Connection Science

ISSN: 0954-0091 (Print) 1360-0494 (Online) Journal homepage: https://www.tandfonline.com/loi/ccos20

Binary atom search optimisation approaches for feature selection

Jingwei Too & Abdul Rahim Abdullah

To cite this article: Jingwei Too & Abdul Rahim Abdullah (2020): Binary atom search optimisation approaches for feature selection, Connection Science, DOI: 10.1080/09540091.2020.1741515

To link to this article: https://doi.org/10.1080/09540091.2020.1741515

Published online: 17 Mar 2020.

CONNECTION SCIENCE
https://doi.org/10.1080/09540091.2020.1741515

Binary atom search optimisation approaches for feature selection

Jingwei Too and Abdul Rahim Abdullah

Faculty of Electrical Engineering, Universiti Teknikal Malaysia Melaka, Malaysia

ABSTRACT

Atom Search Optimisation (ASO) is a recently proposed metaheuristic algorithm that has proved to work effectively on several benchmark tests. In this paper, we propose the binary variants of atom search optimisation (BASO) for wrapper feature selection. In the proposed scheme, eight transfer functions from S-shaped and V-shaped families are used to convert the continuous ASO into the binary version. The proposed BASO approaches are employed to select a subset of significant features for efficient classification. Twenty-two well-known benchmark datasets acquired from the UCI machine learning repository are used for performance validation. In the experiment, the BASO with an optimal transfer function that contributes to the best classification performance is presented. The particle swarm optimisation (PSO), binary differential evolution (BDE), binary bat algorithm (BBA), binary flower pollination algorithm (BFPA), and binary salp swarm algorithm (BSSA) are used to evaluate the efficacy and efficiency of proposed approaches in feature selection. Our experimental results reveal the superiority of proposed BASO not only in high prediction accuracy but also in the minimal number of selected features.

ARTICLE HISTORY
Received 7 September 2019
Accepted 9 March 2020

KEYWORDS
Feature selection; atom search optimisation; binary atom search optimisation; classification; binary optimisation

1. Introduction

With the rapid growth of information technology, the amount of data is evolving exponentially. Ideally, all the information offered by the features is meaningful. Many pattern recognition applications, extracting the features to describe the target concept in the classification tasks. However, the dataset normally contains irrelevant and redundant features, which significantly degrades the performance of the classification model (Wang et al., 2018; Peng et al., 2005). The excessive number of features not only introduces the extra computational complexity but also increases the prediction error (Labani et al., 2018).

Feature selection is one of the effective ways to resolve the issues above. Briefly, feature selection attempts to select a small subset of significant features that can maintain or improve the prediction accuracy (Liu et al., 2018). In general, feature selection can be categorised into wrapper and filter approaches. Wrapper approaches utilise a learning algorithm to evaluate the optimal feature subset (Mafarja & Mirjalili, 2017). On the one

CONTACT Jingwei Too jamesjames868@gmail.com Faculty of Electrical Engineering, Universiti Teknikal Malaysia Melaka, Melaka, Malaysia

hand, filter approaches use the information theory and statistical analysis over feature space to remove the redundant and irrelevant features (Labani et al., 2018). In comparison with wrapper, filter approaches can usually work faster, and they are independent of the learning algorithm. However, wrapper approaches can often achieve better classification results (Xue et al., 2014). Thus, this study focuses on the wrapper feature selection.

The main goal of feature selection is to improve classification performance and reduce the number of features. Thus it can be considered as a combinatorial optimisation task (Faris et al., 2018). Wrapper feature selection is performed using the metaheuristic algorithms such as genetic algorithm (GA) (Huang & Wang, 2006), ant colony optimisation (ACO) (Al-Ani, 2005), and binary particle swarm optimisation (BPSO) (Chuang et al., 2008), binary grey wolf optimisation (BGWO) (Emary et al., 2016a), binary salp swarm algorithm (BSSA) (Faris et al., 2018), binary tree growth algorithm (BTGA) (Too et al., 2018), multi-verse optimiser (BMVO) (Faris et al., 2017), and binary differential evolution (BDE) (Zorarpacı & Özel, 2016). Previous works showed that metaheuristic algorithms were having high potential when solving the feature selection problem, which made them received a lot of attention from researchers. However, according to the No Free Lunch theorem (NFL), there is no universal metaheuristic algorithm that can solve all the feature selection problems effectively (Wolpert & Macready, 1997). Therefore, more and more new algorithms are required for efficient feature selection.

In the past study, Wang et al. (2017) proposed a modified binary-coded ant colony optimisation (MBACO) in which the GA was used to generate a population of high-quality initial solutions for image classification. Rodrigues et al. (2014) applied the binary bat algorithm (BBA) with the optimum-path forest as part of evaluations for feature selection. Moreover, Mafarja et al. (2019) introduced the binary grasshopper optimisation algorithm (BGOA) for feature selection tasks. The authors indicated that the utilisation of S-shaped and V-shaped transfer functions allowed the BGOA to search around the binary search space. Another study shows the implementation of transfer function into the binary butterfly optimisation algorithm (BBOA) can effectively tackle the feature selection problems (Arora & Anand, 2019). More recent studies of wrapper feature selection can be found in (Al-Tashi et al., 2019; Emary et al., 2016b; Faris et al., 2018; Mafarja et al., 2018; Mirhosseini & Nezamabadi-pour, 2018; Pashaei & Aydin, 2017).

Atom Search Optimisation (ASO) is a recently established metaheuristic algorithm inspired by the concept of molecular dynamics (Zhao et al., 2019a). To date, ASO has attracted the attention of various researchers due to its efficacy in solving global optimisation for different applications. In comparison with the Water Drop Algorithm (WDA) (Siddique & Adeli, 2014), Particle Swarm Optimisation (PSO) (Kennedy, 2011), GA and Gravitational Search Algorithm (GSA) (Rashedi et al., 2009), ASO can usually find promising solutions. Hence, ASO can be a potential metaheuristic algorithm for other real-world applications such as feature selection. To the best of our knowledge, there is no study to apply the ASO for feature selection, which becomes the motivation of this work. This encourages us to develop the binary version of ASO to tackle the feature selection problem in classification tasks.

In this study, we propose the new binary version of atom search optimisation (BASO) for wrapper feature selection. The BASO integrates the S-shaped or V-shaped transfer function, which allows the search agent to move on the binary search space. Twenty-two benchmark datasets acquired from the UCI machine learning repository are used to validate the

performance of proposed BASO, and the results are compared with the other five recent and popular algorithms. From the experiment, it shows that BASO is highly capably in evaluating the optimal feature subset, which leads to promising results.

The rest of paper is organised as follows: Section 2 details the standard atom search optimisation. Section 3 describes the proposed binary atom search optimisation approaches. Section 4 depicts the application of proposed approaches for feature selection. Section 5 discusses the findings of the experiments. Finally, Section 6 concluded the findings of the research work.

2. The atom search optimisation

Atom Search Optimisation (ASO) is a new metaheuristic algorithm proposed by Zhao and his colleagues in 2019 (Zhao et al., 2019a). The ASO mimics the basic concept of molecular dynamics and movement principle of atoms such as characteristics of the potential function, interaction force, and geometric constraint force. In ASO, the population of solutions is called atoms, and each atom maintains two vectors, namely, position and velocity.

All the atoms are moving at all times by following the movement principle. Mathematically, the acceleration of atom is defined as:

$$a_i = \frac{F_i + G_i}{m_i} \tag{1}$$

where F is the interaction force, G is the constraint force, and m is the mass of the atom.

The interaction force between the ith atom and jth atom is described by the Lennard-Jones (L-J) potential as:

$$F_{ij}^{d}(t) = \frac{24\varepsilon(t)}{\sigma(t)}\left[2\left(\frac{\sigma(t)}{r_{ij}(t)}\right)^{13} - \left(\frac{\sigma(t)}{r_{ij}(t)}\right)^{7}\right]\frac{r_{ij}(t)}{r_{ij}^{d}(t)} \tag{2}$$

and

$$F'_{ij}(t) = \frac{24\varepsilon(t)}{\sigma(t)}\left[2\left(\frac{\sigma(t)}{r_{ij}(t)}\right)^{13} - \left(\frac{\sigma(t)}{r_{ij}(t)}\right)^{7}\right] \tag{3}$$

where ϵ is the depth of potential, σ is the length scale, r is the distance between two atoms, d is the dimension and t is the current iteration. However, Equation (3) cannot directly apply for optimisation tasks. Thus, a revised version of Equation (3) is designed as follow:

$$F'_{ij}(t) = -\eta(t)[2(h_{ij}(t))^{13} - (h_{ij}(t))^{7}] \tag{4}$$

where η is the depth function to regulate the attraction or repulsion region, and it can be expressed as:

$$\eta(t) = \alpha\left(1 - \frac{t-1}{T}\right)^{3} e^{-\frac{20t}{T}} \tag{5}$$

where α is the depth weight and T is the maximum number of iterations. The function h can be computed as follow:

$$h_{ij}(t) = \begin{cases} h_{\min}, \text{ if } \dfrac{r_{ij}(t)}{\sigma(t)} < h_{\min} \\ h_{\max}, \text{ elseif } \dfrac{r_{ij}(t)}{\sigma(t)} > h_{\max} \\ \dfrac{r_{ij}(t)}{\sigma(t)}, \text{ otherwise} \end{cases} \tag{6}$$

where h_{max} and h_{min} are the upper and lower limit of h, and they are set to 1.1 and 2.4, respectively (Zhao et al., 2019b). The length scale σ^t can be calculated as:

$$\sigma(t) = \left\| X_{ij}(t), \frac{\sum_{j\in K} X_{ij}(t)}{k(t)} \right\|_2 \tag{7}$$

and

$$\begin{cases} h_{\min} = g_0 + g(t) \\ h_{\max} = u \end{cases} \tag{8}$$

where K is the subset of atoms with better fitness values, g_0 and μ are equal to 1.1 and 2.4, respectively. The g is the drift factor that controls the balance between exploration and exploitation, and it is defined as:

$$g(t) = 0.1 \times \sin\left(\frac{\pi}{2} \times \frac{t}{T}\right) \tag{9}$$

The total interaction force can be expressed as:

$$F_i^d(t) = \sum_{j\in K} r_j F_{ij}^d(t) \tag{10}$$

where r is a random number in [0,1]. According to Newton's third law, the force on the jth atom for the same pairwise interaction is the opposite of force on the ith atom as (Zhao et al., 2019b):

$$F_{ij}(t) = -F_{ij}(t) \tag{11}$$

Furthermore, geometric constraint of the ith atom and constraint force can be expressed as follows:

$$\theta_i(t) = [|X_i(t) - X_{best}(t)|^2 - b_{i,best}^2] \tag{12}$$

$$G_i^d(t) = -\lambda(t)\nabla\theta_i^d(t) = -2\lambda(t)(X_i^d(t) - X_{best}^d(t)) \tag{13}$$

where X_{best} is the position of the best atom, $b_{i,best}$ is the fixed bond length between ith atom and the best atom, and λ is the Lagrangian multiplier. By making $2\lambda = \lambda$, the constraint

force can be represented as:

$$G_i^d(t) = \lambda(t)(X_{best}^d(t) - X_i^d(t)) \tag{14}$$

The Lagrangian multiplier can be expressed as:

$$\lambda(t) = \beta e^{-\frac{20t}{T}} \tag{15}$$

where β is the multiplier weight. Finally, the acceleration of the atom can be written as:

$$\begin{aligned} a_i^d(t) &= \frac{F_i^d(t)}{m_i^d(t)} + \frac{G_i^d(t)}{m_i^d(t)} \\ &= -\alpha\left(1 - \frac{t-1}{T}\right)^3 e^{-\frac{20t}{T}} \sum_{j \in K} \frac{r_j[2(h_{ij}(t))^{13} - (h_{ij}(t))^7]}{m_i(t)} \\ &\quad \frac{(X_j^d(t) - X_i^d(t))}{||X_i(t), X_j(t)||_2} + \beta e^{-\frac{20t}{T}} \frac{X_{best}^d(t) - X_i^d(t)}{m_i(t)} \end{aligned} \tag{16}$$

where m is the mass of the atom, which can be calculated by

$$M_i(t) = e^{-\frac{Fit_i(t) - Fit_{best}(t)}{Fit_{worst}(t) - Fit_{best}(t)}} \tag{17}$$

$$m_i(t) = \frac{M_i(t)}{\sum_{j=1}^N M_j(t)} \tag{18}$$

where *Fit* is the fitness value. Considering the minimisation problem, the Fit_{best} and Fit_{worst} are represented as:

$$Fit_{best}(t) = \min_{i=1}^{N} Fit(t) \tag{19}$$

$$Fit_{worst}(t) = \max_{i=1}^{N} Fit(t) \tag{20}$$

Then, the velocity and position of the atom are updated as follows:

$$V_i^d(t+1) = r_1 V_i^d(t) + a_i^d(t) \tag{21}$$

$$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1) \tag{22}$$

where X_i and V_i are the position and velocity of the ith atom, a is the acceleration, d is the dimension of search space, r_1 is a random number in [0,1], and t is the current iteration.

