Journal of Ambient Intelligence and Humanized Computing
https://doi.org/10.1007/s12652-021-03226-5

ORIGINAL RESEARCH

Binary Jaya algorithm based on binary similarity measure for feature selection

Abhilasha Chaudhuri[1] · Tirath Prasad Sahu[1]

Received: 1 October 2020 / Accepted: 25 March 2021

Abstract
Feature selection (FS) has become an indispensable data preprocessing task because of the huge amount of high dimensional data being generated by current technologies. These high dimensional data contains irrelevant, redundant, and noisy features that deteriorate classification accuracy. FS reduces dimensionality by removing the unwanted features thus improves classification accuracy. FS can be considered as a binary optimization problem. In order to solve this problem, this work proposes a new wrapper feature selection technique based on the Jaya algorithm. Three binary variants of the Jaya algorithm are proposed, the first and second ones are based on transfer functions namely BJaya-S and BJaya-V. The third variant (BJaya-JS) explores the search space on the basis of the Jaccard Similarity index. In addition, a probability-based local search technique, namely Neighbourhood Search is proposed to balance the exploration and exploitation. The variants of Jaya algorithm are evaluated and the best variant is selected. The best variant is further compared with six state-of-the-art feature selection techniques. All the performances are tested on 18 high dimensional standard UCI datasets. Experimental result comparison shows that the proposed feature selection technique performs better than other competitors.

Keywords Jaya algorithm · Jaccard similarity · Feature selection

1 Introducton

In the knowledge discovery process, data pre-processing is the fundamental step in which data is prepared to be used by various data mining algorithms. Classification and clustering are the two main classes of data mining algorithms (Han et al. 2011). Both of these classes of algorithms depend upon the features of the underlying dataset to make predictions. However, when the dimensionality of the dataset is very high, performance of the classification and clustering algorithm is affected (Bennasar et al. 2015). High dimensionality in datasets not only causes large time for model construction and redundancy in data but also causes poor generalization capability of clustering and classification algorithms. In order to improve the performance of these algorithms, the selection of important features is required. This process is called *feature selection*. In the feature selection process, relevant and unambiguous features of the dataset are selected. Whereas, noisy and redundant features are removed from the dataset. Feature selection reduces the dimensionality of the dataset, thereby reducing the computational costs and time, at the same time improves the performance of classification and clustering algorithms.

Feature selection techniques are of three types: Filter, Wrapper and Hybrid techniques (Ang et al. 2015). Filter methods decide the importance of features on the basis of some information-theoretic or statistical measures like information gain, correlation, variance, etc. then the most important features as per the user's threshold are used for classification (or other machine learning tasks). In the wrapper method, some search technique is applied to find the feature subset which gives maximum classification accuracy, i.e. the classifier is involved in searching the best feature subset. Therefore wrapper techniques give better classification accuracy than the filter methods. However, filter methods are considered generalized methods because they are independent of classifiers hence they are computationally less expensive than wrapper methods (Mafarja and Mirjalili

Abhilasha Chaudhuri
achaudhuri.phd2018.it@nitrr.ac.in

Tirath Prasad Sahu
tpsahu.it@nitrr.ac.in

[1] National Institute of Technology, Raipur 492010, India

2017). Hybrid feature selection techniques are a combination of filter and wrapper techniques (Peng et al. 2010).

Feature selection techniques based on meta-heuristic optimization algorithms have attracted the efforts of researchers in past two decades (Yang 2010). For a dataset with N features, there are 2^N combination of possible feature subsets. The subset which gives the best performance needs to be chosen by feature selection technique. Therefore feature selection (FS) is a combinatorial problem. Metaheuristic algorithms have been proven better for combinatorial problems (Talbi 2009). They have the ability to explore 2^N possible feature subset in order to find the near-optimal feature subset in a reasonable amount of time. Wrapper feature selection based on meta-heuristic algorithms like Genetic Algorithm (GA) (Kabir et al. 2011; Sikora and Piramuthu 2007) , Ant Colony Optimization(ACO) (Kashef and Nezamabadi-pour 2015) , Particle Swarm Optimization (PSO) (Faris et al. 2016) and Artificial Bee Colony (ABC) (Karaboga and Basturk 2008), are popular feature selection methods available in literature. Performance improvement has been achieved in recent years in the field of wrapper-based feature selection techniques. Various nature-inspired optimization algorithms have been used for wrapper-based feature selection methods. Recently a feature selection algorithm based on Grey Wolf Optimization algorithm, which is based on the behavior of grey wolf, have been proposed (Al-Tashi et al. 2019). Whale optimization algorithm(WOA) (Mafarja and Mirjalili 2017) is another wrapper FS technique which is based on the behavior of Whale. The binary variant of the Dragonfly algorithm (Mafarja et al. 2017b) is also used for wrapper FS. Later, Binary Dragonfly optimization for feature selection using time-varying transfer functions (Mafarja et al. 2018) and an improved dragonfly algorithm (Hammouri et al. 2020) are developed as an extension of (Mafarja et al. 2017b) resulting in the improved performance. Dynamic Salp Swarm algorithm (Tubishat et al. 2020) is an improvement to the Salp Swarm Optimization algorithm (Ahmed et al. 2018) for feature selection problem. Binary Crow search algorithm with time-varying flight lenght (Chaudhuri and Sahu 2020b) is also used for wrapper based FS techniques. Binary butterfly optimization algorithm (Arora and Anand 2019) is yet another recent nature-inspired algorithm that has given promising results in the field of feature selection. The navigation method of Moths is mimicked in Moth flame optimizer, it is used recently for FS (Zawbaa et al. 2016). Antlion optimizer based feature selection method (Mafarja et al. 2017a) is based on the hunting behavior of the Antlions. The binary version of Grasshopper optimization algorithm (Hichem et al. 2019) based on different transfer functions is recently proposed in the literature. A new approach based on unsupervised feature selection by biased random-key genetic algorithm is proposed recently (Martarelli and Nagano 2020). Binary β -hill climbing algorithm (Al-Betar et al. 2020), binary social spider algorithm (Emine and Ülker 2020) and flower pollination based on rough set approach (Tawhid and Ibrahim 2020) are some of the latest algorithms for Feature selection problems.

The meta-heuristic algorithms mentioned in the above paragraph have some specific parameters, that need to be tuned in order to achieve the best performance. For example, three parameters namely - inertia weight, social parameter and cognitive parameter, need to be tuned for achieving the best performance in PSO. Similarly, GA requires adjustment of crossover rate, mutation rate, type of crossover and mutation operator. Proper adjustment of algorithm-specific parameters is required in all the algorithms mentioned above. Parameter adjustment is a very crucial task. Failing to properly tune the algorithm-specific parameters causes the problem of local optima and increased computational cost. Keeping this difficulty in mind Rao et al. have developed the Teaching Learning Based Optimization algorithm (TLBO) (Rao et al. 2011). TLBO algorithm does not require any algorithm-specific parameter tuning. After TLBO, Rao proposed another algorithm-specific parameter-free algorithm named Jaya algorithm(Venkata Rao 2016). Jaya is a simple yet effective optimization algorithm that has been successfully applied to many engineering problems (Rao 2019).

Jaya algorithm is chosen in this work to address the feature selection problem as it has the following advantages over other well-known meta-heuristic techniques: (1) it does not require any parameter tuning, (2) it is very simple to apply, (3) It converges quickly. Jaya algorithm has the potential to effectively solve the feature selection problem with the mentioned advantages, this is the main motivation behind this work.

Li and Yang 2017 and Chaudhuri and Sahu 2020a have applied the Jaya algorithm for feature selection. However, their effort was limited to the specific application area only. Awadallah et al. (Awadallah et al. 2020) have proposed a Jaya based feature selection technique called BJAM. They have used an S-shaped transfer function for binarization of solution space and single bit-flip based on an adaptive mutation operator for improving the exploration capability of the basic Jaya algorithm. This work proposes three new feature selection techniques based on the Jaya algorithm. The first two techniques are based on S-shaped and V-shaped transfer functions they are named BJaya-S and BJaya-V respectively. The third feature selection technique namely BJaya-JS works only in binary search space hence no transfer function is required for binarization. Distance between solutions is measured on the basis of Jaccard similarity. Furthermore, a local search technique is also proposed to enhance the balance between exploration and exploitation in BJaya-JS and thus finding the global optimal solution faster. The proposed local search technique uses three neighbours instead of one i.e. efficient and fast way to utilize the search space. The main aim and goal of this work are discussed in the next subsection.

1.1 Goals

The comprehensive goal of this work is to develop a new wrapper feature selection technique based on Jaya algorithm Venkata Rao (2016). Jaya algorithm works in continuous space, but feature selection is a discrete problem so we need to convert the continuous space into discrete space. Two different approaches have been applied in this work for this conversion: (1) logistic transfer functions and (2) similarity-based approach. Further, a neighborhood search mechanism is introduced in order to improve the solutions and rate of convergence. The contributions of the current work are as follows:

1. A new wrapper feature selection technique is proposed based on Jaya algorithm
2. Three variants of the binary Jaya algorithm have been proposed
3. A neighborhood search mechanism is proposed
4. The performance of all the variants is evaluated on the basis of different performance metrics over 18 benchmark datasets
5. The performance of the best variant is compared with other popular wrapper feature selection techniques over 18 benchmark datasets and improvement in results are obtained the majority of the times

1.2 Organization of this paper

The rest of the paper is organized as follows: Section 2 describes the background of Jaya Algorithm and Jaccard binary similarity index, Sect. 3 explains the proposed approach in detail. Experiments and results are described in Sect. 4. Section 5 concludes the paper and the references used in this study are mentioned after Sect. 5.

