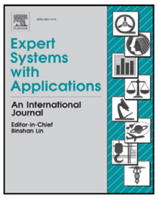
Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



Time-varying hierarchical chains of salps with random weight networks for feature selection

Hossam Faris[a,*], Ali Asghar Heidari[b,e], Ala' M. Al-Zoubi[a], Majdi Mafarja[c], Ibrahim Aljarah[a], Mohammed Eshtay[a], Seyedali Mirjalili[d]

[a] King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan
[b] School of Surveying and Geospatial Engineering, University of Tehran, Tehran, Iran
[c] Department of Computer Science, Birzeit University, Birzeit, Palestine
[d] Torrens University Australia, Brisbane, QLD 4006, Australia
[e] Department of Computer Science, School of Computing, National University of Singapore, Singapore

ARTICLE INFO

Article history:
Received 15 October 2018
Revised 3 August 2019
Accepted 24 August 2019
Available online 25 August 2019

Keywords:
Feature selection
Salp swarm algorithm
Optimization
Evolutionary algorithms

ABSTRACT

Feature selection (FS) is considered as one of the most common and challenging tasks in Machine Learning. FS can be considered as an optimization problem that requires an efficient optimization algorithm to find its optimal set of features. This paper proposes a wrapper FS method that combines a time-varying number of leaders and followers binary Salp Swarm Algorithm (called TVBSSA) with Random Weight Network (RWN). In this approach, the TVBSSA is used as a search strategy, while RWN is utilized as an induction algorithm. The objective function is formulated in a manner to aggregate three objectives: maximizing the classification accuracy, maximizing the reduction rate of the selected features, and minimizing the complexity of generated RWN models. To assess the performance of the proposed approach, 20 well-known UCI datasets and a number of existing FS methods are employed. The comparative results show the ability of the proposed approach in outperforming similar algorithms in the literature and its merits to be used in systems that require FS.

1. Introduction

Data dimensionality has a great impact on the performance of data mining and machine learning tasks. With the advanced data collection tools, a huge amount of data becomes available in many fields such as pattern recognition, image processing, and disease diagnosis systems. Dealing with such data requires massive time and space resources. Also, not all features included in those datasets have the same degree of importance for the field of study. Some of those features can be excluded without affecting the amount of information that can be extracted from the original datasets since they may be redundant or irrelevant. Feature selection (FS) has proved to be an effective tool for tackling this problem (Guyon & Elisseeff, 2003; Jain & Zongker, 1997; Liu & Motoda, 2012).

FS process tries to select the most informative features that can represent the whole feature set without information loss. FS methods can be classified to three different categories according to the degree of dependency on the class label in the dataset: supervised methods (correspond to labeled datasets) (Faris et al., 2018; Nie, Xiang, Jia, Zhang, & Yan, 2008), semi-supervised methods (correspond to partially labeled datasets) (Han et al., 2015), and unsupervised methods (correspond to unlabeled datasets) (Nie, Zhu, Li, 2016).

According to the involvement level of a learning algorithm in evaluating a feature subset, FS methods can be divided into filters, wrappers, and embedded methods (Zhao et al., 2010). In the filter methods, learning algorithms are not used, and features are ranked based on the internal relations between the data itself. Filter methods are known as fast methods, but the selection of features is not based on the performance of a specific learning task such as classification. Typical filter models include information gain (IG) (Quinlan, 1986), Chi-Square (Liu & Setiono, 1995), Gain Ratio (Quinlan, 1993), and ReliefF (Robnik-Šikonja & Kononenko, 2003).

* Corresponding author.
E-mail addresses: hossam.faris@ju.edu.jo (H. Faris), aliasghar68@gmail.com, aliasgha@comp.nus.edu.sg, as_heidari@ut.ac.ir (A.A. Heidari), alaah14@gmail.com (A.M. Al-Zoubi), mmafarja@birzeit.edu, mmafarjeh@gmail.com (M. Mafarja), i.aljarah@ju.edu.jo (I. Aljarah), m.eshtay@fgs.ju.edu.jo (M. Eshtay), ali.mirjalili@gmail.com, ali.mirjalili@laureate.edu.au (S. Mirjalili).
URL: http://www.alimirjalili.com (S. Mirjalili)

https://doi.org/10.1016/j.eswa.2019.112898
0957-4174/© 2019 Elsevier Ltd. All rights reserved.

As opposed to the filter methods, wrappers determine a predictor, and the feature subset is selected if it enhances the performance of that predictor (e.g., classification accuracy) (Kohavi & John, 1997; Oh, Lee, & Moon, 2004). Since the wrapper approaches tend to evaluate each feature using a specific learning algorithm (e.g., classifier), they can optimize the objective (e.g., classification accuracy), but the optimization process is usually slow. The embedded methods involve FS into the training process, where a trade-off between learning speed and model performance is desirable (Nie, Huang, Cai, & Ding, 2010; Tibshirani, 1996).

Besides the evaluation method in wrapper models (e.g., classifier), searching for the best feature subset is a challenging problem in FS methods. FS is known to be as an NP-hard problem with an exponential growth where 2^N solutions have to be handled when dealing with a dataset with N features. Three search strategies can be mainly utilized with FS methods:

- Complete (brute-force) search that tends to generate all possible solutions to select the best one
- Random searches that select the subsets randomly in the hope to find the best one
- Heuristic approaches that use the heuristic information to guide the random search process.

Using complete and random methods with FS is impractical with medium and large-scale datasets and a random search becomes a completer search in the worst case. Heuristic search methods provide general ways that tradeoff between local search and global search to find good solutions (not necessarily the best) within a reasonable running time (Talbi, 2009).

Nature-inspired metaheuristic algorithms are regarded as powerful algorithms with superior performances in treating different optimization scenarios including machine learning and FS (Shrivastava, Shukla, Vepakomma, Bhansali, & Verma, 2017). In previous works, different categories of heuristics such as evolutionary methods (e.g., genetic algorithms (GA) (Holland, 1992)) and swarm intelligence-based techniques (e.g., particle swarm optimization (PSO) (Kennedy & Eberhart, 1997)) have been applied to the FS problem (Mafarja, Jarrar, Ahmad, & Abusnaina, 2018c; Mafarja & Sabar, 2018). Swarm intelligence (SI) is mainly inspired by the intelligent, collective behavior of decentralized and self-organized swarms. SI methods include but not limited to Ant Colony Optimization (ACO) (Dorigo, Birattari, & Stutzle, 2006), Grey Wolf Optimizer (GWO) (Mirjalili, Mirjalili, & Lewis, 2014), Dragonfly Algorithm (DA) (Mirjalili, 2016), Water Cycle Algorithm (WCA) (Heidari, Abbaspour, & Jordehi, 2017), Krill Herd (KH) (Gandomi & Alavi, 2012) algorithm, Whale Optimization Algorithm (WOA) (Al-Zoubi, Faris, Alqatawna, Hassonah, 2018; Mirjalili & Lewis, 2016), and Firefly Algorithm (FA) (Yang, 2009). Salp Swarm Algorithm (SSA) (Mirjalili et al., 2017) is a well-established, recent SI algorithm that mimics the collective foraging behavior of salps in oceans and seas. The SSA showed its efficiency in realizing the optimal solutions of many optimization problems including FS (Faris et al., 2018; Sayed, Khoriba, & Haggag, 2018) and parameters tuning of PEM fuel cells (El-Fergany, 2018).

With a large number of algorithms published, one can ask if there is a need to propose new methods to solve FS problem. Referring to No-Free-Lunch (NFL) theorem (Wolpert & Macready, 1997), there is no universally best method that can solve all possible classes of FS problems. Therefore, the door is still open for proposing new optimization methods or improving the existing ones for tackling FS problems more efficiently.

In addition, the majority of the previous works in the area of unsupervised, wrapper based approaches have focused on utilizing k-NN as an induction algorithm due to its simplicity and speed. This is to reduce the computational overhead of the wrapper. However, the simplicity of the k-NN comes at the cost of the prediction power. It is well-shown that k-NN is not competitive in terms of prediction accuracy with more advanced algorithms in machine learning like neural networks and support vector machines.

The above-mentioned gaps at the search and induction levels in feature selection motivated our attempts to propose an improved version of a recent swarm intelligent algorithm called Salp Swarm Algorithm (SSA) in combination with Random Weight Networks (RWN) as a wrapper based approach for feature selection. SSA was recently proposed by Mirjalili et al. (2017). SSA mimics the swimming behavior as a group in a form of chain in oceans. The algorithm with its model of salp chains has shown promising results in a wide range of challenging real-world complex problems which motivated us to utilize it as the search level for feature selection. On the other side, RWN benefits from several advantages such as the rapid training process, high generalization performance, and the need for little human intervention and involvement. These advantages motivated us to investigate its efficiency as an induction algorithm in the wrapper to be a good candidate to replace the classical k-NN algorithm. As a potential drawback, however, there is no rule of thumb to set the number of hidden nodes in its structure. This is perhaps one of the reasons that prevent other researchers from utilizing RWN as an induction algorithm. The contributions in the proposed wrapper-based approach can be summarized as follows:

- A time-varying hierarchal based version of BSSA is proposed (TVBSSA) for the feature selection task. TVBSSA will dynamically change the number of leaders over the course of iterations, as it starts with few leaders to emphasize the exploration process but with the advancement of the iterations it increases them to intensify the exploitation process.
- Unlike most of the previous works, RWN is utilized as an induction algorithm in the proposed wrapper-based approach. To the best of our knowledge, there is a lack work that investigates and experiments the efficiency of RWN for this task on a large number of benchmark datasets.
- Another contribution in the proposed approach, is that the number of neurons in the hidden layer of RWN is automatically tuned by means of TVBSSA operators which eliminate the need for any additional effort to tune this important parameter.

The rest of this paper is organized as follows: Section 2 reviews previous works on wrapper based FS. The preliminaries of the main algorithms used in this work are given in Section 3. The proposed wrapper approach is described in details in Section 4. The experiments and results are discussed in Section 5. Finally, a summary of findings, conclusions, and future work are given in Section 6.

2. Previous works

Nature-inspired metaheuristic algorithms have been used as beneficial and efficient tools for searching the feature space because of their global (exploration) and local search (exploitation) capabilities. Recently, they were extensively used by the community of machine learning to tackle a wide spectrum of FS problems. Different types of metaheuristic algorithms including evolutionary algorithms (e.g. GA), swarm intelligence algorithms (e.g. PSO) and ACO) and others (e.g. Memetic algorithms) were utilized to tackle these problems.

GA is one of the first evolutionary algorithms that were employed in various wrapper FS methods with different classifiers. GA was used with SVM classifier as an evaluator to design wrapper FS approaches as in Chen, Chan, and Wu (2008); Seo, Lee, and Kim (2014); Shoorehdeli, Teshnehlab, and Moghaddam (2006);

Tan, Fu, Zhang, and Bourgeois (2008). While other wrapper-based approaches used KNN (Cho et al., 2008; Da Silva, Ribeiro, Neto, Traina-Jr, & Traina, 2011; Derrac, García, & Herrera, 2009; Emary, Zawbaa, & Hassanien, 2016a; Faris et al., 2018; Jeong, Shin, & Jeong, 2015; Mafarja & Mirjalili, 2018; Seo et al., 2014; Winkler, Affenzeller, Jacak, & Stekel, 2011). ANN also has been combined with GA for FS approaches in (Hong & Cho, 2006; Tan et al., 2008; Yang & Honavar, 1998; Yusta, 2009).

Derrac et al. (2009) introduced a co-evolutionary algorithm based on GA for FS. The population is divided into three parts, the first part was used for FS, the second was used for instance selection, while the third was used for both instance and FS. Rather than the wrapper approaches, two hybrid filter-wrapper approaches that used GA as a search strategy were proposed in Oreski and Oreski (2014); Tan et al. (2008), two variants of GA were employed as wrapper-FS methods (Li, Lu, Zhang, & Zhao, 2010). In those approaches, KNN was used as a base classifier in a dynamic Adaboost learning model. Another approach, that used GA for both FS and optimizing the structure of the MLP classifier, was proposed in Souza, Matias, and Araójo (2011).

