Webology, Volume 18, Special Issue on Computing Technology and Information Management, September, 2021

Feature Selection Method based on Chaotic Salp Swarm Algorithm and Extreme Learning Machine for Network Intrusion Detection Systems

Rana Nazhan Hadi*
Computer Science, Faculty of Computer & Artificial Inteligence, Benha University, Cairo, Egypt.
E-mail: 11112nazhan@gmail.com

Dr. Rasha Orban Mahmoud
Computer Science, Faculty of Computer & Artificial Inteligence, Benha University, Cairo, Egypt.

Dr. Adly S. Tag Eldien
Faculty of Engineering (Shoubra), Benha University, Cairo, Egypt.

Received March 21, 2021; Accepted July 04, 2021
ISSN: 1735-188X
DOI: 10.14704/WEB/V18SI04/WEB18154

Abstract

Network Intrusion Detection Systems (IDSs) have been widely used to monitor and manage network connections and prevent unauthorized connections. Machine learning models have been utilized to classify the connections into normal connections or attack connections based on the users' behavior. One of the most common issues facing the IDSs is the detection system's low classification accuracy and high dimensionality in the feature selection process. However, the feature selection methods are usually used to decrease the datasets' redundancy and enhance the classification performance. In this paper, a Chaotic Salp Swarm Algorithm (CSSA) was integrated with the Extreme Learning Machine (ELM) classifier to select the most relevant subset of features and decrease the dimensionality of a dataset. Each Salp in the population was represented in a binary form, where 1 represented a selected feature, while 0 represented a removed feature. The proposed feature selection algorithm was evaluated based on NSL-KDD dataset, which consists of 41 features. The results were compared with others and have shown that the proposed algorithm succeeded in achieving classification accuracy up to 97.814% and minimized the number of selected features.

Keywords

Feature Selection, Salp Swarm Algorithm, Intrusion Detection System, Extreme Learning Machine.

Introduction

An intrusion detection system (IDS) is a security tool for the efficient security of information and communication systems. The IDS mainly focuses on the detection of traffics that is dangerous to a network. An IDS has the same functionality as other security processes such as firewalls and antivirus software and can also access the control schemes. The IDS classification is based on their detection limit, as are classified as signature and anomaly detection systems. The signature detection IDS can identify the pattern of traffic or application data as dangerous and will require its database to be updated to store the signature of the determined attack. The anomaly detection IDS detects anomalies by comparing all activities against an established defined behavior (Agrawal, 2015; Balamurugan, 2017; Chandola, 2012).

Several machine learning methods have been employed to improve the attack detection limit and the detection accuracy of IDS systems; they have also been used to develop useful classification and clustering models that can distinguish normal behavior from an attack. The process of detecting the intrusion accurately of an IDS system from the total network traffic has always been a classification problem (Bostani, 2015). There may be some irrelevant features of the training data that do not contribute to detection, and these outside features, in most cases, maybe redundant or introduce noise into the classifier's design. Therefore, it is necessary to choose data with useful features, which will improve the classifiers' performance (Xue, 2014). Generally, the feature selection problem is NP-Hard optimization problem. An efficient optimization algorithm is required to select the minimum relevant subset of features and enhance the classification performance (Taha, 2015; Zhang, 2016).

During the last two decades, numerous research papers were published in the field of optimization algorithms. Most of these research papers have suggested tens of metaheuristics inspired by a living organism's natural behavior, such as birds, bees, ants, fireflies, or even bats (Abba, 2020; Eberhart, 1995; Karaboga, 2005; Mirjalili, 2014; Salih, 2018; Yang, 2009, 2010). On the other hands, other researchers have suggested metaheuristics inspired by social behavior of living groups, such as Teaching–Learning-based Optimization (TLBO), Socio Evolution & Learning Optimization Algorithm (SELO), Cultural Evolution Algorithm (CEA), Artificial Memory Optimization (AMO), and Nomadic People Optimizer (NPO) (G. qiu Huang, 2017; Kumar, 2018; Kuo, 2013; Rao, 2011; Salih, 2020). These algorithms have been used recently for the different research areas, such as engineering, machine learning, information security, and data clustering (Afan, 2020; Alzaidi, 2018; Malik, 2020; Salih, 2019, 2019; Tao, 2020; Yaseen, 2020). The main contribution of this study is to design a wrapper feature selection method based on a recently developed nature-inspired algorithm, which is Salp Swarm Algorithm (SSA) (Mirjalili, 2017) and Extreme Learning Machine (ELM). The rest of the paper is structured as follows: Section II presents an overview of the ELM and SSA, while Section III explains the proposed algorithm in detail. Section IV presents the obtained results. Finally, Section V concludes the findings of this study.