In ASO, the number of best atoms in subset K is used to control the exploration and exploitation phase.

$$k(t) = N - (N-2)\sqrt{\frac{t}{T}} \tag{23}$$

where N is the number of atoms in the population. Initially, higher k promotes exploration. At the end of the iteration, lower k ensures higher exploitation. The pseudocode of ASO is demonstrated in Algorithm 1.

Algorithm 1. Atom search optimisation

Input: N, T, α and β
Randomly initialise position and velocity of atoms
Calculate the fitness of atoms.
Define best atom as $\boldsymbol{Z}$
While (T is not met)
  For $i = 1$ to N
    Compute mass using Equations (17) and (18)
    Define subset K atoms as in (23)
    Calculate interaction force, F using Equation (10)
    Compute constraint force, G using Equation (14)
    Determine acceleration using Equation (16)
    Update velocity of atoms as in Equation (21)
    Update position of atoms as in Equation (22)
    Update $\boldsymbol{Z}$ if there is better atom
  End for
End while
Output: Global best solution

3. Binary version of atom search optimisation

In the continuous version of ASO, the atoms are moving around the search space with the continuous real domain. As for binary optimisation, the atoms are dealing with only two numbers ("1" or "0"). Correspondingly, a way should be found to use the velocity of the atom to change the position from "0" to "1" or vice versa. Among the early works, the transfer function has successfully utilised in binary versions of particle swarm optimisation, gravitational search algorithm, bat algorithm, tree growth algorithm and antlion optimiser (Emary et al., 2016b; Kennedy & Eberhart, 1997; Mirjalili et al., 2014; Rashedi et al., 2010; Too et al., 2018). Previous works show that transfer function is one of the powerful tools to convert the continuous optimisation algorithm into the binary version (Mirjalili & Lewis, 2013).

In this paper, we propose eight different binary variants of ASO (BASO). The proposed BASO approaches are classified into two groups, namely S-shaped family and V-shaped family. The former applies an S-shaped transfer function that enables the search agent to move in the binary search space. The latter uses a V-shaped transfer function that allows the atoms to perform the search on the binary search space.

Table 1 displays the S-shaped and V-shaped transfer functions with mathematical equations. In Table 1, the "erf" denotes the error function. The illustrations of the S-shaped and V-shaped families are shown in Figure 1 and Figure 2, respectively. By integrating a transfer function into ASO, the binary version of ASO is achieved. In this framework, the velocity of the atom is first converted into probability using the S-shaped or V-shaped transfer function. Then, the position of the atom is updated.

Table 1. S-shaped and V-shaped transfer functions.

S-shaped family	Transfer function	V-shaped family	Transfer function
S1	$T(x) = \frac{1}{1+e^{-2x}}$	V1	$T(x) = \left\lvert \mathrm{erf}\left(\frac{\sqrt{\pi}}{2}x\right)\right\rvert$
S2	$T(x) = \frac{1}{1+e^{-x}}$	V2	$T(x) = \lvert\tanh(x)\rvert$
S3	$T(x) = \frac{1}{1+e^{(-x/2)}}$	V3	$T(x) = \left\lvert (x)/\sqrt{1+x^2}\right\rvert$
S4	$T(x) = \frac{1}{1+e^{(-x/3)}}$	V4	$T(x) = \left\lvert \frac{2}{\pi}\arctan\left(\frac{\pi}{2}x\right)\right\rvert$

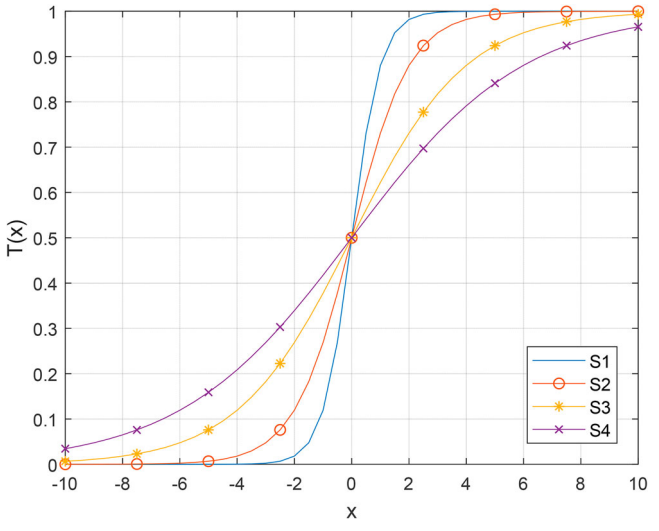

Figure 1. S-shaped transfer functions.

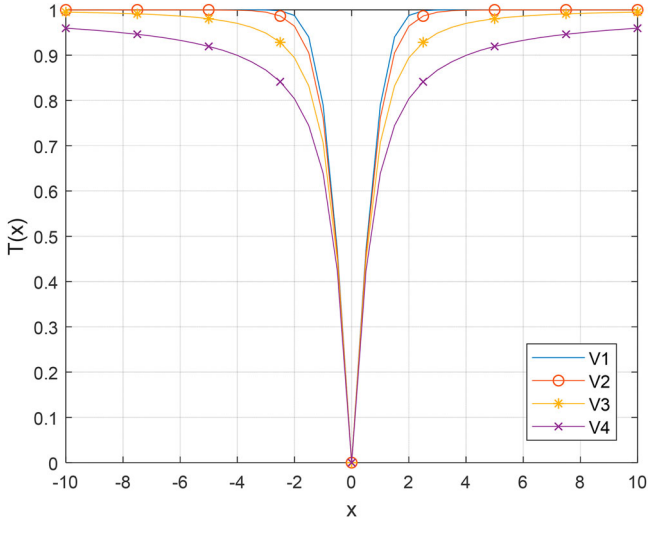

Figure 2. V-shaped transfer functions.

According to literature, the position of the search agent can be updated by using the Equation (24) for S-shaped family or Equation (25) for V-shaped family, which converted the probability into binary representation (Kennedy & Eberhart, 1997; Mirjalili & Lewis, 2013; Rashedi et al., 2010).

$$X_i^d(t+1) = \begin{cases} 1, \text{ if } rand < T(V_i^d(t+1)) \\ 0, \text{ if } rand \geq T(V_i^d(t+1)) \end{cases} \tag{24}$$

$$X_i^d(t+1) = \begin{cases} 1 - X_i^d(t), \text{ if } rand < T(V_i^d(t+1)) \\ X_i^d(t), \quad \text{if } rand \geq T(V_i^d(t+1)) \end{cases} \tag{25}$$

where $X_{i,d}$ and $V_{i,d}$ are the position and velocity of the ith atom at dth dimension, t is the current iteration, and *rand* is a random vector in [0,1]. As can be seen in Equations (24) and (25), the atom randomly changed its position based on the parameter *rand*. Taking Equation (25) as an example, if the *rand* is greater, then a higher value of $T(V)$ is required for the atom to change its position. On the contrary, a lower value of $T(V)$ enables the atom to maintain its position when the *rand* is greater. However, the atom can also change its position when the value of the *rand* is smaller. This indicates that the position of the atom is updated based on a fully random manner, which may lead to unsatisfactory performance. Thus, we propose the new updating rules as follow:

S-shape family

$$X_i^d(t+1) = \begin{cases} 0, & \text{if } T(V_i^d(t+1)) \leq K1 \\ 1, & \text{if } K1 < T(V_i^d(t+1)) \leq K2 \\ X_{best}^d(t), & \text{otherwise} \end{cases} \tag{26}$$

V-shape family

$$X_i^d(t+1) = \begin{cases} X_{best}^d(t), & \text{if } T(V_i^d(t+1)) \leq K1 \\ X_i^d(t), & \text{if } K1 < T(V_i^d(t+1)) \leq K2 \\ 1 - X_i^d(t), & \text{otherwise} \end{cases} \tag{27}$$

The $K1$ and $K2$ are calculated as follows:

$$K1 = \frac{1}{3} rand \tag{28}$$

$$K2 = \frac{1}{3} + \frac{1}{3} rand \tag{29}$$

where *rand* is a random vector in [0,1], $X_{i,d}$ and $V_{i,d}$ are the position and velocity of the ith atom at dth dimension, X_{best} is the position of the best atom, and t is the current iteration. As can be seen in Equation (26), the parameter *rand* has removed and replaced with the $K1$ and $K2$. The atom no longer using a fully random manner to update its position. In our proposed scheme, if the value of $T(V)$ is smaller (less than $K1$), then the atom set its new position to 0; else if the value of $T(V)$ is medium (between $K1$ and $K2$), then the atom changes its position to 1; otherwise, the atom follows the position of the best atom for the next iteration. We involve the best atom in the proposed scheme to guide the atom move toward the global optimal.

The pseudocode of binary atom search optimisation (BASO) is illustrated in Algorithm 2. Firstly, the position and velocity of the N atoms are randomly initialised. Secondly, the fitness values of atoms are calculated, and the best atom is defined. In each iteration, for each atom, the mass is computed using Equations (17) and (18). Then, the interaction and constraint forces are determined using Equations (10) and (14), respectively. Next, the acceleration of the atom is computed as shown in Equation (16). The velocity of the atom is then updated using Equation (21). After that, the velocity of the atom is converted into probability value using the S-shaped or V-shaped transfer function. Later, the position of the atom is updated using Equation (26) or Equation (27). At the end of each iteration, the best atom is updated. The procedure is repeated iteratively until the maximum number of iterations is satisfied. Finally, the global best solution is achieved. The main differences between

Algorithm 2. Binary atom search optimisation

Input: N, T, α and β
Randomly initialise position and velocity of atoms
Calculate the fitness of atoms.
Define best atom as $\boldsymbol{Z}$
While (T is not met)
  For $i = 1$ to N
    Compute mass using (17) and (18)
    Define subset K atoms as in (23)
    Calculate interaction force, F using (10)
    Compute constraint force, G using (14)
    Determine acceleration using (16)
    Update velocity of atoms as in (21)
    Convert velocity into probability using S-shaped or V-shaped transfer function
    Update position of atoms using (26) or (27)
    Update $\boldsymbol{Z}$ if there is better atom
  End for
End while
Output: Global best solution

ASO and BASO are that there is an additional conversion step after velocity update and the position updating step.

4. Proposed binary atom search optimisation for feature selection

The proposed binary atom search optimizations are applied to solve the feature selection problem for classification tasks. Feature selection is a pre-processing step to evaluate the subset of significant features that can significantly enhance the prediction accuracy (Xue et al., 2014). From the optimisation perspective, the feature selection problem is considered as a binary combinatorial optimisation, which represents the solution in binary form. In feature selection, "1" means the feature is selected while "0" represents the unselected feature. Given a solution $X = \{1, 0, 1, 0, 0, 0, 1, 1, 1, 0\}$, it shows that total five features (1st, 3rd, 7th, 8th, and 9th features) are selected.