In addition, various SI algorithms were employed to search the feature space in different FS methods. For instance, the PSO algorithm was combined with SVM classifier in many wrapper FS methods as in Alba, Garcia-Nieto, Jourdan, and Talbi (2007); Tang, Suganthan, and Yao (2005); Unler, Murat, and Chinnam (2011). Other approaches that adopted KNN classifier as an evaluator were proposed as in (Xue, Zhang, & Browne, 2012; 2013). Moreover, some approaches used the ANN classifier (Agrafiotis & Cedeno, 2002; Huang & He, 2007)

An ACO variant was used with SVM for a wrapper FS approach in a face recognition application in (Yan & Yuan, 2004). ACO was combined with DE in the work presented by Khushaba, Al-Ani, Al-Sukker, and Al-Jumaily (2008), in which the former algorithm was used to find initial solutions and the latter algorithm improved the quality of solution obtained. Another work based on ACO that is named (ABACO) was proposed by Kashef and Nezamabadi-pour (2013). A recent FS approach that utilized a hybrid ACO-ABC model (called AC-ABC) was proposed by Shunmugapriya and Kanmani (2017) to benefit from advantages of both ABC and ACO.

A Grey Wolf Optimizer (GWO) based wrapper approach was proposed by Emary, Zawbaa, and Hassanien (2016b) and applied to select an optimal subset of features for classification problems. Two binary versions of GWO were tested and compared to GA and PSO. Another wrapper approach based on KNN used Whale Optimization Algorithm (WOA) was introduced by Mafarja and Mirjalili (2018), in which several variants of binary WOA were considered to search for the optimal feature subset for classification problems. The same authors proposed another wrapper based algorithm that combines WOA with Simulated Annealing (SA) to solve the problem of feature selection (Mafarja & Mirjalili, 2017). The idea was to require SA to exploit the promising regions found by WOA.

A recent binary Ant Lion Optimizer (ALO) eas proposed in Emary et al. (2016a) to handle the feature selection problem. Recently, Faris *et al.* proposed a FS wrapper approach based on a newly proposed metaheuristic called Salp Swarm Algorithm (SSA) and KNN. Many other FS wrapper approaches based on SI algorithms have been proposed such as Binary Cuckoo Search (Rodrigues et al., 2013), Gravitational Search Algorithm (GSA) (Rashedi & Nezamabadi-pour, 2014), Binary Bat Algorithm (BBA) (Nakamura et al., 2013), Harmony Search (HS) (Ramos, Souza, Chiachia, Falcão, & Papa, 2011; Zainuddin, Lai, & Ong, 2016), Competitive Swarm Optimizer (Gu, Cheng, & Jin, 2018) and Binary Grasshopper Optimization Algorithm (BGOA) (Mafarja et al., 2018b).

As previously mentioned, various metaheuristic methods adopted as the wrapper approach for FS. Many of them are using one of the popular classifiers in wrapper approach, that is KNN. However, few research works employed RWN instead of KNN. In Chyzhyk, Savio, and Graña (2014), the authors have used Extreme Learning Machine (ELM) that can be implemented a special type of RWN to evaluate the quality of the selected features. They utilized GA to explore the search space for the best subset of features and applied their method on Alzheimer's disease dataset. Another work was done by Yang, Zhou, and Tsui (2016) that used both DE and ELM to deal with a tool wear estimation application. The developed algorithm is divided into two phases. In the first phase, a discrete DE was employed with ELM to select the input features. In the second phase, a continuous DE was used to optimize the kernel function parameters of ELM. The results showed the superiority of the proposed method in tool wear estimation.

3. Preliminaries

3.1. Salp swarm algorithm

SSA is a new meta-heuristic technique recently proposed by Mirjalili et al. (2017) to efficiently find the optimal solutions for different classes of constrained and unconstrained problems. Salp is known as a type of Salpidae family. Swarming behaviors of salps is a very noticeable phenomenon because salps can build cooperative semi-parallel chains throughout foraging events in the deep oceans. Based on the collaborative strategies of salps, they can preserve more energy when foraging. The SSA method is developed based on the behavior of salps in forming chains and their foraging tactics. From optimization point of view, these chains have a significant impact on balancing the exploration and exploitation inclinations of SSA by assisting it in escaping from local optima (LO) and avoiding stagnation problems.

In SSA, the population is formed using some chains of salps, which consist of two types of salps: leader and followers. When the agent is front-runner, we label it as leader, whereas other salps will be classified as followers. The role of leader salp is to guide and lead the direction and next steps of population, while the follower salps pay attention to other peers. A schematic view of salp chain is shown in Fig. 1.

In SSA, position of salps during the exploration and exploitation phases is defined as an n-dimensional space, where n is total number of variables. For a set of salps X made up of N salps with d dimensions, population of SSA is recorded in a $(N \times d)$-dimensional matrix, as described in Eq. (1):

$$X_i = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_d^1 \\ x_1^2 & x_2^2 & \dots & x_d^2 \\ \vdots & \vdots & \dots & \vdots \\ x_1^N & x_2^N & \dots & x_d^N \end{bmatrix} \tag{1}$$

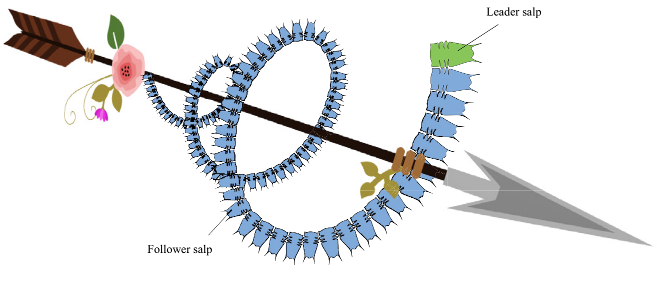


Fig. 1. Illustration of a salp chain.

In SSA, all salps are trying to forage and use the food source using the salp chains. Therefore, food source is the target location of swarm and the position of leader salp is determined based on situation of this source. This rule is expressed in Eq. (2):

$$x_j^1 = \begin{cases} F_j + c_1\left(\left(ub_j - lb_j\right)c_2 + lb_j\right) & c_3 \geq 0.5 \\ F_j - c_1\left(\left(ub_j - lb_j\right)c_2 + lb_j\right) & c_3 < 0.5 \end{cases} \tag{2}$$

where x_j^1 denotes the location of leader, and F_j is the state of food source in the jth dimension, ub_j is the superior restriction of jth dimension, and lb_j shows the inferior limit of jth dimension, c_2 and c_3 are two random values inside [0, 1], and c_1 can be described as a significant parameter in SSA, which is expressed as in Eq. (3):

$$c_1 = 2e^{-\left(\frac{4t}{T_{max}}\right)^2} \tag{3}$$

where t is the iteration, and $T_{\max}$ defines the maximum number of iterations. The parameter c_1 plays a noticeable role in harmonizing the exploration and exploitation propensities of SSA. The position vector of salps are defined and updated in each iteration as in Eq. (4):

$$x_j^i = \frac{x_j^i + x_j^{i-1}}{2} \tag{4}$$

where $i \geq 2$ and x_j^i represents the situation of ith salp at the jth dimension. The pseudo-code of conventional SSA is shown in Algorithm 1.

Algorithm 1 Pseudo-code of SSA.

```
**Input**: Swarm size and total number of iterations ($T_{\max}$).
**Output**: The best salp and its fitness value.
Initialize the random salps $x_i (i = 1, 2, \ldots, n)$
**while** (Termination condition is not true) **do**
    Evaluate the fitness of salps (search agents)
    Determine the fittest agent and set it as **F**
    Update $c_1$ by Eq. (3)
    **for** (each salp ($x_i$)) **do**
        **if** ($i == 1$) **then**
            Update the position vector of leader by Eq. (2)
        **else**
            Update the location of follower salps by Eq. (4)
    Update all agents with regard to the superior and inferior bounds of decision variables
    Return those salps that violated the bounds.
Return **F**
```

Considering Algorithm 1, after defining the population size and possible number of iterations, the SSA algorithm first distributes random salps inside the defined search space. It then evaluates the objective value (fitness) of all salps to sort and detect the best search agent F. All salps are interested to track and follow the prime candidate salp (leader) as shown in Fig. 2. During searching process, the variable c_1 is calculated in each step by Eq. (3). The formula in Eq. (2) can assist SSA in finding and updating the location of leader, whereas Eq. (4) is utilized to inform other salps. Pending the termination state, all phases of SSA, apart from the initialization stage, should be repeated to rise the excellence of agents as much as possible and find a suboptimal or optimum solution for the intended problem.

3.2. Random weight networks (RWN)

In 1992, Schmidt, Kraaijveld, and Duin (1992) introduced RWNs for tackling the training phases of single-hidden layer feed-forward neural networks (SLFNs) (Cao, Wang, Ming, & Gao, 2018). Later, Pao, Park, and Sobajic (1994) also established a similar model termed random vector functional-link networks (RVFLNs). The RWN was developed to generalize the SLFN. Also, it aims to generalize multi hidden-layer feed forward networks where a node is considered a subnetwork connecting additional hidden nodes. In regards to gradient descent approaches used for training of SLFN, we can observe that learning ratio of RWN is particularly fast. It also can provide a better generalization routine. What is more, not like methods such as back propagation, which the user should manually set the parameters such as learning rate, number of epochs, etc., the RWN does not necessitate thoroughgoing human involvement.

However, RWN does not optimize in an iterative manner to handle the parameters of SLFN. As an alternative, input weights and biases will be randomly initialized, and then it concludes the output weighting vector systematically based on Moore-Penrose generalized inverse. There are more learning and generalization characteristics for the RVFLNs. Interested readers are referred to Pao et al. (1994) and Igelnik and Pao (1995). In RVFLNs, direct links are established and initialized from input layer and lead to output layer. This fact is the only dissimilarity among RWNs and RVFLNs. Considering the satisfactory learning rate, generalization and approximation proficiencies of RWNs and RVFLNs, they have involved researchers in diverse areas of engineering and science, significantly.

Suppose N distinct samples represented by (x_i, t_i), where $x_i = [x_{i1}, x_{i2}, \ldots, x_{iN}]^T \in \mathbb{R}^n$ and $t_i = [t_{i1}, t_{i2}, \ldots, t_{iN}]^T \in \mathbb{R}^m$, SLFNs with activation function $g(x)$ and $\tilde{N}$ hidden neurons are formulated as in Eq. (5) (Huang, Zhu, & Siew, 2004):

$$\sum_{i=1}^{\tilde{N}} \beta_i g(\omega_i \cdot x_j + b_i) = o_j, j = 1, \ldots, N \tag{5}$$

where b_i is the threshold of the ith hidden neuron, $w_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^T$ is the vector consists of weighting coefficients for linking the ith hidden neuron to input elements, $\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T$ denotes the weighting coefficients to link the ith hidden element to the output elements (Huang et al., 2004).

The SLFNs with $\tilde{N}$ hidden neurons and activation function $g(x)$ should approximate the original N samples with zero error, that is $\sum_{i=1}^{\tilde{N}} \left\| o_j - t_j \right\| = 0$, i.e., we know w_i, β_i, and b_i, such that (Huang et al., 2004):

$$\sum_{i=1}^{\tilde{N}} \beta_i g(\omega_i \cdot x_j + b_i) = t_j, j = 1, \ldots, N \tag{6}$$

The above-stated N rules are described as in Eq. (7):

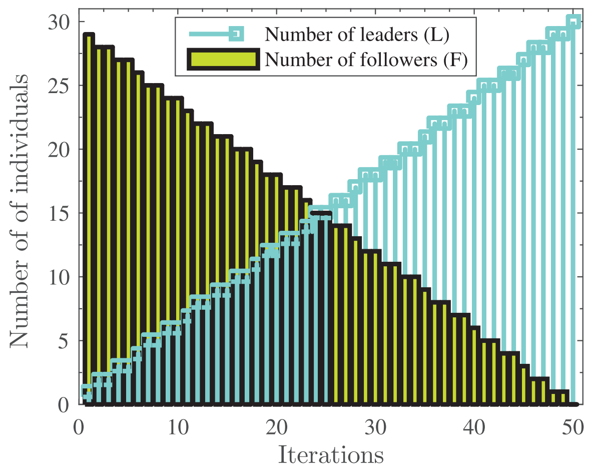
$$H\beta = T \tag{7}$$

where

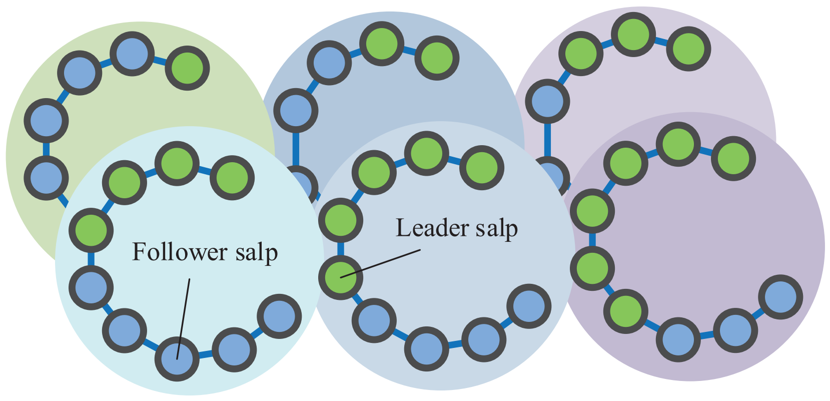
$$H(w_1, \ldots, w_N, b_1, \ldots, b_N, x_1, \ldots, x_N) = \begin{bmatrix} g(w_1.x_1 + b_1) & \ldots & g(w_{\tilde{N}}.x_1 + b_{\tilde{N}}) \\ \vdots & & \vdots \\ g(w_1.x_N + b_1) & \ldots & g(w_{\tilde{N}}.x_N + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}} \tag{8}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \times m}, \quad T = \begin{bmatrix} t_1^T \\ \vdots \\ t_{\tilde{N}}^T \end{bmatrix}_{N \times m} \tag{9}$$

where H shows the hidden layer output matrix, the ith column of H is the ith hidden output vector with respect to $x_1, x_2, \ldots, x_N$



(a) Number of leaders and followers



(b) Time-varying hierarchy in some iterations (order from upper left to lower right)

Fig. 2. Monitoring of the number of leaders and followers and structure of hierarchy over the course of iterations in TVBSSA.