Overview

Extreme Learning Machine (ELM)

Huang et al. (G.B. Huang, 2006) first developed ELM as a single hidden layer feed-forward network where the input weights' selection is made randomly. In contrast, the output weights are estimated analytically. For the hidden neuron layer of ELM, different activation functions are applicable, such as sine, gaussian, sigmoid, and hard limiting functions. For the output neurons, the linear activation function can be employed [25]. Several essential ELM features differed from the conventional popular gradient-based learning frameworks for Feed forward Neural Networks (FFNNs); these include faster learning speed, ability to escape local minima, and good generalization capability.

Consider the ELM network with N number of samples $\{P_i, Q_i\}$, where N represents the n-dimensional feature of sample i while $Q_i = [q_{i1}, \ldots, q_{ic}] \in R^c$ represents its coded class label. Assume that P_i is allocated the class label c_k, then, the kth element of Q_i will be 1 (q_{ik} =1) while the other elements will be -1. For the ELM with C distinct classes and H hidden neurons, the output (Qˆ) is defined thus(G.-B. Huang et al., 2006):

$$\hat{q}_{ik} = \sum_{j=1}^{H} U_{kj} G_j(W, B, P_i), k = 1,2, \ldots, C \qquad (1)$$

Where W is the $H \times n$ input weights, B is the $H \times 1$ bias of the hidden neurons, while U is the $C \times H$ output weights. The j^{th} hidden neuron's output, $G_j(\,)$, is defined thus:

$$G_j = G\left(\sum_{k=1}^{n} W_{jk} p_{ik} + b_j\right), j = 1, 2, \ldots, H \qquad (2)$$

Where $G(\,)$ represents the activation function. When using radial basis function, $G_j(\,)$ is defined thus:

$$G_j = G\left(b_j||P - W||\right), j = 1, 2, \dots, H \qquad (3)$$

Where W and b_j are the centre and width of the radial basis function neuron.

In the case of the sigmoidal activation function, the output of the jth neuron $G_j(\,)$ is defined as follows:

$$G_j = G\left(b_j||P - W||\right), j = 1, 2, \dots, H \qquad (4)$$

The matrix form of Equation (4) can be as follows:

$$\hat{Q} = UQ_H \qquad (5)$$

Where the output of the hidden layer = Q_H, given as:

$$\begin{aligned} &Q_H(W, B, P) \\ &= \begin{bmatrix} G_1(W, b_1, P_1)\; G_1(W, b_1, P_1) \dots G_1(W, b_1, P_1) \\ \cdot \qquad\qquad\qquad \cdot \\ G_H(W, b_H, P_1)\; G_H(W, b_H, P_1) \dots G_H(W, b_1, P_1) \end{bmatrix} \end{aligned} \qquad (6)$$

The selection of the bias B and input weights W of the hidden neurons is done randomly in ELM. Assume that the predicted output, $\hat{Q}$ is equivalent to the coded labels Q, then, the output weights can be analytically estimated thus:

$$\hat{U} = QQ_H^{+} \qquad (7)$$

where Q_H^{+} represents the Moor-Penrose generalized pseudo inverse of the hidden layer output matrix. In general, ELM involves the following steps:

1. Select the number of hidden neurons and a proper activation function that suit the considered problem.
2. Select Randomly the input weight (W) & bias (B) parameters.
3. Calculate the output weight (U) analytically.
4. Estimate the class label using the calculated weights (W, U, B).

The features extracted from the S-Matrix are served as the input to the ELM. At the same time, the integer value of the class label is the output.

