Dragonfly Algorithm: Theory, Literature Review, and Application in Feature Selection

Majdi Mafarja, Ali Asghar Heidari, Hossam Faris, Seyedali Mirjalili and Ibrahim Aljarah

Abstract In this chapter, a wrapper-based feature selection algorithm is designed and substantiated based on the binary variant of Dragonfly Algorithm (BDA). DA is a successful, well-established metaheuristic that revealed superior efficacy in dealing with various optimization problems including feature selection. In this chapter we are going first present the inspirations and methamatical modeds of DA in details. Then, the performance of this algorithm is tested on a special type of datasets that contain a huge number of features with low number of samples. This type of datasets makes the optimization process harder, because of the large search space, and the lack of adequate samples to train the model. The experimental results showed the ability of DA to deal with this type of datasets better than other optimizers in the literature. Moreover, an extensive literature review for the DA is provided in this chapter.

M. Mafarja
Faculty of Engineering and Technology, Department of Computer Science, Birzeit University, PoBox 14, Birzeit, Palestine
e-mail: mmafarja@birzeit.edu

A. A. Heidari
School of Surveying and Geospatial Engineering, University of Tehran, Tehran, Iran
e-mail: as_heidari@ut.ac.ir

H. Faris · I. Aljarah
King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan
e-mail: hossam.faris@ju.edu.jo

I. Aljarah
e-mail: i.aljarah@ju.edu.jo

S. Mirjalili (✉)
Institute of Integrated and Intelligent Systems, Griffith University, Nathan, Brisbane, QLD 4111, Australia
e-mail: seyedali.mirjalili@griffithuni.edu.au

S. Mirjalili et al. (eds.), *Nature-Inspired Optimizers*, Studies in Computational Intelligence 811, https://doi.org/10.1007/978-3-030-12127-3_4

1 Introduction

Some creatures' behaviors have been the inspiration source for many successful optimization algorithms. The main behavior that inspired many researchers to develop new algorithms was the strategy that those creatures use to seek the food sources. Ant Colony Optimization (ACO) [25, 26] and Artificial Bees Colony (ABC) [51] were originally inspired by the behavior of ants and bees respectively in locating food sources and collecting their food. Swarming behavior is another source of inspiration that was used to propose new optimization algorithms. Particle Swarm Optimization (PSO) [25] is a primary swarm based optimization algorithm that mimics the swarming behavior of birds.

The key issue about all the previously mentioned creatures is that they live in groups or folks (called swarms) [53]. An individual in those swarms usually makes a decision based on local information from itself and from the interactions with the other swarm members, also from the environment. Such interactions are the main reason that contribute to the improvement of the social intelligence in these swarms. Most of the swarms contain different organisms from the same species (bees, ants, birds, etc.). By the intelligent collaboration (or swarm intelligence (SI)) between all individuals of the swarm, they have been found to be successful in carrying out specific tasks [35, 95].

Nature-inspired algorithms are population-based metaheuristics algorithms, that tempt to manipulate a set of solutions in each generation of the optimization process [1, 13, 33, 44, 45, 53]. Recently, many such algorithms were proposed and proved their ability to solve different complex optimization problems such as global function optimization [12, 42–44, 46], clustering analysis [4, 8, 10, 90], spam and intrusion detection systems [9, 11, 30, 32], optimizing neural networks [6, 7, 29, 31], link prediction [18], software effort estimation [36], and bio-informatics [16, 23].

As Feature Selection is known to be an NP-hard problem [5], metaheuristic algorithms in general and specially the population-based algorithms showed a superior performance in solving this problem.

PSO has been widely used to tackle FS problems. An improved PSO with local search strategy for FS was proposed in [80]. Another PSO approach with two crossover mechanisms was presented in [22]. Rajamohana and Umamaheswari [82] proposed a hybrid particle swarm optimization with shuffled frog leaping algorithm feature selection approach. An improved ACO was also proposed in [52]. A hybrid FS approach that combines differential evolution (DE) with ABC was proposed in [102].

Grey Wolf Optimizer (GWO) [79] is a recent SI algorithm that mimics the hierarchical organization of the grey wolves in nature. The GWO has been widely used in FS methods with much success [58]. Antlion Optimizer (ALO) was proposed by Mirjalili [74] in 2015, FS was one of the fields where ALO was applied successfully [70]. Whale Optimization Algorithm (WOA), is another SI algorithm that was recently proposed by Mirjalili and Lewis [78]. WOA was successfully applied in many FS methods [72, 73]. Grasshopper Optimization Algorithm (GOA) [87] is

another example of the nature inspired algorithms that was recently proposed and applied to many optimization problems including FS [5, 49, 69]. More recently, a Salp Swarm Algorithm (SSA) was proposed by Mirjalili et al. [76] and was used as a FS selection mechanism in [2].

Dragonfly Algorithm (DA) was recently proposed by Mirjalili [75]. DA is inspired by the behavior of dragonflies, and proved its ability outperform other well-regarded optimizers in the literature [68]. The DA has shown an excellent performance for several continuous, discrete, single-objective and multi-objective optimization problems compared to several state-of-the-art metaheuristic and evolutionary algorithms such as PSO and DE. In this chapter, the binary version of DA is used to serve as a selection mechanism and applied to medical datasets of high dimensional with small number of samples.

2 Feature Selection Problem

With the emergence of the high dimensional data in almost all of the real life fields, the knowledge discovery (KDD) process becomes very complex [61]. The high dimensionality of datasets usually causes many problems for the data mining techniques (e.g., classification), like over-fitting, and hence, decreasing the model performance, and increasing the required computational time. Therefore, reducing data dimensionality becomes highly demanded [59]. FS is a preprocessing step that aims to enhance the performance of the data mining and machine learning techniques by eliminating the redundant irreverent features.

In the FS process, two major steps should be carefully considered when designing a new FS method; selection (search) and evaluation [61]. In the selection step, the feature space has to be searched for a feature subset with a minimal number of features, and reveals the highest model performance (e.g., classification accuracy when considering a classification technique). In the selection step, three different mechanisms can be considered; complete, random, and heuristic search mechanisms. When dealing with high-dimensional datasets, the complete search strategies (i.e., brute force) are impractical since 2^N feature subsets have to be generated and evaluated for a dataset with N features. The random mechanisms are also impractical to be used with FS methods since the search process is controlled by random factors that may lead the search process to be as worst as the complete search, in the worst case. Heuristic search algorithms are the most suitable mechanisms to be used with FS methods since the search process is guided by a heuristic that navigates the process towards finding the near-optimal solutions.

The evaluation criterion is another aspect that should be carefully considered when designing a FS method. In this regard, FS methods can be divided into three main types; filters, wrappers, and embedded methods. In filter approaches [63–66, 68], feature subsets are evaluated depending on the relations between features them-

selves, and their correlation with the class attribute without considering any particular classifier [62]. Thus, a feature with high score is selected to build the classification model in a further step. In the wrapper based methods [2, 34, 73], each feature subset is evaluated based on the performance of a particular classifier. Embedded methods (e.g., regularization) [20, 57, 61] learn which features enhance the performance of the model (e.g., accuracy), while the model is being created.