The optimal feature subset is determined based on two criteria: High classification accuracy and minimal feature size. In this study, the fitness function that used to evaluate the individual search agent is expressed as:

$$\downarrow Fitness = \delta ER + (1 - \delta)\frac{|C|}{|F|} \tag{30}$$

and

$$ER = \frac{\text{No. of wrongly predicted instance}}{\text{Total number of instances}} \tag{31}$$

where ER is the error rate, $|C|$ is the number of selected features, $|F|$ is the total number of features in each dataset, and δ is used to control the influence of classification performance and feature size. Since the classification performance is considered to be the most important metric, we set δ to 0.99 (Emary et al., 2016a; Faris et al., 2018).

Table 2. List of used datasets.

No	Dataset	Number of instances	Number of features
1	Glass	214	10
2	Hepatitis	155	19
3	Iris	150	4
4	Lymphography	148	18
5	Primary Tumour	339	17
6	Soybean	307	35
7	Horse Colic	368	27
8	Ionosphere	351	34
9	Zoo	101	16
10	Wine	178	13
11	Breast Cancer	699	9
12	Lung Cancer	32	56
13	Musk 1	476	166
14	Arrhythmia	452	279
15	Dermatology	366	34
16	SPECT Heart	267	22
17	Libras Movement	360	90
18	ILPD	583	10
19	Seeds	210	7
20	LSVT	126	310
21	Semeion	1593	256
22	Leukemia	72	7128

Table 3. Parameter setting.

Algorithm	Parameter	Value
BASO	Number of search agent, N	10
	Maximum number of iterations, T	100
	Depth weight, α	50
	Multiplier weight, β	0.2
BBA	Number of bats	10
	Maximum number of iterations, T	100
	Maximum frequency, F_{max}	2
	Minimum frequency, F_{min}	0
	Two constants, α and γ	0.9
PSO	Number of particles, N	10
	Maximum number of iterations, T	100
	Acceleration coefficients, c_1 and c_2	2
	Inertia weight, w	0.1
BFPA	Number of flowers, N	10
	Maximum number of iterations, T	100
	Switch probability, P	0.8
	Levy coefficient, β	1.5
BDE	Number of vectors, N	10
	Maximum number of iterations, T	100
	Crossover rate, CR	1
BSSA	Number of salps, N	10
	Maximum number of iterations, T	100

5. Experimental results and discussions

5.1. Data Description

In the experiment, twenty-two datasets acquired from the UCI repository are used to validate the performance of proposed approaches (UCI Machine Learning Repository). Table 2

Table 4. The best fitness value of proposed BASO approaches on 22 datasets.

	Best fitness value							
Dataset	BASO-S1	BASO-S2	BASO-S3	BASO-S4	BASO-V1	BASO-V2	BASO-V3	BASO-V4
1	**0.0040**	**0.0040**	**0.0040**	**0.0040**	**0.0040**	**0.0040**	**0.0040**	**0.0040**
2	**0.1148**	**0.1148**	0.1214	0.1209	0.1280	0.1346	0.1159	0.1275
3	**0.0339**	**0.0339**	**0.0339**	**0.0339**	**0.0339**	**0.0339**	**0.0339**	**0.0339**
4	0.1258	0.1116	0.1181	**0.1105**	0.1312	0.1181	0.1247	0.1187
5	0.5933	0.5926	0.5831	0.5855	0.5735	**0.5705**	0.5836	**0.5705**
6	0.2192	**0.2126**	0.2195	0.2162	0.2313	0.2274	0.2274	0.2304
7	**0.1272**	0.1276	0.1304	0.1284	0.1280	0.1377	0.1397	0.1370
8	**0.0659**	0.0716	0.0691	0.0810	0.0875	0.0853	0.0838	0.0863
9	0.0335	0.0440	**0.0334**	**0.0334**	0.0446	0.0440	0.0440	0.0347
10	0.0388	0.0330	**0.0221**	0.0287	0.0337	0.0388	0.0330	0.0337
11	**0.0303**	**0.0303**	**0.0303**	**0.0303**	0.0307	0.0311	**0.0303**	0.0307
12	0.1987	0.1993	0.2007	0.1682	**0.1368**	0.2018	0.1693	0.1695
13	0.0942	0.0874	0.0872	0.0904	**0.0676**	0.0769	0.0788	0.0705
14	0.3247	**0.3237**	0.3333	0.3382	0.3348	0.3301	0.3347	0.3304
15	0.0151	0.0159	0.0165	0.0165	0.0160	0.0157	0.0163	**0.0135**
16	0.1463	0.1501	0.1501	0.1506	0.1459	0.1506	**0.1421**	0.1450
17	0.2058	**0.1941**	0.1994	0.2059	0.1996	0.1995	0.2024	0.2027
18	0.2727	0.2727	0.2727	0.2727	0.2727	**0.2696**	0.2737	0.2737
19	0.0453	**0.0406**	**0.0406**	**0.0406**	**0.0406**	**0.0406**	**0.0406**	**0.0406**
20	**0.1743**	0.2156	0.2165	0.2484	0.2354	0.2434	0.2434	0.2679
21	0.0745	0.0733	0.0732	0.0751	0.0713	0.0711	0.0711	**0.0696**
22	**0.0153**	0.0158	0.0157	0.0164	0.0188	0.0188	0.0188	0.0188

Table 5. Result of worst fitness value of proposed BASO approaches on 22 datasets.

	Worst fitness value							
Dataset	BASO-S1	BASO-S2	BASO-S3	BASO-S4	BASO-V1	BASO-V2	BASO-V3	BASO-V4
1	0.0209	0.0209	0.0211	0.0209	**0.0181**	0.0191	0.0229	0.0219
2	**0.1523**	0.1727	0.1666	0.1658	0.1631	0.1687	0.1671	0.1621
3	**0.0421**	**0.0421**	**0.0421**	**0.0421**	**0.0421**	**0.0421**	**0.0421**	**0.0421**
4	**0.1611**	0.1699	0.1877	0.1818	0.1736	0.1665	0.1688	0.1753
5	0.6485	0.6485	0.6485	0.6509	**0.6149**	0.6221	0.6323	0.6311
6	0.2717	0.2924	0.2566	**0.2500**	0.2772	0.2766	0.2733	0.2742
7	0.1610	**0.1575**	0.2105	0.2140	0.1993	0.1872	0.1927	0.1986
8	0.1061	**0.0982**	0.1058	0.1130	0.1189	0.1286	0.1209	0.1241
9	0.0960	0.0867	0.0947	0.1151	**0.0755**	**0.0755**	0.0960	0.0836
10	0.0578	0.0570	**0.0555**	0.0570	0.0687	0.0694	0.0644	0.0629
11	0.0354	0.0368	0.0354	0.0354	0.0354	0.0354	0.0346	**0.0339**
12	0.3314	0.3025	0.3370	0.3364	0.3341	0.3341	0.3350	**0.3018**
13	0.1183	0.1170	0.1081	0.1110	**0.0978**	0.1022	0.1020	0.1066
14	0.3641	**0.3553**	0.3632	0.3684	0.3713	0.3635	0.3677	0.3635
15	0.0275	0.0282	**0.0273**	0.0281	0.0306	0.0309	**0.0273**	0.0295
16	0.1929	0.1986	0.1967	0.1948	0.1925	**0.1835**	0.2003	0.1853
17	0.2336	0.2316	0.2371	0.2414	0.2381	**0.2313**	0.2385	0.2351
18	0.2990	0.2986	0.2925	**0.2866**	0.2898	0.2942	0.2908	0.2986
19	0.0609	0.0609	**0.0594**	0.1000	0.0609	0.0609	0.0609	0.0764
20	**0.3019**	0.3073	0.3658	0.3652	0.3672	0.3669	0.3753	0.3670
21	0.0837	0.0808	0.0839	**0.0802**	0.0864	0.0851	0.0837	0.0833
22	**0.0302**	0.0305	0.0344	0.0495	0.0332	0.0332	0.0331	0.0471

listed the utilised datasets. In this study, the *K*-fold cross-validation manner is applied for the performance evaluation. The data is divided into *K* equal parts, in which the *K*-1 parts are used for the training set, and the remaining part is used for the testing set. The procedure is repeated with *K* time by using different parts as the testing set. As for wrapper feature

Table 6. Result of mean fitness value of proposed BASO approaches on 22 datasets.

	Mean fitness value (STD)							
Dataset	BASO-S1	BASO-S2	BASO-S3	BASO-S4	BASO-V1	BASO-V2	BASO-V3	BASO-V4
1	0.0122 (0.0052)	0.0123 (0.0048)	0.0124 (0.0049)	**0.0120** (**0.0050**)	0.0121 (0.0051)	0.0128 (0.0047)	0.0130 (0.0050)	0.0129 (0.0050)
2	**0.1378** (**0.0102**)	0.1385 (0.0120)	0.1405 (0.0122)	0.1402 (0.0109)	0.1477 (0.0089)	0.1493 (0.0093)	0.1497 (0.0125)	0.1450 (0.0099)
3	0.0387 (0.0033)	0.0387 (0.0030)	0.0383 (0.0030)	0.0384 (0.0032)	**0.0382** (**0.0030**)	**0.0382** (**0.0030**)	**0.0382** (**0.0030**)	**0.0382** (**0.0030**)
4	0.1478 (0.0110)	**0.1447** (**0.0162**)	0.1480 (0.0166)	0.1474 (0.0173)	0.1521 (0.0136)	0.1514 (0.0125)	0.1483 (0.0128)	0.1467 (0.0142)
5	0.6055 (0.0133)	0.6061 (0.0125)	0.6047 (0.0148)	0.6028 (0.0150)	0.6011 (0.0108)	0.6032 (0.0148)	0.6044 (0.0129)	**0.6003** (**0.0121**)
6	0.2398 (0.0128)	0.2442 (0.0189)	0.2393 (0.0102)	**0.2370** (**0.0108**)	0.2493 (0.0124)	0.2496 (0.0145)	0.2448 (0.0118)	0.2472 (0.0133)
7	0.1393 (0.0092)	**0.1392** (**0.0096**)	0.1425 (0.0181)	0.1477 (0.0225)	0.1632 (0.0184)	0.1639 (0.0137)	0.1631 (0.0137)	0.1662 (0.0169)
8	**0.0796** (**0.0094**)	0.0846 (0.0082)	0.0882 (0.0082)	0.0970 (0.0102)	0.1082 (0.0084)	0.1021 (0.0107)	0.0995 (0.0103)	0.1041 (0.0095)
9	**0.0594** (**0.0172**)	0.0643 (0.0126)	0.0613 (0.0129)	0.0629 (0.0212)	0.0628 (0.0096)	0.0614 (0.0091)	0.0615 (0.0125)	0.0598 (0.0123)
10	0.0458 (0.0054)	0.0462 (0.0057)	**0.0450** (**0.0070**)	0.0462 (0.0075)	0.0485 (0.0077)	0.0503 (0.0076)	0.0481 (0.0064)	0.0482 (0.0081)
11	0.0324 (0.0012)	0.0324 (0.0015)	0.0324 (0.0014)	**0.0322** (**0.0013**)	0.0326 (0.0012)	0.0324 (0.0010)	0.0323 (0.0011)	0.0323 (0.0008)
12	0.2703 (0.0302)	0.2612 (0.0331)	0.2653 (0.0373)	0.2555 (0.0447)	0.2570 (0.0481)	0.2650 (0.0339)	0.2603 (0.0441)	**0.2504** (**0.0329**)
13	0.1029 (0.0059)	0.1011 (0.0073)	0.1001 (0.0062)	0.1004 (0.0058)	**0.0858** (**0.0089**)	0.0883 (0.0068)	0.0900 (0.0060)	0.0867 (0.0089)
14	**0.3397** (**0.0099**)	0.3428 (0.0079)	0.3477 (0.0076)	0.3538 (0.0067)	0.3524 (0.0110)	0.3483 (0.0097)	0.3503 (0.0101)	0.3469 (0.0082)
15	0.0223 (0.0034)	**0.0212** (**0.0039**)	0.0217 (0.0029)	0.0216 (0.0031)	0.0228 (0.0041)	0.0227 (0.0045)	0.0218 (0.0030)	0.0223 (0.0037)
16	0.1733 (0.0137)	0.1772 (0.0153)	0.1730 (0.0123)	0.1733 (0.0127)	0.1648 (0.0142)	**0.1624** (**0.0094**)	0.1697 (0.0140)	0.1669 (0.0104)
17	0.2189 (0.0074)	0.2177 (0.0109)	0.2193 (0.0095)	0.2239 (0.0102)	0.2157 (0.0101)	**0.2148** (**0.0085**)	0.2172 (0.0105)	0.2170 (0.0091)
18	0.2826 (0.0056)	0.2820 (0.0058)	**0.2816** (**0.0055**)	**0.2816** (**0.0041**)	0.2826 (0.0046)	0.2832 (0.0057)	0.2828 (0.0049)	0.2846 (0.0064)
19	0.0556 (0.0045)	0.0546 (0.0054)	0.0534 (0.0055)	0.0588 (0.0149)	0.0538 (0.0059)	**0.0532** (**0.0057**)	0.0538 (0.0054)	0.0543 (0.0072)
20	**0.2231** (**0.0299**)	0.2494 (0.0199)	0.2807 (0.0431)	0.3170 (0.0298)	0.3238 (0.0347)	0.3224 (0.0284)	0.3299 (0.0283)	0.3299 (0.0235)
21	0.0783 (0.0022)	0.0786 (0.0020)	0.0790 (0.0025)	0.0777 (0.0015)	**0.0763** (**0.0041**)	0.0781 (0.0040)	0.0770 (0.0037)	0.0772 (0.0034)
22	0.0179 (0.0052)	**0.0177** (**0.0043**)	0.0217 (0.0070)	0.0245 (0.0088)	0.0218 (0.0058)	0.0224 (0.0063)	0.0232 (0.0066)	0.0231 (0.0081)