Salp Swarm Algorithm (SSA)

Salps are cylindrical jellyfishes-like creatures that belong to the Salpidae. As shown in Figure 1, they are shaped and moves by pushing water backward to move forward. Salps' biology is still being investigated as they usually live in environments that cannot be easily reached and cannot survive in laboratory environments. Salps' swarming attitude inspired the development of the Salp swarm algorithm (SSA). They form a swarm called Salp chain (shown in Figure 1) in oceans that helps their movement. The formulation of the Salp chain's mathematical model was achieved by dividing the Salps population into two groups - head and followers, where the head is the leading part of the chain. At the same time, the rest is the followers (Mirjalili et al., 2017).

Salps' location in SSA is determined as in other swarm-based techniques; an n-dimensional search space does this by considering the number of parameters (n) in the supposed problem. Therefore, the salps position will be mapped in a 2-D matrix represented as x. Assume a solution space where “F”, the food source, is target of the swarm. Then, the position of the can be upgraded using the following relation (Mirjalili et al., 2017):

$$x_j^1 = \begin{cases} F_j + c_1\left(\left(\mathrm{ub}_j - \mathrm{lb}_j\right)c_2 + \mathrm{lb}_j\right), & c_3 \geq 0, \\ F_j - c_1\left(\left(\mathrm{ub}_j - \mathrm{lb}_j\right)c_2 + \mathrm{lb}_j\right), & c_3 < 0, \end{cases} \tag{8}$$

Where x_j^1 = the position of the leader (the 1st Salp) in the j^{th} dimension, F_j = the location of the food source in the j^{th} dimension, ub_j and lb_j are the upper and lower boundaries of j^{th} dimension, respectively, c_1, c_2, and c_3 = random parameters.

The food source is considered when updating the position of the leader as captured in Eq (8). c_1 is the most significant SSA parameter as it strikes the balance between exploration and exploitation as follows (Mirjalili et al., 2017):

$$c_1 = 2e^{-\left(\frac{rl}{L}\right)^2} \tag{9}$$

Where l = the current iteration; L = the maximum number of iterations; c_2 and c_3 are uniformly generated random numbers in the range of [0,1]; both parameters dictate the next position's alignment in dimension also determine the step size.

The Newtons law of motion is the basis for updating the position of the followers (Mirjalili et al., 2017):

$$x_j^i = \frac{1}{2}at^2 + v_0 t \qquad (10)$$

Where $i \geq 2, x_j^i$ is the position of i^{th} follower Salp in j^{th} dimension, v_0 represents the initial speed, t is time, and $a = \frac{v_{fina}}{v_0}$, where $v = x - x_0/t$.

Time is an iteration in optimization processes; hence, the difference between iterations = 1 and if $v_0 = 0$, this equation can be stated thus (Mirjalili et al., 2017):

$$x_j^i = \frac{1}{2}(x_j^i + x_j^{i-1}) \qquad (11)$$

Where $i \geq 2$ and x_j^i represent the position of i^{th} follower Salp in the j^{th} dimension.

Eqs. (8) and (11) are employed for the simulation of the Salp chain. The shape of a single salp and the chain of salps are given in figure 1.