2 Background

2.1 Jaya algorithm

Venkata Rao (2016) proposed a population-based meta-heuristic algorithm named Jaya in 2016. Unlike other popular meta-heuristic algorithms, the Jaya algorithm does not require any algorithm-specific parameter tuning. This saves a lot of effort because parameter tuning is a complex task. Jaya algorithm assures that, the solution should move towards the best solution and away from the worst solution. At every iteration of the algorithm, the best and worst solutions are tracked down from the population. Jaya algorithm updates the solution of its population based on the best and the worst solutions as given in Eq. (1).

$$X_{k,j}^{new} = X_{k,j}^{current} + r1 * \left(Best_j - \left|X_{k,j}^{current}\right|\right) - r2 * \left(Worst_j - \left|X_{k,j}^{current}\right|\right) \tag{1}$$

The problem has D dimensions and population size of N. The k^{th} solution of the population is represented by X_k. $X_{(k,j)}^{current}$ represents the current value of j^{th} dimension of k^{th} solution. The j^{th} dimension of best and worst solution of the population is represented by $Best_j$ and $Worst_j$ respectively; $r1$ and $r2$ denote the random numbers generated in the range [0, 1]. The flow chart of the Jaya algorithm is shown in Fig. 1.

2.2 Jaccard binary similarity index

Distance between binary vectors can be calculated using any binary distance measure. There are so many binary distance measures available in the literature (Choi et al. 2010). Jaccard similarity index (Jaccard 1912) is the most widely used binary similarity index to measure the binary distance between two binary vectors.

The Jaccard similarity between the two binary vectors $X_i = \{x_{i1}, x_{i2}, x_{i3} \ldots x_{iD}\}$ and $X_j = \{x_{j1}, x_{j2}, x_{j3} \ldots x_{jD}\}$ is defined as:

$$Similarity(X_i, X_j) = \frac{B_{11}}{B_{11} + B_{10} + B_{01}} \tag{2}$$

Here B_{11} = number of bits where $x_{id} = x_{jd} = 1$; B_{10} = number of bits where $x_{id} = 1$ and $x_{jd} = 0$; B_{01} = number of bits where $x_{id} = 0$ and $x_{jd} = 1$. Intuitively measure of dissimilarity between two binary vectors is given as:

$$Dissimilarity(X_i, X_j) = 1 - Similarity(X_i, X_j) = 1 - \frac{B_{11}}{B_{11} + B_{10} + B_{01}} \tag{3}$$

Dissimilarity ranges between 0 and 1 i.e. $0 \le Dissimilarity(X_i, X_j) \le 1$. When two vectors are exactly same their dissimilarity will be 0.

3 The proposed binary Jaya algorithm

As mentioned in Rao (2019), the Jaya optimization algorithm produces competitive results as compared to other well-known optimization algorithms, without any need for tuning algorithm-specific parameters. Jaya is able to avoid premature convergence and local minima. Jaya algorithm balances the trade-off between exploration and exploitation

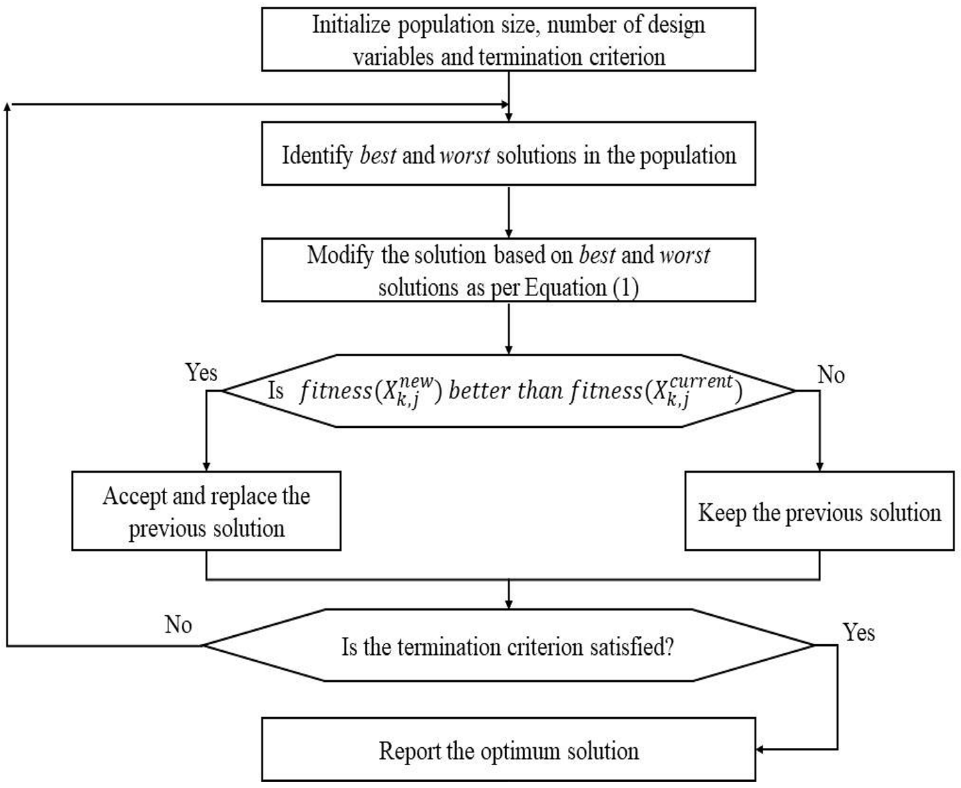
Initialize population size, number of design variables and termination criterion
Identify *best* and *worst* solutions in the population
Modify the solution based on *best* and *worst* solutions as per Equation (1)
Is $fitness(X_{k,j}^{new})$ better than $fitness(X_{k,j}^{current})$
Yes
No
Accept and replace the previous solution
Keep the previous solution
Is the termination criterion satisfied?
No
Yes
Report the optimum solution

Fig. 1 Flow chart of Jaya algorithm (Venkata Rao 2016)

very well. The Jaya algorithm updates the solution on the basis of both, the best and the worst solution. This update mechanism allows the algorithm to reach the optimal solution in less amount of time. These attractive properties motivate us to use the Jaya algorithm for wrapper-based feature selection.

Finding the best feature subset is a challenging task, especially in wrapper-based feature selection. The reason behind this is that each feature subset under consideration needs to be evaluated using a classifier, in each iteration of optimization algorithm. This makes the task computationally very expensive. Therefore, a simple yet powerful and fast converging optimization algorithm is needed. The Jaya algorithm has all these properties, hence it is used in this work.

The original Jaya algorithm (Rao 2019), operates in continuous space. The solutions are updated using Eq.(1). However, feature selection operates in discrete space. The solution of a feature selection problem is a binary vector, where 1 means that the corresponding feature is selected and 0 means the corresponding feature is not selected. This procedure is shown in Fig. 2. Hence, due to the discrete nature of the problem, Eq.(1) can not be used directly to update the solutions. Therefore, we need some mechanism to convert the continuous version of the Jaya algorithm to Binary Jaya algorithm. We have proposed two approaches for this conversion. Approach 1 is based on transfer functions, whereas approach 2 is based on Jaccard binary similarity

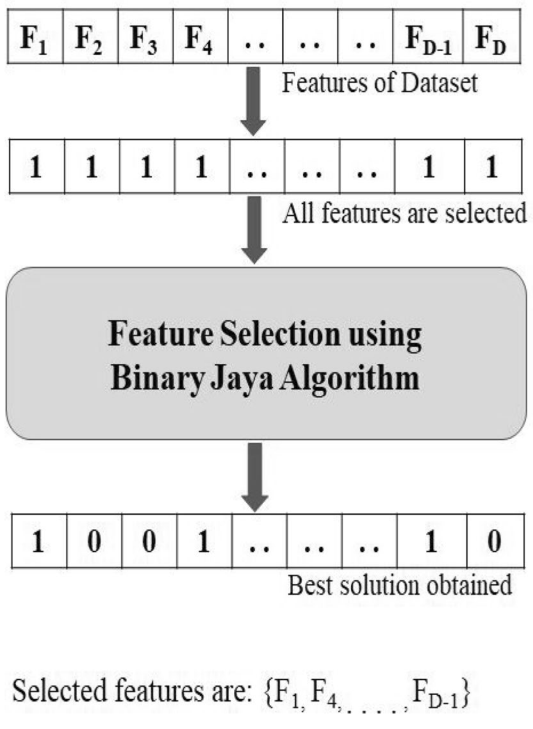
F_1 F_2 F_3 F_4 .. .. .. F_{D-1} F_D
Features of Dataset
1 1 1 1 .. .. .. 1 1
All features are selected
Feature Selection using Binary Jaya Algorithm
1 0 0 1 .. .. .. 1 0
Best solution obtained
Selected features are: $\{F_1, F_4, \ldots, F_{D-1}\}$

Fig. 2 Feature selection using binary Jaya algorithm for a dataset with *D* features. A 1 in the feature vector shows that the corresponding feature is selected and 0 means the corresponding feature is not selected

index. These approaches are described in the following two subsections.

3.1 Binary Jaya optimization algorithm—approach 1 (BJaya-S/BJaya-V)

In the Jaya algorithm solutions are updated according to Eq.(1) in continuous space. In binary Jaya algorithm continuous to binary space conversion need to be done. According to Mirjalili and Lewis (2013), use of transfer function is the easiest way for this conversion. The probability of updation of a solution element from 1 to 0 and vise versa, is defined by the transfer function. In this approach, a sigmoidal (S-shaped) transfer function is used. The sigmoidal transfer function is defined in Eq.(4)

$$S\left(X_j^{new}(t)\right) = \frac{1}{1 + e^{-X_j^{new}(t)}} \tag{4}$$

Where $X_j^{new}(t)$ is the continuous solution at t^{th} iteration. Now to convert its value to discrete (binary) space, a threshold function as described in Eq.(5) is used.

$$X_j^{new}(t+1) = \begin{cases} 0 & \text{if } rand1 < S\left(X_j^{new}(t)\right) \\ 1 & \text{if } rand1 \geq S\left(X_j^{new}(t)\right) \end{cases} \tag{5}$$

Where $rand1$ is a random number drawn from uniform distribution $\in [0, 1]$.