selection, a learning algorithm (classifier) is required to compute the error rate. According to (Emary et al., 2016b; Xue et al., 2014), *k*-nearest neighbour (KNN) with Euclidean distance and $k = 5$ is applied in this work. The KNN is chosen since it can usually achieve satisfactory performance with fast processing speed. In comparison with other classifiers, KNN not only simpler and faster but also easy to implement.

5.2. Comparison algorithms and evaluation metrics

Five recent and popular feature selection methods, namely binary bat algorithm (BBA) (Mirjalili et al., 2014), particle swarm optimisation (PSO) (Mafarja et al., 2019), binary differential

Table 7. Result of accuracy of proposed BASO approaches on 22 datasets.

	Average accuracy (STD)							
Dataset	BASO-S1	BASO-S2	BASO-S3	BASO-S4	BASO-V1	BASO-V2	BASO-V3	BASO-V4
1	0.9912 (0.0056)	0.9907 (0.0052)	0.9910 (0.0053)	0.9912 (0.0056)	**0.9914** (**0.0053**)	0.9907 (0.0050)	0.9905 (0.0054)	0.9907 (0.0052)
2	**0.8633** (**0.0103**)	0.8620 (0.0126)	0.8610 (0.0125)	0.8617 (0.0103)	0.8547 (0.0085)	0.8530 (0.0090)	0.8523 (0.0128)	0.8570 (0.0098)
3	0.9663 (0.0055)	0.9670 (0.0046)	0.9680 (0.0041)	0.9677 (0.0050)	**0.9683** (**0.0043**)	**0.9683** (**0.0043**)	**0.9683** (**0.0043**)	**0.9683** (**0.0043**)
4	0.8557 (0.0115)	**0.8596** (**0.0163**)	0.8561 (0.0169)	0.8568 (0.0173)	0.8507 (0.0139)	0.8518 (0.0127)	0.8550 (0.0129)	0.8568 (0.0146)
5	0.3950 (0.0132)	0.3948 (0.0125)	0.3962 (0.0151)	0.3980 (0.0155)	0.3991 (0.0112)	0.3967 (0.0153)	0.3955 (0.0137)	**0.3997** (**0.0126**)
6	0.7662 (0.0130)	0.7615 (0.0188)	0.7667 (0.0101)	**0.7690** (**0.0108**)	0.7543 (0.0126)	0.7538 (0.0151)	0.7592 (0.0120)	0.7567 (0.0135)
7	**0.8603** (**0.0091**)	**0.8603** (**0.0097**)	0.8574 (0.0182)	0.8522 (0.0226)	0.8376 (0.0183)	0.8371 (0.0137)	0.8378 (0.0137)	0.8347 (0.0169)
8	**0.9209** (**0.0094**)	0.9160 (0.0080)	0.9126 (0.0080)	0.9043 (0.0099)	0.8940 (0.0080)	0.9001 (0.0105)	0.9021 (0.0101)	0.8980 (0.0090)
9	**0.9450** (**0.0167**)	0.9405 (0.0119)	0.9435 (0.0127)	0.9415 (0.0206)	0.9420 (0.0095)	0.9430 (0.0092)	0.9430 (0.0122)	**0.9450** (**0.0124**)
10	0.9594 (0.0057)	0.9591 (0.0056)	**0.9600** (**0.0070**)	0.9588 (0.0076)	0.9562 (0.0080)	0.9538 (0.0079)	0.9562 (0.0062)	0.9562 (0.0086)
11	0.9733 (0.0018)	0.9734 (0.0022)	0.9735 (0.0016)	0.9736 (0.0017)	0.9734 (0.0018)	0.9736 (0.0019)	**0.9738** (**0.0016**)	0.9737 (0.0016)
12	0.7300 (0.0304)	0.7400 (0.0335)	0.7367 (0.0373)	0.7467 (0.0451)	0.7450 (0.0487)	0.7367 (0.0340)	0.7417 (0.0444)	**0.7517** (**0.0333**)
13	0.9018 (0.0056)	0.9048 (0.0067)	0.9053 (0.0059)	0.9049 (0.0055)	**0.9183** (**0.0089**)	0.9157 (0.0070)	0.9140 (0.0060)	0.9176 (0.0089)
14	**0.6590** (**0.0095**)	0.6562 (0.0078)	0.6521 (0.0079)	0.6468 (0.0067)	0.6487 (0.0112)	0.6529 (0.0098)	0.6509 (0.0102)	0.6544 (0.0082)
15	0.9843 (0.0032)	**0.9857** (**0.0039**)	0.9850 (0.0033)	0.9854 (0.0031)	0.9824 (0.0040)	0.9822 (0.0044)	0.9833 (0.0030)	0.9831 (0.0038)
16	0.8310 (0.0132)	0.8279 (0.0146)	0.8315 (0.0119)	0.8315 (0.0120)	0.8387 (0.0141)	**0.8410** (**0.0092**)	0.8338 (0.0145)	0.8367 (0.0101)
17	0.7825 (0.0079)	0.7835 (0.0107)	0.7817 (0.0094)	0.7781 (0.0098)	0.7869 (0.0103)	**0.7879** (**0.0085**)	0.7853 (0.0107)	0.7856 (0.0093)
18	0.7171 (0.0060)	0.7182 (0.0061)	**0.7189** (**0.0054**)	**0.7189** (**0.0045**)	0.7177 (0.0046)	0.7172 (0.0061)	0.7175 (0.0048)	0.7160 (0.0064)
19	0.9471 (0.0043)	0.9481 (0.0053)	0.9493 (0.0054)	0.9440 (0.0145)	0.9488 (0.0058)	**0.9495** (**0.0054**)	0.9488 (0.0053)	0.9483 (0.0068)
20	**0.7758** (**0.0296**)	0.7496 (0.0201)	0.7188 (0.0432)	0.6829 (0.0298)	0.6771 (0.0351)	0.6783 (0.0288)	0.6708 (0.0285)	0.6708 (0.0238)
21	0.9300 (0.0022)	0.9297 (0.0019)	0.9292 (0.0024)	**0.9303** (**0.0013**)	0.9284 (0.0043)	0.9265 (0.0043)	0.9277 (0.0037)	0.9275 (0.0034)
22	0.9836 (0.0052)	**0.9843** (**0.0044**)	0.9814 (0.0067)	0.9800 (0.0085)	0.9829 (0.0059)	0.9821 (0.0063)	0.9814 (0.0067)	0.9814 (0.0082)

evolution (BDE) (Zorarpacı & Özel, 2016), binary salp swarm algorithm (BSSA) (Faris et al., 2018) and binary flower pollination algorithm (BFPA) (Rodrigues et al., 2015) are used to examine the efficacy and efficiency of proposed BASO. Table 3 outlines the detailed parameter settings of utilised algorithms. Note that the α and β of BASO are chosen according to (Zhao et al., 2019a). To ensure a fair comparison, the population size (N) and the maximum number of iterations (T) are fixed at 10 and 100, respectively.

In this paper, six evaluation metrics include the best fitness, worst fitness, mean fitness, accuracy, feature size, and the computational time are used to measure the performance of proposed approaches. Each algorithm is repeated with M independent runs to achieve

Table 8. Result of feature size of proposed BASO approaches on 22 datasets.

	Average number of selected features (STD)							
Dataset	BASO-S1	BASO-S2	BASO-S3	BASO-S4	BASO-V1	BASO-V2	BASO-V3	BASO-V4
1	3.45	**3.15**	3.45	3.25	3.60	3.60	3.60	3.70
	(0.89)	(**0.93**)	(1.39)	(0.97)	(0.50)	(0.94)	(1.05)	(0.57)
2	4.70	**3.50**	5.45	6.20	7.30	7.25	6.75	6.45
	(3.84)	(**2.06**)	(2.63)	(4.05)	(2.03)	(1.94)	(2.36)	(1.93)
3	**2.15**	2.40	2.65	2.55	2.75	2.75	2.75	2.75
	(**0.99**)	(0.88)	(0.75)	(0.94)	(0.79)	(0.79)	(0.79)	(0.79)
4	9.00	10.40	9.85	10.05	**7.75**	8.45	8.55	8.90
	(2.94)	(2.23)	(2.16)	(1.88)	(**1.71**)	(1.70)	(1.99)	(1.55)
5	11.10	11.85	11.85	11.60	10.50	**9.95**	10.05	10.25
	(2.15)	(2.21)	(1.66)	(2.78)	(1.57)	(**2.14**)	(1.96)	(1.80)
6	29.10	28.20	29.10	29.15	21.40	**20.55**	22.20	22.05
	(1.41)	(2.28)	(1.89)	(1.57)	(2.19)	(**2.52**)	(1.58)	(2.04)
7	2.60	**2.40**	3.35	3.70	6.60	7.15	6.65	6.95
	(0.94)	(**0.60**)	(1.42)	(1.30)	(1.60)	(1.53)	(1.31)	(1.43)
8	**4.10**	4.80	5.65	7.50	10.95	10.90	9.05	10.55
	(**1.07**)	(1.51)	(2.25)	(2.46)	(2.52)	(2.13)	(2.19)	(2.96)
9	**7.95**	8.65	8.60	8.00	8.55	8.00	8.15	8.60
	(**2.26**)	(1.90)	(1.88)	(1.89)	(1.05)	(1.49)	(1.53)	(1.35)
10	7.30	7.45	7.00	7.05	6.60	**6.00**	6.10	6.30
	(2.39)	(2.37)	(2.29)	(2.39)	(1.67)	(**1.62**)	(1.71)	(1.69)
11	**5.40**	5.50	5.50	5.45	5.60	5.65	5.70	5.65
	(**1.05**)	(1.28)	(1.05)	(1.10)	(1.05)	(1.09)	(0.92)	(1.18)
12	**16.70**	21.50	25.65	26.15	25.45	24.10	25.30	25.25
	(**11.43**)	(11.49)	(10.11)	(8.99)	(3.12)	(3.19)	(3.80)	(2.92)
13	94.30	113.30	105.30	103.70	81.55	81.20	**81.15**	84.45
	(29.74)	(34.91)	(40.23)	(29.80)	(6.46)	(5.00)	(**5.69**)	(5.25)
14	**57.80**	67.85	91.05	115.70	128.10	131.25	129.75	132.80
	(**25.95**)	(20.08)	(43.42)	(37.06)	(7.84)	(5.65)	(9.00)	(9.35)
15	23.05	24.05	23.25	24.50	18.25	**17.30**	18.10	18.90
	(3.35)	(3.36)	(3.51)	(3.03)	(1.92)	(**2.32**)	(1.62)	(1.77)
16	13.20	15.00	13.70	14.30	11.25	**11.00**	11.55	11.65
	(2.59)	(2.36)	(2.92)	(2.79)	(1.74)	(**2.08**)	(2.28)	(2.06)
17	32.60	29.85	**28.20**	37.35	42.80	43.45	41.40	41.90
	(17.78)	(11.61)	(**10.53**)	(11.57)	(3.38)	(4.52)	(4.27)	(3.97)
18	**2.50**	3.00	3.25	3.30	3.05	3.30	3.15	3.50
	(**1.50**)	(1.75)	(1.59)	(1.49)	(1.36)	(1.53)	(1.42)	(1.57)
19	2.30	2.25	2.20	2.40	**2.15**	2.25	2.20	2.20
	(0.57)	(0.55)	(0.70)	(0.68)	(**0.37**)	(0.55)	(0.41)	(0.52)
20	**37.05**	44.65	69.05	96.60	128.85	123.15	124.90	124.15
	(**31.29**)	(19.92)	(32.27)	(45.44)	(9.71)	(5.87)	(6.79)	(5.44)
21	230.30	229.35	226.95	223.45	136.95	**136.55**	138.45	138.50
	(12.50)	(21.11)	(21.10)	(15.54)	(10.6)	(**9.75**)	(8.03)	(8.09)
22	**1151.55**	1559.8	2329.05	3331.7	3409.15	3397.70	3402.20	3386.55
	(**162.39**)	(244.52)	(770.85)	(944.10)	(51.53)	(43.06)	(32.56)	(39.00)