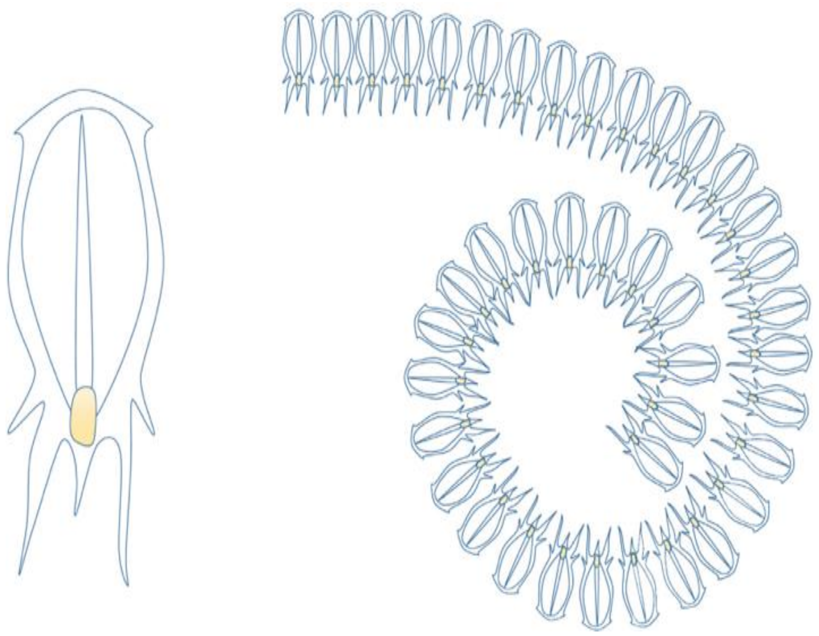

Figure 1. The general structure of Salps

Methodology

As stated previously, this study's main contribution is to design a feature selection method based on SSA. The ELM classifier is utilized for evaluating the Salps/Solutions in the population in terms of the classification accuracy. In this section, the proposed algorithm is explained in detail.

The Proposed Model

The main stages are presented in figure 2; they are given as follows:

Stage 1: Data Preprocessing

The first stage of the proposed algorithm represents the preparation of the dataset. It means reading and pre-processing the dataset using three simple steps, as follows:

- Step 1. Read the Dataset
- Step 2. Convert the Dataset from its original format (i.e., excel format '.xlsx') into a "comma separated value '.csv', which can be easily read by almost any modern programming language.
- Step 3. Normalize the dataset in a fixed range [0,1] using $MinMax$ method, which is formulated as follows

$$N_v = \frac{X_v - Min}{Max - Min} \qquad (12)$$

Where N_v represents the normalized value, while X_v represents the original value. Min and Max denote the maximum and minimum values of a specific feature respectively.

Stage 2: The Inputs

In this stage, the algorithmic parameters such as chaotic initialization values, the population's size, the maximum number of iterations, and other ELM controlling variables are entered.

Stage 3: Implementation

In this stage, the proposed algorithm is executed to select the most relevant minimum number of features, enhancing the classification performance of ELM. The main steps of the proposed algorithm are given in the next subsection.

Stage 4: Evaluation

The best solution obtained using the proposed feature selection algorithm is evaluated in terms of classification accuracy and error rate in this stage.

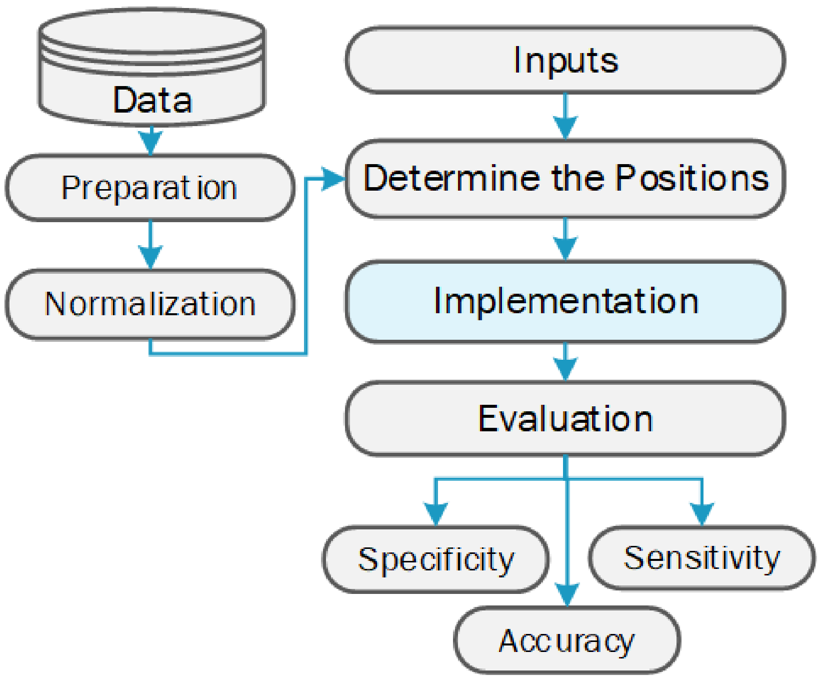

Figure 2 The block diagram of the proposed model

The Proposed Algorithm

The proposed feature selection algorithm in this study is a wrapper method. The Salp Swarm Algorithm (SSA) is used in this research for identifying the relevant subset of features. The proposed algorithm is called Binary Chaotic Salp Swarm Algorithm “BCSSA”. The main steps of BCSSA are given as follows:

Step1: Initialization

Each Salp/Solution in the population is initialized randomly using Chaotic Tent Map. The main reason behind using a chaotic map instead of the uniform distribution is that the uniform distribution may generate the initial position of the solutions in almost a similar position, which may slow down the algorithm's searching process if these solutions are developed in a wrong or bad positions. Chaotic Tent Map is given as follows:

$$X_{i+1} = \begin{cases} \frac{x_i}{0.7} & if\ x_i < 0.7 \\ \frac{10}{3}(1 - x_i) & otherwise \end{cases} \qquad (13)$$

Where X_i= a real value in range 0 and 1, representing a single dimension of any given problem.

Step2: Convert To Binary

In this step, each solution in the population is converted into a binary form, where 0 represents a removed or unselected feature, while 1 represents a selected or remain feature. The conversion process is done as follows:

$$B_i = \begin{cases} 1\,, sigmoid\,(X_i) > \, u\,[0\,,1] \\ 0, \qquad\qquad\qquad otherwise \end{cases} \qquad (14)$$

Where: X_i = position of each particle, $sigmoid(X_i) = 1\,/\,[1 + e^{-X}]$, u = uniform distribution, B_i = binary sequence, 1 = the chances of selecting a feature while 0 = chances of not selecting a feature.

Step3: Objective Function

Evaluate each generated solution via the following objective function:

$$\min f(B_i) = a * ELM(D) + (1-a) * \frac{\#F}{\#All\ Features} \qquad (15)$$

Where D represents the dataset, ELM represents the classification accuracy calculated via the ELM classifier, and a represents a real value in range[0,1]. While $\#F$ and $\#All\ Features$ represent the number of selected and the original features in the dataset.

Step 4: Ranking

In this step, all solutions are sorted discerningly, where the first solution represents the leader, while the rest represent the followers.

Step 5: Position Updating

The real values of each Salps position are updated via equations (8) and (11). Then, the updated values should be checked in terms of the upper and lower boundaries. Then, each solution should be re-converted into binary via step 2, and evaluated via Step 3.

Step 6: The Stop Condition

The positions for each solution in the population are updated, representing a fixed number of iterations. If the number of iteration is still below $MaxItr$, then go to step 5, otherwise, return the final leader.

Results and Discussion

In developing an IDS that utilizes a machine learning technique, an important aspect is designing appropriate features that will aid the developed system's activities in distinguishing normal behaviors from the system or network attacks (Lin, 2012). Although several features have been suggested, their drawback has been the unavailability of public data sets, making it difficult to objectively compare and evaluate the proposed features' performance and delay the systematic efforts on their effects on the developed IDSs. To solve these problems, MIT Lincoln Laboratory (Lippmann, 2000) provided the 1999 KDDCUP data set while Tavallaee et al. (Tavallaee, 2009) provided its modified version (the NSL_KDD dataset). Several studies were made based on these data to evaluate the performance of their proposed IDS systems objectively. The NSL-KDD dataset consists of 41 features, where the number of samples is equal to 125000.

In this section, the final stage which is the evaluation of the proposed model is explained. In order to test the performance of the proposed algorithm, the NDS-KDD dataset is considered for this purpose. The main code of BCSSA and ELM classifier was developed using MATLAB programming language version 2018a, implemented in the environment with the following specification: Windows 10 with 64bit architecture, CPU Intel 2.4GH, and RAM 8GB. Each experiment has been implemented for 10 run times. Table 1 below presents the parametric settings used for implementing BCSSA method and ELM classifier.