In summary, filter methods are considered as the fastest FS methods, since no external tools are required to be employed; nevertheless the model performance is uncertain. Wrappers are slower than filters, while they usually obtain higher accuracy. Embedded methods come in between filters and wrappers in terms of the model performance and the computational time.

3 Dragonfly Algorithm

Dragonfly Algorithm (DA) is a recently well-established population-based optimizer proposed by Mirjalili in 2016 [75]. The DA was developed based on the hunting and migration strategies of dragonflies. The hunting technique is known as static swarm (feeding), in which all members of a swarm can fly in small clusters over a limited

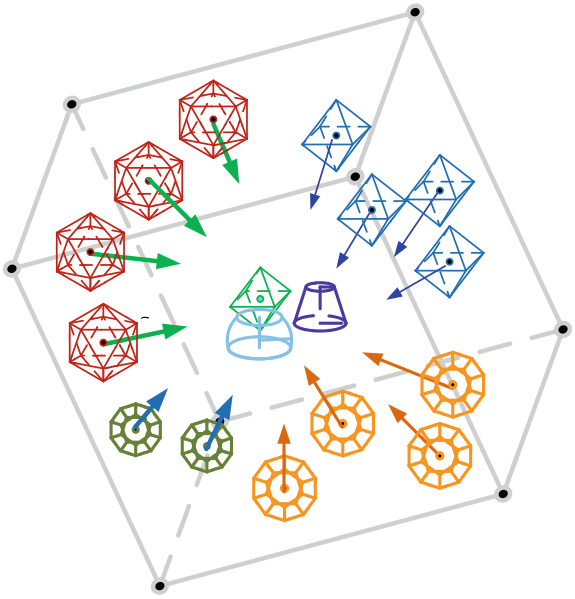

Fig. 1 Dynamic swarming behaviors (each geometric object shows a special type of agents)

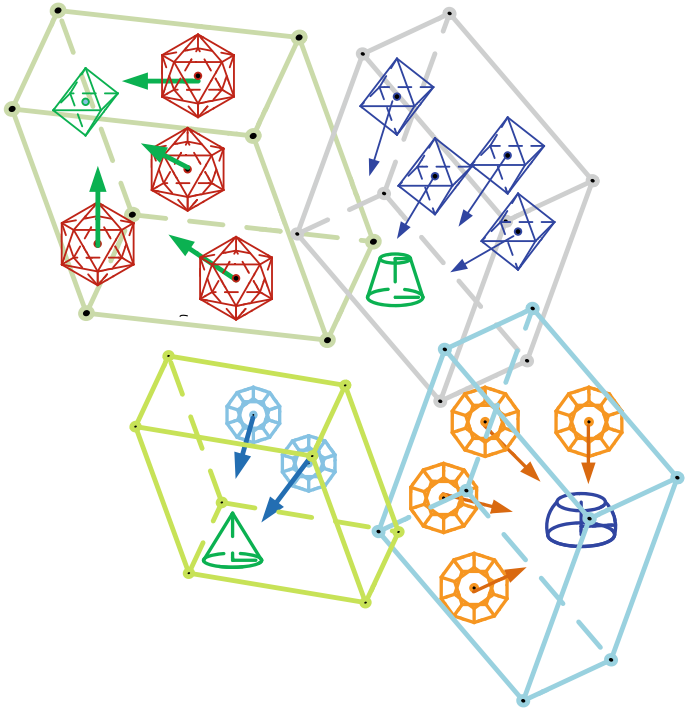

Fig. 2 Static swarming behaviors

space for discovering food sources. The migration strategy of dragonflies is called dynamic swarm (migratory). In this phase, the dragonflies are willing to soar in larger clusters, and as a result, the swarm can migrate. Dynamic and static groups are shown in Figs. 1 and 2. Likewise other swarm-based methods, the operators of DA perform two main concepts: diversification, motivated by the static swarming activities, and intensification, encouraged by the dynamic swarming activities.

In DA, five types of behaviors are designed as follows. In next parts, X is the position vector, X_j is the j-th neighbor of the X, and N denotes the neighborhood size [84]:

- *Separation* is a strategy that dragonflies use to separate themselves from other agents. This procedure is formulated as Eq. (1):

$$S_i = -\sum_{j=1}^{N} X - X_i \tag{1}$$

- *Alignment* shows how an agent will set its velocity with respect to the velocity vector of other adjacent dragonflies. This concept is modeled based on Eq. (2):

$$A_i = \frac{\sum_{j=1}^{N} V_j}{N} \tag{2}$$

where V_j indicates the velocity vector of the j-th neighbor.

- *Cohesion* indicates the inclination of members to move in the direction of the nearby center of mass. This step is formulated as in Eq. (3):

$$C_i = \frac{\sum_{j=1}^{N} x_j}{N} - X \tag{3}$$

- *Attraction* shows to the propensity of members to move towards the food source. The attraction tendency among the food source and the ith agent is performed based on Eq. (4):

$$F_i = F_{loc} - X \tag{4}$$

where F_{loc} is the food source's location.

- *Distraction* shows the proclivity of dragonflies to keep themselves away from an conflicting enemy. The distraction among the enemy and the ith dragonfly is performed according to Eq. (5):

$$E_i = E_{loc} + X \tag{5}$$

where E_{loc} is the enemy's location.

In DA, the fitness of food source and position vectors are updated based on the fittest agent found so far. Moreover, the fitness values and positions of the enemy are computed based on the worst dragonfly. This fact can assist DA in converging towards more promising regions of the solution space and in turn, avoiding from non-promising areas. The position vectors of dragonflies are updated based on two rules: the step vector (ΔX) and the position vector. The step vector indicates the dragonflies' direction of motion and it is calculated as in Eq. (6):

$$\Delta X_{t+1} = (sS_i + aA_i + cC_i + fF_i + eE_i) + wX_t \tag{6}$$

where s, w, a, c, f, and e show the weighting vectors of different components.

The location vector of members is calculated as in Eq. (7):

$$X_{t+1} = X_t + \Delta X_{t+1} \tag{7}$$

where t is iteration.

Pseudocode of DA is shown in Algorithm 1.

Algorithm 1 Pseudocode of DA

Initialize the swarm $X_i (i = 1, 2, \ldots, n)$
Initialize $\Delta X_i (i = 1, 2, \ldots, n)$
while (end condition is not met) **do**
  Evaluate all dragonflies
  Update (F) and (E)
  Update coefficients $(i., e., w, s, a, c, f, and\ e)$
  Attain S, A, C, F, and E (based on Eqs. (9.21 to 4.5))
  Update step vectors (ΔX_{t+1}) by Eq. (4.6)
  Update X_{t+1} by Eq. (4.7)
Return the best agent

4 Literature Review

In this section, previous researches on DA and their core findings are reviewed in detail. From the time of proposing DA, several works have focused on the efficacy of DA or tried to improve its efficacy on tasks such as photovoltaic systems [83], extension of RFID network lifetime [47], economic emission dispatch (EPD) problems [19]. Hamdy et al. [39] used the multi-objective version of DA to solve the design problem of a nearly zero energy building (nZEB), along with six other evolutionary methods. In [85], a hybrid DA approach with extreme learning machine (ELM) was proposed. In this system, DA was used to optimize the number of nodes and their associated weights in the hidden layer of the ELM. Pathania et al. [81] proposed a DA based approach to solve the economic load dispatch (ELD) problem with valve point effect. DA was used in [24] to estimate the location of the unknown wireless nodes that are randomly deployed in a designated area. The binary version of DA was used with a multi-class SVM classifier within an FS method [28]. Hema Sekhar et al. [89] employed DA to optimize the firing angle and the size of the thyristor controlled series capacitor (TCSC). In [86], a self-adaptive DA was used to tackle the multilevel segmentation problem.