(Huang et al., 2004). The general structure of RWN is demonstrated in Fig. 3.

4. Proposed TVBSSA-RWN wrapper

As any wrapper-based feature selection method, our proposed approach consists of three main components as follows:

- The search algorithm: which will be implemented as a dynamic version of the SSA optimizer.
- The induction algorithm: in our case the random weight network will be utilized.
- The evaluation metric: Our proposed evaluation metric will consist of three main parts that combine the classification accuracy, the feature reduction rate and complexity of the induction algorithm.

4.1. Binary SSA

Based on Mirjalili and Lewis (2013), one of the efficient techniques in converting continuous algorithms to binary versions is to utilize transfer functions (TFs). TFs are proposed based on definition of a probability value for updating each individual's element

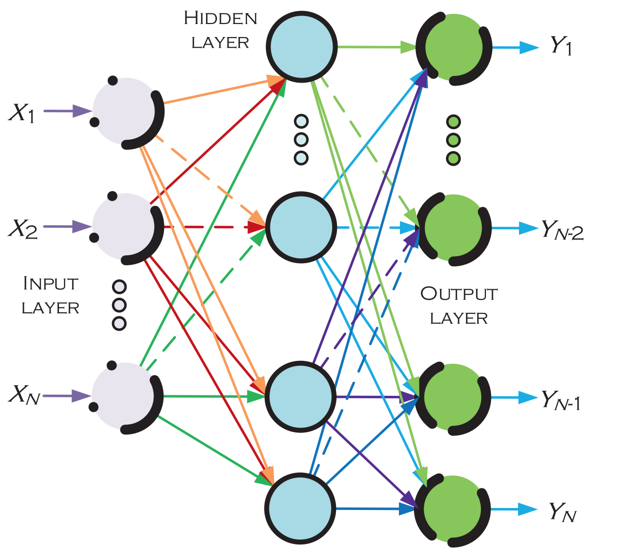


Fig. 3. Overall structure of RWN.

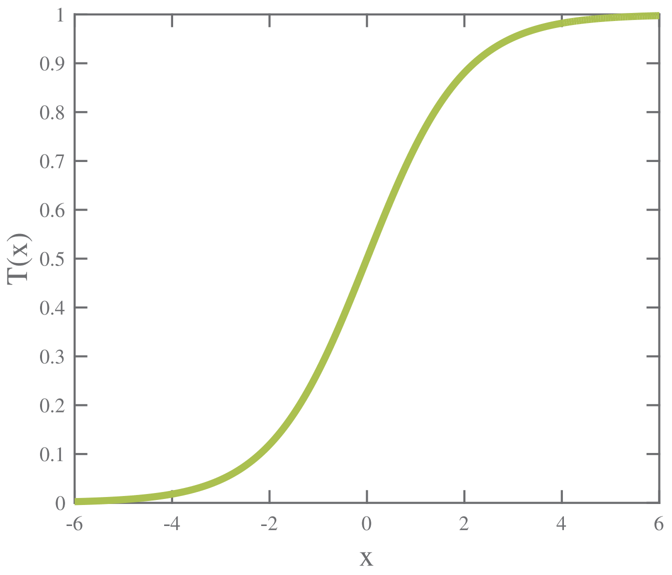


Fig. 4. Utilized S-shaped TF.

representing the selected subset of features. In this work, we utilize one of the most commonly used S-shaped TF to covert the continuous SSA to a binary variant, which is called BSSA. The S-shaped TF can be calculated by Eq. (10).

$$T(x_j^i(t)) = \frac{1}{1+e^{-x_j^i(t)}} \tag{10}$$

where $X_i^d(t+1)$ is the ith element at dth dimension in X solution, $T(x_j^i(t))$ is the probability value, which can be obtained via Eq. (10). Then, the valued of this function is used to generate ones and zeros in the individual as given in Eq. (11).

$$x_i^k(t+1) = \begin{cases} 0 & \text{If } rand < T(v_i^k(t+1)) \\ 1 & \text{If } rand \geq T(v_i^k(t+1)) \end{cases} \tag{11}$$

where x_j^i is the jth element in x solution in the jth dimension, and t is the current iteration. The utilized TF is shown in Fig. 4.

4.2. Time-Varying Binary Salp Swam Algorithm (TVBSSA)

The basic binary SSA (BSSA) has a satisfactory performance in balancing the exploration and exploitation. However, there is room for further improvements in the searching leanings of binary SSA because it may stagnate in local optima (LO), and premature convergence can be happen in dealing with some feature spaces. In order to alleviate the stagnation disadvantages of BSSA, a modified leadership structure is proposed in this work to develop an improved binary variant called TVBSSA. One important phenomenon that has a significant impact on the performance of SSA is the way the follower salps will follow the leader salp during the searching phases. In SSA, there is one leader in each iteration, which is followed by (N−1) salps, where N is the number of all agents. Because we need a high exploration in beginning of the search and more exploitative steps in the last stages, a dynamic time-varying structure for the leadership hierarchy of SSA is utilized here. During the course of iterations, this strategy linearly increases the number of leaders and decreases the number of follower salps. In each iteration, several leaders are determined and then, the rest of population is employed to play the role of follower salps in the exploration and exploitation phases.

Hence, unlike the BSSA, in the time-varying version of BSSA (TVBSSA), the number of leaders and followers change of the course of iterations. In this regard, the number of leaders is determined based on the following rule:

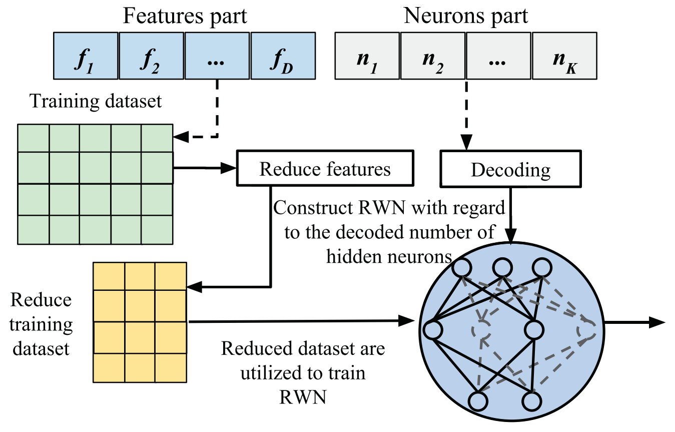


Fig. 5. Solution representation in TVBSSA-RWN.

$$L = ceil\left(N \times \frac{i}{MaxIter+1}\right) \tag{12}$$

where N is the population size, i is the index of each salp, and *MaxIter* is the upper limit for iterations. According to Eq. (12), the number of follower salps (F) in each iteration can be obtained via Eq. (13)

$$F = N - L \tag{13}$$

The pseudo-code of proposed TVBSSA is shown in Algorithm 2.

4.3. Solution representation

In TVBSSA-RWN, each individual (salp) represents a candidate solution for the targeted problem. In our case, the solution is to find the near-optimal subset of features and to tune the number of neurons in the hidden layer of the network, simultaneously. Therefore, the individual in our proposed algorithm is designed to capture these two parts of the solution. As shown in Fig. 5, each individual is represented as a one-dimensional binary array of two parts: the features part (f) and the neurons part (n). The length of the features part equals to the number of features in the dataset D, while the neurons part consists of a number of bits K required to

Algorithm 2 Pseudo-code of TVBSSA.

```
Initialize the salp population $x_i (i = 1, 2, \ldots, n)$ considering $ub$
and $lb$
while (end condition is not satisfied) do
    Calculate the fitness of each search agent (salp)
    Set F as the best search agent
    Update $c_1$ by Eq. 3
    Update number of leaders $L$ by Eq. (12)
    Update number of followers $F$ by Eq. (13)
    for (each salp ($x_i$)) do
        if $x_i$ is a leaser then
            Update the position of the leading salp by Eq. (2)
            Calculate the probabilities using a TF which takes the
output
            of Eq. (2) as its input as in Eq. 10 (S-Shaped)
        else
            Update the position of the follower salp by Eq. (4)
    Update all salps based on the upper and lower bounds of
variables
Return the computed F
```

represent the maximum number of neurons. Therefore, the length of the individual of TVBSSA-RWN is $D + K$.

The process of decoding the individual representation is illustrated in Fig. 5. The features part used to select the features in the dataset that corresponds to the bits that has value of 1, while the other features are eliminated and the training dataset is reduced. The reduced dataset is used to train a RWN that has a number of neurons determined by the neurons part n. The resulted network is evaluated as described next.

4.4. Fitness evaluation

This component of the model is required to evaluate the quality (also known as fitness of in metaheuristic algorithms) of the generated solutions by the swarm optimizer. For this, the fitness function is designed to maximize the accuracy of the induction algorithm (i.e RWN in our case), minimize the number of selected features, and to minimize the complexity of the generated RWN by reducing the number of selected hidden neurons in the network.

$$Fitness = \alpha AccErr + \beta \frac{f}{F} + \gamma \frac{n}{N} \tag{14}$$

where $AccErr$ is the misclassification rate of the generated RWN network, f is the number of selected features by the BSSA, F is the number of features in the dataset, n represents the number of neurons determined by the optimizer, and N is the maximum number of possible neurons in the RWN network. The parameters α, β, and γ are three factors to control the weight of the contribution of their corresponding terms which are the misclassification rate, the reduction rate, and the complexity of the network, respectively. The maximum value of α, β, γ is 1 while minimum is 0. Typically, $\beta = (1 - \alpha)$ as in Emary et al. (2016a).

4.5. Procedure of TVBSSA-RWN

Combining all the parts described earlier, the procedure of the proposed TVBSSA-RWN can be described in the following steps:

- *Initialization*: A predefined number of individuals (i.e. salps) are randomly initialized. Each of these individuals is a candidate solution for the FS problem, and it consists of the feature selection part and the number of hidden neurons in the RWN part.
- *Fitness evaluation*: each individual in the swarm is evaluated using the following steps:
  - Individual split: each individual is divided into the features and neurons parts.
  - Feature selection and reduction of training dataset: The features part is utilized to select its corresponding features and eliminate non-selected features from the training dataset.
  - Network construction: The RWN is built using the specified number of hidden nodes by the individual.
  - Network training: RWN is trained based on the reduced training dataset.
  - Network evaluation: The developed RWN network is evaluated based on a validation set
  - Fitness calculation: the final fitness value is calculated as given in Eq. (14).
- *Hierarchy update*: In this process, the number of leaders and followers are updated according to Eqs. (12) and (13), respectively.
- *Exploit and Explore*: The updating mechanisms described in Eqs. (2) and (4) are applied to the leaders and followers to perform the exploitation and exploration processes.
- *Termination*: The previous steps are performed iteratively until the maximum number of iterations is reached.

Table 1
List of datasets .

No.	Dataset	No. of Features	No. of instances
1.	BreastEW	30	596
2.	Exactly	13	1000
3.	Exactly2	13	1000
4.	HeartEW	13	270
5.	Lymphography	18	148
6.	M-of-n	13	1000
7.	PenglungEW	325	73
8.	SonarEW	60	208
9.	SpectEW	22	267
10.	CongressEW	16	435
11.	KrvskpEW	36	3196
12.	Tic-tac-toe	9	958
13.	Vote	16	300
14.	WaveformEW	40	5000
15.	WineEW	13	178
16.	Clean1	166	476
17.	Semeion	265	1593
18.	Colon	2000	62
19.	Leukemia	7129	72
20.	TOX-171	5749	171

5. Experimental results and discussion

In this section summarizes the results of the proposed SSA with dynamic updating strategies for different FS datasets are presented and analyzed.

