Arabian Journal for Science and Engineering (2020) 45:10875–10890
https://doi.org/10.1007/s13369-020-04871-2

RESEARCH ARTICLE-COMPUTER ENGINEERING AND COMPUTER SCIENCE

Binary JAYA Algorithm with Adaptive Mutation for Feature Selection

Mohammed A. Awadallah[1] · Mohammed Azmi Al-Betar[2,3] · Abdelaziz I. Hammouri[4] · Osama Ahmad Alomari[5]

Received: 17 February 2020 / Accepted: 13 August 2020 / Published online: 1 September 2020

Abstract

In this paper, a new metaheuristic algorithm called JAYA algorithm has been adapted for feature selection. Feature selection is a typical problem in machine learning and data mining domain concerned with determining the subset of high discriminative features from the irrelevant, noisy, redundant, and high-dimensional features. JAYA algorithm is initially proposed for continuous optimization. Due to the binary nature of the feature selection problem, the JAYA algorithm is adjusted using sinusoidal (i.e., S-shape) transfer function. Furthermore, the mutation operator controlled by adaptive mutation rate (R_m) parameter is also utilized to control the diversity during the search. The proposed binary JAYA algorithm with adaptive mutation is called BJAM algorithm. The performance of BJAM algorithm is tested using 22 real-world benchmark datasets, which vary in terms of the number of features and the number of instances. Four measures are used for performance analysis: classification accuracy, number of features, fitness values, and computational times. Initially, a comparison between binary JAYA (BJA) algorithm and the proposed BJAM algorithm is conducted to show the effect of the mutation operator in the convergence behavior. After that, the results produced by the BJAM algorithm are compared against those yielded by ten state-of-the-art methods. Surprisingly, the proposed BJAM algorithm is able to excel other comparative methods in 7 out of 22 datasets in terms of classification accuracy. This can lead to the conclusion that the proposed BJAM algorithm is an efficient algorithm for the problems belonging to the feature selection domain and is pregnant with fruitful results.

Keywords JAYA algorithm · Feature selection · Machine learning · Metaheuristic · Optimization

✉ Mohammed A. Awadallah
ma.awadallah@alaqsa.edu.ps

Mohammed Azmi Al-Betar
mohbetar@bau.edu.jo

Abdelaziz I. Hammouri
aziz@bau.edu.jo

Osama Ahmad Alomari
oalomari@gelisim.edu.tr

1 Department of Computer Science, Al-Aqsa University, P.O. Box 4051, Gaza, Palestine

2 Department of Information Technology - MSAI, College of Engineering and Information Technology, Ajman University, Ajman, UAE

3 Department of Information Technology, Al-Huson University College, Al-Balqa Applied University, P.O. Box 50, Al-Huson, Irbid, Jordan

4 Department of Computer Information Systems, Al-Balqa Applied University, 19117 Al-Salt, Jordan

5 Department of Computer Engineering, Faculty of Engineering and Architecture, Istanbul Gelisim University, Istanbul, Turkey

List of Symbols

Abbreviations

AV	Average of the results
BBA	Binary bat algorithm
BGOA	Binary grasshopper optimization algorithm
BGSA	Binary gravitational search algorithm
BGWO	Binary grey wolf optimizer
BJA	Binary JAYA algorithm
BJAM	Binary JAYA algorithm with adaptive mutation
BSSA	Binary salp swarm algorithm
FS	Feature selection
GA	Genetic algorithm
HGSA	Hybrid gravitational search algorithm
PSO	Particle swarm optimization
SD	Standard derivation of the results
WOA	Whale optimization algorithm

Nomenclature

R_m	The mutation rate
N	The size of the population
$\|A\|$	The number of whole features

$\|R\|$	The number of selected features
α	The role of classification rate and length of feature subset
D	The dimensionality of the solution
FP	False positive
FN	False negative
$\gamma_R(D)$	The classification error rate
Max_itr	The maximum number of iterations
TP	True positive
TN	True negative
$\boldsymbol{X}$	The solution
X_j^{LB}	The lower bound of the decision variable in the j location
X_j^{UB}	The upper bound of the decision variable in the j location
$\boldsymbol{X}_{\text{best}}$	The best solution in the population
$\boldsymbol{X}_{\text{worst}}$	The worst solution in the population

1 Introduction

Feature selection (FS) is the process of removing irrelevant and redundant features in order to obtain the optimal subset of features. The feature selection is a typical process for many problems like image classification [1,2], molecular biology [3], finance [4], text mining [5,6], spam detection [7], and many others reported in [8]. In feature selection, the ultimate goal is to find a solution with high classification accuracy and minimum number of features. In optimization terms, FS is considered as NP-hard category in almost all of its variations [9].

Due to the huge numbers of features, several heuristic and metaheuristic algorithms have been proposed to tackle FS, in particular, the metaheuristic algorithms which are the most successful approaches for FS. These algorithms are categorized into two groups [10]: (i) single solution-based algorithms, and (ii) population-based algorithms. The main concern of the single solution-based algorithms is exploitation, whereas the main concern of the population-based algorithms is the exploration. Exploitation is needed to intensify the search in some promising area using the previous search experience. Exploration is needed to visit new regions in the search space. Single solution-based algorithms used for FS include tabu search [11], GRASP [12], iterated local search [13], β-hill climbing [14], and variable neighborhood search [13]. On the other hand, population-based algorithms used to tackle FS are the competitive swarm optimizer [15], grey wolf optimizer [16], gravitational search algorithm [16], bat algorithm [16], cuckoo search algorithm [17], salp swarm algorithm [18,19], gravitational search algorithm [20], particle swarm optimization [21], whale optimization algorithm, [22], crow search algorithm [23], dragonfly optimization [24], ant lion algorithm [25], firefly algorithm [26], grasshopper optimization algorithm [27], atom search optimization [28], and others reported in [29].

JAYA is one of the most recent population-based metaheuristic algorithms. It has been introduced to solve constrained and unconstrained optimization problems in 2016 by Roa [30]. JAYA is a Sanskrit word meaning victory. The arguable concept of this algorithm is that the solutions in the population should move toward the fittest solutions and should avoid the worst solutions. JAYA has several advantages such as it is simple and easy to implement, it reaches its goal with a minimum number of generations, and it does not any control parameters, which is only dependent on specifying population size and the number of generations [31–33]. These advantages lead to omitting the difficulty of studying the optimal parameter settings and reduce the complexity of the JAYA algorithm on solving any optimization problem [34].

Therefore, JAYA has rapidly been applied and developed for numerous optimization problems such as the knapsack problem [35], facial emotion recognition [36], parameters identification of photovoltaic models [34,37], parameters identification of Bouc–Wen hysteresis model for piezoelectric actuators [38], the maximum power point tracking problem of photovoltaic systems [39], design optimization of truss structures [40], design of the braced dome structure [41], heat exchanger [42,43], job-shop scheduling [44], power flow scheduling [45–47], flow shop scheduling [48], economic load dispatch [49], optimal design of thermal devices [50], reliability–redundancy allocation problems [51], micro-channel heat sink [52], transient stability assessment of power systems [53,54], optimization of plate-fin heat exchanger [55], classification of histopathological tissue images [56], nonlinear channel equalization [57], optimization of submerged arc welding process parameters [58] and so on. However, JAYA has several shortcomings: 1) it faces difficulties when used to solve complex problems with the rugged search space, 2) poor in exploration, and 3) the diversity is uncontrolled during the search [34,35]. These shortcomings lead the researchers to develop modified versions of the JAYA algorithm to cope with the nature of such optimization problems such as: adaptive parameter context [50], population structure context [59,60], hybrid with other optimization component context [31,35,51,61,62], and modified context [43,63,64].

On the other hand, the research communities proposed several binary versions of JAYA algorithm in order to solve some binary optimization problems like economic/emission unit commitment [65], predicting the transient stability status of power systems [54], digital mammogram classification [66], and optimization test functions [63]. These binary versions of JAYA algorithm are able to successfully tackle the mentioned problems. However, these binary versions have transferred the JAYA algorithm to deal with binary variables

in different ways. In [63], the Xor operator is used to transfer the decision variables from continuous to binary. The JAYA algorithm is also hybridized with local search module to deal with a complex uncapacitated facility location problems. Datasets from IEEE-CEC 2015 are also used to further verify the viability of their proposed method. In [56], the continuous variables are discretized into 0 or 1 based on the T threshold and the value of T is extensively studied. Furthermore, instead of using the best solution to generate the other solution, the JAYA algorithm used the first three-best and the last three-worst solutions to construct the next generation. Their proposed algorithm, although tackled image classification successfully, required a careful configuration of T threshold and problem search space to be efficiently worked. In [65], the V-shape transfer function is used to covert the continuous values into 0 or 1. This binary version of JAYA algorithms is used for unit commitment problem in power systems which are constrained problems.

In this paper, a novel binary version of the JAYA algorithm with a mutation operator is proposed for feature selection problem, called BJAM. In BJAM, two main modifications have been utilized in the original JAYA algorithm to deal with FS efficiently: 1) S-shape transfer function is added to the process of JAYA algorithm to convert the continuous variable domain into binary, and 2) The mutation operator is integrated within the framework of the JAYA algorithm in order to enhance the diversity control, therefore empowering the exploration capability. It should be noted that the mutation operator is controlled by the probability of mutation rate (R_m). The higher value of R_m leads to a higher rate of exploration. In order to preserve the parameter-less BJAM, the R_m parameter is tuned in the initial search and deterministically adapted during the search. In order to show the efficiency of the proposed algorithm, it is tested using 22 datasets collected from UCI data repository. Experimentally, the proposed algorithm is evaluated in terms of the classification accuracy, fitness values, number of selected features, and the computational times. Interestingly, the performance of the proposed BJAM algorithm outperforms the other comparative methods by achieving the new results in 7 out of 22 datasets. Furthermore, the performance of the proposed BJAM algorithm is comparable with other methods for the remaining datasets.