In 2017, a memory-based hybrid DA with particle swarm optimization (PSO) has been developed to deal with global optimization [55]. Song et al. [91] developed an enhanced DA with elite opposition learning (EOL) to tackle global optimization tasks. In [3], DA was used to solve the 0-1 knapsack problems. Another application that was solved using DA is the static economic dispatch incorporating solar energy as in [94]. DA was used as a FS search strategy in addition to tune the parameters of ELM in [101]. A multi objective version of DA (MODA) was used as an adaptive engine calibration algorithm to control the engine parameters [38]. Mafarja et al. [71] proposed a FS method that employed the binary version of DA (BDA) as a search strategy. In [96, 97], DA was used as a tool to optimize SVM parameters. In software engineering field, DA was also applied. In [93], DA was used as an optimization tool to select the most important test cases that satisfy all requirements. An improved version of DA based on elite opposition learning strategy was used to solve some global optimization problems [91].

Suresh and Sreejith [94] proposed the use of DA in optimizing the generation costs of all the participating units in a solar thermal economic dispatch system. Memory based Hybrid DA (MHDA) was proposed in [55] to solve numerical optimization problems. DA was used in [37] to optimize the performance of the Pull-in Voltage performance in a MEMS switch. A hybrid model that combined DA with PSO was proposed in [88] to solve the optimal power flow (OPF) problem. DA was also used to select the optimal cluster heads in Radio Frequency Identification (RFID) networks [48]. Jafari and Chaleshtari [50] used the DA to optimize the orthotropic infinite plates with a quasi-triangular cut-out.

In [100], DA was utilized as an efficient optimizer for turn-mill operations. Hariharan et al. [40] proposed an enhanced binary DA to deal with infant cry classification scenarios. Khadanga et al. [54] proposed a hybrid DA with pattern search for developing and studying of tilt integral derivative controllers. Kumar et al. [56] developed an improved fractional DA to be integrated with cluster cloud models. In [99], a modified cumulative DA in cooperation with neural networks was proposed. They also proposed a map-reduce cluster architecture to predict the performance of students using the DA-based approach. In [98], DA was employed to optimize the size and cost of static var compensator for enhancement of voltage profiles in power systems. Amroune et al. [15] proposed a new hybrid DA with support vector regression (SVR) for voltage stability analysis in power systems. Branch and Rey [21] proposed a DA-based load balancing approach to allocate resources in cloud computing. Babayigit [17] applied DA to synthesis of antenna arrays.

To sum up, in the reviewed papers, experimental results show the efficient performance of DA in terms of balancing the exploration and exploitation, speed of convergence, and satisfactory local optima (LO) avoidance.

5 Binary DA (BDA)

The DA algorithm was originally designed to solve the continuous optimization problems. Thus, using DA to tackle the FS problem requires changing the structure of the algorithm to deal with binary solutions. Transfer functions can be employed to convert the continuous solution to a binary one [77], by generating a probability of changing a position's elements to 0 or 1 based on the value of the step vector (velocity) of the ith search agent in the dth dimension in the current iteration (t) as an input parameter. Mirjalili [75] employed the transfer function in Eq. (8) to calculate the probability of changing the continuous positions to binary. This transfer function belongs the V-shaped (see Fig. 3) transfer functions category as denoted in [77].

$$T(v_d^i(t)) = \frac{|v_d^i(t)|}{\sqrt{1 + (v_d^i(t))^2}} \tag{8}$$

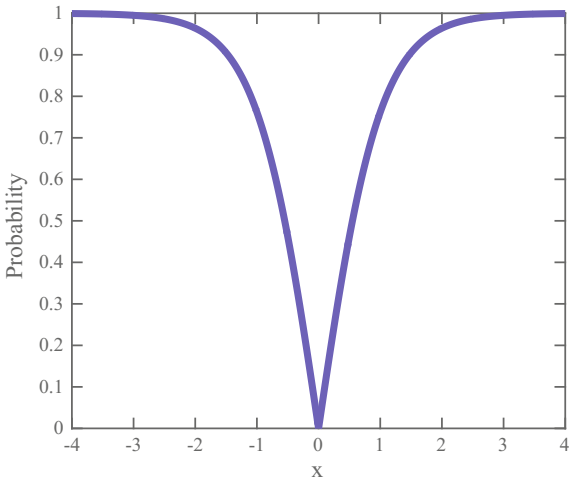

Fig. 3 V-Shaped transfer function [77]

The result $T(v_k^i(t))$, obtained from Eq. (8) is then used to convert the i-th element of the position vector to 0 or 1 according to Eq. (9)

$$X(t+1) = \begin{cases} \neg X_t & r < T(v_k^i(t)) \\ X_t & r \geq T(v_k^i(t)) \end{cases} \tag{9}$$

where r is a random number in the [0,1] interval.

5.1 BDA-Based Wrapper Feature Selection

As the FS is a binary optimization problem, a binary vector of length N (where N is the number of features in the dataset), with 1 for the selected feature and 0 for the non selected feature, is used in the proposed approach (see Fig. 4). The KNN classifier [14] evaluates the selected feature subsets. Here, KNN classifier was used for its simplicity which makes it easy to implement, also it is a nonparametric classifier and hence can be used with different datasets. Moreover, KNN has been widely used with many FS methods and demonstrated to have competitive performance with a wide range of real life datasets [60].

In wrapper FS methods, two contradictory objectives should be considered; classification accuracy and reduction rate. The fitness function Eq. (10) considers both

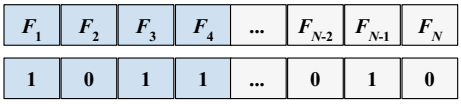

Fig. 4 Solution representation

aspects. α and β are two parameters corresponding to the importance of classification quality and subset length, α is in the [0,1] interval and β =(1 $-\alpha$) is adapted from [69].

$$F(X) = \alpha * \gamma(X) + \beta * (1 - \frac{|R|}{|N|}) \tag{10}$$

where $\gamma(X)$ represents the classification accuracy while using X feature subset, $|R|$ is the number of selected features and $|N|$ is the total number of features in the original dataset. Therefore, α (a number in the interval [0,1]) and β (1-α) [69] parameters were used to represent the importance of classification quality and the reduction rate, respectively.

Pseudocode of BDA-based feature selection technique is shown in Algorithm 2.