5.1. Experimental setup

All experiments are conducted in similar conditions to provide fair comparisons. All methods are coded in MATLAB 2013, and we used a same system with Intel Core(TM) i5-5200U 2.2 GHz of CPU and 4.0GB of RAM.

The parameters of the algorithms are set as follows. In PSO, acceleration constants c_1 and c_2 are set to 2, while the inertia weight is set to 0.9 to 0.4 as in Pacifico and Ludermir (2013). The probabilities of crossover and mutation operators in GA are set to 0.8 and 0.1, respectively, as in Yang, Yi, Zhao, and Dong (2013). For all algorithms, the population/swarm size is set to 50 with same iteration. The maximum number of hidden neurons in TVBSSA is set to 1024, which is represented by 10 elements in the chromosome. For training and testing, we applied 10-folds cross-validation. In this method, each of the training and testing processes is repeated 10 times, and then, the average (Avg) and standard deviation (Std) of the results are calculated.

Table 1 shows the main characteristics of 20 well-studied datasets used in this research to compare and investigate the efficacy of optimizers. These test cases are obtained from the well-regarded UCI repository (Lichman, 2013), which utilized in many works and covers varied characteristics and datasets with various features and instances.

5.2. Selection of (α β γ)

In this subsection, the impact of α, β, and γ values on the performance of TVBSSA is investigated based on the results for Leukemia dataset. These parameters play the role of weighting factors that can change the contribution of each component in the final fitness function.

Based on previous works, researchers always set α and β to 0.99 and 0.01, respectively (Emary, Zawbaa, & Grosan, 2018; Emary et al., 2016a; Faris et al., 2018; Mafarja et al., 2018a). Here, we investigate the impact of different combinations of α, β, and γ values as tabulated in Table 2. Table 2 shows the impact of

Table 2
Impact of α, β and γ on the performance of TVBSSA in terms of accuracy, number of features, and number of neurons for the Leukemia dataset.

Case	α	β	γ	Accuracy		Number of features		Number of neurons	
				Avg	Std	Avg	Std	Avg	Std
S_1	0.999	0.001	0.001	**0.9238**	0.0808	3958.8000	776.5847	493.4000	138.3267
S_2	0.995	0.005	0.005	0.9048	0.0670	3786.0000	1011.5861	505.9000	167.5340
S_3	0.99	0.01	0.01	0.9156	0.0366	4171.2600	405.7624	470.9400	25.7681
S_4	0.95	0.05	0.05	0.8321	0.2200	3570.2000	1268.9346	448.3000	173.2693
S_5	0.8	0.2	0.2	0.7655	0.1443	2096.3000	956.3636	192.4000	133.5808
S_6	0.5	0.5	0.5	0.6435	0.1551	**1359.4000**	374.0280	**33.8000**	42.7000

Table 3
Classification accuracy rates of the TVBSSA-based on s-shape and v-shape transfer functions.

Dataset	TVBSSA-RWN			
	S-Shape		V-Shape	
	Avg	Std	Avg	Std
BreastEW	**0.9616**	0.0053	0.9497	0.0114
Exactly	**0.9986**	0.0013	0.9958	0.0045
Exactly2	**0.7828**	0.0124	0.7780	0.0095
HeartEW	0.8652	0.0067	**0.8807**	0.0169
Lymphography	**0.8706**	0.0199	0.8587	0.0290
M-of-n	0.9922	0.0030	**0.9928**	0.0042
penglungEW	**0.7578**	0.0281	0.6979	0.0578
SonarEW	**0.8176**	0.0256	0.7958	0.0281
SpectEW	0.8750	0.0097	**0.8965**	0.0135
CongressEW	0.9637	0.0086	**0.9647**	0.0129
KrvskpEW	**0.9816**	0.0025	0.9735	0.0024
Tic-tac-toe	**0.9818**	0.0031	0.9534	0.0165
Vote	0.9611	0.0098	**0.9686**	0.0051
WaveformEW	**0.8307**	0.0051	0.8222	0.0039
WineEW	0.9546	0.0243	**0.9548**	0.0200
clean1	**0.8631**	0.0112	0.8593	0.0072
semeion	**0.9704**	0.0039	0.9652	0.0024
Colon	**0.7938**	0.0433	0.6119	0.0204
Leukemia	**0.9156**	0.0366	0.6493	0.0775
TOX-171	0.7625	0.0324	**0.7821**	0.0295

changing α, β, and γ values under six cases (S_1 to S_6) on the accuracy, number of features, and number of neurons obtained by TVBSSA. As per accuracy results in Table 2, we see that the TVBSSA with S_1 case outperforms TVBSSA with other cases in terms of accuracy rate. According to the number of features, we see the S_6 case comes to the first place. Based on number of neurons, S_6 case is the most preferable one. Because the most important criteria for the majority of FS scenarios is the accuracy of classification, however, we validated all methods based on the S_1 cases. Note that if we set any other value for these parameters, the performance of the proposed BDSSA can be repeated again.

5.3. Effect of transfer function

In this subsection, we study the effect of v-shape and s-shape transfer functions (Mirjalili & Lewis, 2013) on the performance of the proposed TVBSSA. Table 3 shows the results in terms of accuracy rate of two versions of TVBSSA one used s-shape and the other used v-shape. The results show that both versions of TVBSSA have very competitive results with a slight advantage for the version that incorporated s-shape transfer function, as it obtained the best average results in 13 datasets out of 20 datasets. Therefore, TVBSSA with s-shape transfer function will be used in the rest of experiments and comparisons.

5.4. Results and discussion

Table 4 shows the results of BSSA-RWN in terms of accuracy measure against other competitors BSSA, GA, and PSO using k-NN classifier. As per results in Table 4, we see that the BSSA with RWN can outperform all competitors on approximately 70% of datasets with satisfying STD results. After BSSA-RWN, the GA-k-NN has reached to the best classification accuracy rates on the M-of-n, SonarEW, and clean1 datasets. For penglungEW case, the PSO-k-NN shows the best rate. As it can be observed in the results of BSSA-k-NN, it cannot show superior efficacy on any dataset, especially compared to BSSA-RWN. However, when comparing BSSA-k-NN with GA-k-NN and PSO-k-NN, we observe that it cannot outperform them on majority of datasets. This observation indicants that the binary SSA outperforms GA and binary PSO when it is joined with a RWN classifier, and a significant improvement in the accuracy rates is seen due to the unique advantages of RWN over k-NN classifier. But when we integrate it with k-NN, the results cannot be better than BSSA-RWN variant. This also shows the significant role of RWN classifier on the accuracy rates.

Table 5 tabulates the results of BSSA-RWN in terms of number of selected features compared to other competitors BSSA, GA, and PSO with k-NN classifier. As per results in Table 5, we see that the GA-k-NN has attained the best rank.

Table 6 demonstrates the results of BSSA-RWN in terms of fitness values compared to other peers with k-NN classifier. The variation of fitness results are also visualized in Fig. 6 Referring to the results in Table 6, it is observed that BSSA with RWN has attained the fittest results on 90% of datasets, which is followed by GA-k-NN, PSO-k-NN, and BSSA-k-NN (see Fig. 6). These results indicate that the BSSA-based feature selection using RWN has increased the quality of found features compared to previous k-NN methods.

The accuracy results of proposed TVBSSA versus BSSA with RWN are compared in Table 7. As per results in Table 7, it is observed that the TVBSSA outperforms the BSSA in dealing with 80% of datasets. Table 8 compares TVBSSA to BSSA based on the number of features. As per results in Table 8, it is seen that the TVBSSA is superior to BSSA in terms of obtained set of features in dealing with Exactly, HeartEW, Lymphography, M-of-n, penglungEW, SonarEW, SpectEW, CongressEW, Vote, WineEW, semeion, and TOX-171.

The convergence curves of TVBSSA-RWN are compared to BSSA-RWN, BSSA-KNN, GA-KNN, and PSO-KNN methods in Figs. 7 and 8. The convergence trends for BreastEW, Exactly2, HeartEW, Lymphography, penglungEW, SonarEW, SpectEW, CongressEW, Tic-tac-toe, Vote, WaveformEW, WineEW, Semeion, Colon, Leukemia, and TOX-171 show that there is a significant gap between the curves of BSSA-KNN, GA-KNN, PSO-KNN and those for BSSA-RWN and TVBSSA-RWN methods, which are equipped with RWN learning schemes. This also indicates the significant impact of RWN' structure on convergence speed of the developed BSSA-based FS methods. The trends also show the unstable balance of exploratory

Table 4
Accuracy results obtained by BSSA-RWN against results of other methods using k-NN .

Dataset	BSSA-RWN		BSSA-k-NN		GA-k-NN		PSO-k-NN	
	Acc	Std	Acc	std	Acc	std	Acc	std
BreastEW	**0.9504**	0.0133	0.9455	0.0085	0.9409	0.0027	0.9409	0.0051
Exactly	0.9980	0.0014	0.9908	0.0054	**1.0000**	0.0000	0.9904	0.0039
Exactly2	**0.7744**	0.0087	0.7502	0.0126	0.7580	0.0000	0.7580	0.0000
HeartEW	**0.8600**	0.0173	0.8052	0.0127	0.7926	0.0166	0.7852	0.0172
Lymphography	**0.8557**	0.0178	0.7796	0.0188	0.8060	0.0166	0.7654	0.0283
M-of-n	0.9898	0.0041	0.9978	0.0015	**1.0000**	0.0000	0.9974	0.0018
penglungEW	0.7600	0.0278	0.8597	0.0268	0.8786	0.0186	**0.8972**	0.0233
SonarEW	0.7988	0.0070	0.8226	0.0081	**0.8239**	0.0071	0.8228	0.0301
SpectEW	**0.8688**	0.0134	0.7887	0.0097	0.7919	0.0212	0.7758	0.0137
CongressEW	**0.9614**	0.0064	0.9394	0.0153	0.9445	0.0064	0.9398	0.0079
KrvskpEW	**0.9804**	0.0024	0.9693	0.0026	0.9758	0.0023	0.9674	0.0024
Tic-tac-toe	**0.9787**	0.0046	0.8109	0.0116	0.8194	0.0038	0.8204	0.0104
Vote	**0.9601**	0.0090	0.9215	0.0068	0.9238	0.0107	0.9322	0.0052
WaveformEW	**0.8324**	0.0039	0.7832	0.0057	0.7970	0.0060	0.7874	0.0074
WineEW	**0.9619**	0.0203	0.9371	0.0106	0.9428	0.0105	0.9437	0.0108
clean1	0.8567	0.0156	0.8545	0.0102	**0.8583**	0.0119	0.8528	0.0084
semeion	**0.9705**	0.0035	0.9702	0.0017	0.9688	0.0039	0.9700	0.0021
Colon	**0.7805**	0.0301	0.7505	0.0456	0.7395	0.0431	0.7548	0.0424
Leukemia	**0.8994**	0.0412	0.8714	0.0126	0.8907	0.0142	0.8868	0.0101
TOX-171	0.7516	0.0343	**0.7528**	0.0467	0.7491	0.0204	0.7308	0.0288

Table 5
Number of features for BSSA-RWN against other methods using k-NN .