Hyperbolic-tan function is the V-shaped transfer function proposed by Rashedi et al., it is described in Eq.(6). This V-shaped function is also used in this work for continuous to discrete conversion.

$$S\left(X_j^{new}(t)\right) = \left|\tanh\left(X_j^{new}(t)\right)\right| \tag{6}$$

Now the following threshold function, described in Eq.(7) will be used for conversion, in case of the V-shaped transfer function.

$$X_j^{new}(t+1) = \begin{cases} \neg X_j^{new}(t) & \text{if } rand2 < S\left(X_j^{new}(t)\right) \\ X_j^{new}(t) & \text{if } rand2 \geq S\left(X_j^{new}(t)\right) \end{cases} \tag{7}$$

Where $rand2$ is a random number drawn from the uniform distribution $\in [0, 1]$. The pseudo-code of the approach-1 is mentioned in Algorithm 1.

Algorithm 1: Pseudocode of Binary Jaya algorithm - approach 1 (BJaya-S/BJaya-V)

```
1  begin
2      Set the population size $N$, the maximum number of iterations $max\_iter$ and the
       dimension of the population $D$
3      Randomly generate initial poputation $X_i = \{x_{i1}, x_{i2}, x_{i3}...x_{iD}\}$ where $i = 0, 1, 2, 3, ...N$
4      while $iteration \leq max\_iter$ do
5          Evaluate each solution in the population using fitness function
6          Identify $Best_i$ and $Worst_i$
7          Determine $X_i^{new}$ using Eq. (1)
8          Convert $X_i^{new}$ into binary using Eq. (4) and Eq.(5) for all solutions //BJaya-S
9          OR
10         Convert $X_i^{new}$ into binary using Eq. (6) and Eq.(7) for all solutions //BJaya-V
11         if $fitness(X_i^{new}) \geq fitness(X_i^{current})$ then
12             $X_i^{current} = X_i^{new}$
13     Return the best solution $Best_i$
```

3.2 Binary Jaya optimization algorithm—approach 2 (BJaya-JS)

Approach 2 is based on a binary similarity measure named Jaccard Similarity, hence the name BJaya-JS. All the steps of BJaya-JS are described in detail in the following paragraphs.

Initialization The initial population for Jaya algorithm is generated randomly in discrete uniform distribution. Each solution X_i in the population is a binary vector of size D, where D is the dimensionality of the problem. A solution X_i is represented as $X_i = \{x_{i1}, x_{i2}, x_{i3} ... x_{iD}\}$, where $i = 0, 1, 2, ...N$, N is the population size.

Population Update In each iteration of the algorithm, all the solutions need to be updated. Solutions are updated using Eq. (1) in the Jaya algorithm. Our proposed method works in binary search space therefore Eq. (1) needs to be rewritten accordingly. Equation (1) can be represented as:

$$X_i^{new} = X_i^{current} + r1 * \left(Best_i - X_i^{current}\right) - r2 * \left(Worst_i - X_i^{current}\right) \tag{8}$$

Where $X_i^{current}$ is the current solution vector, X_i^{new} is the updated solution vector, $Best_i$ and $Worst_i$ represents the binary vectors corresponding to best and worst solutions respectively. The above equation can be rewritten as:

$$\begin{aligned} X_i^{new} &- X_i^{current} \\ &= r1 * \left(Best_i - X_i^{current}\right) - r2 * \left(Worst_i - X_i^{current}\right) \end{aligned} \tag{9}$$

The difference between two binary vectors is represented by the dissimilarity between them. Here we are using the Jaccard dissimilarity measure. Above Eq.(9) can be rewritten as:

$$\begin{aligned} &Dissimilarity\left(X_i^{new}, X_i^{current}\right) \approx r1 * Dissimilarity\left(Best_i, X_i^{current}\right) - \\ &\quad r2 * Dissimilarity\left(Worst_i, X_i^{current}\right) \end{aligned} \tag{10}$$

According to the above equation dissimilarity between X_i^{new} and $X_i^{current}$ should be approximately equal to the right-hand side of the equation. Therefore the above equation can be written as

$$\begin{aligned} \mathbf{Minimize}\{&Dissimilarity\left(X_i^{new}, X_i^{current}\right) \\ &- [r1 * Dissimilarity\left(Best_i, X_i^{current}\right) \\ &- r2 * Dissimilarity\left(Worst_i, X_i^{current}\right)]\} \end{aligned} \tag{11}$$

Let us represent the term in square brackets as **Target** for simplicity in discussion. The variable *Target* signifies that the solution should move towards the best solution and at the same time away from the worst solution.

$$\begin{aligned} Target = [&r1 * Dissimilarity\left(Best_i, X_i^{current}\right) \\ &- r2 * Dissimilarity\left(Worst_i, X_i^{current}\right)] \end{aligned} \tag{12}$$

Now Eq.(11) can be re written as:

$$\mathbf{Minimize}\{Dissimilarity\left(X_i^{new}, X_i^{current}\right) - Target\} \tag{13}$$

Dissimilarity between $\left(Best_i, X_i^{current}\right)$ and $\left(Worst_i - X_i^{current}\right)$ can easily be computed using Eq.(3), after this computation the term *Target* will become a neumerical term. Now, in order to evaluate Eq.(12) we need to compute $Dissimilarity\left(X_i^{new}, X_i^{current}\right)$. The vector $X_i^{current}$ is known and the value of vector X_i^{new} is needed. The value of vector X_i^{new} should be generated in such a manner so that it's dissimilarity with vector $X_i^{current}$ should be close to *Target*. We need a procedure to generate the new solution vector X_i^{new} which is described in the next paragraph.

New Solution Generation The New Solution Generator (NSG) generates the new solution X_i^{new} . NSG receives two inputs n and P_{NS}. The variable n represents the number of candidate solutions to be tried. Out of these n candidate solutions, the solution which gives minimum value as per Eq.(12) will be selected as X_i^{new}. The NSG procedure is described in Algorithm 2. Candidate solutions are generated in two ways:

- Neighborhood search
- Random solution generation

In the Neighborhood search method, three neighbours are used in order to generate a mutant solution. Instead of one bit, randomly selected three bits of the current solution $X_i^{current}$ are toggled (changed from 1 to 0 or 0 to 1). Since this method explores the neighboring search space of current solution, it is called the Neighborhood search method. n candidate solutions are generated using this method. In the Random solution generation method, n candidate solutions are generated randomly. If only the neighborhood search method will be used to generate new solution, there is a chance of getting stuck in local minima or pre-mature convergence. In order to avoid this, a variable P_{NS} is introduced which takes care of exploration whereas, neighborhood search represents exploitation.

Algorithm 2: New_Solution_Generator

```
   Input: $n$, $P_{NS}$
   Result: New solution $X_i^{new}$
 1 if $rand \leq P_{NS}$ then
 2     for $j = 0$ to $n$ do
 3         Randomly generate 3 indexes $k, l$ and $m$
 4         Generate a new solution $X'$ by toggling bits of $k^{th}$, $l^{th}$ and $m^{th}$ position of
             $X_i^{current}$
 5         candidate_solution[j]= $X'$
 6 else
 7     Generate $n$ random solutions and add them to candidate_solution
 8 for $j = 0$ to $n$ do
 9     temp[j] = $Dissimilarity(candidate\_solution[j], X_i^{current})$ - $Target$
10 $X_i^{new} = j^{th}$ candidate_solution which have minimum value of temp[j] return $X_i^{new}$
```

The variable P_{NS} represents the probability of using the neighborhood search method for generating candidate solutions and it's value ranges between 0 to 1. In other words, this variable decides the rate of random solutoin generation. Whenever the value of *rand* is less than P_{NS} the algorithm exploits the solution in the neighborhood that are known to be good solutions so far and when the value of *rand* is greater than P_{NS} the algorithm explores the new random solutions. The use of random solution generation avoids being stuck in local minima. Hence the variable P_{NS} is responsible to maintain the balance between exploration and exploitation. The selection of the value of the variable P_{NS} is an important task because if the value of P_{NS} is too big there is a chance of being stuck in local optima on the other hand, if the value is too small it leads to random search.

The variable *Target* from Eq. (12) guides the selection of one solution from the n candidate solutions. The solution, whose dissimilarity with the current solution $X_i^{current}$ is closest to the value *Target* is chosen as the new solution X_i^{new}.

In this way, NSG ensures that the current solution is updated in such a way that the new solution moves towards the best solution and at the same time moves away from the worst solution. The pseudo-code of the proposed approach is given in Algorithm 3.

The computational time complexity of the BJaya-JS in the best case and worst case is equal to $O(I(d \times N + Cof \times N + n))$ where I is the number of iterations in BJaya-JS, d is the dimensionality of the problem to be solved, N is the population size, Cof is the complexity of the fitness function, and n is the number of candidate solutions to be tried by the New Solution Generator. The best case and average case time complexity of the BJaya-S and BJaya-V is equal to $O(I(d \times N + Cof \times N))$.

3.3 Binary Jaya algorithm for feature selection

Feature selection problem is a binary optimization problem. The solution obtained from the binary optimization

Algorithm 3: Pseudocode of Binary Jaya algorithm - approach 2 (BJaya-JS)

```
 1 begin
 2     Set the population size $N$ and the dimension of the population $D$
 3     Set the maximum number of iterations $max\_iter$
 4     Set the neighborhood search probability $P_{NS}$
 5     Set the number of candidate solutions $n$
 6     Randomly generate initial poputation $X_i = \{x_{i1}, x_{i2}, x_{i3}...x_{iD}\}$ where
         $i = 0, 1, 2, 3, ...N$
 7     while $iteration \leq max\_iter$ do
 8         Evaluate each solution in the population using fitness function
 9         Identify $Best_i$ and $Worst_i$
10         foreach $i \leq N$ do
11             $X_i^{new} = New\_Solution\_Generator(n, P_{NS})$ //Refer algorithm 2
12             if $fitness\,(X_i^{new}) \geq fitness\,(X_i^{current})$ then
13                 $X_i^{current} = X_i^{new}$
14     Return the best solution $Best_i$
```

algorithm for FS is a one-dimensional feature vector that can have binary values i.e. only 1 or 0. The size of the feature vector is equal to the dimensionality of the problem. For D dimensional problem feature vector will have D columns and one row. A 1 in the feature vector represents that the corresponding feature is selected and vice versa. The concept of the feature vector is clearly depicted in Fig. 2.