Table 1 Parameterc Settings for BCSSA and ELM

Parameter	**Value**
BCSSA : Population Size ($P.S$)	50
BCSSA : Iterations ($MaxGent$)	25, 50, 100
BCSSA : Chaotic Initial Value (X_0)	Rand(0,1)
ELM : Number of Neurons in hidden layer	$2 * N + 1 = 83$
ELM : Training Ratio	0.66

The results in details for all experiments are presented in the following tables. These tables presents the results in terms of the error rate, the classification accuracy, number of selected features, and time required for executing the algorithm (in minutes). The classification accuracy could be calculated via the following equation:

$$\text{Accuracy} = \frac{(TP + TN)}{TP + FP + TN + FN} \qquad (16)$$

Where TP, FP, TN, and FN represent the True Positive, False Positive, True Negative, and False Negative. These parameters are calculated based on the confusion matrix. While the error rate could be calculated as follows:

$$\text{Error} = \frac{(FP + FN)}{TP + FP + TN + FN} \qquad (16)$$

The obtained results showed an improved performance in terms of classification accuracy as compared to the original accuracy based on all features, i.e., when the dataset with all features were used as inputs to ELM. Moreover, the number of selected features was also presented in the tables. The number of features chosen was in the range [20,30], meaning that the worst results showed an enhancement in the detection system's performance.

Table 2 The Results when Swarm Size = 50 and Itrations = 25

Run	Error	F	Acc.	Time
1-	0.035794	28	0.97074	8
2-	0.034741	27	0.97156	8
3-	0.035199	23	0.97011	8
4-	0.031523	23	0.97382	7
5-	0.033085	24	0.97249	8
6-	0.034012	26	0.97205	8
7-	0.033315	25	0.97217	7
8-	0.036485	26	0.96955	7
9-	0.033865	29	0.97294	7
10-	0.033136	28	0.97343	8

Table 3 The Results when Swarm Size = 50 and Iterations = 50

Run	Error	F	Acc	Time
1-	0.030025	27	0.97632	16
2-	0.032024	26	0.97406	16
3-	0.030542	24	0.97506	15
4-	0.030472	24	0.97513	15
5-	0.029219	29	0.97763	14
6-	0.029062	24	0.97656	15
7-	0.029596	26	0.97651	15
8-	0.036905	27	0.97544	15
9-	0.032719	28	0.97305	15
10-	0.031698	24	0.97389	16

Table 4 The Results when Swarm Size = 50 and Iterations = 100

Run	Error	F	Acc	Time
1-	0.033462	29	0.97137	22
2-	0.030575	23	0.97478	22
3-	0.028484	24	0.97714	23
4-	0.02871	21	0.97814	23
5-	0.0283	24	0.97733	22
6-	0.029258	23	0.97611	23
7-	0.028726	23	0.97665	22
8-	0.029609	27	0.97674	23
9-	0.028307	30	0.9788	22
10-	0.030385	28	0.97621	23

To benchmark the performance of the proposed algorithm against Naïve Bayesian Classifier (NBC), Extreme Learning Machine (ELM), Binary Bat Algorithm (BBA), Binary Particle Swarm Optimization (BPSO), Binary Firefly Algorithm (BFA), and the proposed Binary Salp Swarm Algorithm (BSSA). Table V shows the result of the comparison.

Table 5 The Results Comparison

Algorithm	Acc. Rate	Err. Rate	No. Features
NBC	89.5%	10.1%	41 (ALL)
ELM	93.421%	6.579%	41 (ALL)
BBA (Najeeb, 2018)	91.62%	8.38%	15
BPSO (Najeeb et al., 2018)	90.63%	9.37%	22
BFA (Najeeb et al., 2018)	92.02%	7.98%	14
BSSA	**97.814%**	**2.871%**	**21**

BSSA has obtained the best accuracy with the lowest number of selected features. However, the results in terms of the selected subset of features were acceptable. BSSA handled the most relevant subset of features to the accuracy classification more than choosing the minimum subset of features. In contrast, the other algorithms did the opposite. BFA found the minimum subset of features; however, the classification accuracy was lower than the one obtained by BSSA.