Dataset	BSSA-RWN		BSSA-k-NN		GA-k-NN		PSO-k-NN	
	Avg	Std	Avg	std	Avg	std	Avg	std
BreastEW	16.3600	0.7403	17.9800	1.3217	**12.6600**	0.8019	14.6400	0.7162
Exactly	12.8800	0.3271	6.6000	0.1225	**6.0000**	0.0000	6.6200	0.1304
Exactly2	13.9200	0.6419	1.9600	0.3286	**1.0000**	0.0000	**1.0000**	0.0000
HeartEW	10.8800	1.3773	8.7200	0.2168	7.5200	0.4025	**7.0200**	0.4438
Lymphography	13.3800	1.8580	12.0400	0.7335	10.5000	0.6325	**9.8400**	0.6348
M-of-n	12.8000	0.6519	6.8200	0.0837	**6.0000**	0.0000	6.9200	0.2049
penglungEW	**124.0200**	12.6458	191.3800	1.2398	150.2800	2.1476	155.2200	0.9935
SonarEW	40.9800	2.0705	38.5800	1.4096	**29.6800**	1.1189	30.5400	0.9633
SpectEW	16.3000	1.7776	15.0600	0.6731	**9.8000**	0.4690	10.5200	0.4207
CongressEW	11.0800	1.2071	7.0600	0.5899	**3.6400**	0.3362	4.9200	0.6573
KrvskpEW	34.1600	1.0455	23.5400	0.5941	**18.1600**	0.7232	20.0800	0.5263
Tic-tac-toe	14.6800	0.2490	6.9800	0.0447	**6.9600**	0.0894	6.9600	0.0548
Vote	10.7000	1.0173	7.0600	0.6731	**3.2000**	0.3674	4.3800	0.5167
WaveformEW	32.5800	0.9149	25.8200	1.1212	**19.6000**	0.5050	20.8200	0.8556
WineEW	7.6200	0.3701	8.6200	0.5263	**6.8000**	0.1732	7.1400	0.4037
clean1	115.4200	3.3214	107.0000	3.3830	**81.6600**	1.6832	82.6800	0.6140
semeion	162.1800	8.5377	168.2000	5.1841	132.4200	1.1946	**131.7800**	2.5956
Colon	1188.9000	87.6461	1272.9400	72.7751	**986.2800**	7.0871	988.1000	6.1053
Leukemia	4022.1600	211.6794	4744.3600	200.2358	**3538.5400**	8.8141	3542.4200	8.7833
TOX-171	3392.6400	142.5020	3851.4000	141.6246	**2863.8800**	6.0521	2868.9600	6.0521

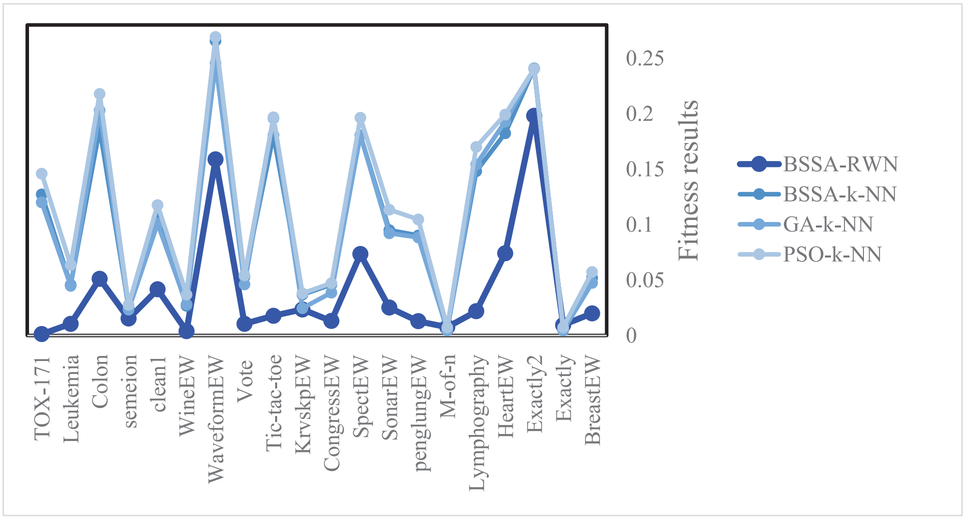
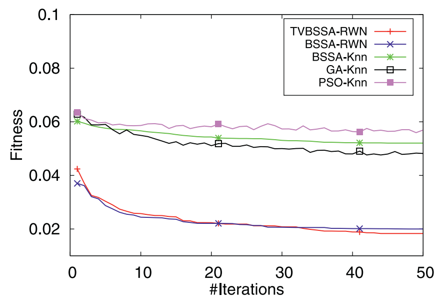
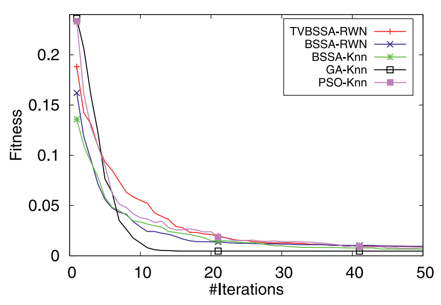


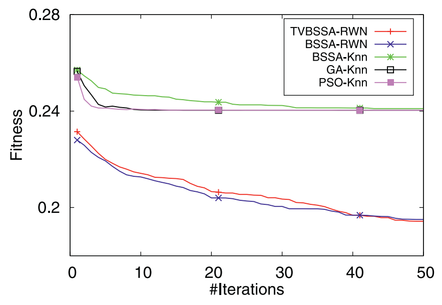
Fig. 6. Variation of fitness results.



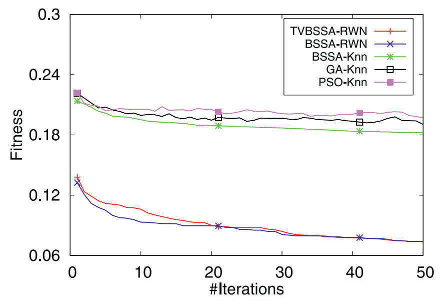
(a) BreastEW



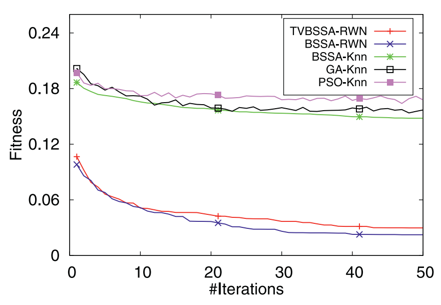
(b) Exactly



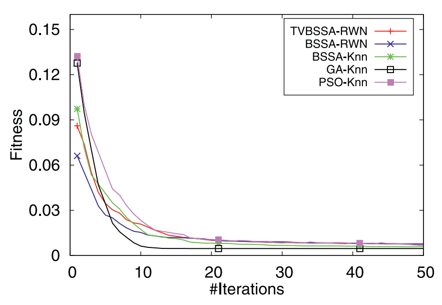
(c) Exactly2



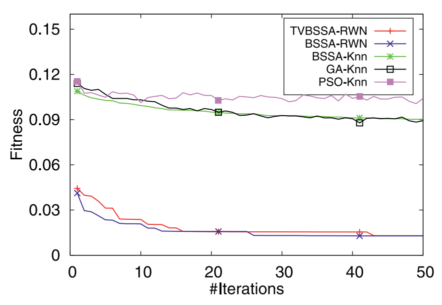
(d) HeartEW



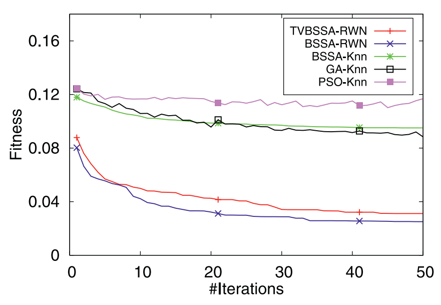
(e) Lymphography



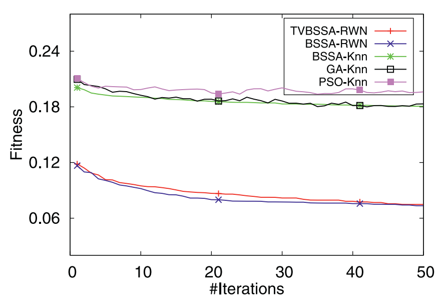
(f) M-of-n



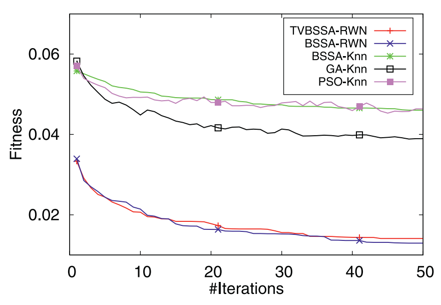
(g) penglungEW



(h) SonarEW

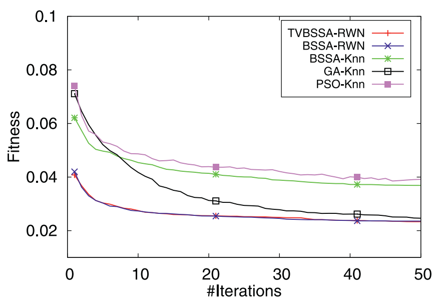


(i) SpectEW

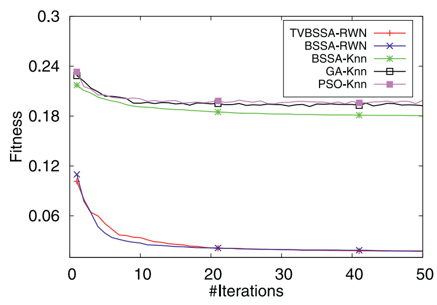


(j) CongressEW

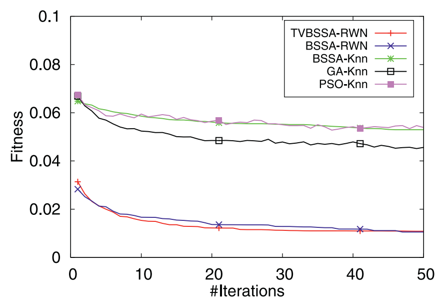
Fig. 7. Convergence curves for TVBSSA-RWN and other methods for BreastEW, Exactly, Exactly2, HeartEW, Lymphography, M-of-n, PenglungEW, SonarEW, SpectEW, and CongressEW benchmark datasets.



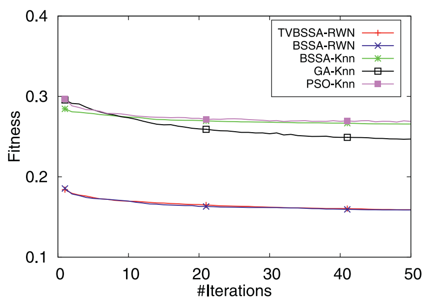
(a) KrvskpEW



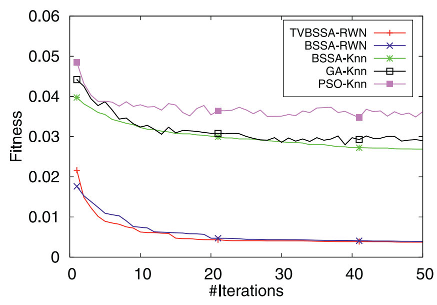
(b) Tic-tac-toe



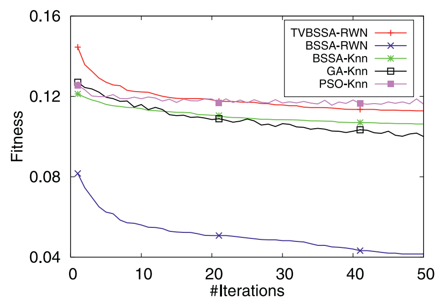
(c) Vote



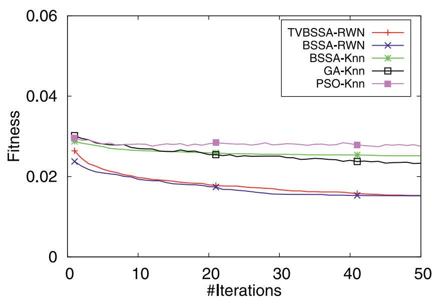
(d) WaveformEW



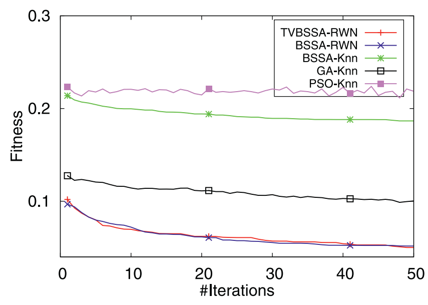
(e) WineEW



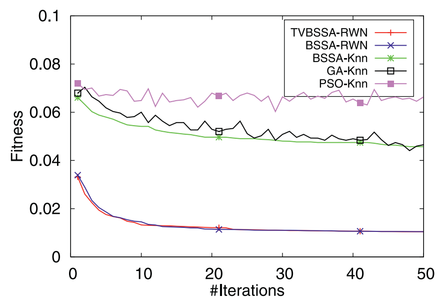
(f) Clean1



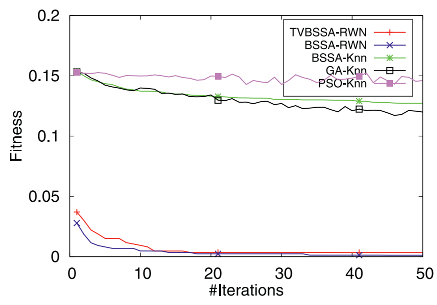
(g) Semeion



(h) Colon



(i) Leukemia



(j) TOX-171

Fig. 8. Convergence curves for TVBSSA-RWN and other methods for KrvskpEW, Tic-tac-toe, Vote, WaveformEW, WineEW, Clean1, Semeion, Colon, Leukemia, and TOX-171 datasets.

Table 6
Fitness for BSSA-RWN against other methods using k-NN.