Feature selection problem has two objectives. The first objective is to improve classification accuracy by selecting the important features. Whereas, the second objective is to select the minimum number of features. These two conflicting objectives are taken care of by using a fitness function described in Eq. (13)

$$Fitness(X) = \alpha\lambda_R(X) + \beta\frac{|X|}{|D|} \tag{14}$$

Where λ_R is the classification error rate, $|X|$ represents the number of features selected by the solution X, $|D|$ represents total number of features in the dataset, α and β are two constant parameter that decide the degree of importance of the two objectives of the FS problem. Values of α and β are in range [0,1], the relationship between α and β i.e. $\beta = (1 - \alpha)$ should be maintained. In this work, value of $\alpha = 0.99$ and $\beta = 0.01$ are adapted as per the standard found in literature (Mafarja et al. 2018).

4 Experimental results and discussion

Implementations of the algorithms are done in Python 3.7; the Naïve Bayes classifier is used for the wrapper-based feature selection. Simulations are executed on an Intel Core i5-8265U-1.6 GHz CPU with 8GB of RAM. The 10-fold cross-validation for feature selection is applied to each dataset in the same manner as majority of papers in this field (Neggaz et al. 2020; Ibrahim et al. 2019; Aljarah et al. 2018). In order to get statistically significant results, each algorithm is executed 30 times on each dataset and the average of these 30 runs is reported in this work. For fair comparison population size is set 30 and number of iterations is kept at 100 for all the algorithms for all the experiments. The results (accuracy, fitness and number of features selected) reported here are based on test data.

4.1 Experimental design

Experiments are performed in two phases. In phase-I, we investigate the best variant of the binary Jaya algorithm from Sect. 3 and in phase-II we compare the proposed approach with other competitive wrapper feature selection methods available in the literature.

Table 1 Parameter values

Algorithm	Parameter	Values
BPSO	Inertia of particle	0.9
	$c1$	0.5
	$c2$	0.5
BCSA	Flight length fl	2
	Awareness Probability AP	0.1
BDFO	Separation s	0.1
	Alignment a	0.1
	Cohesion c	0.7
	Attraction f	1
	Distraction e	1
GA	Crossover probability	0.8
	Mutation Probability	0.02
BGWO	a	[2, 0]
BGOA	c_Max	1
	c_Min	0.00004
BJaya-JS	Number of candidate solutions n	5
	Neighborhood search probability P_{NS}	0.9

Table 2 Summary of datasets

S. No.	Dataset	Number of Features	Number of classes	Number of Instances
1	Diabetes	8	2	786
2	Contraceptive	9	3	1473
3	Glass	9	7	214
4	Breast cancer	10	2	683
5	Vowel	13	11	990
6	Australian	14	2	690
7	Zoo	16	7	101
8	SPECT	22	2	160
9	WDBC	31	2	569
10	Inosphere	33	2	351
11	Sonar	60	2	208
12	Semeion	265	2	1593
13	Madelon	500	2	4400
14	PD speech	754	2	756
15	CANE9	856	9	1080
16	Colon	2000	2	62
17	CNS	7129	2	60
18	Lung	12533	2	181

4.1.1 Phase-I

In the first phase, we compare the approaches based on the Jaya algorithm in order to find out the best approach. We have proposed three approaches based on the Jaya algorithm to address the feature selection problem, as mentioned in Sect. 3. These three approaches have been compared on the basis of metrics defined in sub-Sect. 4.3, the best approach is identified and then it is further compared with other meta-heuristic approaches, as mentioned in the next sub-section.

4.1.2 Phase-II

In this phase, the best approach identified from *phase-I* is compared with other well-regarded metaheuristic approaches. Comparison is done with Binary Particle Swarm Optimization Algorithm (BPSO) (Mirjalili and Lewis 2013), Binary Crow Search Algorithm (BCSA) (De Souza et al. 2018), Binary Dragonfly Optimization Algorithm (BDFO) (Mafarja et al. 2017b), Genetic Algorithm (GA) (Kabir et al. 2011), Binary Grey Wolf Optimization Algorithm (BGWO) (Emary et al. 2016) , and Binary Grasshopper Optimization

Table 3 Average accuracy and fitness comparison among Jaya based methods

		Accuracy			Fitness		
		BJaya-V	BJaya-S	BJaya-JS	BJaya-V	BJaya-S	BJaya-JS
Diabetes	Avg	0.8740	0.8701	**0.8823**	0.1461	0.1572	**0.1162**
	Std	0.6289	0.9647	1.1277	0.0000	0.0132	0.0133
Contraceptive	Avg	0.6123	0.6090	**0.6198**	0.3845	0.3970	**0.3628**
	Std	1.4548	0.4300	1.3603	0.0205	0.0067	0.0155
Glass	Avg	0.7884	0.7543	**0.8078**	0.2407	0.2778	**0.2068**
	Std	3.5269	1.9962	1.6081	0.0239	0.0175	0.0202
Breast cancer	Avg	**1.0000**	0.9998	**1.0000**	0.0411	0.0604	**0.0001**
	Std	0.0000	0.1333	0.0000	0.0042	0.0017	0.0000
Vowel	Avg	0.7586	0.7702	**0.7966**	0.2671	0.2843	**0.2274**
	Std	0.7377	1.1770	1.0222	0.0038	0.0125	0.0115
Australian	Avg	0.9543	**0.9623**	0.9510	0.0677	0.0982	**0.0499**
	Std	0.6633	0.5515	0.3117	0.0194	0.0050	0.0058
Zoo	Avg	**1.0000**	**1.0000**	**1.0000**	0.0750	0.0688	**0.0079**
	Std	0.0000	0.0000	0.0000	0.0000	0.0000	0.0052
SPECT	Avg	0.9615	0.9677	**0.9906**	0.0767	0.1063	**0.0293**
	Std	1.7760	0.5705	1.4565	0.0174	0.0000	0.0144
WDBC	Avg	**0.9991**	0.9982	0.9988	0.0271	0.0559	**0.0246**
	Std	0.2677	0.4248	0.3033	0.0025	0.0027	0.0040
Inosphere	Avg	0.9944	**0.9972**	0.9901	0.0368	0.0777	**0.0330**
	Std	0.7018	0.5730	0.7535	0.0081	0.0039	0.0088
Sonar	Avg	0.9429	0.9310	**0.9579**	0.0857	0.1178	**0.0595**
	Std	2.1372	1.9119	1.7237	0.0157	0.0175	0.0174
Semeion	Avg	**1.0000**	**1.0000**	**1.0000**	0.0525	0.0675	**0.0431**
	Std	0.0000	0.0000	0.0000	0.0000	0.0000	0.0013
Madelon	Avg	0.6547	0.6627	**0.6885**	**0.3150**	0.3672	0.3250
	Std	0.4526	0.5406	0.0000	0.0010	0.0024	0.0010
PD speech	Avg	**0.8776**	0.8691	0.8732	**0.1536**	0.1831	0.1597
	Std	0.3397	0.8168	0.5164	0.0044	0.0063	0.0054
CANE9	Avg	0.9315	0.9231	**0.9387**	0.1304	0.1265	**0.1148**
	Std	0.7808	0.7463	1.0847	0.0076	0.0068	0.0114
Colon	Avg	0.9800	0.8928	**0.9993**	0.0598	0.1530	**0.0480**
	Std	0.6875	0.0234	0.3651	0.0020	0.0312	0.0036
CNS	Avg	0.7666	0.7426	**0.9041**	0.2581	0.2920	**0.1353**
	Std	0.9562	0.5856	2.0126	0.0002	0.0034	0.0184
Lung	Avg	0.9071	0.8688	**0.9878**	0.1427	0.1825	**0.0607**
	Std	0.0054	0.4461	0.0043	0.0002	0.0048	0.0001

Bold values indicate the best results

Table 4 Best accuracy and fitness comparison among Jaya based methods

	Accuracy			Fitness		
	BJaya-V	BJaya-S	BJaya-JS	BJaya-V	BJaya-S	BJaya-JS
Diabetes	**0.8766**	**0.8766**	**0.8766**	**0.1461**	0.1969	**0.1461**
Contraceptive	0.6237	0.6102	**0.6373**	0.4156	0.4217	**0.3864**
Glass	0.8140	0.7674	**0.8605**	0.2747	0.3067	**0.2204**
Breast cancer	**1.0000**	**1.0000**	**1.0000**	0.0566	0.0666	**0.0000**
Vowel	**0.7980**	0.7778	0.7778	0.2874	0.3301	**0.2612**
Australian	0.9565	**0.9638**	0.9493	0.1009	0.1165	**0.0801**
Zoo	**1.0000**	**1.0000**	**1.0000**	0.0750	0.0688	**0.0250**
SPECT	0.9688	0.9688	**1.0000**	0.1253	**0.1063**	0.1116
WDBC	**1.0000**	**1.0000**	**1.0000**	**0.0337**	0.0674	0.0402
Inosphere	**1.0000**	**1.0000**	**1.0000**	**0.0466**	0.0854	0.0587
Sonar	0.9524	0.9524	**0.9762**	**0.1224**	0.1821	**0.1224**
Semeion	**1.0000**	**1.0000**	**1.0000**	0.0525	0.0675	**0.0479**
Madelon	0.6668	0.6692	**0.6750**	**0.3437**	0.3709	0.3572
PD speech	**0.8816**	0.8750	0.8750	**0.1571**	0.1945	0.1768
CANE9	0.9352	0.9259	**0.9491**	0.1476	**0.1467**	0.1643
Colon	0.9800	0.8743	**1.0000**	0.0589	0.1530	**0.0471**
CNS	0.7778	0.7593	**0.9166**	0.2580	0.2822	**0.1238**
Lung	0.9071	0.8783	**0.9878**	0.1427	0.1769	**0.0606**

Bold values indicate the best results

Table 5 Comparison on the basis of average feature selection ratio for Jaya based methods