the statistical meaningful result. Finally, the average metrics obtained from M independent runs are presented as the experimental results. Note that the total number of function evaluations (NFE) for the proposed BASO is $N \times T \times K \times M$. In this study, we set the K and M at 10 and 20. All the analysis is done in MATLAB 9.3 using a computer with Intel Core i5-9400 CPU 2.90 GHz and 16.0GB RAM.

5.3. Assessments of proposed BASO approaches in feature selection

In the first part of the experiment, the assessments on eight proposed BASO algorithms are performed. Tables 4, 5, and 6 exhibit the best, worst and mean fitness values of

Table 9. The *p*-value of Wilcoxon signed rank test for BASO-S1 and other BASO approaches.

	P-value						
Dataset	BASO-S2	BASO-S3	BASO-S4	BASO-V1	BASO-V2	BASO-V3	BASO-V4
1	1.00000	1.00000	1.00000	1.00000	1.00000	0.50000	0.75000
2	0.84375	0.49451	0.87610	0.00793	0.00291	0.00405	0.07292
3	0.62500	0.06250	0.12500	0.03125	0.03125	0.03125	0.03125
4	0.16943	0.86007	0.62471	0.32693	0.29054	0.98962	0.91862
5	0.86084	0.65466	0.38187	0.87872	0.65219	0.97015	0.17463
6	0.34819	0.90557	0.71063	0.01254	0.00993	0.05351	0.01186
7	1.00000	0.77930	0.07104	0.00062	0.00018	0.00017	0.00012
8	0.07065	0.00313	0.00012	0.00013	0.00023	0.00110	0.00013
9	0.22314	0.78955	0.54346	0.45648	0.78979	0.62402	0.95215
10	0.79688	0.81250	0.64063	0.13232	0.00452	0.18457	0.20471
11	1.00000	0.87842	0.55469	1.00000	0.74951	0.17578	0.28906
12	0.30005	0.28784	0.09404	0.25745	0.59473	0.30518	0.00537
13	0.02708	0.03358	0.11983	0.00010	0.00015	0.00012	0.00018
14	0.18378	0.00612	0.00115	0.00047	0.05209	0.03583	0.13150
15	0.13647	0.30902	0.15381	0.08838	0.12793	0.26953	0.40240
16	0.31183	0.88748	0.91663	0.03703	0.00647	0.40670	0.02708
17	0.74304	0.77497	0.02102	0.02008	0.00837	0.12993	0.19347
18	0.65625	0.30859	0.19141	0.82471	0.80176	0.76855	0.41504
19	0.75000	0.12500	0.71875	0.26563	0.06250	0.25000	0.40625
20	0.00669	0.00018	9.00E-05	0.00010	9.00E-05	9.00E-05	9.00E-05
21	0.45334	0.16387	0.74616	0.22662	0.00103	0.01288	0.01322
22	1.00000	0.37500	0.18750	1.00000	0.62500	0.37500	0.37500
w/t/l	1/20/1	3/18/1	4/18/0	6/12/4	7/11/4	6/14/2	5/13/4

Table 10. Result of best fitness value of six different algorithms on 22 datasets.

	Best fitness value					
Dataset	BASO	BBA	BDE	BFPA	PSO	BSSA
1	**0.0040**	**0.0040**	**0.0040**	**0.0040**	**0.0040**	**0.0040**
2	**0.1148**	0.1280	0.1576	0.1423	0.1214	0.1270
3	**0.0339**	**0.0339**	**0.0339**	**0.0339**	**0.0339**	**0.0339**
4	0.1258	0.1252	0.1622	**0.1198**	0.1317	0.1388
5	0.5933	**0.5681**	0.6017	0.5831	0.5987	0.5993
6	**0.2192**	0.2216	0.2579	0.2214	0.2293	0.2607
7	**0.1272**	0.1276	0.2056	0.1694	0.1284	0.1280
8	**0.0659**	0.0810	0.1192	0.1170	0.0691	0.0935
9	0.0335	0.0335	0.0570	**0.0254**	0.0440	0.0353
10	0.0388	0.0330	0.0426	0.0337	**0.0221**	0.0330
11	**0.0303**	**0.0303**	0.0307	**0.0303**	**0.0303**	**0.0303**
12	0.1987	0.2021	0.2701	0.2365	**0.1354**	0.2683
13	0.0942	0.0778	0.1053	0.0986	**0.0568**	0.0993
14	0.3247	0.3324	0.3532	0.3543	**0.3228**	0.3408
15	**0.0151**	0.0163	0.0266	0.0175	**0.0151**	0.0188
16	0.1463	**0.1425**	0.1506	0.1506	0.1450	0.1454
17	0.2058	0.1995	0.2269	0.2173	**0.1929**	0.2129
18	0.2727	**0.2696**	0.2822	**0.2696**	0.2727	**0.2696**
19	0.0453	**0.0406**	0.0547	**0.0406**	**0.0406**	**0.0406**
20	**0.1743**	0.2767	0.3035	0.2944	0.2835	0.2589
21	0.0745	0.0731	0.0796	0.0771	**0.0641**	0.0856
22	**0.0153**	0.0188	0.0190	0.0190	0.0179	0.0179

proposed BASO approaches. Note that the best result for each approach is bolded. As can be observed, BASO-S1 and BASO-S2 scored the lowest best fitness values on most of the datasets (8 datasets), followed by BASO-S3 and BASO-V4 (6 datasets). Out of twenty-two datasets, BASO-S1 and BASO-V1 perceived the smallest worst fitness value in 5 datasets. As

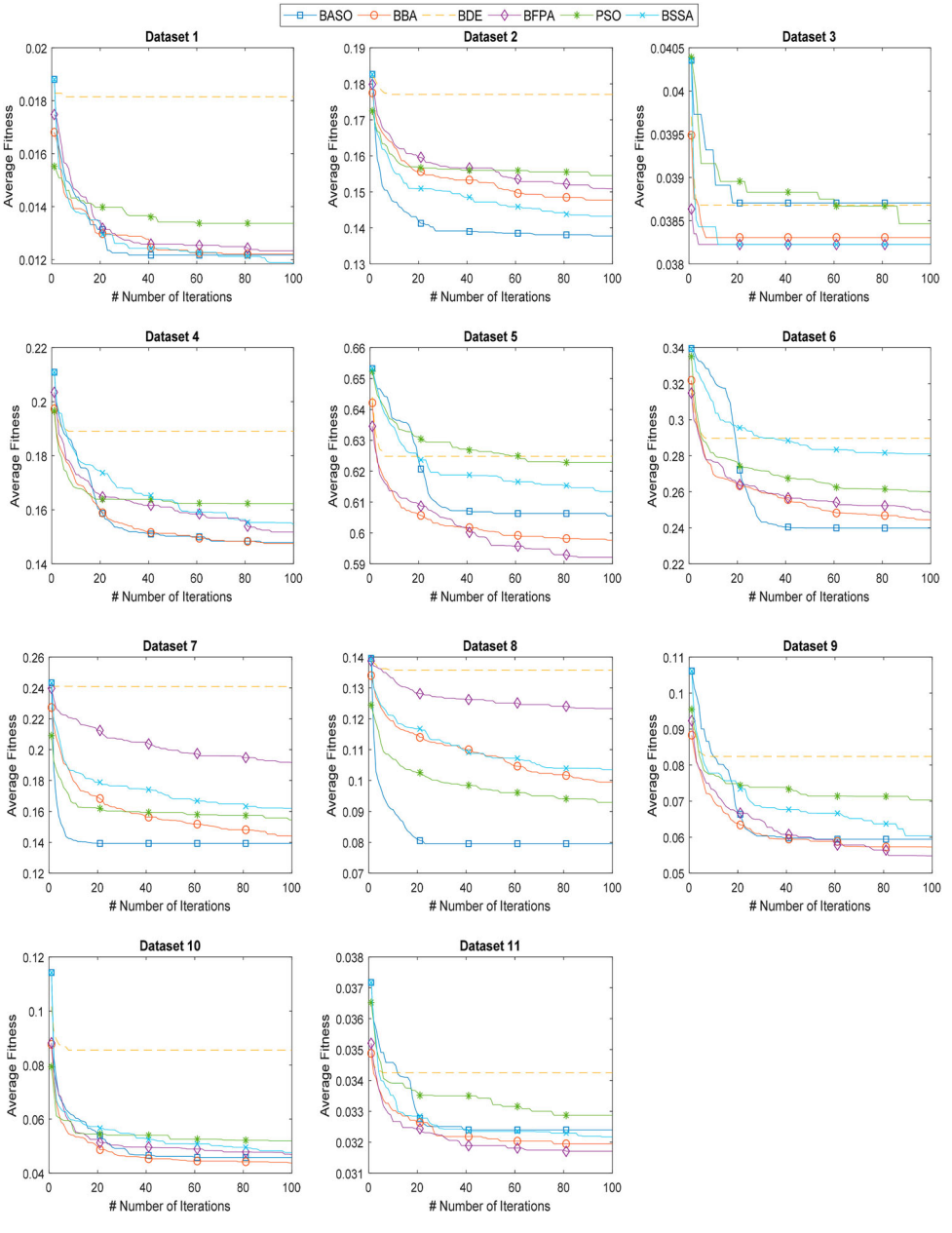

Figure 3. Convergence curve of six different algorithms for dataset 1–11.

for mean fitness value, the best approach was found to be BASO-S1, which provided better performance than other approaches in feature selection.