Data	BSSA-RWN		BSSA-k-NN		GA-k-NN		PSO-k-NN	
	Avg	Std	Avg	std	Avg	std	Avg	std
BreastEW	**0.0200**	0.0014	0.0520	0.0014	0.0474	0.0008	0.0572	0.0023
Exactly	0.0094	0.0014	0.0066	0.0011	**0.0046**	0.0000	0.0073	0.0012
Exactly2	**0.1980**	0.0038	0.2410	0.0002	0.2404	0.0000	0.2404	0.0000
HeartEW	**0.0740**	0.0070	0.1823	0.0013	0.1930	0.0025	0.1992	0.0047
Lymphography	**0.0218**	0.0103	0.1481	0.0049	0.1542	0.0025	0.1700	0.0018
M-of-n	0.0075	0.0004	0.0058	0.0003	**0.0046**	0.0000	0.0073	0.0012
penglungEW	**0.0128**	0.0109	0.0903	0.0020	0.0884	0.0025	0.1048	0.0049
SonarEW	**0.0251**	0.0071	0.0951	0.0024	0.0921	0.0034	0.1134	0.0034
SpectEW	**0.0734**	0.0046	0.1807	0.0017	0.1806	0.0037	0.1965	0.0031
CongressEW	**0.0130**	0.0023	0.0460	0.0015	0.0387	0.0015	0.0471	0.0024
KrvskpEW	**0.0235**	0.0007	0.0369	0.0012	0.0247	0.0007	0.0379	0.0021
Tic-tac-toe	**0.0178**	0.0021	0.1806	0.0013	0.1960	0.0023	0.1969	0.0023
Vote	**0.0105**	0.0034	0.0530	0.0009	0.0462	0.0025	0.0543	0.0025
WaveformEW	**0.1587**	0.0009	0.2657	0.0013	0.2455	0.0012	0.2692	0.0015
WineEW	**0.0039**	0.0003	0.0269	0.0015	0.0281	0.0017	0.0370	0.0030
clean1	**0.0416**	0.0049	0.1063	0.0012	0.1009	0.0027	0.1177	0.0023
semeion	**0.0152**	0.0006	0.0252	0.0003	0.0233	0.0006	0.0276	0.0006
Colon	**0.0511**	0.0021	0.1866	0.0019	0.2029	0.0041	0.2179	0.0045
Leukemia	**0.0105**	0.0004	0.0457	0.0027	0.0447	0.0011	0.0636	0.0051
TOX-171	**0.0012**	0.0001	0.1273	0.0032	0.1200	0.0084	0.1460	0.0065

Table 7
Accuracy for BSSA-RWN versus TVBSSA-RWN.

Dataset	BSSA-RWN		TVBSSA-RWN	
	Acc	Std	Acc	Std
BreastEW	0.9504	0.0133	**0.9616**	0.0053
Exactly	0.9980	0.0014	**0.9986**	0.0013
Exactly2	0.7744	0.0087	**0.7828**	0.0124
HeartEW	0.8600	0.0173	**0.8652**	0.0067
Lymphography	0.8557	0.0178	**0.8706**	0.0199
M-of-n	0.9898	0.0041	**0.9922**	0.0030
penglungEW	**0.7600**	0.0278	0.7578	0.0281
SonarEW	0.7988	0.0070	**0.8176**	0.0256
SpectEW	0.8688	0.0134	**0.8750**	0.0097
CongressEW	0.9614	0.0064	**0.9637**	0.0086
KrvskpEW	0.9804	0.0024	**0.9816**	0.0025
Tic-tac-toe	0.9787	0.0046	**0.9818**	0.0031
Vote	0.9601	0.0090	**0.9611**	0.0098
WaveformEW	**0.8324**	0.0039	0.8307	0.0051
WineEW	**0.9619**	0.0203	0.9546	0.0243
clean1	0.8567	0.0156	**0.8631**	0.0112
semeion	**0.9705**	0.0035	0.9704	0.0039
Colon	0.7805	0.0301	**0.7938**	0.0433
Leukemia	0.8994	0.0412	**0.9156**	0.0366
TOX-171	0.7516	0.0343	**0.7625**	0.0324

Table 8
Number of features for BSSA-RWN versus TVBSSA-RWN.

Dataset	BSSA-RWN		TVBSSA-RWN	
	Avg	Std	Avg	Std
BreastEW	**16.3600**	0.7403	16.7600	1.7401
Exactly	12.8800	0.3271	**12.8200**	0.2864
Exactly2	**13.9200**	0.6419	14.5400	0.8849
HeartEW	10.8800	1.3773	**10.5200**	0.9365
Lymphography	13.3800	1.8580	**13.1000**	1.1336
M-of-n	12.8000	0.6519	**12.6400**	0.5320
penglungEW	124.0200	12.6458	**107.7200**	15.7311
SonarEW	40.9800	2.0705	**39.4800**	1.9202
SpectEW	16.3000	1.7776	**15.9800**	0.4764
CongressEW	11.0800	1.2071	**10.6400**	1.5662
KrvskpEW	**34.1600**	1.0455	34.8000	1.6248
Tic-tac-toe	**14.6800**	0.2490	14.8200	0.4324
Vote	10.7000	1.0173	**10.0400**	1.6009
WaveformEW	**32.5800**	0.9149	33.5600	1.1781
WineEW	7.6200	0.3701	**7.0800**	0.4658
clean1	**115.4200**	3.3214	127.5200	3.9442
semeion	162.1800	8.5377	**160.5800**	2.8648
Colon	**1188.9000**	87.6461	1218.5000	35.8552
Leukemia	**4022.1600**	211.6794	4171.2600	405.7624
TOX-171	3392.6400	142.5020	**3353.4400**	124.2197

and exploitative trends in PSO-KNN and GA-KNN for majority of datasets such as BreastEW, HeartEW, Lymphography, penglungEW, SonarEW, SpectEW, CongressEW, and TOX-171, while the proposed TVBSSA-RWN shows an stable and superior performance for majority of cases. The convergence curves of TVBSSA-RWN and BSSA-RWN are very competitive and both superior to other peers.

As mentioned in the previous sections, one the main features of the proposed TVBSSA-RWN is that it automatically tunes the number of neurons in the hidden layer of its RWN. This feature makes the algorithm needs little human intervention/involvement and saves time required for tuning this important parameter. Figs. 9 and 10 demonstrate the change of average number of neurons over the course of iterations in TVBSSA-RWN for all benchmark datasets. Although the algorithm was given a large space to search for this number, which is [1,1024], TVBSSA-RWN was capable in narrowing down this range over the course of iterations, until it reaches a number that minimizes the fitness value.

Results of Wilcoxon statistical rank-sum test are presented in Table 9. According to the p-values in this table, we can see that the difference between the accuracy rates of the proposed TVBSSA-RWN versus other peers are significantly meaningful in the majority of cases.

5.5. Comparison with other meta-heuristics in literature

In this section, the classification accuracy rates obtained by the proposed TVBSSA combined with RWN is compared with those reported in previous literature. The results are obtained from three recognized papers, which utilized same datasets for comparison purposes. In Table 10, the efficacy of TVBSSA in terms of classification accuracy is compared with the performance of Genetic Algorithm (GA1) and Particle Swarm Optimization (PSO1) in Kashef and Nezamabadi-pour (2015) performed based on the source code developed by the same researchers. In addition, the accuracy rates are compared with rates of the binary Grey Wolf Optimization - approach 1 (bGWO1), binary Grey Wolf Optimization - approach 2 (bGWO2), GA2 and PSO2 from the work in

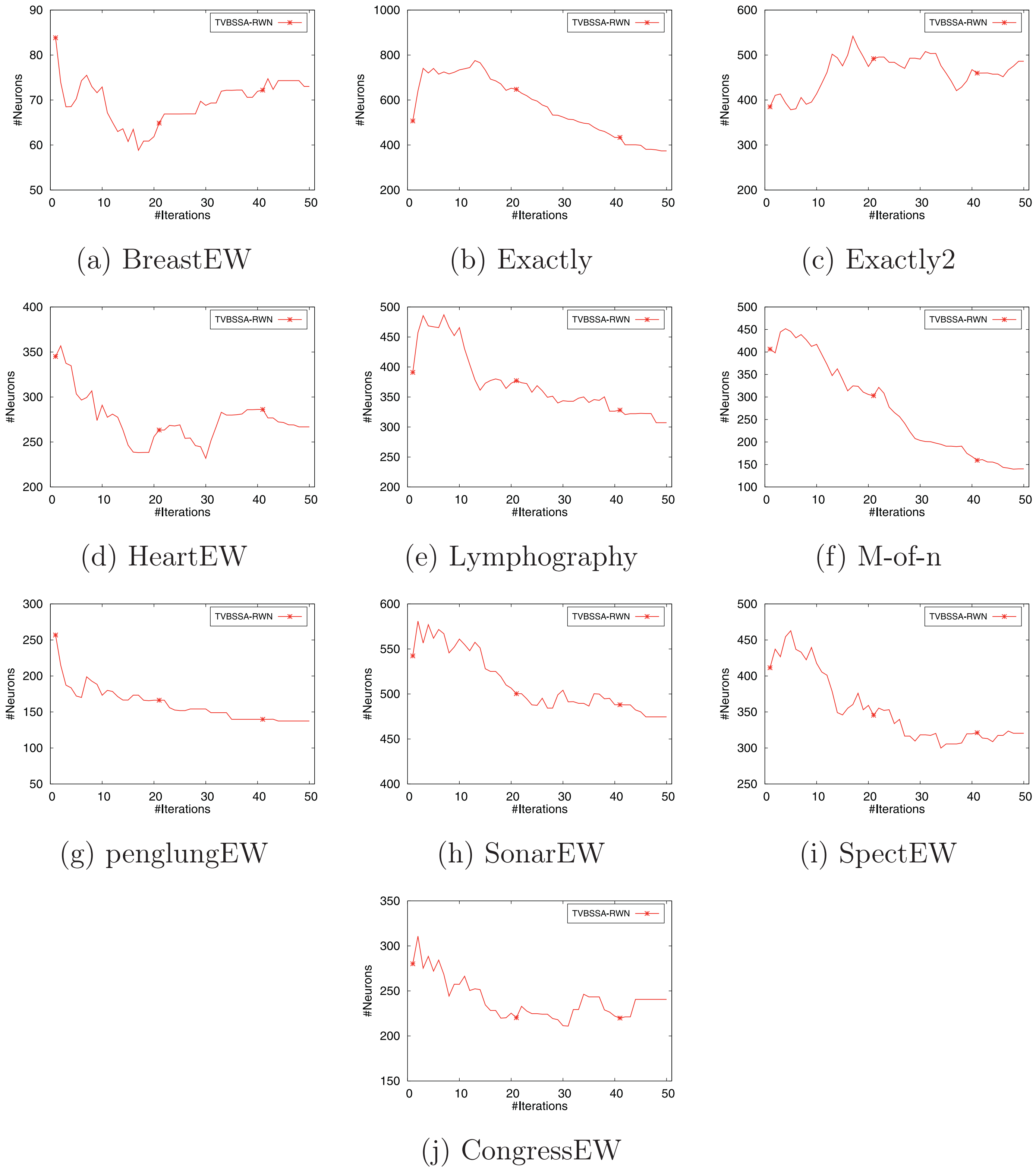


Fig. 9. Average number of neurons over the course of iterations in TVBSSA-RWN.