	BJaya-V	BJaya-S	BJaya-JS		BJaya-V	BJaya-S	BJaya-JS
Diabetes	0.6250	**0.3750**	0.8750	**Inosphere**	0.7273	0.7576	**0.5152**
Contraceptive	0.5556	**0.4444**	0.5556	**Sonar**	0.7500	0.6333	**0.5333**
Glass	0.5556	0.6667	**0.4444**	**Semeion**	0.7396	0.7396	**0.4679**
Breast cancer	**0.6000**	0.8000	0.7000	**Madelon**	0.5420	0.7120	**0.5400**
Vowel	0.8462	0.6923	**0.3846**	**PD speech**	0.6419	0.6618	**0.5172**
Australian	**0.6429**	**0.6429**	0.7143	**CANE9**	0.6752	0.6682	**0.5152**
Zoo	0.7500	0.7500	**0.3750**	**Colon**	**0.4685**	0.5660	0.4800
SPECT	0.6364	0.8182	**0.5909**	**CNS**	0.6844	0.6558	**0.4973**
WDBC	0.5161	0.6452	**0.2903**	**Lung**	0.5951	0.6418	**0.5012**

Bold values indicate the best results

Algorithm (BGOA) (Mafarja et al. 2019). The performance is evaluated on the evaluation criteria mentioned in sub-section 4.3. Experiments have been conducted to determine the optimal values of parameters n and P_{NS} which give the best fitness in reasonable computational cost. It has been found that larger values of n made the algorithm slow without any significant improvement in fitness value and very small values of n were not able to provide the diversity in candidate solutions; $n = 5$ is giving the optimal result. The value for parameter $P_{NS} = 0.9$ is mainataining the balance between exploration and exploitation very well. Parameter of different algorithms used here are selected using trial and error method and the best setting obtained for each algorithm is given in Table 1.

4.2 Dataset description

The performance of the proposed algorithm is evaluated on 18 benchmark datasets for feature selection. All the datasets are collected from the UCI repository. Number of features in the datasets ranges from 8 to 12533 and the instances also ranges from 100 to 4500. This variation in number of features and number of instances poses different challenges on classifiers and feature selection algorithms. These datasets are carefully chosen to test all sorts of challenges faced by the feature selection algorithm. The summary of the datasets used in this work is given in Table 2.

Table 6 Average accuracy comparison with other competitor algorithms

		BPSO	BCSA	BDFO	GA	BGWO	BGOA	BJaya-JS
Diabetes	Avg	0.8766	0.8768	0.5720	0.8816	0.8286	0.7856	**0.8823**
	Std	0.0000	0.8776	0.0000	0.2212	0.9776	0.0127	1.1277
Contraceptive	Avg	0.5945	0.6193	**0.6277**	0.6102	0.5936	0.5951	0.6198
	Std	0.2085	1.3118	0.2940	0.9292	1.1462	0.0123	1.3603
Glass	Avg	0.7798	0.7659	0.7686	0.7760	**0.8907**	0.7128	0.8078
	Std	1.3287	1.0460	1.2910	1.4369	2.2062	0.0213	1.6081
Breast cancer	Avg	**1.0000**	**1.0000**	0.9748	**1.0000**	**1.0000**	0.9844	**1.0000**
	Std	0.0000	0.0000	0.1947	0.0000	0.0000	0.0061	0.0000
Vowel	Avg	0.7444	0.7739	0.7685	0.7712	0.7865	0.4557	**0.7966**
	Std	1.1065	0.4906	0.7640	0.6273	0.2440	0.0119	1.0222
Australian	Avg	0.9466	**0.9551**	0.8734	0.9470	0.9370	0.7525	**0.9510**
	Std	0.3552	0.5515	0.0000	0.4281	0.7676	0.0135	0.3117
Zoo	Avg	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.6001	**1.0000**
	Std	0.0000	0.0000	0.0000	0.0000	0.0000	0.0005	0.0000
SPECT	Avg	0.9635	**0.9948**	0.8436	0.9192	0.9688	0.7507	0.9906
	Std	1.1845	1.1845	0.6534	0.9190	1.4731	0.0248	1.4565
WDBC	Avg	0.9947	0.9953	0.9392	0.9982	0.9982	0.8382	**0.9988**
	Std	0.4530	0.5012	0.5498	0.4248	0.3699	0.6018	0.3033
Inosphere	Avg	0.9925	**0.9972**	0.9623	0.9962	0.9831	0.8323	0.9901
	Std	1.2118	0.6820	0.2099	0.6059	1.1110	0.0157	0.7535
Sonar	Avg	0.9190	0.9381	0.7212	0.8296	0.9388	0.7802	**0.9579**
	Std	1.1864	1.1864	0.8444	1.0154	1.2549	0.0202	1.7237
Semeion	Avg	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
	Std	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Madelon	Avg	0.6594	0.6592	0.6328	0.6627	0.6721	**0.7732**	0.6885
	Std	0.6503	0.5424	0.5235	0.5406	0.7916	0.0648	0.0000
PD speech	Avg	0.8632	0.8500	0.8547	0.8691	0.8572	0.7953	**0.8732**
	Std	1.2328	0.6795	0.6854	0.8168	1.2031	0.5498	0.5164
CANE9	Avg	0.8796	0.9176	0.9277	**0.9387**	0.8935	0.8657	**0.9387**
	Std	1.2726	0.5691	0.7328	0.7463	1.4962	0.7623	1.0847
Colon	**Avg**	0.9489	0.9776	0.9925	0.9962	0.9982	0.9897	**0.9993**
	Std	0.0061	0.0054	1.2118	0.6059	0.3325	0.8741	0.3651
CNS	**Avg**	0.8896	0.8396	0.8646	0.8968	0.8947	**0.9173**	0.9041
	Std	4.0254	2.5743	3.2998	3.0190	4.6258	2.4812	2.0126
Lung	**Avg**	0.9798	0.8765	0.9282	0.9838	0.9776	0.9427	**0.9878**
	Std	0.0054	0.0876	0.0465	0.0049	0.3425	0.5424	0.0043

Bold values indicate the best results

4.3 Evaluation criteria

This section shows the metrics used for the performance evaluation of the proposed approach. In order to get statistically significant results, 30 independent runs of each approach are considered. Following metrics are used for quantitative evaluation purpose:

– Average accuracy and standard deviation over 30 independent runs
– Average fitness and standard deviation over 30 independent runs
– Average feature selection ratio
– Average computational time over 30 independent runs
– Statistical best accuracy obtained in 30 independent runs of algorithm

Further, to ensure the statistical significance of the results the pair-wise Wilcoxon signed-rank test at significance level 95% is applied on all the algorithms for each of the datasets. The *P*-values obtained from the comparison of average accuracies of BJaya-JS with all other competitor algorithms for each of the 18 datasets using the Wilcoxon signed-rank test are listed in Table 12.

In order to evaluate the overall performance of BJaya-JS with other algorithms across all the datasets, Wilcoxon text is again applied on the average accuracies listed in Table 6 and the obtained P-values are listed in Table 13. Furthermore, we have performed the Friedman test which is used to compare the performance of multiple methods simultaneously. The null hypothesis H_0 of the Friedman test states that there is no difference between the variables. The null hypothesis is rejected as we have got P-value less than 0.05. This indicates that the results are statistically significant. The result of the Friedman test is also listed in Table 13.

4.4 Numerical results and discussion

4.4.1 Comparison between Jaya based approaches

This subsection discusses the results of three Jaya based feature selection approaches. The experimental results are shown in Tables 3, 4, 5. The results in terms of average classification accuracy, average fitness and their respective standard deviation, over 30 independent runs are shown in Table 3. The bold values indicate the best results.

It is evident from Table 3 that Jaccard Similarity-based approach i.e. BJaya-JS outperforms the other two methods fourteen times, in terms of average classification accuracy. Transfer function based approaches i.e. BJaya-V and BJaya-S give better results only five and four times respectively.

BJaya-JS achieves the best average fitness sixteen out of eighteen times. Whereas, BJaya-V achieved the best fitness only two times. Analysis of results in terms of average accuracy and average fitness implies that the Jaccard similarity based approach outperforms transfer function based approaches. Further, the standard deviation is also inspected, and it is observed that BJaya-JS have consistently small values of standard deviation. This indicates that the performance of the algorithm is consistent throughout the different datasets.

Best accuracy and best fitness obtained by the three Jaya based FS approaches, over the 30 independent runs are listed in Table 4. BJaya-JS secured first place by outperforming other methods on 15 datasets in terms of *best accuracy*. BJaya-V comes second, it has achieved the highest accuracy in 8 datasets. BJaya-S is in the last position as it has performed better in terms of *best accuracy* than other approaches in only 7 datasets. When we compare in terms of *best fitness*, BJaya-JS wins the race twelve times by achieving better results than other methods, BJaya-V and BJaya-S performed well in only 6 and 2 datasets respectively. A careful study of Table 4 suggests that BJaya-JS has shown better performance than other Jaya based methods in terms of *best fitness* and *best accuracy*.