Tables 7 and 8 outline the experimental results of accuracy and feature size of proposed BASO approaches. Inspecting the result in Table 7, BASO-S1 perceived the highest accuracy on 6 datasets, followed by BASO-S2 and BASO-V2 (4 datasets). Eventually, BASO-S1 outperformed other approaches in evaluating the significant features, thus leading to excellent

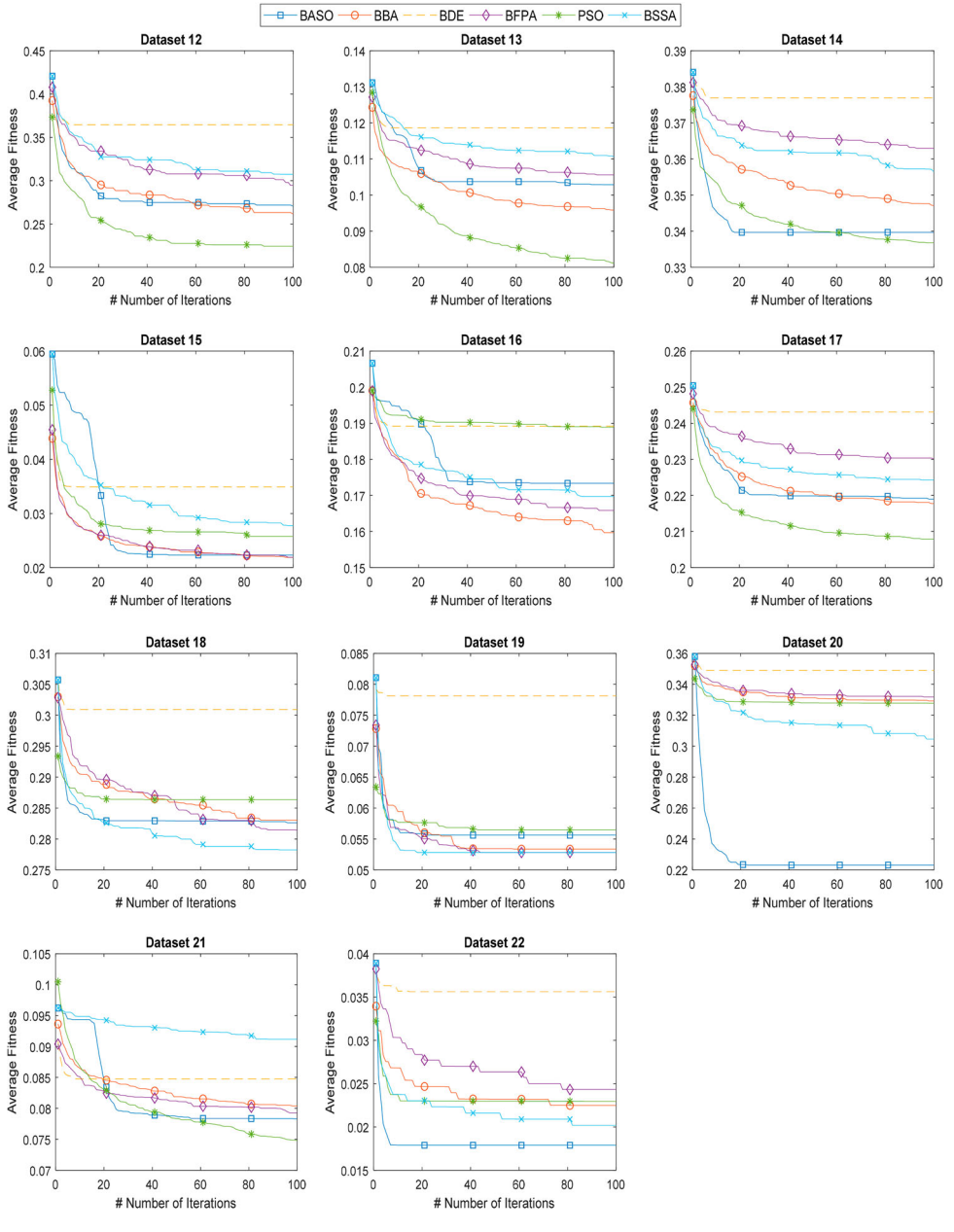

Figure 4. Convergence curve of six different algorithms for dataset 12–22.

results. In Table 8, BASO-S1 obtained the smallest number of selected features (feature size) on 9 datasets. In comparison with other approaches, BASO-S1 can usually provide a smaller feature size. Based on the results obtained, the BASO that contributed to the optimal performance was BASO-S1.

Furthermore, the Wilcoxon signed-rank test with a 5% significant level ($\alpha = 0.05$) is used to examine whether the performance of BASO is significantly better than other algorithms.

Table 11. Result of worst fitness value of six different algorithms on 22 datasets.

	Worst fitness value					
Dataset	BASO	BBA	BDE	BFPA	PSO	BSSA
1	0.0209	**0.0181**	0.0276	**0.0181**	0.0209	**0.0181**
2	**0.1523**	0.1803	0.1977	0.1576	0.1864	0.1600
3	**0.0421**	**0.0421**	0.0446	**0.0421**	**0.0421**	**0.0421**
4	**0.1611**	0.1823	0.2112	0.1699	0.1889	0.1753
5	0.6485	0.6161	0.6587	**0.6095**	0.6527	0.6239
6	0.2717	0.2694	0.3067	**0.2667**	0.2953	0.3019
7	**0.1610**	0.1857	0.3264	0.2186	0.2206	0.1852
8	**0.1061**	0.1167	0.1501	0.1323	0.1107	0.1133
9	0.0960	0.0861	0.1176	**0.0663**	0.1126	0.0928
10	0.0578	**0.0527**	0.1700	0.0570	0.0737	0.0570
11	0.0354	0.0339	0.0376	**0.0336**	0.0365	0.0354
12	0.3314	0.3345	0.4046	0.3364	**0.2997**	0.3668
13	0.1183	0.1137	0.1293	0.1141	**0.1032**	0.1161
14	0.3641	0.3613	0.3987	0.3717	**0.3533**	0.3651
15	0.0275	0.0273	0.0560	**0.0257**	0.0350	0.0353
16	0.1929	0.1835	0.2097	**0.1806**	0.2070	0.1826
17	0.2336	0.2308	0.2556	0.2425	**0.2242**	0.2406
18	0.2990	0.2986	0.3231	0.2925	0.3037	**0.2843**
19	0.0609	0.0609	0.1080	**0.0594**	0.0689	**0.0594**
20	**0.3019**	0.3668	0.3927	0.3690	0.3738	0.3591
21	0.0837	0.0875	0.0886	**0.0823**	0.0843	0.0952
22	**0.0302**	0.0331	0.0616	0.0485	0.0463	0.0322

In this test, if the *p*-value is less than 0.05, then there is a significant difference between the performance of two algorithms; otherwise, the performances of the two algorithms are similar. Table 9 outlines the *p*-value of the Wilcoxon signed-rank test for BASO-S1 and other approaches. In Table 9, the symbols of "*w* / *t* / *l*" indicate the proposed BASO-S1 was superior to (win), equal to (tie), and inferior to (lose) other algorithms. The result shows that the classification performances of BASO-S1 and BASO-S2 were similar. However, by making a one to one comparison (BASO-S1 versus BASO-S2), the algorithm that contributed to the optimal classification performance was found to be BASO-S1. From the point of view, BASO-S1 showed superior performance against other approaches in feature selection. Hence, only BASO with transfer function S1 is used in the rest of this paper.

5.4. Comparison with Other Algorithms

In the second part of the experiment, the performance of the proposed BASO is further compared with BBA, BDE, BFPA, PSO, and BSSA. Figure 3 and Figure 4 illustrate the convergence curves of proposed BASO and other algorithms on 22 datasets. From Figure 3 and Figure 4, BASO contributed better convergence on most of the datasets, especially on dataset 2, 7, 8, 20, and 22. This finding indicates that BASO is highly capable of searching for an optimal solution, which leads to satisfactory performance. Taking dataset 22 (Leukemia) as an example, BASO converged faster than other algorithms to find the global optimum. In comparison with other algorithms, BASO offered a very high diversity, thus resulting in optimal fitness value.

Tables 10, 11, and 12 show the experimental results of the best, worst, and mean fitness values of six different algorithms. In these Tables, the best result is highlighted with bold text. As can be seen, the performance of BASO was competitive against others. Among

Table 12. Result of mean fitness value of six different algorithms on 22 datasets.

	Mean fitness value (STD)					
Dataset	BASO	BBA	BDE	BFPA	PSO	BSSA
1	0.0122 (0.0052)	0.0122 (0.0049)	0.0182 (0.0055)	0.0123 (0.0051)	0.0134 (0.0051)	**0.0118** (**0.0049**)
2	**0.1378** (**0.0102**)	0.1477 (0.0111)	0.1770 (0.0124)	0.1509 (0.0059)	0.1545 (0.0186)	0.1432 (0.0083)
3	0.0387 (0.0033)	0.0383 (0.0031)	0.0387 (0.0031)	**0.0382** (**0.0030**)	0.0385 (0.0031)	**0.0382** (**0.0030**)
4	0.1478 (0.0110)	**0.1476** (**0.0144**)	0.1890 (0.0146)	0.1518 (0.0114)	0.1623 (0.0137)	0.1545 (0.0107)
5	0.6055 (0.0133)	0.5977 (0.0122)	0.6248 (0.0152)	**0.5921** (**0.0081**)	0.6228 (0.0164)	0.6134 (0.0085)
6	**0.2398** (**0.0128**)	0.2443 (0.0138)	0.2897 (0.0136)	0.2482 (0.0105)	0.2601 (0.0173)	0.2809 (0.0116)
7	**0.1393** (**0.0092**)	0.1442 (0.0145)	0.2408 (0.0259)	0.1917 (0.0120)	0.1545 (0.0227)	0.1620 (0.0118)
8	**0.0796** (**0.0094**)	0.0995 (0.0114)	0.1358 (0.0080)	0.1233 (0.0041)	0.0929 (0.0103)	0.1035 (0.0053)
9	0.0594 (0.0172)	0.0573 (0.0149)	0.0824 (0.0164)	**0.0548** (**0.0101**)	0.0703 (0.0192)	0.0604 (0.0141)
10	0.0458 (0.0054)	**0.0437** (**0.0066**)	0.0855 (0.0414)	0.0468 (0.0057)	0.0519 (0.0117)	0.0476 (0.0071)
11	0.0324 (0.0012)	0.0319 (0.0011)	0.0342 (0.0019)	**0.0317** (**0.0010**)	0.0329 (0.0015)	0.0322 (0.0012)
12	0.2703 (0.0302)	0.2615 (0.0380)	0.3645 (0.0448)	0.2946 (0.0318)	**0.2234** (**0.0401**)	0.3074 (0.0252)
13	0.1029 (0.0059)	0.0959 (0.0099)	0.1186 (0.0064)	0.1056 (0.0041)	**0.0811** (**0.0121**)	0.1107 (0.0042)
14	0.3397 (0.0099)	0.3471 (0.0088)	0.3769 (0.0096)	0.3630 (0.0051)	**0.3368** (**0.0084**)	0.3565 (0.0058)
15	0.0223 (0.0034)	**0.0219** (**0.0023**)	0.0349 (0.0072)	**0.0219** (**0.0020**)	0.0258 (0.0063)	0.0278 (0.0045)
16	0.1733 (0.0137)	**0.1597** (**0.0115**)	0.1892 (0.0138)	0.1658 (0.0068)	0.1889 (0.0207)	0.1697 (0.0099)
17	0.2189 (0.0074)	0.2178 (0.0079)	0.2431 (0.0086)	0.2301 (0.0077)	**0.2079** (**0.0071**)	0.2243 (0.0079)
18	0.2826 (0.0056)	0.2831 (0.0080)	0.3010 (0.0097)	0.2814 (0.0058)	0.2864 (0.0072)	**0.2782** (**0.0046**)
19	0.0556 (0.0045)	0.0534 (0.0056)	0.0781 (0.0200)	**0.0528** (**0.0053**)	0.0565 (0.0063)	**0.0528** (**0.0053**)
20	**0.2231** (**0.0299**)	0.3293 (0.0231)	0.3490 (0.0225)	0.3318 (0.0185)	0.3278 (0.0245)	0.3046 (0.0295)
21	0.0783 (0.0022)	0.0804 (0.0042)	0.0847 (0.0023)	0.0793 (0.0014)	**0.0749** (**0.0059**)	0.0912 (0.0024)
22	**0.0179** (**0.0052**)	0.0225 (0.0063)	0.0356 (0.0122)	0.0243 (0.0081)	0.0229 (0.0095)	0.0202 (0.0052)

rivals, BASO achieved the optimal best and mean fitness values on 10 and 6 datasets, respectively. In terms of consistency, BFPA yielded the lowest STD of fitness value on most of the datasets, which showed high consistency in this work.

Based on the results obtained, BASO has shown to be the best algorithm that contributed to the optimal performance in this work. The observed improvements in searching the global optimum (best solution) are attributed to the attractive force or repulsive force between the atom and its neighbour in BASO. Initially, the attractive force promotes exploration and enable the atoms to search globally. At the end of the iteration, the repulsive force encourages the atoms to search locally around the promising region. This, in turn,

Table 13. Result of accuracy of six different algorithms on 22 datasets.