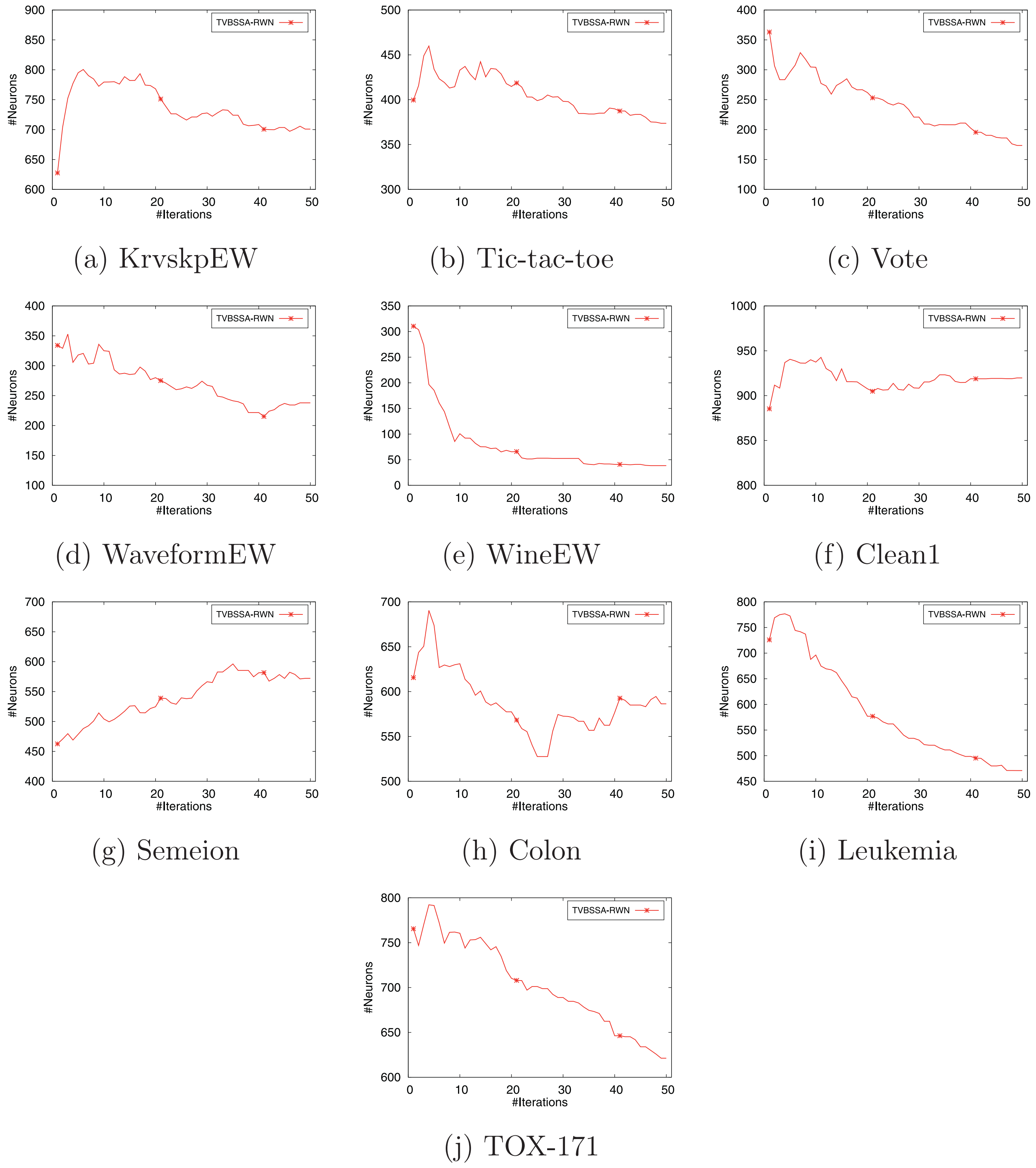


Fig. 10. Average number of neurons over the course of iterations in TVBSSA-RWN.

Table 9
P-values of the Wilcoxon test for the classification accuracy results of the TVBSSA-RWN versus other approaches ($p \geq 0.05$ are underlined).

Dataset	BSSA-kNN	GA-kNN	PSO-kNN
BreastEW	1.32E−5	1.15E−7	2.92E−6
Exactly	1.59E−5	4.04E−3	1.75E−6
Exactly2	2.48E−34	8.80E−36	1.90E−34
HeartEW	3.05E−7	2.17E−9	1.49E−9
Lymphography	2.08E−8	2.73E−6	3.60E−11
M-of-n	6.09E−4	1.68E−7	2.64E−3
penglungEW	9.86E−6	1.05E−8	4.00E−11
SonarEW	9.97E−1	8.72E−1	7.77E−1
SpectEW	2.24E−14	1.21E−12	2.13E−15
CongressEW	1.35E−5	4.72E−6	4.60E−6
KrvskpEW	3.71E−15	5.24E−7	6.96E−18
Tic-tac-toe	1.92E−34	1.92E−34	1.88E−34
Vote	4.14E−7	2.07E−9	3.97E−6
WaveformEW	1.68E−30	3.07E−23	7.02E−29
WineEW	9.52E−3	8.65E−3	1.32E−2
Clean1	2.90E−1	3.16E−1	1.93E−1
semeion	8.76E−1	2.77E−1	6.04E−1
Colon	2.87E−1	5.86E−3	4.65E−2
Leukemia	2.41E−3	7.28E−2	3.45E−2
TOX-171	8.07E−1	6.25E−1	4.17E−2

Emary et al. (2016a), Binary Butterfly Optimization Algorithm (S-bBOA) (Arora & Anand, 2019), Binary Gravitational Search Algorithm (BGSA), and Binary Bat Algorithm (BBA) (Mafarja et al., 2018b).

As can be seen in Table 10, the proposed TVBSSA-RWN can reach to highest accuracy rates compared to previous methods on around 78% of datasets. The proposed TVBSSA-RWN have shown a significant superiority over GWO, GA, and PSO algorithms on several datasets. The accuracy rates of the TVBSSA-RWN are better than the rates of GA, binary PSO, and binary GWO in Emary et al. (2016b) on all datasets. It also provides superior results compared to GA1-based FS method in Kashef and Nezamabadi-pour (2015) for 92% of datasets, while it outperforms the PSO1-based technique for 85% of datasets. These results also indicate the significant role of time-varying hierarchy and merits of RWN instead of K-NN in enriching the exploitation and exploitation inclinations and alleviating the stagnation drawbacks of the BSSA. The comparative study with previous works show that the proposed mechanisms have enhanced the quality of the resulted feature sets in dealing with different cases.

All in all, we can summarize the advantages of the proposed TVBSSA-RWN algorithm upon the well-known wrapper based feature selection algorithms as follows: first, it utilizes the model of salp chains in SSA for the search process in the wrapper which proved to be very efficient different complex optimization problems. Second, and unlike most of the well-known feature selection algorithms, TVBSSA-RWN incorporates RWN as a powerful induction algorithm. The experiments reveal that the combination of time varying chain of salps and RWN showed very promising results for the feature selection tasks.

6. Conclusion and future directions

In this work, an improved binary SSA was utilized as a search strategy combined with RWN classifier as an induction algorithm to design an improved wrapper FS method (called TVBSSA). In TVBSSA, a dynamic strategy was adopted to control the number of leaders and followers and replace the static strategy in the original SSA. Three objectives were utilized in the fitness function; maximizing the classification accuracy, maximizing the reduction rates, and minimizing the complexity of RWN classifier. The extensive experiments using 14 well-regarded benchmark datasets revealed that the developed approach beat other peers from the literature in terms of prediction power. Based on the Wilcoxon statistical test, TVBSSA significantly outperformed other approaches in terms of the classification accuracy.

The main conclusion drawn from the results and findings is the merits of the proposed method in handling the challenges that an optimization algorithm faces when solving feature selection problems. The proposed method is worthy of integration in different machine learning systems to find an optimal set of features. However, it is important to note that this work comes with limitations. First, all datasets that were used in this work range from small to large datasets, however, nowadays there are many applications with much larger scale worth of investigation based on frameworks specialized for this type of datasets with extremely high-dimensional feature space. Moreover, we investigated only binary classification problems. Other types of machine learning tasks like multiclass classification problems and regression problems have additional issues at the level of design and formulation that need to be carefully addressed.

As a future direction, researchers can adopt other dynamic strategies to control other parameters in SSA, especially c_3 parameter which controls the balance between exploration and exploitation phases. Moreover, it would be very interesting to handle the problem of optimization in this work as a multi-objective problem by separating the three objectives with the use of the dominance relation to finding the best feature subsets. Also, the findings of this work motivate us to investigate a parallel implementation of the salp chains model to target high-dimensional large-scale datasets of different real-world applications.

Table 10
Classification accuracies of the TVBSSA-based approach compared to other optimizers from literature.

Dataset	TVBSSA	GA1	PSO1	bGWO1	bGWO2	GA2	PSO2	S-bBOA	BBA	BGSA
Exactly	**0.999**	0.822	0.973	0.708	0.776	0.674	0.688	0.972	0.610	0.697
Exactly2	**0.783**	0.677	0.666	0.745	0.750	0.746	0.730	0.760	0.628	0.706
HeartEW	**0.865**	0.732	0.745	0.776	0.776	0.780	0.787	0.824	0.754	0.777
Lymphography	**0.871**	0.758	0.759	0.744	0.700	0.696	0.744	0.868	0.701	0.781
M-of-n	0.992	0.916	**0.996**	0.908	0.963	0.861	0.921	0.972	0.722	0.835
penglungEW	0.758	0.672	0.879	0.600	0.584	0.584	0.584	0.878	0.795	**0.919**
SonarEW	0.818	0.833	0.804	0.731	0.729	0.754	0.737	**0.936**	0.844	0.888
SpectEW	**0.875**	0.756	0.738	0.820	0.822	0.793	0.822	0.846	0.800	0.783
CongressEW	**0.964**	0.898	0.937	0.935	0.938	0.932	0.928	0.959	0.872	0.951
KrvskpEW	**0.982**	0.940	0.949	0.944	0.956	0.920	0.941	0.966	0.816	0.908
Tic-tac-toe	**0.982**	0.764	0.750	0.728	0.727	0.719	0.735	0.798	0.665	0.753
Vote	0.961	0.808	0.888	0.912	0.920	0.904	0.904	**0.965**	0.851	0.931
WaveformEW	**0.831**	0.712	0.732	0.786	0.789	0.773	0.762	0.743	0.669	0.695
WineEW	0.955	0.947	0.937	0.930	0.920	0.937	0.933	**0.984**	0.919	0.951
W|T|L	9|0|5	0|0|14	1|0|13	0|0|14	0|0|14	0|0|14	0|0|14	3|0|11	0|0|14	1|0|13

Declaration of Competing Interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Credit authorship contribution statement

Hossam Faris: Conceptualization, Supervision, Methodology, Writing - original draft. **Ali Asghar Heidari:** Formal analysis, Writing - original draft. **Ala' M. Al-Zoubi:** Software, Investigation, Formal analysis. **Majdi Mafarja:** Writing - original draft. **Ibrahim Aljarah:** Writing - original draft. **Mohammed Eshtay:** Software, Investigation. **Seyedali Mirjalili:** Writing - original draft, Supervision.

References

Agrafiotis, D. K., & Cedeno, W. (2002). Feature selection for structure- activity correlation using binary particle swarms. *Journal of Medicinal Chemistry, 45*(5), 1098–1107.

Al-Zoubi, A. M., Faris, H., Alqatawna, J., & Hassonah, M. A. (2018). Evolving Support Vector Machines using Whale Optimization Algorithm for spam profiles detection on online social networks in different lingual contexts. *Knowledge-Based Systems, 153*, 91–104. doi:10.1016/j.knosys.2018.04.025.

Alba, E., Garcia-Nieto, J., Jourdan, L., & Talbi, E.-G. (2007). Gene selection in cancer classification using pso/svm and ga/svm hybrid algorithms. In *Evolutionary computation, 2007. CEC 2007. IEEE congress on* (pp. 284–290). IEEE.

Arora, S., & Anand, P. (2019). Binary butterfly optimization approaches for feature selection. *Expert Systems with Applications, 116*, 147–160.

Cao, W., Wang, X., Ming, Z., & Gao, J. (2018). A review on neural networks with random weights. *Neurocomputing, 275*, 278–287.

Chen, D., Chan, K. C., & Wu, X. (2008). Gene expression analyses using genetic algorithm based hybrid approaches. In *Evolutionary computation, 2008. cec 2008.(ieee world congress on computational intelligence). ieee congress on* (pp. 963–969). IEEE.

Cho, H.-W., Kim, S. B., Jeong, M. K., Park, Y., Ziegler, T. R., & Jones, D. P. (2008). Genetic algorithm-based feature selection in high-resolution NMR spectra. *Expert Systems with Applications, 35*(3), 967–975.

Chyzhyk, D., Savio, A., & Graña, M. (2014). Evolutionary elm wrapper feature selection for Alzheimer's disease cad on anatomical brain MRI. *Neurocomputing, 128*, 73–80.

Da Silva, S. F., Ribeiro, M. X., Neto, J. d. E. B., Traina-Jr, C., & Traina, A. J. (2011). Improving the ranking quality of medical image retrieval using a genetic feature selection method. *Decision Support Systems, 51*(4), 810–820.

Derrac, J., García, S., & Herrera, F. (2009). A first study on the use of coevolutionary algorithms for instance and feature selection. In *International conference on hybrid artificial intelligence systems* (pp. 557–564). Springer.

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence magazine, 1*(4), 28–39.

El-Fergany, A. A. (2018). Extracting optimal parameters of pem fuel cells using salp swarm optimizer. *Renewable Energy, 119*, 641–648.

Emary, E., Zawbaa, H. M., & Grosan, C. (2018). Experienced gray wolf optimization through reinforcement learning and neural networks. *IEEE Transactions on Neural Networks and Learning Systems, 29*(3), 681–694.