Algorithm 2 Pseudocode of BDA-based feature selection algorithm

Initialize the candidate features $X_i (i = 1, 2, \ldots, n)$
Initialize $\Delta X_i (i = 1, 2, \ldots, n)$
while (end condition is not met) **do**
  Evaluate all dragonflies based on Eq. (11.10)
  Update (F) and (E)
  Update coefficients ($i.$, $e.$, w, s, a, c, f, *and* e)
  Attain S, A, C, F, and E (based on Eqs. (9.21 to 4.5))
  Update step vectors (ΔX_{t+1}) by Eq. (4.6)
  Calculate $T(\Delta X)$ using Eq. (4.8)
  Update X_{t+1} by Eq. (4.9)
Return the best set of features

6 Results and Discussions

To test the proposed approach, 9 high dimensional datasets with low number of samples were used. As can be seen form Table 1, nine datasets have 226 distinct categories, 50308 samples (patients) and 2308–15009 variables (genes). All datasets are accessible in a public source.[1] A train/test model is utilized to validate the performance of the proposed BDA approach, where 80% of the data were used for training purposes, while the remaining 20% were used for testing [73]. All approaches were implemented using MATLAB 2017a and the results were repeated for 30 independent times on an Intel Core i5 machine, 2.2 GHz CPU and 4 GB of RAM. Table 2 presents the detailed parameter settings for utilized methods.

[1] http://www.gems-system.org/.

Table 1 Details of high-dimensional small instances datasets [92]

Dataset name	No. of samples	No. of features	No. of classes
11_Tumors	174	12533	11
14_Tumors	308	15009	26
Brain_Tumor1	90	5920	5
Brain_Tumor2	50	10367	4
Leukemia1	72	5327	3
Leukemia2	72	11225	3
SRBCT	83	2308	4
Prostate_Tumor	102	10509	2
DLBCL	77	5469	2

Table 2 The parameter settings

Parameter	Value
Population size	10
Number of iterations	100
Dimension	Number of features
Number of runs for each technique	30
α in fitness function	0.99
β in fitness function	0.01
a in bGWO	[2 0]
Q_{min} Frequency minimum in BBA	0
Q_{max} Frequency maximum in BBA	2
A Loudness in BBA	0.5
r Pulse rate in BBA	0.5
G_0 in BGSA	100
α in BGSA	20

6.1 Results and Analysis

In order to assess the performance of BDA algorithm on the high dimensional datasets, we tested six well-know metaheuristics algorithms for comparison purposes. All algorithms were implemented and ran on the same environment in order to make a fair comparison. For all algorithms, the average classification accuracy, average selected features, and average fitness values for the 30 independent runs are reported. Note that the best results in the subsequent results are highlighted in boldface. Moreover, a Wilcoxon signed-rank test is executed to state if there is a significant difference between the reported results with the significance interval 95% ($\alpha = 0.05$).

Table 3 shows the average classification accuracy over the 30 runs of different algorithms. As can be seen from Table 3, the proposed BDA shows the best performance compared to all other approaches. BGSA comes in the second place by obtaining the best results in three datasets, and then bGWO outperformed other approaches in only one dataset. Comparing BDA with BGSA; the second best approach, it can be obviously seen that BDA records the best results in 6 datasets, while BGSA is outperforming other approaches in 2 datasets, and both approaches obtained the same results in dealing with Brain_Tumor1 dataset. According to Table 4, which presents the p-values of BDA versus other approaches, there is a significant difference in the performance of the proposed approach and the other peers. These results verify that the proposed BDA maintains a more stable balance between diversification and intensification, and this leads to higher classification rates compared to other peers.

Table 3 Average classification accuracy for all approaches

Dataset	bGWO	BGSA	BBA	BGA	BPSO	BDA
11_Tumors	0.858	0.819	0.656	0.616	0.758	**0.879**
14_Tumors	0.576	0.481	0.556	0.428	0.477	**0.724**
Brain_Tumor1	**0.944**	**0.944**	0.894	0.848	0.726	0.889
Brain_Tumor2	0.720	0.800	0.543	0.745	0.416	**0.857**
DLBCL	0.754	**0.875**	0.848	0.868	0.818	**0.875**
Leukemia1	0.869	0.658	0.756	0.761	0.663	**0.933**
Leukemia2	0.867	0.867	0.922	0.821	0.818	**0.933**
Prostate_Tumor	0.878	**0.943**	0.921	0.834	0.806	0.902
SRBCT	0.882	0.884	0.859	0.760	0.746	**0.941**

Table 4 p-values between BDA and other approaches for the classification accuracy

Dataset	BDA versus				
	bGWO	BGSA	BBA	BGA	BPSO
11_Tumors	**1.61E-05**	**1.86E-11**	**1.20E-11**	**1.27E-11**	**1.19E-11**
14_Tumors	**2.54E-11**	**1.53E-11**	**2.64E-11**	**2.52E-11**	**2.84E-11**
Brain_Tumor1	**1.69E-14**	**1.69E-14**	1.70E-01	**7.24E-13**	**9.13E-13**
Brain_Tumor2	**1.03E-10**	**1.43E-06**	**8.50E-12**	**2.53E-11**	**8.71E-12**
DLBCL	**4.16E-14**	NaN	**2.98E-04**	**1.75E-02**	**9.21E-13**
Leukemia1	**1.17E-13**	**1.57E-13**	**3.37E-13**	**9.38E-13**	**1.02E-12**
Leukemia2	**1.69E-14**	**1.69E-14**	**2.14E-02**	**7.65E-13**	**5.57E-13**
Prostate_Tumor	**1.19E-04**	**2.98E-09**	**4.36E-03**	**9.49E-12**	**2.88E-12**
SRBCT	**1.69E-14**	**1.17E-13**	**3.48E-11**	**8.41E-13**	**8.09E-13**

Table 5 Average selected features for all approaches

Dataset	bGWO	BGSA	BBA	BGA	BPSO	BDA
11_Tumors	6233.23	6258.00	**5082.73**	6261.87	6261.50	5660.17
14_Tumors	8511.53	7564.97	**6207.97**	7500.77	7489.47	7206.77
Brain_Tumor1	2926.63	2896.13	2389.60	2960.17	2950.57	**2269.77**
Brain_Tumor2	5141.93	5106.47	**4173.83**	5156.87	5196.80	4285.23
DLBCL	2691.43	2699.57	2208.83	2731.20	2733.30	**1884.90**
Leukemia1	2620.63	2662.03	2077.73	2660.40	2664.53	**1850.47**
Leukemia2	5530.73	5523.27	4472.83	5613.60	5605.80	**4466.07**
Prostate_Tumor	5391.43	5242.90	**4124.50**	5269.43	5247.00	4372.77
SRBCT	1126.73	1144.80	925.37	1161.10	1148.97	**909.57**

Table 6 p-values between BDA and other approaches for the number of selected features