The rest of the paper is organized as follows: Sect. 2 reviews the related work of the feature selection. The procedural steps of the JAYA algorithm are described in Sect. 3. The proposed BJAM algorithm is described in Sect. 4. Section 5 discusses the experimental results obtained by running the proposed algorithm on UCI datasets. Finally, the conclusion and some future ideas are given in Sect. 6.

2 Literature Review

Conventionally, in pattern recognition application, raw data consist of irrelevant, redundant, and noisy features, which entail an effective procedure or technique that has the capability of removing such features without missing information. In order to achieve that purpose, feature selection approaches from different metric emerged to remove unwanted features from raw data and maintain the most worthy and informative features. Feature selection provides several benefits to the pattern recognition researchers [67]: (1) it can reduce the dimensionality of the data. (2) it can maintain or enhance the classification performance of machine learning algorithms. (3) it can reduce clinical setting cost.

Feature selection process relies on four main steps [68]: (1) subset generation, (2) subset assessment, (3) ending criterion, (4) validation. Subset generation relies on different directions and procedures in searching for exploring feature space and seeking for the best feature subset for the problem in hand. These search techniques fall under three categories: (1) exhaustive. (2) random. (3) heuristic. There are a few who used exhaustive search to address the feature selection problem. This can be justified as the process of exhaustive search entails examining all possible feature subset, which is computationally an expensive task, and most of the raw datasets of the real-life problems are high dimensional. Random search generates feature subsets randomly, and then add or remove features in a random manner. In the long term, running random search may perform similar to the complete search. Heuristic search can be illustrated as 'depth first' search guided by heuristics. In the recent decades, researchers have shifted toward using metaheuristics approaches due to their effectiveness in solving feature selection problems, for instance, particle swarm optimization [21], whale optimization algorithm [22], crow search algorithm [23], flower pollination algorithm [69], and bat algorithm [70]. Each generated candidate feature subset is passed to the subset assessment step. In this step, the assessment process of feature subset has two scenarios: 1) assess the features based on the principle characteristics of the dataset, called the filter method. On the other hand, machine learning is served as evaluator to each candidate feature subset. This schema is known as the wrapper method. Both subset generation and subset assessment are represented as an iterative optimization process and are stopped until the stopping constraint is met (i.e., obtaining the required classification accuracy or reaching a certain number of iterations). The main target of this process is to check the validity of the selected feature subset by running various experiments and drawing a comparison with those produced by other methods.

Feature selection has proved to be an effective and significant role in the pattern recognition task such as electroencephalogram classification [71], intrusion detection [72],

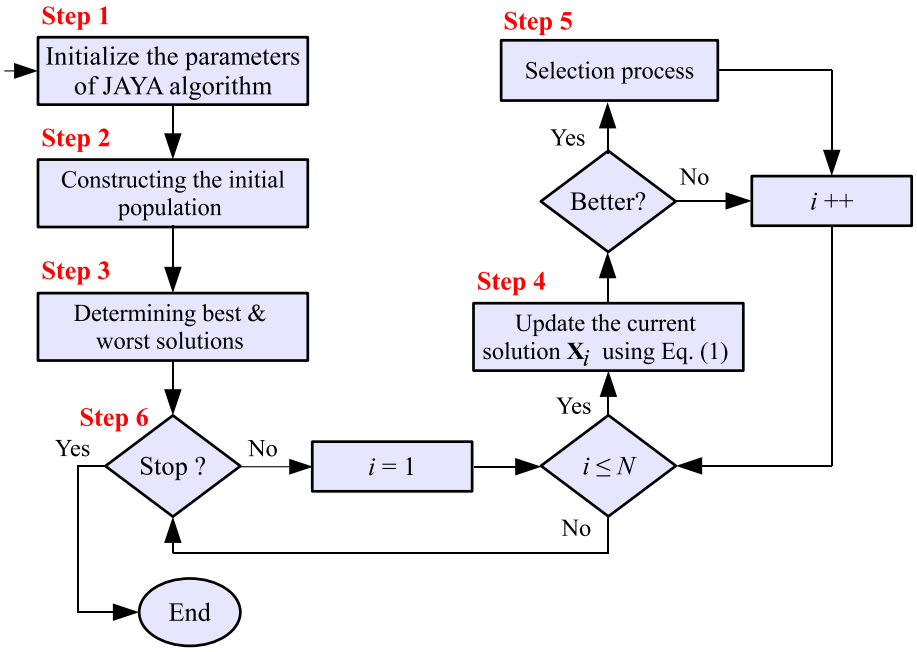

Fig. 1 The flowchart of the JAYA algorithm

cancer classification [69], and financial distress prediction [4]. Methods of feature selection have been widely classified into two main groups: filter and wrapper approach. As mentioned in the previous paragraph, the filter evaluates the features based on the intrinsic characteristics of datasets. Some examples of filter approach are Chi-square [73], information gain [74], fisher score [75], and minimum redundancy maximum relevancy [76]. In essence, the wrapper approach exploits machine learning to evaluate candidate feature subsets.

In the literature, many researchers proposed wrapper approaches to seek for the optimal feature subset using metaheuristics algorithms. For example, grey wolf optimization is a recent algorithm [77] that has been integrated with several update strategies; it was proposed to serve as a search engine in wrapper approach to select the most distinct features. In another study, gravitational search algorithm [20], along with evolutionary crossover and mutation operators, is proposed to tackle feature selection problem. A binary version ant lion optimizer [25] with a crossover operator is integrated to empower its exploitation process to obtain the optimal feature subset. The authors in [70] proposed a hybridization optimization framework between bat algorithm and β-hill climbing to optimize effectively feature search space. In [78], the researchers proposed a multi-objective feature selection method based on the artificial bee colony algorithm integrated with the non-dominated sorting procedure and genetic operators.

3 JAYA Algorithm

The JAYA algorithm is one of the most recent metaheuristic algorithms; it was introduced by Rao [30] for solving constrained and unconstrained optimization problems. This algorithm is a population-based metaheuristic inspired by the natural concept of the "survival of the fittest," where the solutions in the population should aim toward the best solutions while avoiding the worst solution. In other words, the search process of the JAYA algorithm tires to get closer to success in order to reach the best solutions, and tries to avoid failure by moving far from the worst solutions. JAYA algorithm has several advantages, which include easy to implement, no algorithm-specific parameters by depending only on two parameters (i.e., the size of the population and the maximum number of iterations) [40,52]. The pseudo-code of the JAYA algorithm is illustrated in Algorithm 1, while the flowchart of the JAYA algorithm is shown in Fig. 1. Furthermore, the procedural steps of the JAYA algorithms are described as follows:

Step 1: Initialize the parameters of JAYA algorithm. The two parameters of the JAYA algorithm have to be initialized for any optimization problems. These parameters include i) the size of the population (N), and the maximum number of iterations (Max_itr). The data related to the problem are extracted from datasets, and the solution representation, as well as the objective function, is normally defined.

Step 2: Constructing the initial population. In this step, the initial solutions are generated and stored in the pop-

ulation. The population consists of N solutions with D-dimensional solution $\boldsymbol{X}_i = (X_{i,1}, X_{i,2}, \cdots, X_{i,D})$. Each solution is generated randomly using Eq. (1)

$$X_{i,j} = X_j^{LB} + (X_j^{UB} - X_j^{LB}) \times U(0,1) \tag{1}$$

where $i \in (1, 2, \cdots, N)$, and $j \in (1, 2, \cdots, D)$. $X_{i,j}$ represents the j^{th} decision variable in the i^{th} solution. X_j^{LB}, X_j^{UB} are the lower and upper bounds of the decision variable in the j location. $U(0, 1)$ is a function that generates a random value uniformly distributed in (0,1). Later on, the fitness cost of the generated solutions is calculated using the objective function $f(\boldsymbol{X})$.

Step 3: Determining best and worst solutions. In this step, the best and the worst solutions (i.e., $\boldsymbol{X}_{\text{best}}$, $\boldsymbol{X}_{\text{worst}}$) in the population are identified. In minimization context, $f(\boldsymbol{X}_{\text{best}}) \leq f(\boldsymbol{X}_i)$, $\forall i = 1, 2, \ldots, N$. In addition, $f(\boldsymbol{X}_{\text{worst}}) \geq f(\boldsymbol{X}_i)$, $\forall i = 1, 2, \ldots, N$.

Step 4: Improvement process. In this step, every decision variable of every solution at each iteration is stochastically modified using JAYA operator formulated in Eq. (2):

$$X'_{i,j} = X_{i,j} + r_1 \times (X_{\text{best},j} - |X_{i,j}|) - r_2 \times (X_{\text{worst},j} - |X_{i,j}|) \tag{2}$$

where $\boldsymbol{X}'_i$, and $\boldsymbol{X}_i$ are the new and the current solutions, respectively. $X'_{i,j}$ is the modified value of the decision value $X_{i,j}$. r_1 and r_2 are two random numbers in the range of (0,1). These random numbers are used to ensure good exploration. The term $r_1 \times (X_{\text{best},j} - |X_{i,j}|)$ indicates the tendency of the current solution to move closer to the best solution ($X_{\text{best},j}$). On the other hand, the term $-r_2 \times (X_{\text{worst},j} - |X_{i,j}|)$ indicates the tendency of the current solution to avoid the worst solution ($X_{\text{worst},j}$). The absolute values $|X_{i,j}|$ in Eq. (2) are subtracted from $X_{\text{best},j}$ and $X_{\text{worst},j}$ to enhance the exploration capability of the JAYA algorithm [79].