Emary, E., Zawbaa, H. M., & Hassanien, A. E. (2016a). Binary ant lion approaches for feature selection. *Neurocomputing, 213*, 54–65.

Emary, E., Zawbaa, H. M., & Hassanien, A. E. (2016b). Binary grey wolf optimization approaches for feature selection. *Neurocomputing, 172*, 371–381.

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

Gandomi, A. H., & Alavi, A. H. (2012). Krill herd: A new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation, 17*(12), 4831–4845. doi:10.1016/j.cnsns.2012.05.010.

Gu, S., Cheng, R., & Jin, Y. (2018). Feature selection for high-dimensional classification using a competitive swarm optimizer. *Soft Computing, 22*(3), 811–822.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research, 3*, 1157–1182. (Mar)

Han, Y., Yang, Y., Yan, Y., Ma, Z., Sebe, N., & Zhou, X. (2015). Semisupervised feature selection via spline regression for video semantic recognition. *IEEE Transactions on Neural Networks and Learning Systems, 26*(2), 252–264.

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.

Holland, J. H. (1992). Genetic algorithms. *Scientific American, 267*(1), 66–73.

Hong, J.-H., & Cho, S.-B. (2006). Efficient huge-scale feature selection with speciated genetic algorithm. *Pattern Recognition Letters, 27*(2), 143–150.

Huang, G.-B., Zhu, Q.-Y., & Siew, C.-K. (2004). Extreme learning machine: a new learning scheme of feedforward neural networks. In *Neural networks, 2004. proceedings. 2004 ieee international joint conference on: 2* (pp. 985–990). IEEE.

Huang, R., & He, M. (2007). Feature selection using double parallel feedforward neural networks and particle swarm optimization. In *Evolutionary computation, 2007. cec 2007. ieee congress on* (pp. 692–696). IEEE.

Igelnik, B., & Pao, Y.-H. (1995). Stochastic choice of basis functions in adaptive function approximation and the functional-link net. *IEEE Transactions on Neural Networks, 6*(6), 1320–1329.

Jain, A., & Zongker, D. (1997). Feature selection: evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 19*(2), 153–158.

Jeong, Y.-S., Shin, K. S., & Jeong, M. K. (2015). An evolutionary algorithm with the partial sequential forward floating search mutation for large-scale feature selection problems. *Journal of The Operational Research Society, 66*(4), 529–538.

Kashef, S., & Nezamabadi-pour, H. (2013). A new feature selection algorithm based on binary ant colony optimization. In *Information and knowledge technology (ikt), 2013 5th conference on* (pp. 50–54). IEEE.

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

Kennedy, J., & Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. In *Systems, man, and cybernetics, 1997. computational cybernetics and simulation., 1997 IEEE international conference on: 5* (pp. 4104–4108). IEEE.

Khushaba, R. N., Al-Ani, A., AlSukker, A., & Al-Jumaily, A. (2008). A combined ant colony and differential evolution feature selection algorithm. In *International conference on ant colony optimization and swarm intelligence* (pp. 1–12). Springer.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence, 97*(1–2), 273–324.

Li, R., Lu, J., Zhang, Y., & Zhao, T. (2010). Dynamic adaboost learning with feature selection based on parallel genetic algorithm for image annotation. *Knowledge-Based Systems, 23*(3), 195–201.

Lichman, M. (2013). UCI machine learning repository. URL http://archive.ics.uci.edu/ml.

Liu, H., & Motoda, H. (2012). *Feature selection for knowledge discovery and data mining*: 454. Springer Science & Business Media.

Liu, H., & Setiono, R. (1995). Chi2: Feature selection and discretization of numeric attributes.

Mafarja, M., Aljarah, I., Heidari, A. A., Hammouri, A. I., Faris, H., Al-Zoubi, A. M., et al. (2018a). Evolutionary population dynamics and grasshopper optimization approaches for feature selection problems. *Knowledge-Based Systems, 145*, 25–45. doi:10.1016/j.knosys.2017.12.037.

Mafarja, M., Aljarah, I., Heidari, A. A., Hammouri, A. I., Faris, H., Ala'M, A.-Z., et al. (2018b). Evolutionary population dynamics and grasshopper optimization approaches for feature selection problems. *Knowledge-Based Systems, 145*, 25–45.

Mafarja, M., Jarrar, R., Ahmad, S., & Abusnaina, A. (2018c). Feature selection using binary particle swarm optimization with time varying inertia weight strategies. *The 2nd international conference on future networks & distributed systems, Amman, Jordan*: 2. ACM.

Mafarja, M., & Mirjalili, S. (2018). Whale optimization approaches for wrapper feature selection. *Applied Soft Computing, 62*, 441–453.

Mafarja, M., & Sabar, N. R. (2018). Rank based binary particle swarm optimisation for feature selection in classification. *The 2nd international conference on future networks & distributed systems, amman, jordan*: 2. ACM.

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

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. doi:10.1007/s00521-015-1920-1.

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.

Mirjalili, S., & Lewis, A. (2013). S-shaped versus v-shaped transfer functions for binary particle swarm optimization. *Swarm and Evolutionary Computation, 9*, 1–14.

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software, 95*, 51–67.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46–61.

Nakamura, R. Y. M., Pereira, L. A. M., Rodrigues, D., Costa, K. A. P., Papa, J. P., & Yang, X.-S. (2013). Binary bat algorithm for feature selection. In *Swarm intelligence and bio-inspired computation* (pp. 225–237). Elsevier.

Nie, F., Huang, H., Cai, X., & Ding, C. H. (2010). Efficient and robust feature selection via joint l2, 1-norms minimization. In *Advances in neural information processing systems* (pp. 1813–1821).

Nie, F., Xiang, S., Jia, Y., Zhang, C., & Yan, S. (2008). Trace ratio criterion for feature selection. In *AAAI: 2* (pp. 671–676).

Nie, F., Zhu, W., & Li, X. (2016). Unsupervised feature selection with structured graph optimization. In *Thirtieth AAAI conference on artificial intelligence* (pp. 1302–1308).

Oh, I.-S., Lee, J.-S., & Moon, B.-R. (2004). Hybrid genetic algorithms for feature selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 26*(11), 1424–1437.

Oreski, S., & Oreski, G. (2014). Genetic algorithm-based heuristic for feature selection in credit risk assessment. *Expert Systems with Applications, 41*(4), 2052–2064.

Pacifico, L. D., & Ludermir, T. B. (2013). Evolutionary extreme learning machine based on particle swarm optimization and clustering strategies. In *Neural networks (ijcnn), the 2013 international joint conference on* (pp. 1–6). IEEE.

Pao, Y.-H., Park, G.-H., & Sobajic, D. J. (1994). Learning and generalization characteristics of the random vector functional-link net. *Neurocomputing, 6*(2), 163–180.

Quinlan, J. (1986). Induction of decision trees. *Machine Learning, 1*(1), 81–106.

Quinlan, J. (1993). *C4. 5: programs for machine learning*. Morgan Kaufmann.

Ramos, C. C., Souza, A. N., Chiachia, G., Falcão, A. X., & Papa, J. P. (2011). A novel algorithm for feature selection using harmony search and its application for non-technical losses detection. *Computers & Electrical Engineering, 37*(6), 886–894.

Rashedi, E., & Nezamabadi-pour, H. (2014). Feature subset selection using improved binary gravitational search algorithm. *Journal of Intelligent & Fuzzy Systems, 26*(3), 1211–1221.

Robnik-Šikonja, M., & Kononenko, I. (2003). Theoretical and empirical analysis of relieff and rrelieff. *Machine learning, 53*(1), 23–69.

Rodrigues, D., Pereira, L. A., Almeida, T., Papa, J. P., Souza, A., Ramos, C. C., & Yang, X.-S. (2013). Bcs: A binary cuckoo search algorithm for feature selection. In *Circuits and systems (iscas), 2013 IEEE international symposium on* (pp. 465–468). IEEE.

Sayed, G. I., Khoriba, G., & Haggag, M. H. (2018). A novel chaotic salp swarm algorithm for global optimization and feature selection. *Applied Intelligence*, 1–20.

Schmidt, W. F., Kraaijveld, M. A., & Duin, R. P. (1992). Feedforward neural networks with random weights. In *Pattern recognition, 1992. vol. ii. conference b: Pattern recognition methodology and systems, proceedings., 11th iapr international conference on* (pp. 1–4). IEEE.

Seo, J.-H., Lee, Y. H., & Kim, Y.-H. (2014). Feature selection for very short-term heavy rainfall prediction using evolutionary computation. *Advances in Meteorology, 2014*.

Shoorehdeli, M. A., Teshnehlab, M., & Moghaddam, H. A. (2006). Feature subset selection for face detection using genetic algorithms and particle swarm optimization. In *Networking, sensing and control, 2006. icnsc'06. proceedings of the 2006 ieee international conference on* (pp. 686–690). IEEE.

Shrivastava, P., Shukla, A., Vepakomma, P., Bhansali, N., & Verma, K. (2017). A survey of nature-inspired algorithms for feature selection to identify parkinson's disease. *Computer Methods and Programs in Biomedicine, 139*, 171–179.

Shunmugapriya, P., & Kanmani, S. (2017). A hybrid algorithm using ant and bee colony optimization for feature selection and classification (ac-abc hybrid). *Swarm and Evolutionary Computation, 36*, 27–36.

Souza, F., Matias, T., & Araójo, R. (2011). Co-evolutionary genetic multilayer perceptron for feature selection and model design. In *Emerging technologies & factory automation (etfa), 2011 ieee 16th conference on* (pp. 1–7). IEEE.

Talbi, E.-G. (2009). *Metaheuristics: from design to implementation*: 74. John Wiley & Sons.

Tan, F., Fu, X., Zhang, Y., & Bourgeois, A. G. (2008). A genetic algorithm-based method for feature subset selection. *Soft Computing, 12*(2), 111–120.

Tang, E. K., Suganthan, P. N., & Yao, X. (2005). Feature selection for microarray data using least squares svm and particle swarm optimization. In *Computational intelligence in bioinformatics and computational biology, 2005. cibcb'05. proceedings of the 2005 ieee symposium on* (pp. 1–8). IEEE.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 267–288.

Unler, A., Murat, A., & Chinnam, R. B. (2011). mr2pso: A maximum relevance minimum redundancy feature selection method based on swarm intelligence for support vector machine classification. *Information Sciences, 181*(20), 4625–4641.

Winkler, S. M., Affenzeller, M., Jacak, W., & Stekel, H. (2011). Identification of cancer diagnosis estimation models using evolutionary algorithms: A case study for breast cancer, melanoma, and cancer in the respiratory system. In *Proceedings of the 13th annual conference companion on genetic and evolutionary computation* (pp. 503–510). ACM.

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

Xue, B., Zhang, M., & Browne, W. N. (2012). New fitness functions in binary particle swarm optimisation for feature selection. In *Evolutionary computation (cec), 2012 ieee congress on* (pp. 1–8). IEEE.

Xue, B., Zhang, M., & Browne, W. N. (2013). Novel initialisation and updating mechanisms in pso for feature selection in classification. In *European conference on the applications of evolutionary computation* (pp. 428–438). Springer.

Yan, Z., & Yuan, C. (2004). Ant colony optimization for feature selection in face recognition. In *Biometric authentication* (pp. 221–226). Springer.

Yang, H., Yi, J., Zhao, J., & Dong, Z. (2013). Extreme learning machine based genetic algorithm and its application in power system economic dispatch. *Neurocomputing, 102*, 154–162.

Yang, J., & Honavar, V. (1998). Feature subset selection using a genetic algorithm. In *Feature extraction, construction and selection* (pp. 117–136). Springer.

Yang, W.-A., Zhou, Q., & Tsui, K.-L. (2016). Differential evolution-based feature selection and parameter optimisation for extreme learning machine in tool wear estimation. *International Journal of Production Research, 54*(15), 4703–4721.

Yang, X.-S. (2009). Firefly algorithms for multimodal optimization. In *SAGA 2009* (pp. 169–178)). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-04944-6_14.

Yusta, S. C. (2009). Different metaheuristic strategies to solve the feature selection problem. *Pattern Recognition Letters, 30*(5), 525–534.

Zainuddin, Z., Lai, K. H., & Ong, P. (2016). An enhanced harmony search based algorithm for feature selection: Applications in epileptic seizure detection and prediction. *Computers & Electrical Engineering, 53*, 143–162.

Zhao, Z., Morstatter, F., Sharma, S., Alelyani, S., Anand, A., & Liu, H. (2010). Advancing feature selection research. *ASU Feature Selection Repository*, 1–28.