Dataset	BDA versus				
	bGWO	BGSA	BBA	BGA	BPSO
11_Tumors	**3.02E-11**	**3.02E-11**	**3.52E-07**	**3.01E-11**	**3.02E-11**
14_Tumors	**3.46E-10**	**1.46E-10**	**6.68E-11**	**2.22E-09**	**4.36E-09**
Brain_Tumor1	**3.02E-11**	**3.00E-11**	3.37E-01	**3.02E-11**	**3.01E-11**
Brain_Tumor2	**3.01E-11**	**3.01E-11**	1.02E-01	**3.01E-11**	**3.01E-11**
DLBCL	**3.01E-11**	**3.00E-11**	**2.60E-08**	**3.01E-11**	**3.02E-11**
Leukemia1	**3.00E-11**	**3.01E-11**	**7.20E-05**	**3.01E-11**	**3.01E-11**
Leukemia2	**3.01E-11**	**3.01E-11**	8.36E-01	**3.02E-11**	**3.01E-11**
Prostate_Tumor	**3.02E-11**	**3.01E-11**	**2.87E-02**	**3.01E-11**	**3.02E-11**
SRBCT	**3.00E-11**	**3.00E-11**	6.41E-01	**2.99E-11**	**3.00E-11**

Inspecting the results in Table 5, it can be obviously seen that BDA selects the minimal number of features on 55% of the datasets. It is followed by BBA which came in the second place by outperforming other approaches in 45% of the datasets. we see that no other approach is capable to compete with those two approaches. P-values in Table 6 also show that BDA can significantly outperform other approaches (namely bGWO, BGSA, BGA, and BPSO) in all datasets, while on five datasets out of nine, it can significantly outperform BBA approach.

As mentioned before, the adopted fitness function considers two objectives of the wrapper FS approach, i.e., the classification accuracy and the number of selected features. Observing the results in Tables 7 and 8, it can be seen the BDA significantly outperforms bGWO, BGSA, BGA, and BPSO approaches on all datasets, and on 55% of the datasets when compared to BBA. In addition to the superior performance of BDA approach, it recorded the fastest convergence rates among other approaches in most of the datasets as can be seen in Fig. 5, which shows the convergence speed of bGWO, BGSA, BBA, and BDA over the iterations. The reasons for satisfying

Table 7 Average fitness values for all approaches

Dataset	bGWO	BGSA	BBA	BGA	BPSO	BDA
11_Tumors	0.145	0.184	0.321	0.757	0.777	**0.124**
14_Tumors	0.425	0.519	0.417	0.473	0.504	**0.278**
Brain_Tumor1	0.060	**0.060**	0.080	0.837	0.876	0.114
Brain_Tumor2	0.282	0.203	0.393	0.600	0.653	**0.146**
DLBCL	0.248	0.129	**0.125**	0.843	0.845	0.127
Leukemia1	0.135	0.344	0.212	0.862	0.875	**0.069**
Leukemia2	0.137	0.137	**0.069**	0.863	0.884	0.070
Prostate_Tumor	0.126	0.062	**0.024**	0.846	0.859	0.102
SRBCT	0.121	0.119	0.105	0.864	0.889	**0.062**

Table 8 p-values between BDA and other approaches for the fitness values

Dataset	BDA versus				
	bGWO	BGSA	BBA	BGA	BPSO
11_Tumors	**7.67E-09**	**2.99E-11**	**3.01E-11**	**2.91E-11**	**2.94E-11**
14_Tumors	**3.01E-11**	**3.00E-11**	**3.01E-11**	**2.98E-11**	**2.96E-11**
Brain_Tumor1	**3.01E-11**	**3.00E-11**	**1.59E-07**	**2.82E-11**	**2.54E-11**
Brain_Tumor2	**2.94E-11**	**2.95E-11**	**3.01E-11**	**2.83E-11**	**2.68E-11**
DLBCL	**2.92E-11**	**2.95E-11**	1.20E-01	**2.57E-11**	**2.77E-11**
Leukemia1	**2.95E-11**	**3.01E-11**	**3.00E-11**	**2.76E-11**	**2.69E-11**
Leukemia2	**2.91E-11**	**2.91E-11**	**1.67E-06**	**2.80E-11**	**2.63E-11**
Prostate_Tumor	**1.28E-09**	**9.79E-05**	**4.97E-11**	**2.75E-11**	**2.80E-11**
SRBCT	**2.98E-11**	**2.96E-11**	**2.77E-05**	**2.79E-11**	**2.80E-11**

convergence curves in majority of datasets is that BDA has a high capacity for performing the focused intensification tendencies in last steps and extra diversification jumps in initial phases.

7 Conclusions and Future Directions

In this chapter, a wrapper FS approach (called BDA) that employs SSA as a selection strategy and KNN as an evaluator was proposed. BDA was tested using nine large scale medical datasets with low number of samples. Dealing with high dimensional datasets with low number of samples is challenging since the model cannot has no enough samples to be trained, in addition to having a large search space. BDA showed a good performance on those datasets when compared with five well known wrapper FS approaches.