Step 5: Selection process.
When the new solution $\boldsymbol{X}'_i$ is completely generated, it compares to the current one $\boldsymbol{X}_i$ stored in the i^{th} position in the population. If the fitness value of $\boldsymbol{X}'_i$ is better than the fitness value of $\boldsymbol{X}_i$, then the new solution replaces the current one in the population (i.e., $\boldsymbol{X}_i = \boldsymbol{X}'_i$).

Step 6: Stop criterion. Steps 3 to 5 are repeated until the maximum number of iterations (i.e., Max_itr) are satisfied.

Algorithm 1 The pseudo-code of JAYA algorithm

```
Initialize the parameters of the JAYA algorithm: population size (N), and maximum number of iterations (Max_itr)
———— Initialize the initial population ————
X_{i,j} = X_j^{LB} + (X_j^{UB} − X_j^{LB}) × U(0,1)   ∀i = 1, 2, ..., N, and ∀j = 1, 2, ..., D
Calculate f(X_i)   ∀i = 1, 2, ..., N
———— Improvements loop ————
itr=1
while (itr ≤ Max_itr) do
    Determine the best solution in the population (X_best)
    Determine the worst solution in the population (X_worst)
    for i = 1, ··· , N do
        for j = 1, ··· , D do
            Set r1 ∈[0,1]
            Set r2 ∈[0,1]
            X'_{i,j} = X_{i,j} + r1 × (X_{best,j} − |X_{i,j}|) − r2 × (X_{worst,j} − |X_{i,j}|)
        end for
        if f(X'_i) ≤ f(X_i) then
            X_i = X'_i   {Update process}
        end if
    end for
    itr = itr + 1
end while
```

4 The Proposed Modified Binary JAYA Algorithm

The proposed modified binary JAYA algorithm (BJAM) has the main JAYA optimization framework, including its operators. In addition, BJAM has two major improvements to the basic JAYA algorithm. The first improvement is the way of transferring the decision variables constructed based on the original JAYA algorithm operators into binary form. The second improvement is the utilization of a mutation operator controlled by the mutation rate (R_m) to control the diversity aspects during the search to manage the features space. Therefore, the two operators are discussed in the following subsections. The solution of feature selection is represented as a binary vector of features ($\boldsymbol{X} = (X_1, X_2, \ldots, X_D)$) where each variable in the solution takes a value of one if it is selected while zero if it is not so. The solution is evaluated based on a fitness function formulated in Eq. (3).

$$f(\boldsymbol{X}) = \alpha \gamma_R(D) + (1 - \alpha) \frac{|R|}{|A|} \tag{3}$$

where the classification error rate is represented as $\gamma_R(D)$. In this study, the kNN classifier is used for finding the classification error rate [80]. $|R|$ is the number of features selected and $|A|$ is the number of whole features, and α denotes the role of classification rate and length of feature subset. The value range of $\alpha \in [0, 1]$.

In order to compute the classification accuracy and error rate measurements, Eqs. (4) and (5) are used. Classification accuracy refers to a statistical measure which defines how well the classifier can correctly use the picked features to correctly label a given tuple into a class. It can be computed using Eq.(4).

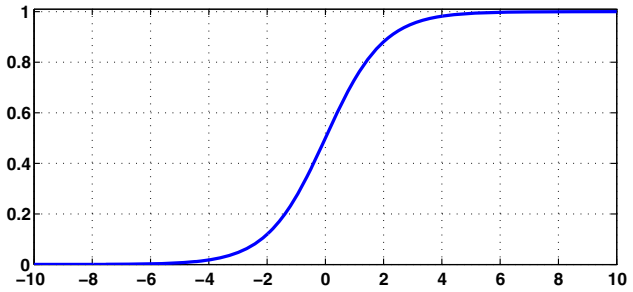

Fig. 2 *S*-shape function

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

where TP (true positive) denotes identifying correctly the class using a correct set of features. TN (true negative) denotes identifying correctly that it is not the class using a correct set of features. FP (false positive) denotes incorrectly identifying that it is the class. Finally, FN (false negative) denotes incorrectly identifying that it is not the class.

The classification error rate is computed in Eq.(5) used to measure the percentage of features that are incorrectly assigned. It will be part of the formulated objective function.

$$\gamma_R(D) = 1 - \text{Accuracy}. \tag{5}$$

4.1 *S*-shape Transfer Operator

The JAYA population is reconstructed using JAYA operator shown in Step 4 shown in the previous section. However, the resulting values of the solutions are continuous. In order to keep up with the feature space, the sinusoidal transfer function (or *S*-shape transfer function as drawn in Fig. 2) proposed in [81] is used to convert them to binary values. In practice, the new solution vector $\boldsymbol{X}'_i$ has to be entered to Eq. (6) in order to calculate its transfer vector T as follows:

$$T\left(X'_{i,j}\right) = \frac{1}{1 + e^{-X'_{i,j}}} \quad \forall i = (1, 2, \ldots, D). \tag{6}$$

Thereafter, each variable in any solution takes a binary value based on its transfer vector T using Eq. (7).

$$X'_{i,j} = \begin{cases} 1 & r < T\left(X'_{i,j}\right) \\ 0 & \text{Otherwise} \end{cases} \tag{7}$$

where $X'_{i,j}$ is the transferred variable to binary. r is a random value generated by a function yielded a value between 0 and 1 (i.e., $r \in [0, 1]$).

4.2 Mutation Operator

In fact, the search space of the features selection problems normally has a huge number of features, and a few relevant features shall be selected to represent the whole features. However, the search space of the features selection problems is not deep, and the relevant subset of features can be found on the surface of the search space regions. Furthermore, the improvement loop in JAYA algorithm performance is concerned with an intensification search in which the new population is improved based on the distance of the current solution and the best and worst solutions. Therefore, the mutation operator is utilized after *S*-shape transfer operator in the proposed BJAM to improve diversity. This operator is controlled by the adaptive mutation rate parameter (R_m) that takes a large value in the initial search and is gradually reduced during the search until a steady state is deducted (see Eq. (8)).

$$R_m = 0.9 + \frac{-0.9 \times (itr - 1)}{\text{Max\_itr} - 1}. \tag{8}$$

In practical terms, based on the adaptive mutation rate parameter (R_m), each binary value of the feature vectors stored in the population is checked for whether it is flipped as in Eq. (9).

$$X'_{i,j} = \begin{cases} \text{flip}(X'_{i,j}) & r < R_m \\ \text{no} - \text{flip}(X'_{i,j}) & \text{Otherwise} \end{cases}. \tag{9}$$

Note that $\text{flip}(X'_{i,j})$ is a function transpose 1 to 0 and vise-versa. r is a uniform distributed function which generates a value between 0 and 1. As can be noted, the higher the value of R_m leads to higher rate of exploration, and thus higher rate of diversity.

The flowchart of the BJAM is drawn in Fig. 3, while the pseudo-code of BJAM is given in Algorithm 2.

5 Results and Discussion

To examine the efficiency of the proposed BJAM algorithm for solving the FS problems, 22 datasets collected from UCI data repository well-circulated by the literature are used. The characteristics of these datasets are summarized in Table 1. In this table, the number of features varies from 9 to 7129, and the number of instances is in the range of 62 to 5000.

In order to make a fair comparison, the proposed BJAM algorithm as well as the binary JAYA (BJA) algorithm runs 20 independent replications as recommended in [14,22]. The settings of the parameters include the population size (N) which is set to 50, while the maximum number of iterations (Max_itr) is set 200, where these settings are recommended