In Table 5, the three Jaya based approaches are compared in terms of average feature selection ratio. BJaya-JS has a minimum average feature selection ratio in 13 datasets. Observation of Tables 3, 4, 5 clearly suggests that among the three Jaya based feature selection approaches, the Jaccard Similarity based approach i.e. BJaya-JS can be declared the best approach because it gives the best result in terms of average accuracy, average fitness, best accuracy, best fitness and average feature selection ratio. With these results, it is

Table 7 Best accuracy comparison with other competitor algorithms

	BPSO	BCSA	BDFO	GA	BGWO	BGOA	BJaya-JS
Diabetes	0.8766	0.8831	0.5720	**0.8901**	0.8506	0.8142	0.8766
Contraceptive	0.5966	**0.6407**	0.6298	0.6287	0.6102	0.6249	0.6373
Glass	0.7907	0.7907	0.7812	0.7728	**0.9302**	0.7389	0.8605
Breast cancer	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9953	**1.0000**
Vowel	0.7525	0.7378	0.7783	0.7893	**1.0000**	0.4752	0.7778
Australian	0.9493	0.9492	0.8734	**0.9521**	0.9493	0.7718	0.9493
Zoo	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.6008	**1.0000**
SPECT	0.9688	**1.0000**	0.8734	0.9234	**1.0000**	0.7985	**1.0000**
WDBC	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Inosphere	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.8582	**1.0000**
Sonar	0.9286	0.9524	0.7485	0.8476	0.9624	0.8248	**0.9762**
Semeion	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Madelon	0.6692	0.6692	0.6428	0.6747	0.6873	**0.7787**	0.6750
PD speech	0.8816	0.8553	0.8675	**0.8908**	0.8618	0.8093	0.8750
CANE9	0.9028	0.9259	0.9290	0.9439	0.9167	0.8862	**0.9491**
Colon	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
CNS	0.9032	0.8553	0.8675	0.9064	0.9071	**0.9389**	0.9166
Lung	0.9873	0.8865	0.9369	0.9875	0.98716	0.9562	**0.9878**

Bold values indicate the best results

Table 8 Average fitness comparison with other competitor algorithms

		BPSO	BCSA	BDFO	GA	BGWO	BGOA	BJaya-JS
Diabetes	Avg	0.1610	0.1206	0.8982	0.1520	0.1580	0.1540	**0.1162**
	Std	0.0000	0.0147	0.0453	0.0300	0.0173	0.0076	0.0133
Contraceptive	Avg	0.4004	0.3581	0.1205	0.2393	**0.0137**	**0.0137**	0.3628
	Std	0.0036	0.0121	0.0140	0.0131	0.0192	0.0079	0.0155
Glass	Avg	0.2068	0.2217	0.2584	0.2401	0.5574	0.3138	**0.2068**
	Std	0.0159	0.0121	0.0211	0.0166	0.0287	0.0069	0.0202
Breast Cancer	Avg	0.0398	0.0006	0.0427	0.0217	0.0180	0.0178	**0.0001**
	Std	0.0000	0.0032	0.0000	0.0016	0.0080	0.0081	0.0000
Vowel	Avg	0.3020	0.2281	0.2532	0.2618	0.0169	**0.0115**	0.2274
	Std	0.0074	0.0047	0.0062	0.0095	0.0094	0.0070	0.0115
Australian	Avg	0.1009	0.0509	0.0556	0.0583	0.0165	**0.0147**	0.0499
	Std	0.0000	0.0112	0.0110	0.0111	0.0148	0.0073	0.0058
Zoo	Avg	0.0750	0.0196	0.0463	0.0456	0.0177	0.0188	**0.0079**
	Std	0.0000	0.0039	0.0043	0.0025	0.0069	0.0069	0.0052
SPECT	Avg	0.0805	**0.0273**	0.0327	0.0545	0.0416	0.0387	0.0293
	Std	0.0158	0.0137	0.0100	0.0120	0.0211	0.0066	0.0144
WDBC	Avg	0.0418	0.0332	0.0246	0.0246	**0.0214**	0.3593	0.0246
	Std	0.0012	0.0025	0.0021	0.0011	0.0117	0.0043	0.0040
Inosphere	Avg	0.0602	0.0323	0.4327	0.2325	0.0416	**0.0215**	0.0330
	Std	0.0094	0.0080	0.0045	0.0063	0.0175	0.0065	0.0088
Sonar	Avg	0.1199	0.0913	0.1924	0.0983	0.0814	0.1164	**0.0595**
	Std	0.0091	0.0120	0.0560	0.0139	0.0164	0.0040	0.0174
Semeion	Avg	0.0537	0.0423	0.0432	0.0467	0.0871	0.0875	**0.0431**
	Std	0.0027	0.0010	0.0012	0.0013	0.0071	0.0047	0.0013
Madelon	Avg	0.3551	0.3477	0.3215	0.3762	0.3904	0.3581	**0.3250**
	Std	0.0068	0.0033	0.0043	0.0042	0.0113	0.0121	0.0010
PD speech	Avg	0.1612	0.1767	0.1657	0.1692	0.1441	**0.1385**	0.1597
	Std	0.0109	0.0012	0.0012	0.0012	0.0019	0.0054	0.0054
CANE9	Avg	0.1443	0.1176	0.1211	0.1343	0.1753	0.1614	**0.1148**
	Std	0.0102	0.0034	0.0034	0.0234	0.0110	0.0023	0.0114
Colon	Avg	0.0490	0.0488	0.0898	0.1052	0.0892	0.0718	**0.0480**
	Std	0.0003	0.0324	0.0453	0.0300	0.0316	0.0101	0.0036
CNS	Avg	0.1496	0.1492	0.1383	0.1553	0.0184	0.0175	**0.1353**
	Std	0.0001	0.0002	0.0040	0.0021	0.0212	0.0088	0.0184
Lung	Avg	0.0489	0.0493	**0.0427**	0.0643	0.0892	0.0718	0.0607
	Std	0.0004	0.0001	0.0000	0.0016	0.0316	0.0101	0.0001

Bold values indicate the best results

clear that the binary Jaya algorithm based on the Jaccard Similarity index is better than the transfer function based approaches. Also, the neighborhood search method along with the probability term P_{NS} is very effective in exploring the binary search space and locating the (near) optimal solution.

4.4.2 Comparison with other meta-heuristic feature selection approaches

This subsection compares the best Jaya based feature selection approach obtained from the previous section with other popular meta-heuristic feature selection approaches. BJaya-JS is compared with BPSO, BCSA, BDFO, GA, BGWO and BGOA. These algorithms are considered state-of-the-art, well-regarded metaheuristic based wrapper feature selection techniques. Naive Bayes classifier is used to evaluate the fitness of solutions according to Eq. (13).

Average accuracy and standard deviation of all the algorithms over 30 independent runs for all the 18 datasets are shown in Table 6. We can see from Table 6 that the proposed approach BJaya-JS produces the best average accuracy for 12 datasets. Whereas, BPSO, BCSA, BDFO, GA, BGWO and

Table 9 Best fitness comparison with other competitor algorithms

	BPSO	BCSA	BDFO	GA	BGWO	BGOA	BJaya-JS
Diabetes	0.1610	0.1661	0.1923	0.1520	**0.0196**	0.0215	0.1461
Contraceptive	0.4136	0.3772	0.3049	0.2593	0.0314	**0.0214**	0.3864
Glass	0.3080	0.2734	0.2684	0.2601	0.5957	0.4314	**0.2204**
Breast cancer	0.0398	0.0175	0.0427	0.0227	0.0232	0.0222	**0.0001**
Vowel	0.3192	0.2476	0.2682	0.2638	0.0272	**0.0172**	0.2612
Australian	0.1002	0.0944	0.0556	0.0596	**0.0265**	0.0315	0.0801
Zoo	0.0775	0.0250	0.0467	0.0576	**0.0218**	0.0219	0.0250
SPECT	0.1153	0.0554	0.0433	0.0657	**0.0472**	0.0439	0.1116
WDBC	0.0434	0.0402	0.0427	0.0425	**0.0321**	0.3959	0.0402
Inosphere	0.0769	0.0617	0.4370	0.2385	0.0462	**0.0352**	0.0587
Sonar	0.1340	0.1093	0.1924	**0.0983**	0.0864	0.2116	0.1224
Semeion	0.0652	**0.0442**	0.0452	0.0538	0.0969	0.0949	0.0479
Madelon	0.3649	**0.3496**	0.3921	0.3862	0.4339	0.4636	0.3572
PD speech	0.1872	0.1796	0.1784	0.1768	0.1844	0.2139	**0.1768**
CANE9	0.1680	0.1261	**0.1147**	0.1534	0.2218	0.1961	0.1643
Colon	**0.0491**	0.0547	0.0938	0.2105	0.0929	0.0827	0.0548
CNS	0.1996	0.1549	0.2334	0.2355	**0.0218**	0.0318	0.2135
Lung	0.0549	0.0535	**0.0427**	0.0694	0.0949	0.0762	0.0651

Bold values indicate the best results

Table 10 Comparison on the basis of average feature selection ratio with other competitor algorithms

	BPSO	BCSA	BDFO	GA	BGWO	BGOA	BJaya-JS
Diabetes	**0.5000**	**0.5000**	0.7500	0.7500	0.6250	0.8750	0.8750
Contraceptive	**0.4444**	0.8889	**0.4444**	0.5556	0.5556	0.6667	0.5556
Glass	**0.4444**	0.5556	0.6667	0.6667	0.5556	0.7778	**0.4444**
Breast cancer	0.7000	0.5000	**0.4000**	0.5000	0.8000	0.7000	0.7000
Vowel	0.9231	0.7692	0.6154	0.6154	0.4615	0.5385	**0.3846**
Australian	0.6429	0.6429	0.4286	0.6429	**0.3571**	0.6429	0.7143
Zoo	0.7500	0.5625	0.8125	0.6250	0.5000	0.5625	**0.3750**
SPECT	0.6364	0.7273	0.6364	0.6818	0.6818	0.8182	**0.5909**
WDBC	0.4194	0.5161	0.6452	0.3226	0.3548	0.4839	**0.2903**
Inosphere	0.5758	0.5455	0.6061	0.6667	0.5758	0.6364	**0.5152**
Sonar	0.4833	0.6500	0.6000	**0.4333**	0.4667	0.5500	0.5333
Semeion	0.5358	0.6377	**0.4679**	0.7396	0.5094	0.6189	**0.4679**
Madelon	0.6420	0.6380	0.5460	0.7120	0.7520	0.5720	**0.5400**
PD speech	0.5252	0.6340	**0.5172**	0.6618	0.5225	0.5663	**0.5172**
CANE9	0.5210	0.6320	**0.5152**	0.6682	0.7617	0.6379	**0.5152**
Colon	0.5040	0.4965	0.5105	0.5620	0.5020	0.4890	**0.4800**
CNS	0.4982	0.6395	0.5169	0.5193	0.5034	0.5047	**0.4973**
Lung	0.5031	0.6409	0.5632	0.5753	0.6137	0.5376	**0.5012**

Bold values indicate the best results

BGOA gives best performance in terms of average accuracy three, six, three, four, three and three datasets respectively.