	Average accuracy (STD)						
Dataset	Full features	BASO	BBA	BDE	BFPA	PSO	BSSA
1	0.9776 (0.0060)	0.9912 (0.0056)	0.9910 (0.0053)	0.9867 (0.0055)	**0.9914** (**0.0053**)	0.9895 (0.0057)	**0.9914** (**0.0053**)
2	0.7750 (0.0068)	**0.8633** (**0.0103**)	0.8543 (0.0113)	0.8267 (0.0128)	0.8523 (0.0058)	0.8467 (0.0190)	0.8580 (0.0084)
3	0.9647 (0.0065)	0.9663 (0.0055)	0.9680 (0.0046)	0.9680 (0.0041)	**0.9683** (**0.0043**)	0.9673 (0.0048)	**0.9683** (**0.0043**)
4	0.7618 (0.0141)	**0.8557** (**0.0115**)	0.8554 (0.0144)	0.8157 (0.0148)	0.8529 (0.0117)	0.8404 (0.0138)	0.8486 (0.0110)
5	0.3161 (0.0081)	0.3950 (0.0132)	0.4023 (0.0132)	0.3764 (0.0159)	**0.4092** (**0.0085**)	0.3762 (0.0163)	0.3865 (0.0088)
6	0.6957 (0.0088)	**0.7662** (**0.0130**)	0.7590 (0.0142)	0.7158 (0.0135)	0.7560 (0.0108)	0.7425 (0.0173)	0.7213 (0.0121)
7	0.6704 (0.0044)	**0.8603** (**0.0091**)	0.8560 (0.0145)	0.7618 (0.0254)	0.8107 (0.0120)	0.8457 (0.0225)	0.8392 (0.0115)
8	0.8420 (0.0046)	**0.9209** (**0.0094**)	0.9024 (0.0109)	0.8679 (0.0079)	0.8801 (0.0040)	0.9086 (0.0101)	0.8981 (0.0054)
9	0.8735 (0.0195)	0.9450 (0.0167)	0.9470 (0.0145)	0.9235 (0.0163)	**0.9500** (**0.0103**)	0.9335 (0.0195)	0.9440 (0.0143)
10	0.6982 (0.0143)	0.9594 (0.0057)	**0.9609** (**0.0064**)	0.9206 (0.0417)	0.9579 (0.0061)	0.9515 (0.0119)	0.9559 (0.0075)
11	0.9717 (0.0021)	0.9733 (0.0018)	0.9736 (0.0017)	0.9725 (0.0024)	**0.9742** (**0.0017**)	0.9725 (0.0020)	0.9731 (0.0017)
12	0.5167 (0.0315)	0.7300 (0.0304)	0.7400 (0.0384)	0.6383 (0.0449)	0.7083 (0.0322)	**0.7767** (**0.0406**)	0.6933 (0.0256)
13	0.8687 (0.0075)	0.9018 (0.0056)	0.9080 (0.0102)	0.8872 (0.0066)	0.8997 (0.0042)	**0.9221** (**0.0123**)	0.8923 (0.0043)
14	0.5319 (0.0247)	0.6590 (0.0095)	0.6542 (0.0090)	0.6254 (0.0099)	0.6397 (0.0051)	**0.6636** (**0.0084**)	0.6441 (0.0057)
15	0.8650 (0.0092)	**0.9843** (**0.0032**)	0.9832 (0.0025)	0.9724 (0.0074)	**0.9843** (**0.0021**)	0.9790 (0.0066)	0.9769 (0.0048)
16	0.7788 (0.0122)	0.8310 (0.0132)	**0.8438** (**0.0115**)	0.8160 (0.0139)	0.8381 (0.0068)	0.8115 (0.0227)	0.8335 (0.0101)
17	0.7392 (0.0092)	0.7825 (0.0079)	0.7847 (0.0081)	0.7603 (0.0092)	0.7736 (0.0079)	**0.7937** (**0.0071**)	0.7774 (0.0080)
18	0.6678 (0.0074)	0.7171 (0.0060)	0.7174 (0.0083)	0.7006 (0.0086)	0.7191 (0.0060)	0.7136 (0.0069)	**0.7216** (**0.0052**)
19	0.8810 (0.0076)	0.9471 (0.0043)	0.9493 (0.0054)	0.9262 (0.0195)	**0.9498** (**0.0052**)	0.9464 (0.0062)	**0.9498** (**0.0052**)
20	0.6108 (0.0255)	**0.7758** (**0.0296**)	0.6713 (0.0235)	0.6533 (0.0230)	0.6708 (0.0186)	0.6717 (0.0248)	0.6962 (0.0298)
21	0.9159 (0.0016)	**0.9300** (**0.0022**)	0.9241 (0.0042)	0.9228 (0.0023)	0.9265 (0.0015)	0.9291 (0.0061)	0.9124 (0.0025)
22	0.9457 (0.0136)	**0.9836** (**0.0052**)	0.9821 (0.0063)	0.9693 (0.0125)	0.9814 (0.0082)	0.9807 (0.0096)	**0.9836** (**0.0052**)

will make a well stable balance between global and local search, thus resulting in superior performance.

Table 13 exhibits the result of the accuracy of six different algorithms on 22 datasets. As can be seen, the classification performance has shown great improvement when the feature selection algorithms are utilised. The results clearly show the effectiveness of feature selection in classification tasks. In Table 13, BASO contributed the highest accuracy on dataset 2, 4, 6, 7, 8, 15, 20, 21, and 22 (nine datasets). The second best algorithm was found to be BFPA (seven datasets), followed by BSSA (five datasets). Correspondingly, BASO overtakes other algorithms in evaluating the significant features from the large available feature set.

Table 14. Result of feature size of six different algorithms on 22 datasets.

	Average number of selected features (STD)					
Dataset	BASO	BBA	BDE	BFPA	PSO	BSSA
1	3.45	3.25	4.95	3.85	**3.00**	3.35
	(0.89)	(0.91)	(1.32)	(0.59)	(**1.03**)	(0.88)
2	**4.70**	6.65	10.35	8.95	5.15	5.05
	(3.84)	(2.37)	(2.23)	(2.52)	(1.81)	(1.99)
3	**2.15**	2.65	2.80	2.75	2.45	2.75
	(**0.99**)	(0.88)	(0.70)	(0.79)	(0.89)	(0.79)
4	9.00	7.85	11.85	10.95	**7.70**	8.30
	(2.94)	(2.06)	(2.06)	(1.32)	(**1.66**)	(1.53)
5	11.10	10.05	12.65	12.35	**8.90**	10.30
	(2.15)	(2.06)	(2.43)	(1.18)	(**1.83**)	(1.66)
6	29.10	20.15	29.20	23.10	18.10	**17.45**
	(1.41)	(1.95)	(2.98)	(2.36)	(1.89)	(**2.58**)
7	**2.60**	4.25	13.55	11.60	4.80	7.50
	(**0.94**)	(1.12)	(2.63)	(1.35)	(1.58)	(2.35)
8	**4.10**	9.80	16.80	15.80	8.25	9.20
	(**1.07**)	(2.80)	(4.80)	(2.84)	(2.59)	(1.67)
9	7.95	7.65	10.65	8.50	**7.15**	7.90
	(2.26)	(1.69)	(1.95)	(1.24)	(**1.14**)	(1.68)
10	7.30	6.50	8.95	6.65	**5.00**	5.05
	(2.39)	(1.47)	(1.23)	(1.69)	(**1.03**)	(1.23)
11	5.40	5.25	6.35	5.55	5.05	**5.00**
	(1.05)	(1.16)	(1.57)	(1.23)	(1.28)	(**1.17**)
12	**16.70**	22.90	36.10	32.80	17.75	21.45
	(**11.43**)	(3.77)	(5.09)	(2.98)	(3.24)	(3.10)
13	94.30	78.85	115.65	104.20	**67.10**	68.60
	(29.74)	(8.22)	(22.27)	(5.72)	(**6.03**)	(4.68)
14	**57.80**	133.40	170.95	174.80	102.75	115.25
	(**25.95**)	(8.48)	(31.90)	(10.07)	(7.61)	(12.06)
15	23.05	17.85	25.60	21.65	17.10	**16.75**
	(3.35)	(1.53)	(2.95)	(2.13)	(2.20)	(**3.01**)
16	13.20	11.25	15.35	12.20	**5.10**	10.65
	(2.59)	(2.12)	(2.70)	(1.77)	(**4.40**)	(1.84)
17	**32.60**	41.70	52.15	53.65	33.50	34.60
	(**17.78**)	(4.52)	(7.26)	(4.38)	(3.82)	(4.98)
18	**2.50**	3.30	4.55	3.35	2.85	2.65
	(**1.50**)	(1.45)	(1.67)	(1.35)	(1.35)	(0.99)
19	2.30	2.20	3.55	2.15	**2.40**	2.15
	(0.57)	(0.52)	(1.05)	(0.49)	(**0.60**)	(0.49)
20	**37.05**	118.95	178.25	184.85	84.60	119.25
	(**31.29**)	(8.10)	(23.22)	(7.66)	(6.77)	(12.20)
21	230.30	134.30	213.60	166.85	118.25	**112.55**
	(12.50)	(6.65)	(21.95)	(6.19)	(7.75)	(**11.60**)
22	**1151.55**	3415.00	3718.60	4228.20	2747.80	2793.60
	(**162.39**)	(56.60)	(382.75)	(307.49)	(49.72)	(46.76)

Table 14 displays the result of feature size (number of selected features) of six different algorithms on 22 datasets. In terms of feature size, BASO can usually affirm the minimal number of selected features, followed by BSSA. Based on the result obtained, BASO gave the smallest feature size on 10 datasets. The results validate that BASO was good at eliminating the irrelevant and redundant features, thus, providing high prediction accuracy. Among rivals, BASO is highly capable of finding significant features, which can contribute to better classification performance.

Table 15 outlines the *p*-value of the Wilcoxon signed-rank test for BASO and its competitors. The experimental results clearly show that proposed BASO was a useful tool for

Table 15. The *p*-value of Wilcoxon signed rank test for BASO and other algorithms.

	P-value				
Dataset	BBA	BDE	BFPA	PSO	BSSA
1	1.00000	0.00024	1.00000	0.03125	1.00000
2	0.02312	8.00E-05	0.00081	0.00397	0.02075
3	0.06250	0.12500	0.03125	0.25000	0.03125
4	0.96399	8.00E-05	0.36023	0.00218	0.01930
5	0.12686	0.00067	0.00022	0.00019	0.02281
6	0.05721	0.00010	0.01214	0.00101	9.00E-05
7	0.29761	9.00E-05	9.00E-05	0.00710	0.00023
8	0.00017	9.00E-05	9.00E-05	0.00177	0.00015
9	0.55957	0.00040	0.16235	0.03479	0.87305
10	0.68359	0.00018	0.50781	0.01910	0.10535
11	0.55957	0.31238	0.04297	0.12109	0.57422
12	0.34930	0.00017	0.02832	0.00072	0.00049
13	0.01953	0.00011	0.11445	0.00010	0.00024
14	0.14573	9.00E-05	0.00014	0.04674	0.00043
15	0.32361	8.00E-05	0.77246	0.00465	0.00032
16	0.00044	0.00013	0.01415	0.00260	0.44436
17	0.28513	9.00E-05	0.00018	0.00020	0.01421
18	0.88477	0.00032	0.35205	0.08960	0.00195
19	0.12500	0.00012	0.03125	0.64063	0.03125
20	9.00E-05	9.00E-05	8.00E-05	0.00010	9.00E-05
21	0.00011	9.00E-05	0.00011	0.40892	8.00E-05
22	0.62500	6.00E-05	0.53125	0.37500	1.00000
w/t/l	4/16/2	20/2/0	9/8/5	12/6/4	13/6/3

Table 16. Result of computational time of six different algorithms on 22 datasets.