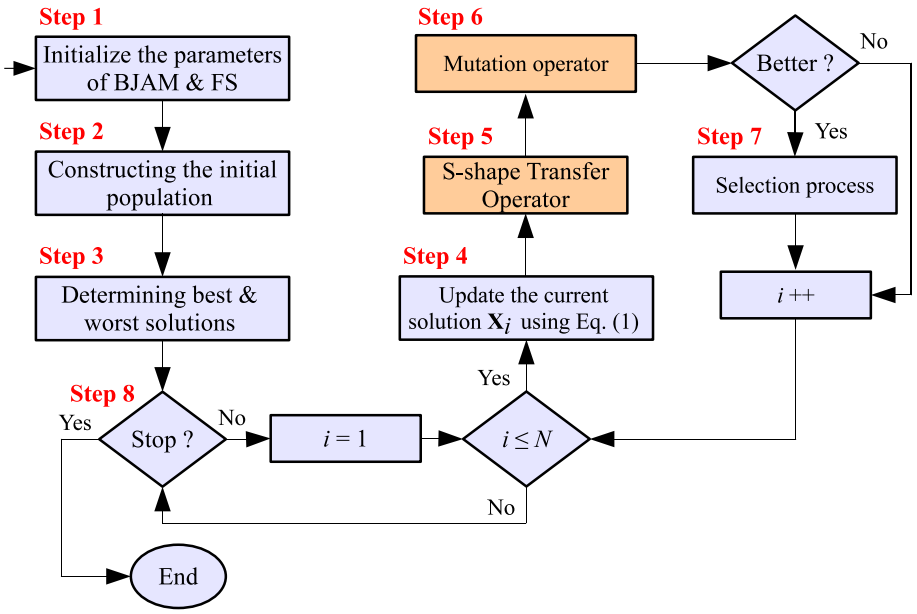

Fig. 3 The flowchart of the proposed BJAM algorithm for FS

Algorithm 2 The pseudo-code of the proposed BJAM algorithm for FS

Initialize the parameters of the JAYA algorithm: population size (N), and maximum number of iterations (Max_itr)
———— Constructing the initial population ————
$X_{i,j} \in [0,1]$ $\forall i = 1, 2, \ldots, N$, and $\forall j = 1, 2, \ldots, D$
Calculate $f(X_i)$ $\forall i = 1, 2, \ldots, N$
———— Improvements loop ————
itr=1
while ($itr \leq$ Max_itr) **do**
  Determine the best solution in the population (X_{best})
  Determine the worst solution in the population (X_{worst})
  for $i = 1, \cdots, N$ **do**
    for $j = 1, \cdots, D$ **do**
      Set $r_1 \in [0,1]$
      Set $r_2 \in [0,1]$
      $X'_{i,j} = X_{i,j} + r_1 \times (X_{best,j} - |X_{i,j}|) - r_2 \times (X_{worst,j} - |X_{i,j}|)$
      ———— S-shape Transfer Operator ————
      $T(X'_{i,j}) = \frac{1}{1+e^{-X'_{i,j}}}$
      Set $r \in [0,1]$
      if $r < T(X'_{i,j})$ **then**
        $X'_{i,j} = 1$
      else
        $X'_{i,j} = 0$
      end if
      ———— Mutation Operator ————
      for $j = 1, \cdots, D$ **do**
        Set $r \in [0,1]$
        $R_m = 0.9 + \frac{-0.9 \times (itr-1)}{\text{Max\_itr}-1}$
        if $r < R_m$ **then**
          if $X'_{i,j} > 0$ **then**
            $X'_{i,j} = 0$
          else
            $X'_{i,j} = 1$
          end if
        end if
      end for
    end for
    ———— Selection process ————
    if $f(X'_i) \leq f(X_i)$ **then**
      $X_i = X'_i$
    end if
  end for
  $itr = itr + 1$
end while

in [32]. In experiments, each dataset is divided into two parts for training and testing purposes, where 50% of the samples are used for training, and 50% of the samples for testing. All the experiments are run using a laptop with 2.80 Intel Core i7 with 16 GB RAM. The operating system was Microsoft Windows 10, while the Matlab (R2014a) is the programming language. The performance of the proposed BJAM algorithm against the BJA algorithm is provided in Sect. 5.1. Finally, the performance of the proposed BJAM algorithm is compared with ten other comparative methods in Sect. 5.2.

5.1 Comparison Between BJA and BJAM Algorithms

In this section, the performance of the proposed BJAM algorithm is compared with the BJA algorithm. Table 2 shows the experimental results of running the two JAYA-based algorithms in terms of the classification accuracy, the fitness values, the selected features, and the computational time required to converge, respectively. It is noteworthy to mention that the values in bold indicate best results.

Table 2 outlines the average (AV) and the standard derivation (SD) of the results obtained by the proposed BJAM algorithm, as well as the BJA algorithm in terms of the classification accuracy. The highest values of accuracy are the best, which are highlighted using bold font. We can remark that the BJA algorithm obtained the best results in 5 out of 22 datasets. On the other hand, the proposed BJAM algorithm achieved the best results in 12 out of 22 datasets, while both BJA and BJAM are able to achieve the same results for the remaining five datasets. This is due to the fact that the JAYA algorithm has a tendency to be an exploiter rather than explorer when biasing to the best solution to guide the search. Furthermore, the mutation operator is combined with

Table 1 The characteristics of the datasets

Dataset	No. of features	No. of instances
Breastcancer	9	699
BreastEW	30	596
Clean1	166	476
Colon	2000	62
CongressEW	16	435
Exactly	13	1000
Exactly2	13	1000
HeartEW	13	270
IonosphereEW	34	351
KrvskpEW	36	3196
Leukemia	7129	72
Lymphography	18	148
M-of-n	13	1000
PenglungEW	325	73
Semeion	265	1593
SonarEW	60	208
SpectEW	22	267
Tic-tac-toe	9	958
Vote	16	300
WaveformEW	40	5000
WineEW	13	178
Zoo	16	101

the JAYA algorithm in BJAM, and this is used to empower the exploration capability and thus skip the problem of getting stuck in local optima by well-controlling the diversify.

Apparently, the results produced by the BJA and BJAM are almost stable. This is ensured by the value of the standard deviation (SD) computed, where it almost tends to zero. This because the objective function drives the search to reach the optimal solution, and the binary search space of the feature selection components is well defined in the JAYA operators.

Figure 4 shows the notched boxplots that used to show the distribution of the results in terms of the classification accuracy of running the proposed BJAM algorithm, as well as the BJA algorithm for 20 times on all datasets. The *x-axis* represents the algorithm, while the *y-axis* represents the classification accuracy. Some of these boxplots show similar quartiles and wishers such as Breast cancer, Colon, Leukemia, HeartEW, Tic-tac-toe, and Zoo datasets in Fig. 4. In general, very small gap between the best, median, and the worst results indicates the stability of the algorithm. This is because the algorithm is able to achieve converged results on all times of runs. In-depth, it can be observed from the plots that there is a gap between the best, median, and the worst results produced by proposed BJAM algorithm for some boxplots such as Exactly2, Lymphography, PenglungEW, and WineEW. This is due to the fact that the mutation operator in BJAM can affect the diversity behavior; therefore, the performance can be also affected. In general, the gaps appeared in inner and outer quartiles whiskers of the boxplots are very small for all datasets since the differences between whiskers less than 0.1 which means that the performance is robust.

With regard to the fitness function values, the comparison results between the BJA and BJAM algorithms in terms of the average (AV) and the standard derivation (SD) are also recorded in Table 2. The best results are highlighted in bold (the minimum is the best). From the results in Table 2, we can see that the performance of the proposed BJAM algorithm outperforms the BJA algorithm by getting the best results in 20 out of 22 datasets. This is because the proposed algorithm is able to strike the right balance between the exploration and exploitation capabilities and thus obtained better results.

Figure 5 plots the convergence behavior of the BJA and BJAM using all 22 datasets. The *x-axis* represents the number of iterations, while the *y-axis* represents the fitness values. As can be seen from the figures, the convergence of the BJA algorithm is faster than the proposed BJAM algorithm. This leads to getting stuck in local optima in the early stages of the search process. On the other hand, the proposed BJAM algorithm is more robust than the BJA algorithm, where the fitness values are stagnated in the final stages of the search process.

Table 2 summarizes the average (AV) and the standard derivation (SD) of the results achieved by BJA and BJAM in terms of the number of the selected features. The best results are again highlighted using bold font (the minimum is the best). As shown in Table 2, the proposed BJAM algorithm is able to achieve the best results with minimum number of selected features for almost all datasets. On the other hand, the standard derivation values in Table 2 indicate that the proposed BJAM is more robust than the BJA by getting the lowest values for almost all datasets.

Finally, in Table 2, the average (AV) and the standard derivation (SD) of the results obtained by BJA and BJAM are summarized in terms of computational time required to converge (in seconds). It should be noted that the minimum computational time is the best, and it is highlighted using bold font. From the comparative results recorded in Table 2, it can be observed that the BJA algorithm achieved the minimum computational time on three datasets, while the BJAM algorithm obtained the minimum computational times for the other 19 datasets. This proves that the proposed BJAM algorithm is able to achieve the best results for almost all of the datasets in terms of classification accuracy, the fitness values, the selected attributes, and the computational time.

5.2 Comparison with Others

In order to evaluate the performance of the proposed BJAM algorithm against others, it is compared with ten other meth-