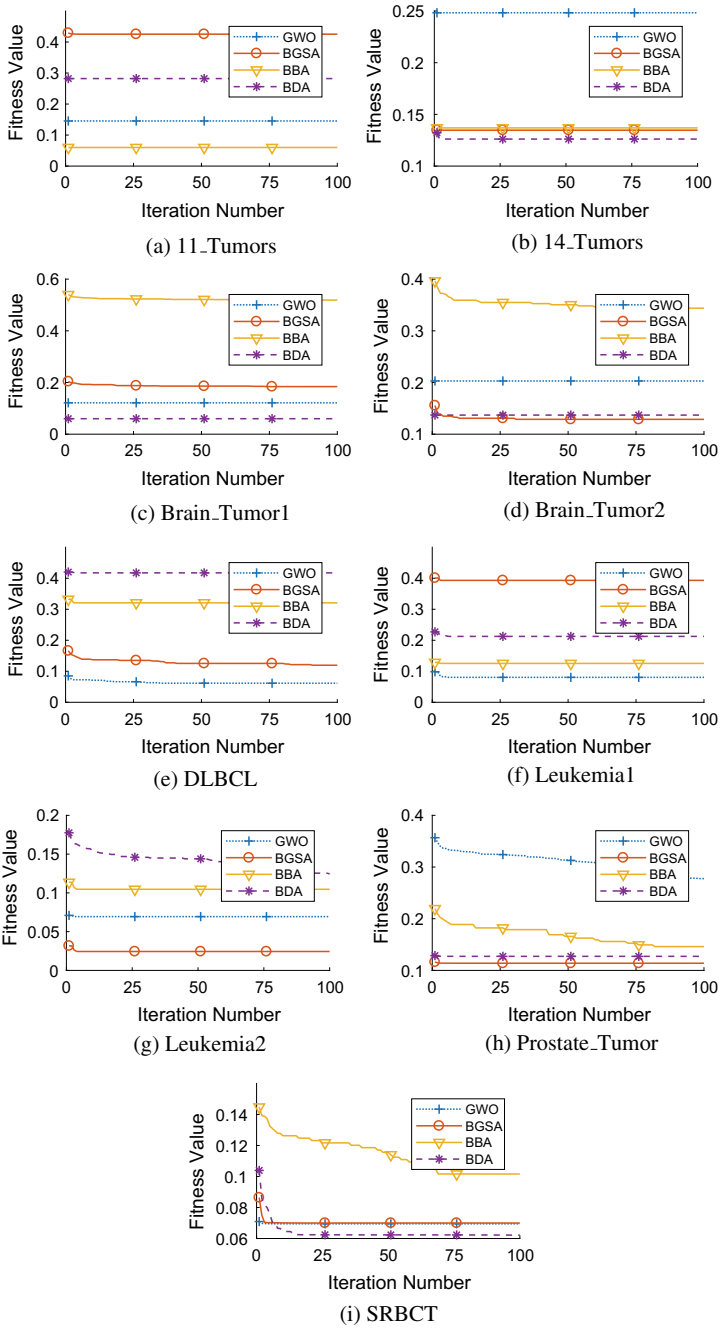

Fig. 5 Convergence curves for BGWO, BGSA, BBA, and BDA for all datasets

References

1. Abbassi, R., Abbassi, A., Heidari, A. A., & Mirjalili, S. (2019). An efficient salp swarm-inspired algorithm for parameters identification of photovoltaic cell models. *Energy Conversion and Management, 179*, 362–372.
2. An efficient binary salp swarm algorithm with crossover scheme for feature selection problems. (2018). *Knowledge-Based Systems, 154*, 43–67.
3. Abdel-Basset, M., Luo, Q., Miao, F., & Zhou, Y. (2017). Solving 0-1 knapsack problems by binary dragonfly algorithm. In *International Conference on Intelligent Computing* (pp. 491–502). Springer.
4. Al-Madi, N., Aljarah, I., & Ludwig, S. (2014). Parallel glowworm swarm optimization clustering algorithm based on mapreduce. In *IEEE Symposium Series on Computational Intelligence (IEEE SSCI 2014)*. IEEE Xplore Digital Library.
5. Aljarah, I., AlaM, A. Z., Faris, H., Hassonah, M. A., Mirjalili, S., & Saadeh, H. (2018). Simultaneous feature selection and support vector machine optimization using the grasshopper optimization algorithm. *Cognitive Computation*, 1–18.
6. Aljarah, I., Faris, H., & Mirjalili, S. (2018). Optimizing connection weights in neural networks using the whale optimization algorithm. *Soft Computing*, *22*(1), 1–15.
7. Aljarah, I., Faris, H., Mirjalili, S., & Al-Madi, N. (2018). Training radial basis function networks using biogeography-based optimizer. *Neural Computing and Applications*, *29*(7), 529–553.
8. Aljarah, I., & Ludwig, S. A. (2012). Parallel particle swarm optimization clustering algorithm based on mapreduce methodology. In *Proceedings of the Fourth World Congress on Nature and Biologically Inspired Computing (IEEE NaBIC12)*. IEEE Explore.
9. Aljarah, I., & Ludwig, S. A. (2013). A mapreduce based glowworm swarm optimization approach for multimodal functions. In *IEEE Symposium Series on Computational Intelligence, IEEE SSCI 2013*. IEEE Xplore.
10. Aljarah, I., & Ludwig, S. A. (2013). A new clustering approach based on glowworm swarm optimization. In *Proceedings of 2013 IEEE Congress on Evolutionary Computation Conference (IEEE CEC13)*. Cancun, Mexico: IEEE Xplore.
11. Aljarah, I., & Ludwig, S. A. (2013). Towards a scalable intrusion detection system based on parallel PSO clustering using mapreduce. In *Proceedings of Genetic and Evolutionary Computation Conference (ACM GECCO13)*. Amsterdam: ACM.
12. Aljarah, I., & Ludwig, S. A. (2016). A scalable mapreduce-enabled glowworm swarm optimization approach for high dimensional multimodal functions. *International Journal of Swarm Intelligence Research (IJSIR)*, *7*(1), 32–54.
13. Aljarah, I., Mafarja, M., Heidari, A. A., Faris, H., Zhang, Y., & Mirjalili, S. (2018). Asynchronous accelerating multi-leader salp chains for feature selection. *Applied Soft Computing, 71*, 964–979.
14. Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, *46*(3), 175–185.
15. Amroune, M., Bouktir, T., & Musirin, I. (2018). Power system voltage stability assessment using a hybrid approach combining dragonfly optimization algorithm and support vector regression. *Arabian Journal for Science and Engineering*, 1–14.
16. Aminisharifabad, M., Yang, Q., & Wu, X. (2018). A penalized autologistic regression with application for modeling the microstructure of dual-phase high strength steel. *Journal of Quality Technology,* in-press.
17. Babayigit, B. (2018). Synthesis of concentric circular antenna arrays using dragonfly algorithm. *International Journal of Electronics*, *105*(5), 784–793.
18. Barham, R., & Aljarah, I. (2017). Link prediction based on whale optimization algorithm. In *The International Conference on new Trends in Computing Sciences (ICTCS2017)*. Amman: Jordan.

19. Bhesdadiya, R., Pandya, M. H., Trivedi, I. N., Jangir, N., Jangir, P., & Kumar, A. (2016). Price penalty factors based approach for combined economic emission dispatch problem solution using dragonfly algorithm. In *2016 International Conference on Energy Efficient Technologies for Sustainability (ICEETS)* (pp. 436–441). IEEE.
20. Blum, A. L., & Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence*, *97*(1–2), 245–271.
21. Branch, S. R., & Rey, S. (2018). Providing a load balancing method based on dragonfly optimization algorithm for resource allocation in cloud computing. *International Journal of Networked and Distributed Computing*, *6*(1), 35–42.
22. Chen, Y., Li, L., Xiao, J., Yang, Y., Liang, J., & Li, T. (2018). Particle swarm optimizer with crossover operation. *Engineering Applications of Artificial Intelligence*, *70*, 159–169.
23. Chitsaz, H., & Aminisharifabad, M. (2015). Exact learning of rna energy parameters from structure. *Journal of Computational Biology*, *22*(6), 463–473.
24. Daely, P. T., & Shin, S. Y. (2016). Range based wireless node localization using dragonfly algorithm. In *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)* (pp. 1012–1015). IEEE.
25. Dorigo, M., & Birattari, M. (2011). Ant colony optimization. In *Encyclopedia of machine learning* (pp. 36–39). Springer.
26. Dorigo, M., & Di Caro, G. (1999). Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 Congress on Evolutionary Computation, 1999. CEC 99*, vol. 2 (pp. 1470–1477). IEEE.
27. Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995. MHS'95* (pp. 39–43). IEEE.
28. Elhariri, E., El-Bendary, N., & Hassanien, A. E. (2016). Bio-inspired optimization for feature set dimensionality reduction. In *2016 3rd International Conference on Advances in Computational Tools for Engineering Applications (ACTEA)* (pp. 184–189). IEEE.
29. Faris, H., Aljarah, I., Al-Madi, N., & Mirjalili, S. (2016). Optimizing the learning process of feedforward neural networks using lightning search algorithm. *International Journal on Artificial Intelligence Tools*, *25*(06), 1650033.
30. Faris, H., Aljarah, I., & Al-Shboul, B. (2016). A hybrid approach based on particle swarm optimization and random forests for e-mail spam filtering. *International Conference on Computational Collective Intelligence* (pp. 498–508). Cham: Springer.
31. Faris, H., Aljarah, I., & Mirjalili, S. (2017). Evolving radial basis function networks using moth–flame optimizer. In *Handbook of Neural Computation* (pp. 537–550).
32. Faris, H., Aljarah, I., et al. (2015). Optimizing feedforward neural networks using krill herd algorithm for e-mail spam detection. In *2015 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)* (pp. 1–5). IEEE.
33. Faris, H., Ala'M, A. Z., Heidari, A. A., Aljarah, I., Mafarja, M., Hassonah, M. A., & Fujita, H. (2019). An intelligent system for spam detection and identification of the most relevant features based on evolutionary random weight networks. *Information Fusion, 48*, 67–83.
34. Faris, H., Mafarja, M. M., Heidari, A. A., Aljarah, I., AlaM, A. Z., Mirjalili, S., et al. (2018). An efficient binary salp swarm algorithm with crossover scheme for feature selection problems. *Knowledge-Based Systems*, *154*, 43–67.
35. Fisher, L. (2009). *The perfect swarm: The science of complexity in everyday life*. Basic Books.
36. Ghatasheh, N., Faris, H., Aljarah, I., & Al-Sayyed, R. M. (2015). Optimizing software effort estimation models using firefly algorithm. *Journal of Software Engineering and Applications*, *8*(03), 133.
37. Guha, K., Laskar, N., Gogoi, H., Borah, A., Baishnab, K., & Baishya, S. (2017). Novel analytical model for optimizing the pull-in voltage in a flexured mems switch incorporating beam perforation effect. *Solid-State Electronics*, *137*, 85–94.
38. Guo, S., Dooner, M., Wang, J., Xu, H., & Lu, G. (2017). Adaptive engine optimisation using NSGA-II and MODA based on a sub-structured artificial neural network. In *2017 23rd International Conference on Automation and Computing (ICAC)* (pp. 1–6). IEEE.

