm optimization for feature selection, *Applied Soft Computing* 11 (2011) 239–248.
36. Liu B, Wang L, Jin YH, Tang F, Huang DX. Improved particle swarm optimization combined with chaos, *Chaos Solitons Fractals* 25(5) (2005) 1261–71.
37. Talatahari S, Azar BF, Sheikholeslami R, Gandomi AH. Imperialist competitive algorithm combined with chaos for global optimization. *Commun Nonlinear Sci Numer Simul*, 17(3), (2012) 1312–9.
38. Alatas B. Uniform big bang-chaotic big crunch optimization. *Commun Nonlinear Sci Numer Simul* 16(9), (2011) 3696–703.
39. K. Tatsumi, T. Ibuki and K. Tatsumi, A chaotic particle swarm optimization exploiting a virtual quartic objective function based on the personal and global best solutions, *Applied Mathematics and Computation*, 2013(219): 8991–9011.
40. J.H. Holland, Genetic algorithms, *Scholarpedia* 7 (12) (2012) 1482.
41. J. Yun-Won, P. Jong-Bae, J. Se-Hwan, K.Y. Lee, A new quantum-inspired binary pso: application to unit commitment problems for power systems, IEEE Trans. Power Syst. 25 (3) (2010) 486–1495.
42. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The weka data mining software: an update, *SIGKDD Explor. Newsl.* 11 (1), (2009), 10–18.
43. M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, *Ann. Math. Stat.* 11 (1940) 86–92.