Table 2 The comparison results between BJA and BJAM algorithms

Dataset		Accuracy		Fitness		Features		Computational time	
		BJA	BJAM	BJA	BJAM	BJA	BJAM	BJA	BJAM
Breastcancer	AV	**0.969**	**0.969**	**0.038**	**0.038**	**6**	**6**	180.014	**166.688**
	SD	0.000	0.000	0.000	0.000	0.000	0.000	1.188	1.316
BreastEW	AV	**0.978**	0.974	**0.029**	0.031	21.55	**16.65**	179.116	**170.228**
	SD	0.003	0.003	0.003	0.002	2.364	4.395	1.579	1.432
Clean1	AV	**0.892**	0.891	0.114	**0.113**	117	**76.15**	305.254	**255.427**
	SD	0.006	0.005	0.006	0.005	6.618	15.012	2.213	1.085
Colon	AV	0.839	**0.871**	0.166	**0.131**	1232.5	**734.5**	199.365	**170.246**
	SD	0.000	0.007	0.001	0.007	215.539	128.614	0.941	1.070
CongressEW	AV	0.966	**0.97**	0.039	**0.033**	7.85	**4.9**	159.794	**152.769**
	SD	0.003	0.002	0.003	0.002	1.342	0.894	0.811	0.760
Exactly	AV	0.948	**0.969**	0.057	**0.036**	6.55	**6.2**	284.198	**249.461**
	SD	0.027	0.027	0.027	0.027	0.470	0.470	1.183	1.185
Exactly2	AV	**0.772**	**0.772**	**0.229**	**0.229**	3.9	**3.25**	266.385	**217.914**
	SD	0.003	0.004	0.002	0.002	1.761	2.447	1.201	0.808
HeartEW	AV	**0.88**	**0.88**	**0.125**	**0.125**	**8.15**	**8.15**	**131.978**	136.663
	SD	0.003	0.003	0.003	0.003	0.366	0.410	1.103	0.863
IonosphereEW	AV	0.928	**0.942**	0.077	**0.06**	18.95	**9.95**	145.874	**142.124**
	SD	0.004	0.005	0.004	0.005	1.490	2.285	0.882	1.061
KrvskpEW	AV	0.966	**0.968**	0.04	**0.037**	22.35	**18.15**	1981.722	**1572.512**
	SD	0.003	0.004	0.003	0.004	1.917	3.137	5.238	3.112
Leukemia	AV	0.865	**0.889**	0.139	**0.114**	4088.3	**2539.6**	1108.305	**536.773**
	SD	0.009	0.000	0.008	0.000	735.075	85.705	106.143	55.801
Lymphography	AV	0.892	**0.895**	0.113	**0.109**	10.5	**10.1**	123.567	**123.386**
	SD	0.009	0.010	0.009	0.010	1.372	1.281	0.782	0.617
M-of-n	AV	0.986	**0.989**	0.019	**0.016**	6.5	**6.35**	274.216	**247.394**
	SD	0.008	0.005	0.009	0.005	0.503	0.489	1.272	1.010
PenglungEW	AV	0.845	**0.874**	0.16	**0.128**	215.5	**123.8**	146.841	**144.827**
	SD	0.011	0.016	0.011	0.016	22.151	20.143	0.825	0.845
Semeion	AV	**0.977**	0.976	0.03	**0.029**	187.55	**133.6**	3091.689	**2164.767**
	SD	0.001	0.001	0.001	0.001	6.610	25.863	5.867	4.720
SonarEW	AV	0.809	**0.827**	0.195	**0.175**	37.9	**23.85**	132.365	**128.811**
	SD	0.008	0.013	0.008	0.013	4.607	4.428	0.646	0.585
SpectEW	AV	0.87	**0.881**	0.134	**0.121**	10.7	**8**	**130.843**	132.504
	SD	0.006	0.006	0.006	0.006	1.424	1.021	0.599	0.553
Tic-tac-toe	AV	**0.775**	**0.775**	**0.231**	**0.231**	**7**	**7**	270.295	**228.152**
	SD	0.000	0.000	0.000	0.000	0.000	0.000	1.133	0.823
Vote	AV	0.959	**0.96**	0.046	**0.044**	8.6	**7**	**135.287**	136.125
	SD	0.004	0.003	0.004	0.003	1.182	1.031	0.860	0.756
WaveformEW	AV	**0.793**	0.791	**0.212**	0.214	29.4	**26.6**	5103.722	**4090.432**
	SD	0.003	0.003	0.003	0.003	2.291	3.340	10.180	7.725
WineEW	AV	**0.975**	0.974	**0.031**	**0.031**	7.55	**6.8**	126.184	**124.771**
	SD	0.005	0.005	0.004	0.004	0.308	1.814	0.971	0.798
Zoo	AV	**1**	**1**	0.005	**0.001**	8.6	**2.05**	135.654	**133.775**
	SD	0.000	0.000	0.001	0.001	0.999	0.887	0.593	0.820

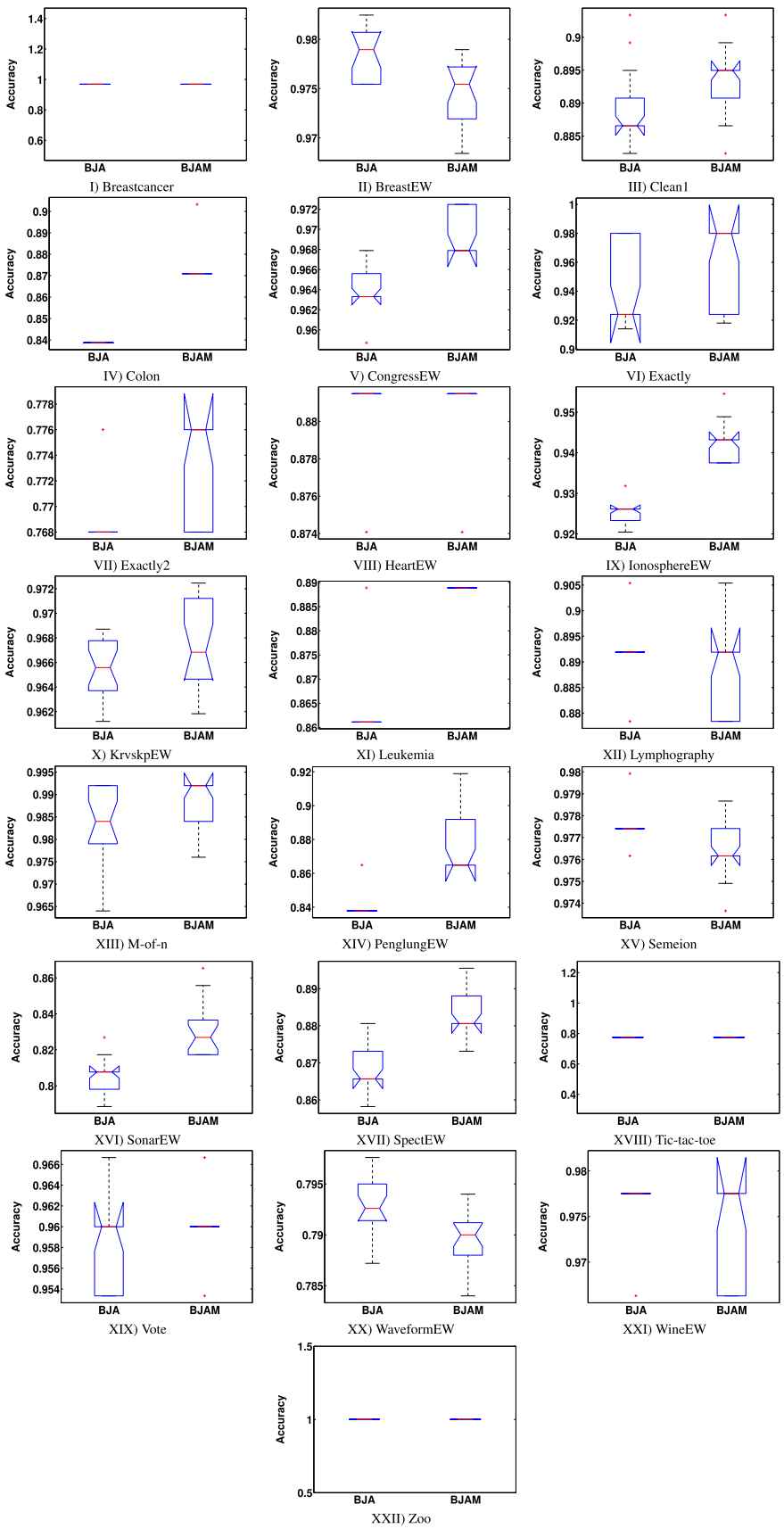

Fig. 4 The boxplots of the classification accuracy results for the proposed BJA and BJAM

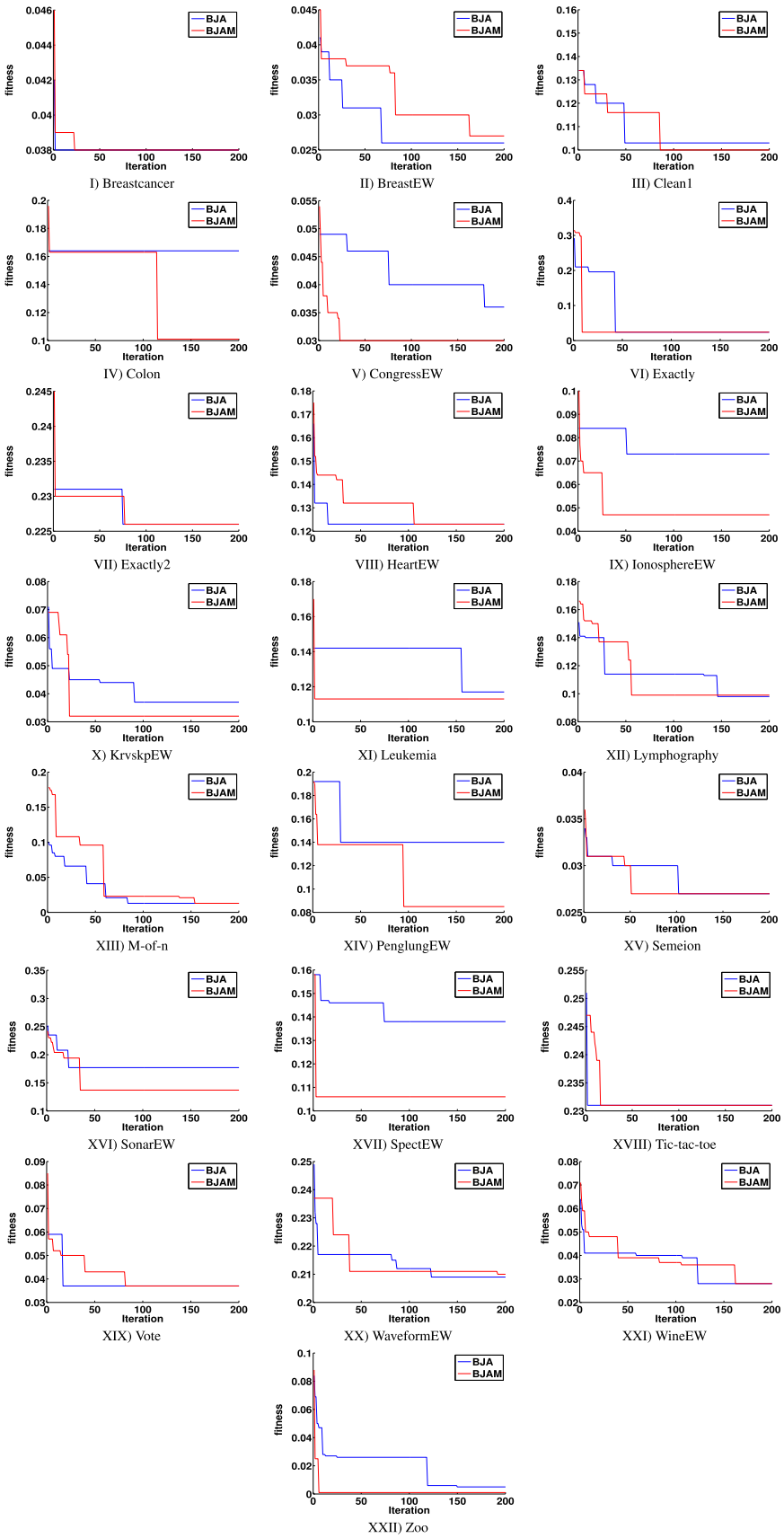

Fig. 5 The convergence plots of BJA and BJAM algorithms

Table 3 Abbreviations of the comparative methods

Method name	Abb.	Reference
Autonomous groups particles swarm optimization	AGPSO	[18]
Binary grasshopper optimization algorithm	BGOA	[16]
Binary grey wolf optimizer	BGWO	[16]
Binary gravitational search algorithm	BGSA	[16]
Binary bat algorithm	BBA	[16]
Binary salp swarm algorithm	BSSA	[18]
Hybrid gravitational search algorithm	HGSA	[20]
Whale optimization algorithm	WOA	[22]
Genetic algorithm	GA	[82]
Particle swarm optimization	PSO	[82]