	Average computational time (s)					
Dataset	BASO	BBA	BDE	BFPA	PSO	BSSA
1	1.8254	1.8174	1.8712	**1.8159**	1.8664	1.8834
2	1.2675	1.2770	1.2716	**1.1550**	1.3391	1.2442
3	1.3596	1.3679	1.3791	1.3641	1.3653	**1.2748**
4	1.5718	1.5405	1.5863	1.5428	**1.4959**	1.5056
5	3.2763	3.2902	3.3005	3.2438	3.1721	**3.0454**
6	3.3393	3.2148	3.2440	3.2011	**3.0942**	3.1008
7	3.4725	3.4256	3.4086	3.3069	**3.1306**	3.4239
8	3.7714	3.9957	3.7992	3.9652	**3.6410**	3.9096
9	1.0513	0.9785	1.0150	1.0100	**0.9508**	0.9807
10	1.6760	1.5991	1.6388	1.6370	**1.5674**	1.6049
11	**8.8065**	9.3924	8.9310	9.3301	9.1552	9.1849
12	0.7488	0.6528	0.6636	0.6657	**0.6360**	0.6745
13	7.3580	7.2602	7.5168	7.5534	**6.5645**	7.3524
14	7.9689	**7.6515**	8.4535	7.7732	7.6605	7.8091
15	4.0536	4.0044	4.3176	3.9508	**3.8166**	4.0245
16	2.5885	2.5977	2.7386	2.6166	**2.3624**	2.4230
17	4.5891	4.4750	4.8109	4.4712	**4.3223**	4.5517
18	6.5592	6.7840	7.0325	**6.3694**	6.6544	6.8848
19	**1.8501**	1.8522	1.8931	1.8680	1.8613	1.9196
20	2.7290	2.3630	2.4706	2.5131	**2.1297**	2.3888
21	84.0517	74.4592	84.6066	78.8079	73.4817	**72.9405**
22	41.2162	31.2371	32.726	34.6249	**28.5456**	32.1246

feature selection problems. Furthermore, the result of computational time of six different algorithms is shown in Table 16. As can be observed, PSO was found to be the fastest feature selection algorithm, which contributes to the lowest computational time. Even though

BASO cannot ensure the fastest processing speed, however, it can usually select a smaller number of significant features to achieve the highest classification accuracy.

There are several limitations can be found within this research. First, the parameter settings of BASO are fixed in this work. For other applications, the users are recommended to test the parameters in order to get the best performance. Second, we only apply the KNN as the learning algorithm to compute the classification performance in feature selection. However, other popular and powerful learning algorithms such as support vector machine (SVM), convolutional neural network (CNN), and random forest (RF) can be implemented to boost the result.

6. Conclusion

In this paper, new binary variants of ASO, namely binary atom search optimisation (BASO), are proposed to solve the feature selection problem in classification tasks. Eight transfer functions (from S-shaped and V-shaped family) are used in this study to convert the continuous ASO into a binary version. Besides, a novel position updating rule is designed to enhance the performance of BASO in feature selection. Twenty-two datasets are utilised to validate the performance of the proposed BASO in feature selection. Our results indicated that BASO with S-shaped transfer function (S1) is highly capable of selecting the relevant features as compared to other BASO approaches. Moreover, the performance of the proposed BASO was further compared with BBA, BDE, BFPA, PSO, and BSSA. The experimental results proved the superiority of proposed BASO in terms of classification accuracy, feature size, and convergence speed. Future research can focus on the efficacy of BASO in the classification problem, as well as other binary optimisation tasks.

Acknowledgement

The authors would like to thank the Skim Zamalah UTeM for supporting this research.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

ORCID

Jingwei Too http://orcid.org/0000-0001-6908-1038

References

Al-Ani, A. (2005). *Feature Subset Selection Using Ant Colony Optimization*. https://opus.lib.uts.edu.au/handle/10453/6181

Al-Tashi, Q., Kadir, S. J. A., Rais, H. M., Mirjalili, S., & Alhussian, H. (2019). Binary optimization using hybrid grey wolf optimization for feature selection. *IEEE Access*, *7*, 39496–39508. https://doi.org/10.1109/ACCESS.2019.2906757

Arora, S., & Anand, P. (2019). Binary butterfly optimization approaches for feature selection. *Expert Systems with Applications*, *116*, 147–160. https://doi.org/10.1016/j.eswa.2018.08.051

Chuang, L.-Y., Chang, H.-W., Tu, C.-J., & Yang, C.-H. (2008). Improved binary PSO for feature selection using gene expression data. *Computational Biology and Chemistry*, *32*(1), 29–38. https://doi.org/10.1016/j.compbiolchem.2007.09.005

Emary, E., Zawbaa, H. M., & Hassanien, A. E. (2016a). Binary grey wolf optimization approaches for feature selection. *Neurocomputing*, *172*, 371–381. https://doi.org/10.1016/j.neucom.2015.06.083

Emary, E., Zawbaa, H. M., & Hassanien, A. E. (2016b). Binary ant lion approaches for feature selection. *Neurocomputing*, *213*, 54–65. https://doi.org/10.1016/j.neucom.2016.03.101

Faris, H., Hassonah, M. A., Al-Zoubi, A. M., Mirjalili, S., & Aljarah, I. (2017). A multi-verse optimizer approach for feature selection and optimizing SVM parameters based on a robust system architecture. *Neural Computing and Applications*, *30*(8), 2355–2369. https://doi.org/10.1007/s00521-016-2818-2

Faris, H., Mafarja, M. M., Heidari, A. A., Aljarah, I., Al-Zoubi, A. M., Mirjalili, S., & Fujita, H. (2018). An efficient binary salp swarm algorithm with crossover scheme for feature selection problems. *Knowledge-Based Systems*, *154*, 43–67. https://doi.org/10.1016/j.knosys.2018.05.009

Huang, C.-L., & Wang, C.-J. (2006). A GA-based feature selection and parameters optimizationfor support vector machines. *Expert Systems with Applications*, *31*(2), 231–240. https://doi.org/10.1016/j.eswa.2005.09.024

Kennedy, J. (2011). Particle swarm Optimization. In Claude Sammut & Geoffrey I. Webb (Eds.), *Encyclopedia of machine learning* (pp. 760–766). Springer.https://doi.org/10.1007/978-0-387-30164-8_630.

Kennedy, J., & Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. *Computational Cybernetics and Simulation 1997 IEEE International Conference on Systems, Man, and Cybernetics*, *5*, 4104–4108. https://doi.org/10.1109/ICSMC.1997.637339

Labani, M., Moradi, P., Ahmadizar, F., & Jalili, M. (2018). A novel multivariate filter method for feature selection in text classification problems. *Engineering Applications of Artificial Intelligence*, *70*, 25–37. https://doi.org/10.1016/j.engappai.2017.12.014

Liu, M., Xu, C., Luo, Y., Xu, C., Wen, Y., & Tao, D. (2018). Cost-Sensitive feature selection by Optimizing F-Measures. *IEEE Transactions on Image Processing*, *27*(3), 1323–1335. https://doi.org/10.1109/TIP.2017.2781298

Mafarja, M., Aljarah, I., Faris, H., Hammouri, A. I., Al-Zoubi, A. M., & Mirjalili, S. (2019). Binary grasshopper optimisation algorithm approaches for feature selection problems. *Expert Systems with Applications*, *117*, 267–286. https://doi.org/10.1016/j.eswa.2018.09.015

Mafarja, M., Aljarah, I., Heidari, A. A., Faris, H., Fournier-Viger, P., Li, X., & Mirjalili, S. (2018). Binary dragonfly optimization for feature selection using time-varying transfer functions. *Knowledge-Based Systems*, *161*, 185–204. https://doi.org/10.1016/j.knosys.2018.08.003

Mafarja, M. M., & Mirjalili, S. (2017). Hybrid Whale Optimization algorithm with simulated annealing for feature selection. *Neurocomputing*, *260*, 302–312. https://doi.org/10.1016/j.neucom.2017.04.053

Mirhosseini, M., & Nezamabadi-pour, H. (2018). BICA: A binary imperialist competitive algorithm and its application in CBIR systems. *International Journal of Machine Learning and Cybernetics*, *9*(12), 2043–2057. https://doi.org/10.1007/s13042-017-0686-4

Mirjalili, S., & Lewis, A. (2013). S-shaped versus V-shaped transfer functions for binary particle swarm Optimization. *Swarm and Evolutionary Computation*, *9*, 1–14. https://doi.org/10.1016/j.swevo.2012.09.002

Mirjalili, S., Mirjalili, S. M., & Yang, X.-S. (2014). Binary bat algorithm. *Neural Computing and Applications*, *25*(3-4), 663–681. https://doi.org/10.1007/s00521-013-1525-5

Pashaei, E., & Aydin, N. (2017). Binary black hole algorithm for feature selection and classification on biological data. *Applied Soft Computing*, *56*, 94–106. https://doi.org/10.1016/j.asoc.2017.03.002

Peng, H., Long, F., & Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(8), 1226–1238. https://doi.org/10.1109/TPAMI.2005.159

Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA: A gravitational search algorithm. *Information Sciences*, *179*(13), 2232–2248. https://doi.org/10.1016/j.ins.2009.03.004

Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2010). BGSA: Binary gravitational search algorithm. *Natural Computing*, *9*(3), 727–745. https://doi.org/10.1007/s11047-009-9175-3

Rodrigues, D., Pereira, L. A. M., Nakamura, R. Y. M., Costa, K. A. P., Yang, X.-S., Souza, A. N., & Papa, J. P. (2014). A wrapper approach for feature selection based on Bat algorithm and optimum-path forest. *Expert Systems with Applications*, *41*(5), 2250–2258. https://doi.org/10.1016/j.eswa.2013.09.023

Rodrigues, D., Yang, X.-S., Souza, A. N. d., & Papa, J. P. (2015). Binary flower pollination algorithm and Its application to feature selection. In X.-S. Yang (Ed.), *Recent Advances in swarm Intelligence and Evolutionary Computation* (pp. 85–100). Springer. https://doi.org/10.1007/978-3-319-13826-8_5

Siddique, N., & Adeli, H. (2014). Water Drop algorithms. *International Journal on Artificial Intelligence Tools*, *23*(06), 1430002. https://doi.org/10.1142/S0218213014300026

Too, J., Abdullah, A. R., Mohd Saad, N., & Mohd Ali, N. (2018). Feature selection based on binary tree growth algorithm for the classification of Myoelectric Signals. *Machines*, *6*(4), 65. https://doi.org/10.3390/machines6040065

UCI Machine Learning Repository. (n.d.). Retrieved March 24, 2019, from https://archive.ics.uci.edu/ml/index.php

Wang, C., Hu, Q., Wang, X., Chen, D., Qian, Y., & Dong, Z. (2018). Feature selection based on Neighborhood Discrimination Index. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(7), 1–14. https://doi.org/10.1109/TNNLS.2017.2710422

Wang, M., Wan, Y., Ye, Z., & Lai, X. (2017). Remote sensing image classification based on the optimal support vector machine and modified binary coded ant colony optimization algorithm. *Information Sciences*, *402*, 50–68. https://doi.org/10.1016/j.ins.2017.03.027

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82. https://doi.org/10.1109/4235.585893

Xue, B., Zhang, M., & Browne, W. N. (2014). Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms. *Applied Soft Computing*, *18*(Supplement C), 261–276. https://doi.org/10.1016/j.asoc.2013.09.018

Zhao, W., Wang, L., & Zhang, Z. (2019a). Atom search optimization and its application to solve a hydrogeologic parameter estimation problem. *Knowledge-Based Systems*, *163*, 283–304. https://doi.org/10.1016/j.knosys.2018.08.030

Zhao, W., Wang, L., & Zhang, Z. (2019b). A novel atom search optimization for dispersion coefficient estimation in groundwater. *Future Generation Computer Systems*, *91*, 601–610. https://doi.org/10.1016/j.future.2018.05.037

Zorarpacı, E., & Özel, S. A. (2016). A hybrid approach of differential evolution and artificial bee colony for feature selection. *Expert Systems with Applications*, *62*, 91–103. https://doi.org/10.1016/j.eswa.2016.06.004