39. Hamdy, M., Nguyen, A. T., & Hensen, J. L. (2016). A performance comparison of multi-objective optimization algorithms for solving nearly-zero-energy-building design problems. *Energy and Buildings*, *121*, 57–71.
40. Hariharan, M., Sindhu, R., Vijean, V., Yazid, H., Nadarajaw, T., Yaacob, S., et al. (2018). Improved binary dragonfly optimization algorithm and wavelet packet based non-linear features for infant cry classification. *Computer Methods and Programs in Biomedicine*, *155*, 39–51.
41. Heidari, A. A., & Abbaspour, R. A. (2018). Enhanced chaotic grey wolf optimizer for real-world optimization problems: A comparative study. In *Handbook of Research on Emergent Applications of Optimization Algorithms* (pp. 693–727). IGI Global.
42. Heidari, A. A., Abbaspour, R. A., & Jordehi, A. R. (2017). An efficient chaotic water cycle algorithm for optimization tasks. *Neural Computing and Applications*, *28*(1), 57–85.
43. Heidari, A. A., Abbaspour, R. A., & Jordehi, A. R. (2017). Gaussian bare-bones water cycle algorithm for optimal reactive power dispatch in electrical power systems. *Applied Soft Computing*, *57*, 657–671.
44. Heidari, A. A., & Delavar, M. R. (2016). A modified genetic algorithm for finding fuzzy shortest paths in uncertain networks. In *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLI-B2* (pp. 299–304).
45. Heidari, A. A., Faris, H., Aljarah, I., & Mirjalili, S. (2018). An efficient hybrid multilayer perceptron neural network with grasshopper optimization. *Soft Computing*, 1–18.
46. Heidari, A. A., & Pahlavani, P. (2017). An efficient modified grey wolf optimizer with lévy flight for optimization tasks. *Applied Soft Computing*, *60*, 115–134.
47. Hema, C., Sankar, S., et al. (2016). Energy efficient cluster based protocol to extend the RFID network lifetime using dragonfly algorithm. In *2016 International Conference on Communication and Signal Processing (ICCSP)* (pp. 0530–0534). IEEE.
48. Hema, C., Sankar, S., et al. (2017). Performance comparison of dragonfly and firefly algorithm in the RFID network to improve the data transmission. *Journal of Theoretical and Applied Information Technology*, *95*(1), 59.
49. Ibrahim, H. T., Mazher, W. J., Ucan, O. N., & Bayat, O. (2018). A grasshopper optimizer approach for feature selection and optimizing SVM parameters utilizing real biomedical data sets. *Neural Computing and Applications*.
50. Jafari, M., & Chaleshtari, M. H. B. (2017). Using dragonfly algorithm for optimization of orthotropic infinite plates with a quasi-triangular cut-out. *European Journal of Mechanics-A/Solids*, *66*, 1–14.
51. Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, *39*(3), 459–471.
52. Kashef, S., & Nezamabadi-pour, H. (2015). *An advanced ACO algorithm for feature subset selection*, *147*, 271–279.
53. Kennedy, J. (2006). Swarm intelligence. In *Handbook of Nature-Inspired and Innovative Computing* (pp. 187–219). Springer.
54. Khadanga, R. K., Padhy, S., Panda, S., & Kumar, A. (2018). Design and analysis of tilt integral derivative controller for frequency control in an islanded microgrid: A novel hybrid dragonfly and pattern search algorithm approach. *Arabian Journal for Science and Engineering*, 1–12.
55. Ks, S. R., & Murugan, S. (2017). Memory based hybrid dragonfly algorithm for numerical optimization problems. *Expert Systems with Applications*, *83*, 63–78.
56. Kumar, C. A., Vimala, R., Britto, K. A., & Devi, S. S. (2018). FDLA: Fractional dragonfly based load balancing algorithm in cluster cloud model. *Cluster Computing*, 1–14.
57. Langley, P., et al. (1994). Selection of relevant features in machine learning. *Proceedings of the AAAI Fall symposium on relevance*, *184*, 245–271.
58. Li, Q., Chen, H., Huang, H., Zhao, X., Cai, Z., Tong, C., et al. (2017). An enhanced grey wolf optimization based feature selection wrapped kernel extreme learning machine for medical diagnosis. *Computational and Mathematical Methods in Medicine*, *2017*.