Table 4 The average classification accuracy results of the proposed BJAM algorithm compared to others

Dataset	BJAM	BGOA	BGWO	BGSA	BBA	BSSA	HGSA	WOA	GA	PSO	AGPSO
Breastcancer	0.969	**0.980**	0.968	0.957	0.937	–	0.974	0.968	0.957	0.949	–
BreastEW	**0.974**	0.947	0.954	0.942	0.931	–	0.971	0.971	0.923	0.933	–
Clean1	0.891	0.863	0.908	0.898	0.826	**0.914**	–	–	0.862	0.845	-
Colon	**0.871**	0.870	0.661	0.766	0.682	0.657	–	–	0.682	0.624	–
CongressEW	**0.970**	0.964	0.948	0.951	0.872	**0.970**	0.966	0.956	0.898	0.937	0.961
Exactly	0.969	0.999	0.809	0.697	0.610	0.997	**1.000**	**1.000**	0.822	0.973	**1.000**
Exactly2	0.772	**0.780**	0.743	0.706	0.628	0.767	0.770	0.742	0.677	0.666	0.761
HeartEW	**0.880**	0.833	0.792	0.777	0.754	0.833	0.856	0.807	0.732	0.745	0.793
IonosphereEW	**0.942**	0.899	0.885	0.881	0.877	0.938	0.934	0.926	0.863	0.876	0.892
KrvskpEW	0.968	0.968	0.934	0.908	0.816	0.969	**0.978**	0.972	0.940	0.949	0.964
Leukemia	0.889	0.931	0.884	0.844	0.877	**0.951**	–	–	0.705	0.862	–
Lymphography	**0.895**	0.868	0.813	0.781	0.701	0.844	0.892	0.852	0.758	0.759	0.825
M-of-n	0.989	**1.000**	0.894	0.835	0.722	0.999	**1.000**	0.991	0.916	0.996	**1.000**
PenglungEW	0.874	0.927	0.850	0.919	0.795	0.907	**0.956**	0.792	0.672	0.879	0.836
Semeion	0.976	0.976	0.972	0.971	0.962	**0.980**	–	–	0.963	0.967	–
SonarEW	0.827	0.912	0.836	0.888	0.844	0.948	**0.958**	0.919	0.833	0.804	0.886
SpectEW	0.881	0.826	0.810	0.783	0.800	0.833	**0.919**	0.866	0.756	0.738	0.703
Tic-tac-toe	0.775	0.808	0.754	0.753	0.665	0.797	0.788	0.785	0.764	0.750	**0.813**
Vote	0.960	0.966	0.944	0.931	0.851	0.955	**0.973**	0.939	0.808	0.888	0.966
WaveformEW	0.791	0.737	0.723	0.695	0.669	0.736	**0.815**	0.753	0.712	0.732	0.738
WineEW	0.974	0.989	0.960	0.951	0.919	**0.998**	0.989	0.959	0.947	0.937	0.969
Zoo	**1.000**	0.993	0.975	0.939	0.874	0.993	0.932	0.980	0.946	0.963	0.971

ods from the literature that used to solve the same FS problems. The abbreviations of the comparative algorithms are summarized in Table 3. However, Table 4 illustrates the average of the results of running the comparative methods 20 times in terms of classification accuracy. The results of the proposed BJAM algorithm are extracted from Table 2. The best results are highlighted using bold font, where the maximum accuracy is the best. According to the results in Table 4, the HGSA algorithm obtained the best results on eight datasets. Additionally, the proposed BJAM algorithm outperforms the other comparative methods in seven datasets, while it can provide very competitive results compared to the other algorithms for the remaining datasets. Furthermore, the performance of BSSA is better than the other comparative methods on five datasets, while the BGOA and AGPSO algorithms outperform the other algorithms on three datasets. Finally, the WOA achieved the optimal results on one dataset. On the other hand, the BGWO, BGSA, BBA, GA, and PSO algorithms can not obtain the best results for any of the datasets.

For further validations, Friedman's statistical test is used to calculate the average rankings of the proposed BJAM algo-

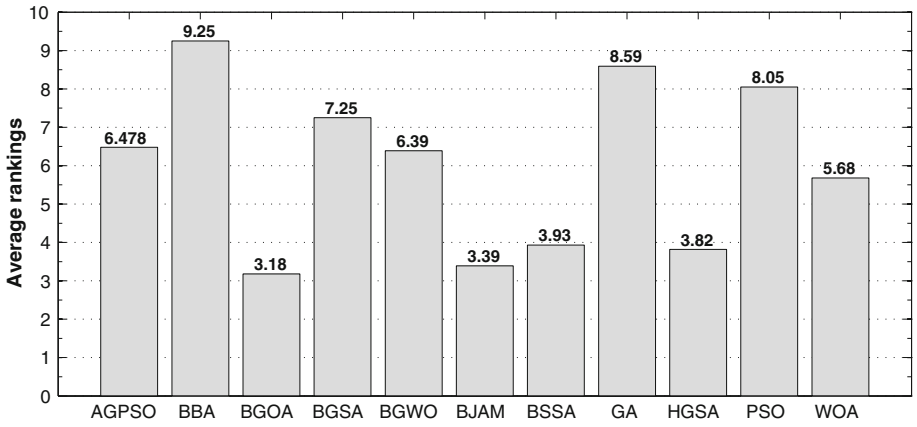

Fig. 6 Average rankings of the algorithms calculated using Friedman test based on the results recorded in Table 4

Table 5 Holm's results between the control method (BGOA) and other comparative methods

Rank	Algorithm	Adjusted ρ-value	(α/Rank)	Hypothesis
10	BBA	1.29E-09	0.00500	Rejected
9	GA	6.33E-08	0.00556	Rejected
8	PSO	1.15E-06	0.00625	Rejected
7	BGSA	4.74E-05	0.00714	Rejected
6	AGPSO	9.83E-04	0.00833	Rejected
5	BGWO	0.00135	0.01000	Rejected
4	WOA	0.01242	0.01250	Rejected
3	BSSA	0.45325	0.01667	Not Rejected
2	HGSA	0.52454	0.02500	Not Rejected
1	BJAM	0.83793	0.05000	Not Rejected

rithm as well as the other comparative algorithms. Figure 6 illustrates the average rankings of all comparative algorithms, where these rankings are calculated based on the results recorded in Table 4. It should be noted that the lower rankings indicate better performance, while significance level α = 0.05. In Friedman's test, the null hypothesis, H_0 affirms the equal behavior of the competitive methods. On the other hand, the alternative hypothesis, H_1, indicates the difference in behaviors of the competitive methods. It can be observed from Fig. 6, that the BGOA algorithm is ranked first, while the proposed BJAM algorithm is able to obtain the second rank. The ρ-value is computed by the Friedman's test is 8.0127E-11, which is below the significant level. This value indicates that there are significant differences between the behaviors of the comparative algorithms, and this leads to reject H_0 and accept H_1.

Using the rankings computed by the Friedman test, the adjusted ρ-value between the controlled algorithm and the other comparative methods is calculated using the Holm's procedure. It should be noted that the controlled algorithm is the BGOA algorithm because it obtained the first rankings. It can be seen from the results recorded in Table 5, there is a significant difference in behavior between the BGOA algorithm and seven of the other methods (i.e., BBA, GA, PSO, BGSA, AGPSO, BGWO, and WOA). However, there is no significant difference in behavior among the BGOA algorithm and the remaining algorithms. Interestingly, the behavior differences are not detected between the BGOA algorithm and the proposed BJAM algorithm using Holm's procedure. This proves that the proposed BJAM algorithm is an alternative method able to succeed in solving the feature selection problems.

6 Conclusion

Feature selection is a typical problem of machine learning and data mining. It is concerned with determining a subset of high discriminative features from the whole set of irrelevant, redundant, high-dimensional, and noisy features. Conventionally, finding the optimal subset of features is a taxing task in feature selection operation. Therefore, an efficient and robust optimization method is required to tackle such a problem. In this paper, the JAYA algorithm, the recent swarm-based algorithm, is utilized for feature selection problems. The JAYA algorithm is initially proposed to tackle continuous optimization problems. Here, the proposed version of JAYA algorithm is utilized to deal with binary search space and called binary JAYA algorithm (BJA). The *S*-shape function is added to the process of BJA to transfer the continuous region into binary. Furthermore, the mutation operator controlled by mutation rate (R_m) is adapted in the BJA algorithm to empower diversity capabilities, and the new version is called binary JAYA algorithm with adaptive mutation (BJAM).