Conclusion

A binary Salp Swarm Algorithm (BSSA) was proposed in this research paper for selecting the most relevant subset of features, which enhance the classification performance of the network intrusion detection system (IDS). Extreme Learning Machine (ELM) showed that the proposed BSSA could select on average 25 features, with average accuracy ∓ 97.5%. For future studies, the proposed algorithm could enhance the classification performance for other case studies, such as email spam filtering, or medical diagnosis case studies.

References

Abba, S.I., Hadi, S.J., Sammen, S.S., Salih, S.Q., Abdulkadir, R.A., Pham, Q.B., & Yaseen, Z.M. (2020). Evolutionary computational intelligence algorithm coupled with self-tuning predictive model for water quality index determination. *Journal of Hydrology*, *587*. https://doi.org/10.1016/j.jhydrol.2020.124974

Afan, H.A., Allawi, M.F., El-Shafie, A., Yaseen, Z.M., Ahmed, A.N., Malek, M.A., Koting, S.B., Salih, S.Q., Mohtar, W.H.M.W., Lai, S.H., Sefelnasr, A., Sherif, M., & El-Shafie, A. (2020). Input attributes optimization using the feasibility of genetic nature inspired algorithm: Application of river flow forecasting. *Scientific Reports*, *10*(1), 4684. https://doi.org/10.1038/s41598-020-61355-x

Agrawal, S., & Agrawal, J. (2015). Survey on anomaly detection using data mining techniques. *In Procedia Computer Science, 60*, 708–713. https://doi.org/10.1016/j.procs.2015.08.220

Alzaidi, A.A., Ahmad, M., Ahmed, H.S., & Solami, E.Al. (2018). Sine-Cosine Optimization-Based Bijective Substitution-Boxes Construction Using Enhanced Dynamics of Chaotic Map. *Complexity*, *2018*. https://doi.org/https://doi.org/10.1155/2018/9389065

Balamurugan, V., & Saravanan, R. (2017). Enhanced intrusion detection and prevention system on cloud environment using hybrid classification and OTS generation. *Cluster Computing*, 1–13. https://doi.org/10.1007/s10586-017-1187-7

Bostani, H., & Sheikhan, M. (2015). Hybrid of binary gravitational search algorithm and mutual information for feature selection in intrusion detection systems. *Soft Computing*, *21*(9), 1–18. https://doi.org/10.1007/s00500-015-1942-8

Chandola, V., Banerjee, A., & Kumar, V. (2012). Anomaly detection for discrete sequences: A survey. *IEEE Transactions on Knowledge and Data Engineering*, *24*(5), 823–839. https://doi.org/10.1109/TKDE.2010.235

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, 39–43.

Huang, G.B., Zhu, Q.Y., & Siew, C.K. (2006). Extreme learning machine: Theory and applications. *Neurocomputing*. https://doi.org/10.1016/j.neucom.2005.12.126

Huang, G. qiu. (2017). Artificial memory optimization. *Applied Soft Computing Journal*, *61*, 497–526. https://doi.org/10.1016/j.asoc.2017.08.021

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. *Technical Report TR06, Erciyes University*, (TR06), 10. https://doi.org/citeulike-article-id:6592152

Kumar, M., Kulkarni, A.J., & Satapathy, S.C. (2018). Socio evolution &amp; learning optimization algorithm: A socio-inspired optimization methodology. *Future Generation Computer Systems*, *81*, 252–272. https://doi.org/10.1016/j.future.2017.10.052

Kuo, H.C., & Lin, C.H. (2013). Cultural evolution algorithm for global optimizations and its applications. *Journal of Applied Research and Technology*, *11*(4), 510–522. https://doi.org/10.1016/S1665-6423(13)71558-X

Lin, S.W., Ying, K.C., Lee, C.Y., & Lee, Z.J. (2012). An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection. *Applied Soft Computing Journal*, *12*(10), 3285–3290. https://doi.org/10.1016/j.asoc.2012.05.004

Lippmann, R., Haines, J.W., Fried, D.., Korba, J., & Das, K. (2000). 1999 DARPA off-line intrusion detection evaluation. *Computer Networks*, *34*(4), 579–595. https://doi.org/10.1016/S1389-1286(00)00139-0