59. Li, Y., Li, T., & Liu, H. (2017). Recent advances in feature selection and its applications. *Knowledge and Information Systems*, *53*(3), 551–577.
60. Liao, T., & Kuo, R. (2018). Five discrete symbiotic organisms search algorithms for simultaneous optimization of feature subset and neighborhood size of KNN classification models. *Applied Soft Computing*, *64*, 581–595.
61. Liu, H., & Motoda, H. (2012). *Feature selection for knowledge discovery and data mining*, vol. 454. Springer Science & Business Media.
62. Liu, H., Setiono, R., et al. (1996). A probabilistic approach to feature selection-a filter solution. In *Thirteenth International Conference on Machine Learning (ICML)*, vol. 96 (pp. 319–327). Citeseer.
63. Mafarja, M., & Abdullah, S. (2011). Modified great deluge for attribute reduction in rough set theory. In *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, vol. 3, (pp. 1464–1469). IEEE.
64. Mafarja, M., & Abdullah, S. (2013). Investigating memetic algorithm in solving rough set attribute reduction. *International Journal of Computer Applications in Technology*, *48*(3), 195–202.
65. Mafarja, M., & Abdullah, S. (2013). Record-to-record travel algorithm for attribute reduction in rough set theory. *Journal of Theoretical and Applied Information Technology*, *49*(2), 507–513.
66. Mafarja, M., & Abdullah, S. (2014). Fuzzy modified great deluge algorithm for attribute reduction. *Recent Advances on Soft Computing and Data Mining* (pp. 195–203). Cham: Springer.
67. Mafarja, M., & Abdullah, S. (2015). A fuzzy record-to-record travel algorithm for solving rough set attribute reduction. *International Journal of Systems Science*, *46*(3), 503–512.
68. 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.
69. Mafarja, M., Aljarah, I., Heidari, A. A., Hammouri, A. I., Faris, H., AlaM, A. Z., et al. (2018). Evolutionary population dynamics and grasshopper optimization approaches for feature selection problems. *Knowledge-Based Systems*, *145*, 25–45.
70. Mafarja, M., Eleyan, D., Abdullah, S., & Mirjalili, S. (2017). 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). ACM.
71. Mafarja, M., Jaber, I., Eleyan, D., Hammouri, A., & Mirjalili, S. (2017). Binary dragonfly algorithm for feature selection. In *2017 International Conference on New Trends in Computing Sciences (ICTCS)* (pp. 12–17).
72. Mafarja, M., & Mirjalili, S. (2017). Hybrid whale optimization algorithm with simulated annealing for feature selection. *Neurocomputing*.
73. Mafarja, M., & Mirjalili, S. (2017). Whale optimization approaches for wrapper feature selection. *Applied Soft Computing*, *62*, 441–453.
74. Mirjalili, S. (2015). The ant lion optimizer. *Advances in Engineering Software*, *83*, 80–98.
75. Mirjalili, S. (2016). Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing and Applications*, *27*(4), 1053–1073.
76. Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software*, *114*, 163–191.
77. Mirjalili, S., & Lewis, A. (2013). S-shaped versus V-shaped transfer functions for binary particle swarm optimization. *Swarm and Evolutionary Computation*, *9*, 1–14.
78. Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software*, *95*, 51–67.
79. Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, *69*, 46–61.
80. Moradi, P., & Gholampour, M. (2016). A hybrid particle swarm optimization for feature subset selection by integrating a novel local search strategy. *Applied Soft Computing*, *43*, 117–130.

81. Pathania, A. K., Mehta, S., & Rza, C. (2016). Economic load dispatch of wind thermal integrated system using dragonfly algorithm. In *2016 7th India International Conference on Power Electronics (IICPE)* (pp. 1–6). IEEE.
82. Rajamohana, S., & Umamaheswari, K. (2018). Hybrid approach of improved binary particle swarm optimization and shuffled frog leaping for feature selection. *Computers & Electrical Engineering*.
83. Raman, G., Raman, G., Manickam, C., & Ganesan, S. I. (2016). Dragonfly algorithm based global maximum power point tracker for photovoltaic systems. In Y. Tan, Y. Shi, & B. Niu (Eds.), *Advances in Swarm Intelligence* (pp. 211–219). Cham: Springer International Publishing.
84. Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *ACM SIGGRAPH Computer Graphics*, vol. 21 (pp. 25–34). ACM.
85. Salam, M. A., Zawbaa, H. M., Emary, E., Ghany, K. K. A., & Parv, B. (2016). A hybrid dragonfly algorithm with extreme learning machine for prediction. In *2016 International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)* (pp. 1–6). IEEE.
86. Sambandam, R. K., & Jayaraman, S. (2016). Self-adaptive dragonfly based optimal thresholding for multilevel segmentation of digital images. *Journal of King Saud University-Computer and Information Sciences*.
87. Saremi, S., Mirjalili, S., & Lewis, A. (2017). Grasshopper optimisation algorithm: Theory and application. *Advances in Engineering Software*, *105*, 30–47.
88. Shilaja, C., & Ravi, K. (2017). Optimal power flow using hybrid DA-APSO algorithm in renewable energy resources. *Energy Procedia*, *117*, 1085–1092.
89. Sekhar, A. H., & Devi, A. L. (2016). Analysis of multi tcsc placement in transmission system by using firing angle control model with heuristic algorithms. *ARPN Journal of Engineering and Applied Sciences*, *11*(21), 12743–12755.
90. Shukri, S., Faris, H., Aljarah, I., Mirjalili, S., & Abraham, A. (2018). Evolutionary static and dynamic clustering algorithms based on multi-verse optimizer. *Engineering Applications of Artificial Intelligence*, *72*, 54–66.
91. Song, J., & Li, S. (2017). Elite opposition learning and exponential function steps-based dragonfly algorithm for global optimization. In *2017 IEEE International Conference on Information and Automation (ICIA)* (pp. 1178–1183). IEEE.
92. Statnikov, A., Aliferis, C. F., Tsamardinos, I., Hardin, D., & Levy, S. (2004). A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. *Bioinformatics*, *21*(5), 631–643.
93. Sugave, S. R., Patil, S. H., & Reddy, B. E. (2017). DDF: Diversity dragonfly algorithm for cost-aware test suite minimization approach for software testing. In *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)* (pp. 701–707). IEEE.
94. Suresh, V., & Sreejith, S. (2017). Generation dispatch of combined solar thermal systems using dragonfly algorithm. *Computing*, *99*(1), 59–80.
95. Surowiecki, J., Silverman, M. P., et al. (2007). The wisdom of crowds. *American Journal of Physics*, *75*(2), 190–192.
96. Tharwat, A., Gabel, T., & Hassanien, A.E. (2017). Classification of toxicity effects of biotransformed hepatic drugs using optimized support vector machine. In A. E. Hassanien, K. Shaalan, T. Gaber, M. F. Tolba (Eds.), *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2017* (pp. 161–170). Cham: Springer International Publishing.
97. Tharwat, A., Gabel, T., & Hassanien, A.E. (2017). Parameter optimization of support vector machine using dragonfly algorithm. In *International Conference on Advanced Intelligent Systems and Informatics* (pp. 309–319). Springer.
98. Vanishree, J., & Ramesh, V. (2018). Optimization of size and cost of static var compensator using dragonfly algorithm for voltage profile improvement in power transmission systems. *International Journal of Renewable Energy Research (IJRER)*, *8*(1), 56–66.
99. VeeraManickam, M., Mohanapriya, M., Pandey, B. K., Akhade, S., Kale, S., Patil, R., et al. (2018). Map-reduce framework based cluster architecture for academic students performance prediction using cumulative dragonfly based neural network. *Cluster Computing*, 1–17.

100. Vikram, K. A., Ratnam, C., Lakshmi, V., Kumar, A. S., & Ramakanth, R. (2018). Application of dragonfly algorithm for optimal performance analysis of process parameters in turn-mill operations-a case study. In *IOP Conference Series: Materials Science and Engineering*, vol. 310 (p. 012154). IOP Publishing.
101. Wu, J., Zhu, Y., Wang, Z., Song, Z., Liu, X., Wang, W., et al. (2017). A novel ship classification approach for high resolution sar images based on the BDA-KELM classification model. *International Journal of Remote Sensing*, *38*(23), 6457–6476.
102. 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.