The proposed BJA and BJAM are tested using 22 datasets collected from UCI data repository well-circulated in the literature. These datasets vary in terms of the number of features and number of instances. In order to evaluate the performance of the proposed method, four measures are used: classification accuracy, number of selected features, fitness values,

and computational times. In order to evaluate the mutation operator, BJA and BJAM are compared. The comparative results suggest that using mutation operator in BJAM for FS can achieve Superior results. For all measures, the proposed BJAM algorithm is able to excel those produced by BJA algorithm.

In terms of comparative evaluations, the classification accuracy results of the proposed BJAM method are compared against others produced by ten comparative methods using the same UCI datasets. The proposed BJAM algorithm is able to outperform the other methods in 7 out of 22 datasets used. Using Friedman statistical test, the proposed method is ranked ***second*** in comparison with the whole comparative methods. Furthermore, the behavior differences are not detected between the BGOA algorithm (i.e., BGOA is ranked first using Friedman's test) and the proposed BJAM algorithm using Holm's procedure as a post hoc method.

Since the proposed BJAM algorithm has revealed very successful outcomes for feature selection domain, the future direction can be guided by testing other machine learning feature selection-based problems from different domains like image classification, molecular biology, etc. The *S*-shape transfer function used for BJAM can be further studied. Finally, hybridizing with other metaheuristic components is needed in order to enhance the exploitation ability of the JAYA algorithm.

References

1. Ma, L.; Manchun Li, Y.; Gao, T.C.; Ma, X.; Lean, Q.: A novel wrapper approach for feature selection in object-based image classification using polygon-based cross-validation. IEEE Geosci. Remote Sens. Lett. **14**(3), 409–413 (2017)
2. Dubey, S.R.; Singh, S.K.; Singh, R.K.: Local wavelet pattern: a new feature descriptor for image retrieval in medical ct databases. IEEE Trans. Image Process. **24**(12), 5892–5903 (2015)
3. Bajcsy, P.: An overview of dna microarray grid alignment and foreground separation approaches. EURASIP J. Adv. Signal Process. **2006**(1), 080163 (2006)
4. Liang, D.; Tsai, C.-F.; Hsin-Ting, W.: The effect of feature selection on financial distress prediction. Knowl.-Based Syst. **73**, 289–297 (2015)
5. Jing, L.-P.; Huang, H.-K.; Shi, H.-B.: Improved feature selection approach tfidf in text mining. In: Proceedings. International Conference on Machine Learning and Cybernetics, vol 2, IEEE, pp. 944–946 (2002)
6. Ravisankar, P.; Ravi, V.; Rao, G.R.; Bose, I.: Detection of financial statement fraud and feature selection using data mining techniques. Decis. Support Syst. **50**(2), 491–500 (2011)
7. Zhang, Y.; Wang, S.; Phillips, P.; Ji, G.: Binary pso with mutation operator for feature selection using decision tree applied to spam detection. Knowl.-Based Syst. **64**, 22–31 (2014)
8. Bolon-Canedo, V.; Alonso-Betanzos, A.: Ensembles for feature selection: a review and future trends. Inf. Fusion **52**, 1–12 (2019)
9. Blum, A.L.; Rivest, R.L.: Training a 3-node neural network is np-complete. In: Machine Learning: From Theory to Applications, pp. 9–28, Springer (1993)
10. BoussaiD, I.; Lepagnot, J.; Siarry, P.: A survey on optimization metaheuristics. Inf. Sci. **237**, 82–117 (2013)
11. Zhang, H.; Sun, G.: Feature selection using tabu search method. Pattern Recognit. **35**(3), 701–711 (2002)
12. Bermejo, P.; Gamez, J.A.; Puerta, J.M.: A grasp algorithm for fast hybrid (filter-wrapper) feature subset selection in high-dimensional datasets. Pattern Recognit. Lett. **32**(5), 701–711 (2011)
13. Marinaki, M.; Marinakis, Y.: A hybridization of clonal selection algorithm with iterated local search and variable neighborhood search for the feature selection problem. Memet. Comput. **7**(3), 181–201 (2015)
14. Al-Betar, M.A.; Hammouri, A.I.; Awadallah, M.A.; Iyad Abu Doush.: Binary β-hill climbing optimizer with s-shape transfer function for feature selection. J. Ambient Intell. Human. Comput. (2020). https://doi.org/10.1007/s12652-020-02484-z
15. Shenkai, G.; Cheng, R.; Jin, Y.: Feature selection for high-dimensional classification using a competitive swarm optimizer. Soft Comput. **22**(3), 811–822 (2018)
16. Mafarja, M.; Aljarah, I.; Heidari, A.A.; Hammouri, A.I.; Faris, H.; Alaa, A.-Z.; Mirjalili, S.: Evolutionary population dynamics and grasshopper optimization approaches for feature selection problems. Knowl.-Based Syst. **145**, 25–45 (2018)
17. El Aziz, M.A.; Hassanien, A.E.: Modified cuckoo search algorithm with rough sets for feature selection. Neural Comput. Appl. **29**(4), 925–934 (2018)
18. Aljarah, I.; Mafarja, M.; Heidari, A.A.; Faris, H.; Zhang, Y.; Mirjalili, S.: Asynchronous accelerating multi-leader salp chains for feature selection. Appl. Soft Comput. **71**, 964–979 (2018)
19. Hegazy, A.E.; Makhlouf, M.A.; El-Tawel, G.S.: Feature selection using chaotic salp swarm algorithm for data classification. Arab. J. Sci. Eng. **44**(4), 3801–3816 (2019)
20. Taradeh, M.; Mafarja, M.; Heidari, A.A.; Faris, H.; Aljarah, I.; Mirjalili, S.; Fujita, H.: An evolutionary gravitational search-based feature selection. Inf. Sci. **497**, 219–239 (2019)
21. Vieira, S.M.; Mendoncca, L.F.; Farinha, G.J.; Sousa, J.M.C.: Modified binary pso for feature selection using svm applied to mortality prediction of septic patients. Appl. Soft Comput. **13**(8), 3494–3504 (2013)
22. Mafarja, M.; Mirjalili, S.: Whale optimization approaches for wrapper feature selection. Appl. Soft Comput. **62**, 441–453 (2018)
23. Sayed, G.I.; Hassanien, A.E.; Azar, A.T.: Feature selection via a novel chaotic crow search algorithm. Neural Comput. Appl. **31**(1), 171–188 (2019)
24. Mafarja, M.; Aljarah, I.; Heidari, A.A.; Faris, H.; Fournier-Viger, P.; Li, X.; Mirjalili, S.: Binary dragonfly optimization for feature selection using time-varying transfer functions. Knowl.-Based Syst. (2018)
25. Emary, E.; Zawbaa, H.M.; Hassanien, A.E.: Binary ant lion approaches for feature selection. Neurocomputing **213**, 54–65 (2016)
26. Zhang, L.; Mistry, K.; Lim, C.P.; Neoh, S.C.: Feature selection using firefly optimization for classification and regression models. Decis. Support Syst. **106**, 64–85 (2018)
27. Aljarah, I.; Ala'M, A.-Z.; Faris, H.; Hassonah, M.A.; Mirjalili, S.; Saadeh, H.: Simultaneous feature selection and support vector machine optimization using the grasshopper optimization algorithm. Cogn. Comput. **10**(3), 478–495 (2018)
28. Too, J.; Abdullah, A.R.: Chaotic atom search optimization for feature selection. Arab. J. Sci. Eng., pp. 1–17 (2020)
29. Li, Y.; Li, T.; Liu, H.: Recent advances in feature selection and its applications. Knowl. Inf. Syst. **53**(3), 551–577 (2017)
30. Rao, R.: Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. Int. J. Ind. Eng. Comput. **7**(1), 19–34 (2016)
31. ViswavandyaViswavandya, M.; Mohanty, A.; Ray, P.K.; Paramita, P.; et al.: Restoration of stable voltage in an isolated hybrid solar