Malik, A., Kumar, A., Salih, S.Q., Kim, S., Kim, N.W., Yaseen, Z.M., & Singh, V.P. (2020). Drought index prediction using advanced fuzzy logic model: Regional case study over Kumaon in India. *PLOS ONE*, *15*(5). https://doi.org/10.1371/journal.pone.0233280

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. https://doi.org/10.1016/j.advengsoft.2017.07.002

Mirjalili, S., Mirjalili, S.M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, *69*, 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007

Najeeb, R.F., & Dhannoon, B.N. (2018). A Feature Selection Approach Using Binary Firefly algorithm for Network Intrusion Detection System. *ARPN Journal of Engineering and Applied Sciences*, *13*(6), 2347–2352.

Rao, R.V., Savsani, V.J., & Vakharia, D.P. (2011). Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, *43*(3), 303–315. https://doi.org/10.1016/j.cad.2010.12.015

Salih, S.Q. (2019). A New Training Method Based on Black Hole Algorithm for Convolutional Neural Network. *Journal of Sourthwest Jiaotong University*, *54*(3), 1–10. https://doi.org/10.1002/9783527678679.dg01121

Salih, S.Q., & Alsewari, A.A. (2020). A new algorithm for normal and large-scale optimization problems: Nomadic People Optimizer. *Neural Computing and Applications*, *32*(14), 10359–10386. https://doi.org/10.1007/s00521-019-04575-1

Salih, S., Alsewari, A.A., Al-Khateeb, B., & Zolkipli, M.F. (2018). Novel Multi-Swarm Approach for Balancing Exploration and Exploitation in Particle Swarm Optimization. In *In Proceesdings of 3rd International Conference of Reliable Information and Communication Technology 2018 (IRICT 2018)*, 196–206.

Salih, S.Q., Alsewari, A.A., & Yaseen, Z.M. (2019). Pressure Vessel Design Simulation: Implementing of Multi-Swarm Particle Swarm Optimization. *Proceedings of the 8th International Conference on Software and Computer Applications*, 120–124.

Taha, A.M., Chen, S.D., & Mustapha, A. (2015). Multi-Swarm bat algorithm. *Research Journal of Applied Sciences, Engineering and Technology*. https://doi.org/10.19026/rjaset.10.1839

Tao, H., Salih, S.Q., Saggi, M.K., Dodangeh, E., Voyant, C., Al-Ansari, N., Yaseen, Z.M., & Shahid, S. (2020). A Newly Developed Integrative Bio-Inspired Artificial Intelligence Model for Wind Speed Prediction. *IEEE Access*, *8*, 83347–83358.

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A.A. (2009). A detailed analysis of the KDD CUP 99 data set. In *IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009*. https://doi.org/10.1109/CISDA.2009.5356528

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

Yang, X.S. (2009). Firefly algorithms for multimodal optimization. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 5792 LNCS*, 169–178. https://doi.org/10.1007/978-3-642-04944-6_14

Yang, X.S. (2010). A new metaheuristic bat-inspired algorithm. In *Studies in Computational Intelligence*, 284, 65–74. https://doi.org/10.1007/978-3-642-12538-6_6

Yaseen, Z.M., Al-Juboori, A.M., Beyaztas, U., Al-Ansari, N., Chau, K.W., Qi, C., Ali, M., Salih, S.Q., & Shahid, S. (2020). Prediction of evaporation in arid and semi-arid regions: a comparative study using different machine learning models. *Engineering Applications of Computational Fluid Mechanics*, *14*(1), 70–89. https://doi.org/10.1080/19942060.2019.1680576

Zhang, L., Shan, L., & Wang, J. (2016). Optimal feature selection using distance-based discrete firefly algorithm with mutual information criterion. *Neural Computing and Applications*, 1–14. https://doi.org/10.1007/s00521-016-2204-0

Noruzi, A. (2018). Folks thesauri or search thesauri: Why semantic search engines need Folks Thesauri?. *Webology, 15*(2), 1-10.