Further, the small values of standard deviation throughout all the datasets indicate the consistent performance of the proposed approach. In Table 7, the best accuracy obtained by each of the algorithms in 30 runs is listed. In ten out of eighteen datasets the proposed algorithm BJaya-JS has achieved better accuracy than other competitors. Genetic Algorithm is the second-best algorithm as per the average accuracy measure because it has achieved the best results nine times.

The convergence behavior of the proposed BJaya-JS and other algorithms on all datasets is illustrated in Fig. 3. The *x*-axis denotes the number of iterations and the *y*-axis denotes the accuracy of the algorithm on a particular iteration. Fig. 3 shows that the BJaya-JS algorithm achieves fast convergence for almost all datasets.

Table 11 Comparison on the basis of average computational time (in minutes) for 30 runs

	BPSO	BCSA	BDFO	GA	BGWO	BGOA	BJaya-JS
Diabetes	**0.0295**	0.0297	0.0304	0.0484	0.0561	0.0306	0.0661
Contraceptive	0.0383	0.0366	0.0548	0.0482	0.0612	**0.0354**	0.0609
Glass	0.0390	0.0332	0.0514	0.0457	0.0661	0.0333	**0.0258**
Breast cancer	0.0277	0.0281	0.0271	0.0470	0.0448	0.0289	**0.0265**
Vowel	0.0550	0.0542	0.0433	0.1582	0.1051	0.0793	**0.0237**
Australian	0.0309	**0.0295**	0.0329	0.0491	0.0624	0.0302	0.0681
Zoo	0.0351	0.0329	0.0331	0.0633	0.0449	0.0541	**0.0253**
SPECT	0.0375	**0.0239**	0.0612	0.0341	0.0456	0.0241	0.0440
WDBC	0.0589	0.0602	0.0624	**0.0511**	0.0792	0.0318	0.0704
Inosphere	0.0296	0.0295	0.0291	0.0471	0.0722	0.0286	**0.0266**
Sonar	0.0345	0.0288	0.0410	0.0532	0.0958	**0.0227**	0.0838
Semeion	0.4618	1.2954	1.1423	0.6102	0.9115	0.4736	**0.4578**
Madelon	**1.3883**	3.2087	2.6115	2.6186	3.2785	1.6245	1.6126
PD speech	0.9015	**0.6687**	0.8162	0.7414	0.9588	0.7315	0.7513
CANE9	1.6389	1.5876	1.7077	1.5464	1.6245	1.5726	**1.5202**
Colon	**0.1150**	0.1367	0.1550	0.2562	0.2328	0.2641	0.2483
CNS	0.4133	0.3378	0.3860	0.2224	0.2454	0.2448	**0.2000**
Lung	3.0239	2.6000	2.5460	2.0956	**1.2428**	1.8829	2.3083

Bold values indicate the best results

The average fitness of all the algorithms for all the datasets over 30 independent runs is listed in Table 8. In ten out of eighteen datasets, BJaya-JS gives the best average fitness value. Whereas other competitors could not perform well in more than five datasets. Moreover, the standard deviation of the BJaya-JS is also consistently low, indicating the consistent performance of the algorithm over all datasets.

Calculation of fitness measures includes both the classification accuracy and the number of features selected. Since the proposed method is giving better performance than its competitors 80% of the time, it can be concluded as the best feature selection method.

Table 9 compares the feature selection algorithms on the basis of the best fitness value obtained in 30 runs. In this table also BJaya-JS outperforms other methods three times. It is evident from Table 10 that the proposed method has a minimum feature selection ratio of 72.22% times. In order to study the computational time taken by the algorithms, the average computational time over 30 runs for all the algorithms is listed in Table 11. The BJaya-JS algorithm outperforms algorithms 8 out of 18 times. The reason behind this speed is that unlike other algorithms, BJaya-JS operates in binary space throughout the algorithm, i.e. it does not require conversion from continuous to binary. Moreover, Jaya is also a faster algorithm as compared to the other algorithms because it converges in less number of function evaluations (Rao 2019).

Further, the statistical Wilcoxon signed rank test (Demšar 2006) is performed on accuracy results of the proposed method BJaya-JS versus all other competitors at significance level 0.05 (refer to Table 12). *P*-values less than 0.05 indicates that the results are statistically significant. From Table 12 it is evident that we have got statistically significant results most of the time.

Therefore, on the basis of all the comparative results presented in Tables 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 and Fig. 3 it is confirmed that the proposed method BJaya-JS is able to produce better results than its competitors in terms of classification accuracy, fitness, computational time and the selected number of features. The proposed approach is based on Jaya algorithm which does not require any algorithm-specific parameter tuning, so as a result of this BJaya-JS requires only two parameters (n and P_{NS}) to be tuned. However, the limitation of the proposed approach is that its effectiveness is dependent on the values of the parameter n and P_{NS}, the large value of n can make the algorithm slow.

The proposed approach is tested on a wide variety of datasets (number of features ranging from 8 to 12533) and results confirm that the approach has the adaptability for small as well as a large number of features in the datasets. It is also evident that operating in binary space from start to end, without any conversion taking place (through transfer function) is an effective technique for feature selection based on metaheuristic optimization algorithms. Moreover, considering 3 neighbours in generating a solution is an efficient way of searching the solution space. The neighborhood search probability P_{NS} also plays an important role in finding the global optimal solution, avoiding local optima and balancing the exploration & exploitation. Furthermore,

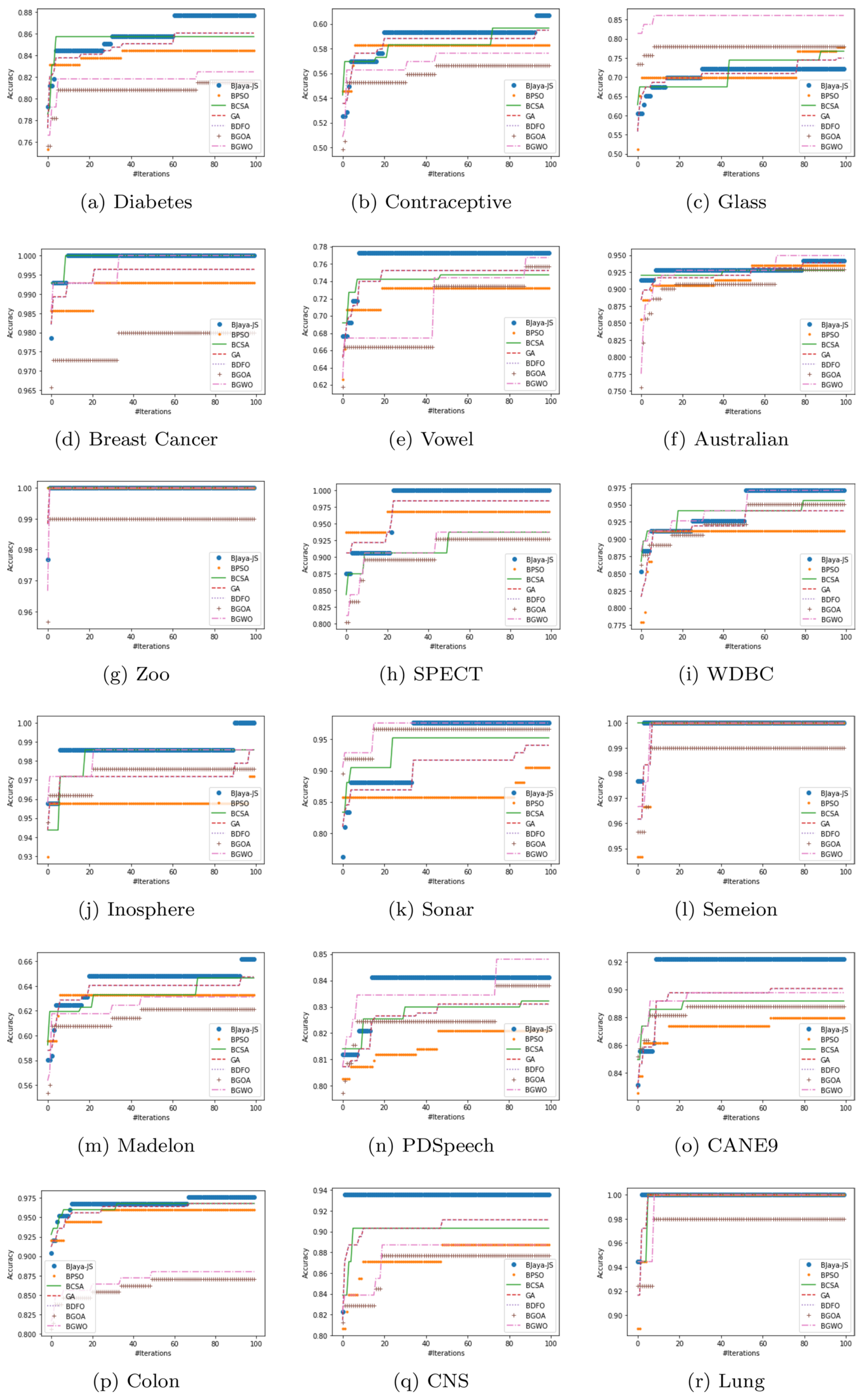

Fig. 3 Comparision of convergence behavior of proposed BJaya-JS with other competitor algorithms

Table 12 P-values of Wilcoxon test for Accuracy of BJaya-JS Vs. other methods

	BPSO	BCSA	BDFO	GA	BGWO	BGOA
Diabetes	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05
Contraceptive	<0.05	<0.05	**<u>0.064</u>**	<0.05	<0.05	<0.05
Glass	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05
Breast cancer	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05
Vowel	<0.05	<0.05	<0.05	<0.05	**<u>0.062</u>**	<0.05
Australian	<0.05	**<u>0.066</u>**	<0.05	<0.05	<0.05	<0.05
Zoo	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05
SPECT	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05
WDBC	**<u>0.059</u>**	<0.05	<0.05	<0.05	<0.05	<0.05
Inosphere	<0.05	<0.05	<0.05	<0.05	<0.05	**<u>0.084</u>**
Sonar	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05
Semeion	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05
Madelon	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05
PD speech	<0.05	<0.05	<0.05	<0.05	<0.05	**<u>0.054</u>**
CANE9	<0.05	**<u>0.063</u>**	<0.05	<0.05	<0.05	<0.05
Colon	<0.05	<0.05	<0.05	**<u>0.073</u>**	<0.05	<0.05
CNS	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05
Lung	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05

Bold and underlined values depicts statistically insignificant results

Table 13 Overall comparison of algorithms using statistical tests

BJaya-JS Vs	Wilcoxon test (*P*-value)	Friedman test
BPSO	0.0008	There is no difference between variables. P = 0.0000002 (i.e. Reject H0)
BCSA	0.0076	
BDFO	0.0006	
GA	1	
BGWO	0.0106	
BGOA	0.0016	

Hence, it can be concluded that BJaya-JS is a very efficient wrapper-based feature selection technique.