power system with combined jaya-de algorithm. Optik **180**, 536–548 (2019)
32. Du, D.-C.; Vinh, H.-H.; Trung, V.-D.; Hong Quyen, N.-T.; Trung, N.-T.: Efficiency of jaya algorithm for solving the optimization-based structural damage identification problem based on a hybrid objective function. Eng. Optim. **50**(8), 1233–1251 (2018)
33. Singh, S.P.; Prakash, T.; Singh, V.P.; Ganesh Babu, M.: Analytic hierarchy process based automatic generation control of multi-area interconnected power system using jaya algorithm. Eng. Appl. Artif. Intell. **60**, 35–44 (2017)
34. Yu, K.; Liang, J.J.; Qu, B.Y.; Chen, X.; Wang, H.: Parameters identification of photovoltaic models using an improved jaya optimization algorithm. Energy Convers. Manag. **150**, 742–753 (2017)
35. Congcong, W.; He, Y.: Solving the set-union knapsack problem by a novel hybrid jaya algorithm. Soft Comput. **24**(3), 1883–1902 (2020)
36. Wang, S.-H.; Phillips, P.; Dong, Z.-C.; Zhang, Y.-D.: Intelligent facial emotion recognition based on stationary wavelet entropy and jaya algorithm. Neurocomputing **272**, 668–676 (2018)
37. Wang, L.; Huang, C.: A novel elite opposition-based jaya algorithm for parameter estimation of photovoltaic cell models. Optik **155**, 351–356 (2018)
38. Son, N.N.; Van Kien, C.; Anh, H.P.H.: Parameters identification of bouc–wen hysteresis model for piezoelectric actuators using hybrid adaptive differential evolution and jaya algorithm. Eng. Appl. Artif. Intell. **87**, 103317 (2020)
39. Huang, C.; Wang, L.; Yeung, R.S.-C.; Zhang, Z.; Chung, H.S.-H.; Bensoussan, A.: A prediction model-guided jaya algorithm for the pv system maximum power point tracking. IEEE Trans. Sustain. Energy **9**(1), 45–55 (2017)
40. Degertekin, S.O.; Lamberti, L.; Ugur, I.B.: Sizing, layout and topology design optimization of truss structures using the jaya algorithm. Appl. Soft Comput. **70**, 903–928 (2018)
41. Dede, T.; Grzywinski, M.; Rao, R.V.: Jaya: A new meta-heuristic algorithm for the optimization of braced dome structures. In: Rao, R.V., Taler, J. (eds.) Advanced Engineering Optimization Through Intelligent Techniques, pp. 13–20. Springer, Singapore (2020)
42. Rao, R.V.; Saroj, A.; Oclon, P.; Taler, J.; Taler, D.: Single-and multi-objective design optimization of plate-fin heat exchangers using jaya algorithm. Heat Trans. Eng. **39**(13–14), 1201–1216 (2018)
43. RaoRao, R.V.; Saroj, A.: Multi-objective design optimization of heat exchangers using elitist-jaya algorithm. Energy Syst. **9**(2), 305–341 (2018)
44. Gao, K.; Yang, F.; Zhou, M.C.; Pan, Q.; Suganthan, P.N.: Flexible job-shop rescheduling for new job insertion by using discrete jaya algorithm. IEEE Trans. Cybern. **49**(5), 1944–1955 (2018)
45. Warid, W.; Hizam, H.; Mariun, N.; Wahab, N.I.A.: A novel quasi-oppositional modified jaya algorithm for multi-objective optimal power flow solution. Appl. Soft Comput. **65**, 360–373 (2018)
46. Warid, W.; Hizam, H.; Mariun, N.; Abdul-Wahab, N.: Optimal power flow using the jaya algorithm. Energies **9**(9), 678 (2016)
47. Warid, W.: Optimal power flow using the amtpg-jaya algorithm. Appl. Soft Comput., p. 106252 (2020)
48. Buddala, R.; Mahapatra, S.S.: Improved teaching-learning-based and jaya optimization algorithms for solving flexible flow shop scheduling problems. J. Ind. Eng. Int. **14**(3), 555–570 (2018)
49. Bhoye, M.; Pandya, M.H.; Valvi, S.; Trivedi, I.N.; Jangir, P.; Parmar, S.A.: An emission constraint economic load dispatch problem solution with microgrid using jaya algorithm. In: 2016 International Conference on Energy Efficient Technologies for Sustainability (ICEETS), IEEE, pp. 497–502 (2016)
50. Rao, R.V.; More, K.C.: Design optimization and analysis of selected thermal devices using self-adaptive jaya algorithm. Energy Convers. Manag. **140**, 24–35 (2017)
51. Ghavidel, S.; Azizivahed, A.; Li, L.: A hybrid jaya algorithm for reliability-redundancy allocation problems. Eng. Optim. **50**(4), 698–715 (2018)
52. Rao, R.V.; More, K.C.; Taler, J.; Oclon, P.: Dimensional optimization of a micro-channel heat sink using jaya algorithm. Appl. Thermal Eng. **103**, 572–582 (2016)
53. Gu, X.; Li, Y.; Jia, J.: Feature selection for transient stability assessment based on kernelized fuzzy rough sets and memetic algorithm. Int. J. Electr. Power and Energy Syst. **64**, 664–670 (2015)
54. Li, Y.; Yang, Z.: Application of eos-elm with binary jaya-based feature selection to real-time transient stability assessment using pmu data. IEEE Access **5**, 23092–23101 (2017)
55. Kiran Chunilal More and R Venkata Rao. Design optimization of plate-fin heat exchanger by using modified jaya algorithm. In: Advanced Engineering Optimization Through Intelligent Techniques, Springer, Berlin, pp. 165–172 (2020)
56. Tiwari, Varun; Jain, S.C.: An optimal feature selection method for histopathology tissue image classification using adaptive jaya algorithm. Evol. Intell. 1–14 (2019)
57. Ingle, K.K.; Jatoth, R.K.: An efficient jaya algorithm with lévy flight for non-linear channel equalization. Expert Syst. Appl. **145**, 112970 (2020)
58. Rao, R.V.; Rai, D.P.: Optimization of submerged arc welding process parameters using quasi-oppositional based jaya algorithm. J. Mech. Sci. Technol. **31**(5), 2513–2522 (2017)
59. Rao, R.V.; Saroj, A.: A self-adaptive multi-population based jaya algorithm for engineering optimization. Swarm Evol. Comput. **37**, 1–26 (2017)
60. Wang, L.; Zhang, Z.; Huang, C.; Tsui, K.L.: A gpu-accelerated parallel jaya algorithm for efficiently estimating li-ion battery model parameters. Appl. Soft Comput. **65**, 12–20 (2018)
61. Luo, X.; Cao, L.; Wang, L.; Zhao, Z.; Huang, C.: Parameter identification of the photovoltaic cell model with a hybrid jaya-nm algorithm. Optik **171**, 200–203 (2018)
62. Azizi, M.; Ghasemi, S. A. M.; Ejlali, R.G.; Talatahari, S.: Optimum design of fuzzy controller using hybrid ant lion optimizer and jaya algorithm. Artif. Intell. Rev., pp. 1–32, (2019)
63. Aslan, M.; Gunduz, M.; Kiran, M.S.: Jayax: Jaya algorithm with xor operator for binary optimization. Appl. Soft Comput. **82**, 105576 (2019)
64. Oclon, P.; Cisek, P.; Rerak, M.; Taler, D.; Rao, R.V.; Vallati, A.; Pilarczyk, M.: Thermal performance optimization of the underground power cable system by using a modified jaya algorithm. Int. J. Thermal Sci. **123**, 162–180 (2018)
65. Yang, Z.; Guo, Y.; Niu, Q.; Ma, H.; Zhou, Y.; Zhang, L.: A novel binary jaya optimization for economic/emission unit commitment. In: 2018 IEEE Congress on Evolutionary Computation (CEC), pp. 1–6 (2018)
66. Mohanty, F., Rup, S., Dash, B.: Compound local binary pattern and enhanced jaya optimized extreme learning machine for digital mammogram classification. In: International Conference on Intelligent Data Engineering and Automated Learning, Springer, pp. 1–8 (2018)
67. Mohamad, M.S.; Omatu, S.; Deris, S.; Yoshioka, M.: A modified binary particle swarm optimization for selecting the small subset of informative genes from gene expression data. IEEE Trans. Inf. Technol. Biomed. **15**(6), 813–822 (2011)
68. Dash, M.; Liu, H.: Feature selection for classification. Intell. data Anal. **1**(1–4), 131–156 (1997)
69. Osama, A.A.; Khader, A.T.; Al-Betar, M.A.; Alyasseri, Z.A.A.: A hybrid filter-wrapper gene selection method for cancer classification. In: 2nd International Conference on BioSignal Analysis, Processing and Systems (ICBAPS), pp. 32–37 (2018)
70. Alomari, O.A.; Khader, A.T.; Al-Betar, M.A.; Awadallah, M.A.: A novel gene selection method using modified mrmr and hybrid

bat-inspired algorithm with β-hill climbing. Appl. Intell. **48**(11), 4429–4447 (2018)
71. Rodrigues, D.; Silva, G.F.A.; Papa, J.P.; Marana, A.N.; Yang, X.-S.: Eeg-based person identification through binary flower pollination algorithm. Expert Syst. Appl. **62**, 81–90 (2016)
72. Alomari, O.; Othman, Z.A.: Bees algorithm for feature selection in network anomaly detection. J. Appl. Sci. Res. **8**(3), 1748–1756 (2012)
73. Chao-Ton, S.; Hsu, J.-H.: An extended chi2 algorithm for discretization of real value attributes. IEEE Trans. Knowl. Data Eng. **17**(3), 437–441 (2005)
74. Lai, C.-M.; Yeh, W.-C.; Chang, C.-Y.: Gene selection using information gain and improved simplified swarm optimization. Neurocomputing **218**, 331–338 (2016)
75. Gu, Q.; Li, Z.; Han, J.: Generalized fisher score for feature selection. (2012) arXiv:1202.3725
76. Alomari, O.A.; Khader, A.T.; Al-Betar, M.A.; Abualigah, L.M.: Mrmr ba: a hybrid gene selection algorithm for cancer classification. J. Theory Appl. Inf. Technol. **95**(12), 2610–2618 (2017)
77. Qiang, T.; Chen, X.; Liu, X.: Multi-strategy ensemble grey wolf optimizer and its application to feature selection. Appl. Soft Comput. **76**, 16–30 (2019)
78. Hancer, E.; Xue, B.; Zhang, M.; Karaboga, D.; Akay, B.: Pareto front feature selection based on artificial bee colony optimization. Inf. Sci. **422**, 462–479 (2018)
79. Rao, R.V.; Waghmare, G.G.: A new optimization algorithm for solving complex constrained design optimization problems. Eng. Optim. **49**(1), 60–83 (2017)
80. Liao, Y.; Vemuri, V.R.: Use of k-nearest neighbor classifier for intrusion detection. Comput. Secur. **21**(5), 439–448 (2002)
81. Kennedy, J.; Eberhart, R.C.: A discrete binary version of the particle swarm algorithm. In: 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, vol. 5, IEEE, pp. 4104–4108 (1997)
82. Kashef, S.; Nezamabadi-pour, H.: An advanced aco algorithm for feature subset selection. Neurocomputing **147**, 271–279 (2015)