5 Conclusion

In this work, three variants of the Binary Jaya algorithm were introduced to solve the feature selection problem. The three variants were based on the S-shaped transfer function (named BJaya-S), the V-shaped transfer function (named BJaya-V) and the Jaccard Similarity (named BJaya-JS). All three variants were wrapper based feature selection technique, associated with the Naive Bayes classifier. Unlike the common practice of using transfer functions in meta-heuristic feature selection algorithms, the proposed BJaya-JS does not use the transfer function. The BJaya-JS operates only in binary space throughout all the phases of algorithm (this saves the continuous to binary conversion time). The Jaccard similarity index was used to find the distance between two solutions in the search space. Furthermore, a new local search technique was proposed, we named it New Solution Generator (NSG). The NSG produces diverse solutions, reduces the chances of being stuck in local minima and maintains the balance between exploration and exploitation. The NSG makes use of three neighboring solutions instead of one to generate new solutions which give it the capability to explore the solution space efficiently in less amount of time. The proposed approach was evaluated on 18 datasets and compared with the BPSO, BCSA, BDFO, GA, BGWO, and BGOA. The BJaya-JS outperformed other algorithms in terms of average accuracy and average fitness. In addition, the performance was consistent throughout all the datasets as suggested by the small values of standard deviation. Moreover, the BJaya-JS produced a very small feature selection ratio and took very less computational time across the majority of the datasets. Furthermore, the statistical significance of the results was proven by Wilcoxon signed rank test and Friedman test. Also, BJaya-JS demonstrated a faster convergence rate than other competing algorithms in most of the datasets.

For future work, application areas could be explored for this proposed approach. The suitability of this approach to gene expression datasets, intrusion detection datasets, or other high-dimensional datasets could be investigated.

References

Ahmed S, Mafarja M, Faris H, Aljarah I (2018) Feature selection using salp swarm algorithm with chaos. In: Proceedings of the 2nd international conference on intelligent systems, metaheuristics & swarm intelligence, p 65–69

Al-Betar MA, Hammouri AI, Awadallah MA, Doush IA (2020) Binary $\beta $-hill climbing optimizer with s-shape transfer function for feature selection. J Ambient Intell Hum Comput 1:1–29

Aljarah I, Mafarja M, Heidari AA, Faris H, Zhang Y, Mirjalili S (2018) Asynchronous accelerating multi-leader salp chains for feature selection. Appl Soft Comput 71:964–979

Al-Tashi Q, Kadir SJA, Rais HM, Mirjalili S, Alhussian H (2019) Binary optimization using hybrid grey wolf optimization for feature selection. IEEE Access 7:39496–39508

Ang JC, Mirzal A, Haron H, Hamed HNA (2015) Supervised, unsupervised, and semi-supervised feature selection: a review on gene selection. IEEE/ACM Trans Comput Biol Bioinf 13(5):971–989

Arora S, Anand P (2019) Binary butterfly optimization approaches for feature selection. Expert Syst Appl 116:147–160

Awadallah MA, Al-Betar MA, Hammouri AI, Alomari OA (2020) Binary Jaya algorithm with adaptive mutation for feature selection. Arab J Sci Eng 1:1–16

Bennasar M, Hicks Y, Setchi R (2015) Feature selection using joint mutual information maximisation. Expert Syst Appl 42(22):8520–8532

Chaudhuri A, Sahu T (2020a) Promethee-based hybrid feature selection technique for high-dimensional biomedical data: application to parkinson's disease classification. Electron Lett 56(25):1403–6

Chaudhuri A, Sahu TP (2020b) Feature selection using binary crow search algorithm with time varying flight length. Expert Syst Appl 1:114288

Choi S-S, Cha S-H, Tappert CC (2010) A survey of binary similarity and distance measures. J Syst Cybern Inf 8(1):43–48

De Souza RC T, dos Santos Coelho L, De Macedo C A, Pierezan J (2018) A v-shaped binary crow search algorithm for feature selection. In 2018 IEEE congress on evolutionary computation (CEC), p 1–8. IEEE

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

Emary E, Zawbaa HM, Hassanien AE (2016) Binary ant lion approaches for feature selection. Neurocomputing 213:54–65

Emine B, Ülker E (2020) An efficient binary social spider algorithm for feature selection problem. Expert Syst Appl 146:113185

Faris H, Aljarah I, Al-Shboul B (2016) A hybrid approach based on particle swarm optimization and random forests for e-mail spam filtering. In International conference on computational collective intelligence, p 498–508. Springer

Hammouri AI, Mafarja M, Al-Betar MA, Awadallah MA, Abu-Doush I (2020) An improved dragonfly algorithm for feature selection. Knowl Based Syst 203:106131

Han J, Pei J, Kamber M (2011) Data mining: concepts and techniques. Elsevier, Amsterdam

Hichem H, Elkamel M, Rafik M, Mesaaoud MT, Ouahiba C (2019) A new binary grasshopper optimization algorithm for feature selection problem. J King Saud Univ Comput Inf Sci. https://doi.org/10.1016/j.jksuci.2019.11.007

Ibrahim RA, Ewees AA, Oliva D, Abd Elaziz M, Lu S (2019) Improved salp swarm algorithm based on particle swarm optimization for feature selection. J Ambient Intell Hum Comput 10(8):3155–3169

Jaccard P (1912) The distribution of the flora in the alpine zone. New Phytol 11(2):37–50

Kabir MM, Shahjahan M, Murase K (2011) A new local search based hybrid genetic algorithm for feature selection. Neurocomputing 74(17):2914–2928

Karaboga D, Basturk B (2008) On the performance of artificial bee colony (abc) algorithm. Appl Soft Comput 8(1):687–697

Kashef S, Nezamabadi-pour H (2015) An advanced aco algorithm for feature subset selection. Neurocomputing 147:271–279

Li Y, Yang Z (2017) Application of eos-elm with binary Jaya-based feature selection to real-time transient stability assessment using pmu data. IEEE Access 5:23092–23101

Mafarja MM, Mirjalili S (2017) Hybrid whale optimization algorithm with simulated annealing for feature selection. Neurocomputing 260:302–312

Mafarja M, Eleyan D, Abdullah S, Mirjalili S (2017a) S-shaped vs. v-shaped transfer functions for ant lion optimization algorithm in feature selection problem. In Proceedings of the international conference on future networks and distributed systems, p 1–7

Mafarja M M, Eleyan D, Jaber I, Hammouri A, Mirjalili S (2017b) Binary dragonfly algorithm for feature selection. In 2017 International conference on new trends in computing sciences (ICTCS), p 12–17. IEEE

Mafarja M, Aljarah I, Heidari AA, Faris H, Fournier-Viger P, Li X, Mirjalili S (2018) Binary dragonfly optimization for feature selection using time-varying transfer functions. Knowl Based Syst 161:185–204

Mafarja M, Aljarah I, Faris H, Hammouri AI, Ala'M A-Z, Mirjalili S (2019) Binary grasshopper optimisation algorithm approaches for feature selection problems. Expert Syst Appl 117:267–286

Martarelli NJ, Nagano MS (2020) Unsupervised feature selection based on bio-inspired approaches. Swarm Evolut Comput 52:100618

Mirjalili S, Lewis A (2013) S-shaped versus v-shaped transfer functions for binary particle swarm optimization. Swarm Evolut Comput 9:1–14

Neggaz N, Ewees AA, Abd Elaziz M, Mafarja M (2020) Boosting salp swarm algorithm by sine cosine algorithm and disrupt operator for feature selection. Expert Syst Appl 145:113103

Peng Y, Wu Z, Jiang J (2010) A novel feature selection approach for biomedical data classification. J Biomed Inf 43(1):15–23

Rao RV (2019) Jaya: an advanced optimization algorithm and its engineering applications. Springer, Berlin

Rao RV, Savsani VJ, Vakharia D (2011) Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput Aided Des 43(3):303–315

Sikora R, Piramuthu S (2007) Framework for efficient feature selection in genetic algorithm based data mining. Eur J Oper Res 180(2):723–737

Talbi E-G (2009) Metaheuristics: from design to implementation, vol 74. Wiley, Hoboken

Tawhid M A, Ibrahim A M (2020) Hybrid binary particle swarm optimization and flower pollination algorithm based on rough set approach for feature selection problem. In Nature-inspired computation in data mining and machine learning, p 249–273. Springer

Tubishat M, Ja'afar S, Alswaitti M, Mirjalili S, Idris N, Ismail MA, Omar MS (2020) Dynamic salp swarm algorithm for feature selection. Expert Syst Appl 164:113873

Venkata Rao R (2016). Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *International Journal of Industrial Engineering Computations*

Yang X-S (2010) Nature-inspired metaheuristic algorithms. Luniver press, UK

Zawbaa H M, Emary E, Parv B, Sharawi M (2016) Feature selection approach based on moth-flame optimization algorithm. In 2016 IEEE congress on evolutionary computation (CEC), p 4612–4617. IEEE